# Anomaly Detection Dataset for Industrial Control Systems

Alireza Dehlaghi-Ghadim* Mahshid Helali Moghadam†, Ali Balador‡ Hans Hansson§
School of Innovation, Design and, Engineering
Mälardalens University, Västerås, Sweden
Research Institute of Sweden (RISE)

Email: *alireza.dehlaghi.ghadim@(mdu.se/ri.se), †mahshid.helali.moghadam@mdu.se,
‡li.balador@mdu.se §hans.hansson@(mdu.se/ri.se)



***Abstract*—Over the past few decades, Industrial Control Systems (ICSs) have been targeted by cyberattacks and are becoming increasingly vulnerable as more ICSs are connected to the internet. Using Machine Learning (ML) for Intrusion Detection Systems (IDSs) is a promising approach for ICS cyber protection, but the lack of suitable datasets for evaluating ML algorithms is a challenge. Although there are a few commonly used datasets, they may not reflect realistic ICS network data, lack necessary features for effective anomaly detection, or be outdated. This paper presents the *'ICS-Flow'* dataset, which offers network data and process state variables logs for supervised and unsupervised ML-based IDS assessment. The network data includes normal and anomalous network packets and flows captured from simulated ICS components and emulated networks. The anomalies were injected into the system through various attack techniques commonly used by hackers to modify network traffic and compromise ICSs. We also proposed open-source tools, "ICSFlowGenerator" for generating network flow parameters from Raw network packets. The final dataset comprises over 25,000,000 raw network packets, network flow records, and process variable logs. The paper describes the methodology used to collect and label the dataset and provides a detailed data analysis. Finally, we implement several ML models, including the decision tree, random forest, and artificial neural network to detect anomalies and attacks, demonstrating that our dataset can be used effectively for training intrusion detection ML models.**

**Keywords:** Anomaly Detection, Dataset, Industrial Control System, Intrusion Detection, Cyberattack, Artificial Intelligence.

## I. Introduction

Industrial Control Systems (ICSs) are used to control various industrial processes, such as power plants, power grids, railways, and factories [1]. However, many of the protocols used lack encryption or authentication mechanisms, making them vulnerable to cyberattacks [2]. Recent years have seen an increase in cyber threats to industrial systems [3], with notable examples including the Stuxnet malware attack on Iranian Uranium enrichment facilities [4], hacking Ukrainian power grid [5], the Irongate malware attack on Siemence's ICS [6], and the Triton malware attack on the Saudi Arabian petrochemical plant [7]. Furthermore, Kaspersky ICS-CERT report [8] shows that 33.8% of ICS computers were compromised in 2021, indicating that cybersecurity is a major concern for modern ICSs. This is why ICS security has become a major topic of research in recent years [9], [10].

To protect industrial systems against the increasing risk of cyberattacks, Intrusion Detection Systems (IDSs) are introduced [11]. These systems monitor networks or hosts to detect cyberattacks or malicious activities [12], notifying security administrators or event management systems of any detected threats. An effective IDS should detect attacks accurately in a minimum time with the least number of false-positive alerts. To achieve these goals, using Machine Learning (ML) [13] and Deep Learning (DL) [14] in IDSs seems promising. However, due to security concerns and the risk of interrupting industrial processes, it is not feasible to test intrusion detection methods on operational industrial systems. Alternately, a dataset containing a range of cyberattacks could provide a benchmark to compare and evaluate intrusion detection algorithms [15] before deploying them in real-world scenarios.

Creating a comprehensive and representative dataset of intrusions in ICSs would provide an invaluable resource for the development and testing of new detection methods, as well as for training and validating ML algorithms. It would enable researchers to evaluate the effectiveness of different approaches to intrusion detection, identify areas for improvement, and enhance the overall robustness and effectiveness of cybersecurity measures for ICSs. However, despite its potential benefits, using such datasets can be challenging due to various obstacles. Here, we outline some of these challenges:

- Currently, only a limited number of datasets are available to evaluate ML-based anomaly detection in ICSs [16]. However, some of these datasets are based on unrealistic implementations.
- An important consideration for anomaly detection is the nature of the anomaly. For intrusion detection tasks, only a small subset of available datasets are relevant because they inject anomalies through cyberattacks.
- Some ICS testbeds lack crucial details or have implemented them incorrectly, which can impact the accuracy and effectiveness of anomaly detection methods.
- Some datasets are highly anonymized and cannot be shared due to confidentiality concerns, while others do not reflect current market trends [11].
- A significant challenge in training ML models for

anomaly detection is the lack of labeled data. Some ICS anomaly datasets are unlabeled, making it challenging to train effective models.
- Network trace labeling in available datasets is typically based on the automatic generation of synthetic network traces, which eliminate necessary details for accurately distinguishing between legitimate and malicious activity.
- Available datasets for anomaly detection tasks vary in the type of data logged, with some only recording process state variables, some only recording controlling commands, and others capturing entire network packets. This study focuses specifically on the latter type of data, which can limit the availability of suitable datasets.
- Due to the highly specialized nature of IDSs, transferring datasets between them is difficult, and customized datasets for each domain are preferred [16].

Moreover, in network-based intrusion detection for the ICS domain, the existing datasets are often unsuitable for training intrusion detection models since they are designed for labeling individual network traces, and most are based on the automatic generation of synthetic network traces [17]. Anomaly detection in ICSs differs from anomalous network packet detection because anomalies may exist across entire traffic, not just in a specific network packet. Therefore, there is a need to redefine the intrusion detection task to include anomaly detection in the network traffic pattern rather than just a specific network packet. Therefore, we aim to provide our dataset in both forms of raw packet files and network flow records, which is suitable for studying network flow patterns.

This article aims to provide an intrusion detection dataset that addresses the mentioned issues by providing a realistic benchmark to compare ML-based IDSs in the ICS domain. To achieve this goal, we leverage the ICSSIM framework [18], a tool for simulating customized virtual ICS security testbeds, to investigate cyber threats and attacks. In particular:

1) We implement four types of cyberattacks: 'Reconnaissance', 'Distributed Denial of Service' (DDoS), 'false data injection' using Man-in-the-Middle (MitM) technique, and 'Replay' attacks on an ICS testbed.
2) We develop ICSFlowGenerator, a reusable open-source tool to extract network flow features from raw network packets.
3) We propose the publicly available 'ICS-Flow' dataset[1] as a public resource. This dataset is unique in several ways:
    a) To further enhance the dataset's practicality, we have implemented multiple labeling strategies.
    b) It contains a diverse set of network flow features that capture different aspects of ICS network behavior.
    c) The anomalies in the dataset are due to the realistic implementation of network attack scenarios. This differs from other datasets that employ the creation of synthetic network anomalies, making our dataset more representative of real-world conditions.
    d) We offer a complete dataset without any anonymization to provide comprehensive support for anomaly detection, including a network flow dataset, process state snapshots, attack logs, and ICS components logs.

Moreover, to demonstrate the practicality of our proposed dataset, we evaluate various ML-based intrusion detection models on the 'ICS-Flow' dataset and compare their performance.

The remainder of this paper is organized as follows. Section II presents the current state of the art for available testbeds. Section III proposes the ICS-Flow intrusion detection dataset by implementing a scenario using our proposed testbed in another paper and injecting different types of attacks. Section IV provides a detailed analysis of the generated dataset. Section V shows the implementation of anomaly detection techniques on the proposed dataset. Section VI compares and discusses the results of the anomaly detection methods. Section VII identifies the primary threats to the study's validity and applied mitigation techniques. Finally, Section VIII concludes the paper and provides future research directions.

## II. Related Work

This section briefly surveys related work on network intrusion detection datasets to highlight the prerequisites for future datasets. A comprehensive survey of available datasets and their strengths and weaknesses for anomaly detection is beyond the scope of this paper. However, there are several surveys in this area, such as [19]–[21], to mention a few.

Many datasets are available for anomaly detection in computer networks, but the KDD Cup '99[2] dataset is the most well-known and widely used. In 1998, a DARPA dataset [22] was collected using TCPdump of US air force LAN simulation. A subset of the DARPA dataset used to extract features by the Massachusetts Institute of Technology (MIT) Lincoln Laboratory results in KDD Cup '99 dataset [23]. This dataset contains 41 features per connection, categorized into three groups: basic features, traffic features, and content features. These features are computed for normal and attack records, belonging to four types of attacks: User to Root (U2R), Remote to Local attack (R2L), Probing, and Denial of Service attack. However, there is some criticism against this dataset. Firstly underlying network traffic backs to a few decades ago. Secondly, the dataset records are not refined well since there are many redundant records, and record classes are not balanced across the dataset [24].

In 2009, Mahbod Tavallaee et al. [25] conducted a statistical analysis on the KDDCUP'99, finding that some issues with the dataset adversely affected the anomaly detection experiences. They enhanced the KDD Cup dataset by resolving major

[1]https://www.kaggle.com/datasets/alirezadehlaghi/icssim

[2]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

criticism such as duplicate or irrelevant records and data asymmetry, which led to a new dataset titled NSL-KDD[3]. This dataset collected approximately 150,000 data points and divided them into training and test subsets with the same attributes as KDD-CUP 99. This means that this data set does not represent modern network attack scenarios.

Morris et al. [26] introduced comprehensive datasets encompassing network traffic, process control, and process measurement features gathered from two laboratory-scale ICSs - a gas pipeline and a water storage tank - utilizing the Modbus over serial lines protocol. Their research involved executing various attacks on their experimental setups, such as reconnaissance, response injection, command injection, and Denial-of-Service (DoS) attacks. Nonetheless, it is important to note that their network traffic dataset solely comprises the features of Modbus commands utilized within the ICSs.

To generate a dataset that reflects modern network traffic scenarios and contains varieties of low-footprint attacks, the Australian centre for cybersecurity researchers generated the UNSW-NB 15[4] dataset [27]. This dataset contains 41 features, categorized into six groups: flow, basic, content, time, additional general, and connection features. The first four feature categories were derived directly from dumped traffic, and the auxiliary C# program was developed to compute the other two additional groups of features. They used IXIA PefectStorm Tool [5] to generate normal and abnormal traffic; therefore, despite the simulation of 9 different attack types, their attack implementation restricts to predefined synthetic attack types in the IXIA tool.

Canadian Institute for cybersecurity proposed the CICIDS2017[6] dataset based on profiling the human users' abstract behavior [11]. To generate realistic background traffic, they profiled the behavior of 25 human users, considering eight famous attacks, namely brute force, DoS, botnet, and Heartbleed. Finally, they extracted more than 80 network parameters and then, using Random Forest Regressor, selected the best features for each attack. Although the CICIDS2017 dataset has been used in many intrusion detection experiments, it is not suitable for ICS since it does not reflect the ICS network traffic patterns.

Mathur and Tippenhauer created a small-scale, but fully operational water treatment system (SWaT) testbed for cybersecurity research [28]. Since SWaT is large and geographically dispersed, Programmable Logic Controllers (PLC) use Wireless and wired Fieldbus communications to control the sensors and actuators. Several attack scenarios are defined for this testbed which targets single or multi-points on single or multi-stages. They released the final dataset containing process state variables, selected packet features, and logs of performed attacks [29][7].

Gómez et al. presented Electra[8], an anomaly detection dataset [16] for heterogeneous ICS scenarios. They selected the railway industry, and the Electra dataset was conducted using network traffic generated from normal and attack situations at a traction substation. Although this work has some highlighted features, such as using realistic devices (PLCs and SCADA), network protocols (Modbus), and scenarios, their extracted features are limited to Modbus-related features such as function code and errors. Therefore, this dataset lacks general network features to detect intrusion.

TON_IoT dataset, presented by Alsaedi et al. [31][9], comprises Telemetry data of IoT/IIoT devices collected in a controlled environment during both normal operations and in the presence of different cyber-attacks. In addition, the dataset also includes operating systems logs (such as disk or memory usage and process information) and network traffic of an IoT network, acquired from a realistic representation of a medium-scale network at the Cyber Range and IoT Labs at the UNSW Canberra (Australia). This dataset can be employed to create and assess data-driven intrusion detection systems for IoT and IIoT environments. While this dataset includes various attack types, it falls short in terms of comprehensive network features. It solely consists of fundamental network features like the network addresses, the amount of transferred traffic, and a few protocol-specific features such as HTTP, DNS, and SSH. In order to gain a more holistic understanding of network behavior during attacks, additional enriched network features should be incorporated into the dataset. Moreover, although the TON_IoT Dataset provides telemetry data for IoT/IIoT devices, it differs considerably from the controlling system's domain targeted by our article, where controllers (PLCs) issue controlling commands in a loop.

A Water Distribution Testbed (WDT) proposed by Faramondi et al. [32] was used to generate an intrusion detection dataset for ICSs. They emulated water flowing between 8 tanks as Hardware in a Loop (HIL) and used miniCPS [33] as a simulation tool to simulate the control system and networking infrastructure. This testbed had real hardwired subsystems, virtually connected to a simulated one. Multi-controller implementation is a key component of this study since it enables attacks that target communication between controllers. The proposed dataset includes both system state variables and network data to reveal attack consequences on physical processes and network traffic. This dataset has some issues, including the fact that the network dataset only contains packet-based features, which means network flow parameters are not included.

Table I summarizes the differences between the available datasets for intrusion detection experiments in industrial systems, including variations in logged data, implemented attacks, and industry domain. Furthermore, the dataset's quality can be significantly affected by the specific implementation of the testbed and attacks. Consequently, the next section offers

[3]http://www.unb.ca/cic/datasets/nsl.html
[4]https://research.unsw.edu.au/projects/unsw-nb15-dataset
[5]http://www.ixiacom.com/products/perfectstorm
[6]https://www.unb.ca/cic/datasets/ids-2017.html
[7]https://itrust.sutd.edu.sg/itrust-labs_datasets/dataset_info/
[8]http://perception.inf.um.es/ICS-datasets/
[9]https://research.unsw.edu.au/projects/toniot-datasets

Table I
SUMMARY OF INTRUSION DETECTION DATASETS RELATED TO OUR PROPOSED DATASET

| Dataset | Year | Data Source | Records | Attacks |
|---|---|---|---|---|
| DARPA [22] | 1998 | Network traffic and audit logs collected using a simulated network (air force LAN) | Raw Pcap files of simulated network dump | - U2R<br>- R2L<br>- DoS<br>- Probe |
| KDD Cup '99 [23] | 1999 | A subset of the DARPA dataset used to extract features | Connection features (41 Features):<br>- Basic features<br>- Traffic features (same host/service)<br>- content feature | - U2R<br>- R2L<br>- DoS<br>- Probe |
| NSL-KDD [25] | 2009 | Enhanced version of KDD Cup '99 dataset using statistical analysis | Connection features (41 Features):<br>- Basic features<br>- Traffic features (same host/service)<br>- content feature | - U2R<br>- R2L<br>- DoS<br>- Probe |
| Gas Pipeline [26], [30] | 2014 | Two laboratory-scale ICS<br>- a gas pipeline<br>- a water storage tank | - Network traffic,<br>- Process control,<br>- Process measurement features | - Reconnaissance<br>- Response Injection<br>- Command Injection<br>- Denial-of-Service |
| UNSW-NB15 [27] | 2015 | Used IXIA PefectStorm Tool to generate normal and malicious traffic | Flow features (47 Features):<br>- Flow features<br>- Basic features<br>- Content features<br>- Time features<br>- additional general features<br>- additional connection features | - Fuzzers<br>- Backdoors<br>- DoS<br>- Exploits<br>- Generic<br>- Reconnaissance<br>- Shellcode<br>- Worms |
| SWaT [29] | 2016 | Small-scale water treatment system (SWaT) | Packet features (18 Features):<br>- Packet basic features<br>- TCP flags<br>- Modbus (code, value)<br>Process state Variables (51 features) | Multiple customized attack scenarios, with various attacks per scenario |
| CICIDS2017 [11] | 2017 | Profiling the abstract behaviour human users | Flow features (80 Features):<br>- Flow features<br>- Basic features<br>- Content features<br>- Time features | - Brute Force FTP, SSH<br>- DoS, DDoS<br>- Heartbleed<br>- Infiltration<br>- Botnet |
| Electra [16] | 2019 | Network traffic generated from normal and attack situations at a traction substation | Modbus features (10 Features):<br>- Timestamp<br>- IP and MAC addresses<br>- Function code and data<br>- Error, memory address | - False Data Injection<br>- Replay<br>- Reconnaissance |
| TON_IoT [31] | 2020 | Telemetry data of IoT/IIoT Sevices | Telemetry data of Devices<br>Operating Systems logs<br>Network traffic<br>- General features<br>- HTTP, DNS, SSH features | - MitM - Password<br>- Scan -Ransomware<br>- (D)DoS - Injection<br>- Backdoor - XSS |
| WDT [32] | 2021 | Emulated water distribution testbed, Including HIL and controlling network | Packet features (14 Features):<br>- Packet basic features<br>- TCP flags<br>- Modbus (code, value)<br>Process state Variables (21 features) | - MitM<br>- Scan<br>- DoS<br>- Physical attacks |
| ICS-Flow [This article] | 2022 | Emulated bottle filling factory testbed, Including HIL and controlling network | Flow features (54 Features):<br>- Flow features<br>- General features<br>- TCP features<br>- Extended labeling features<br>Process state variables<br>Attack logs<br>Network packets | - MitM (Injection)<br>- Scan<br>- DoS<br>- Replay Attack |

more information on the development of the ICS-Flow dataset to illustrate its potential value as a resource for evaluating industrial IDSs.

## III. ICS-Flow Dataset

In this section, we introduce the ICS-Flow dataset, which is designed as an evaluation dataset for ML-based IDSs intended for industrial systems. We first present our industrial testbed environment. Then, we describe selected attack types and explain how the attacks are implemented. Next, we detail the creation of the intrusion detection dataset, including the network features derived and the network packet processing techniques used to generate network flow features. Finally, we introduce the labeling process and our different strategies for labeling.

### A. Testbed Environment

We used the bottle filling factory simulation provided by ICSSIM [18] as a virtual testbed to investigate cyber threats and attacks. The simulation, which is illustrated in Figure 1, includes an ICS that controls the equipment within the factory - such as pipes, valves, a conveyor belt, and a water tank - to fill empty bottles with water from the tank. The input valve regulates the water level in the tank to ensure that it remains within the permissible range, while the output valve controls the flow of water for filling bottles. The conveyor belt engine ensures empty bottles are positioned correctly beneath the filler. PLC-1 reads the tank water level and pipe water flow sensors and sends control commands to turn input and output valves *'On'* and *'Off'*. PLC-2 monitors the water level in filling bottles and the distance between the filler and the next bottle using sensors. It also issues control commands for the conveyor belt based on sensor readings and communications with PLC-1.

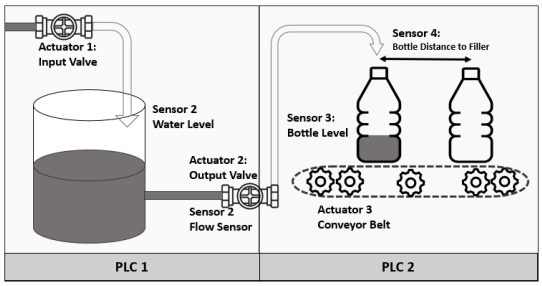


Figure 1. The Sample bottle filling factory presented in ICSSIM framework [18]

We enhanced the virtual testbed from [18] by introducing new types of Human Machine Interfaces (HMI), control logic, and attacks. The final architecture of the testbed, depicted in Figure 2, consists of two PLCs that manage a bottle filling factory and three HMIs that supervise the system and send manual operational commands. All the components of the ICS are connected through a network, and Table II provides detailed network configurations of the ICS nodes. In this testbed, all ICS components operate on Docker containers[10] in automatic mode since the simulation is conducted using ICSSIM. However, the use of automated HMIs in this testbed may create uniform network traffic during regular operation, which could decrease the difficulty of anomaly detection in the dataset. To address this concern, we added random parameters to the HMIs' behavior.

Table II
ICS nodes network configurations in our testbed

| Name | Mac | IP |
|---|---|---|
| PLC1 | 02:42:c0:a8:00:0b | 192.168.0.11 |
| PLC1 | 02:42:c0:a8:00:0c | 192.168.0.12 |
| HMI1 | 02:42:c0:a8:00:15 | 192.168.0.21 |
| HMI2 | 02:42:c0:a8:00:16 | 192.168.0.22 |
| HMI3 | 02:42:c0:a8:00:17 | 192.168.0.23 |
| Attacker | 02:42:c0:a8:00:29 | 192.168.0.41 |

- HMI1: constantly reads all controlling signals.
- HMI2: sends write commands based on a predefined scenario to simulate HMIs' user commands.
- HMI3: sends eligible write commands with uniform random values.

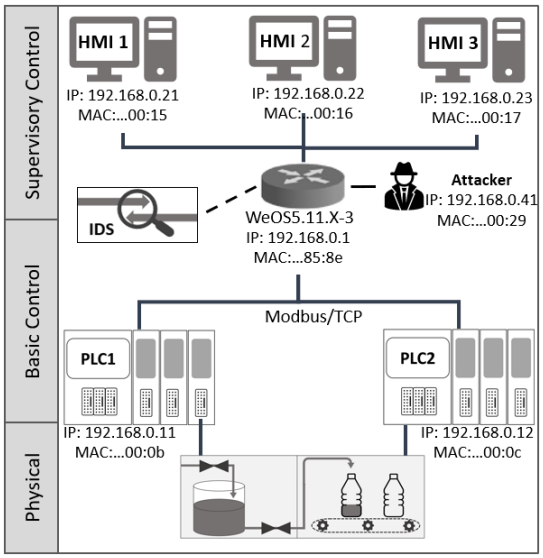


Figure 2. The testbed architecture

In order to simulate network communication realistically, the Modbus protocol has been employed for all communications. Modbus is a communication protocol widely used in operational ICSs [34], originally designed for PLCs and later adopted as the de facto standard for connecting industrial electronic devices [35]. We use Modbus TCP to ensure compatibility with a variety of industrial devices. PLCs function as Modbus-TCP servers, while ICS components establish communication by creating Modbus-TCP client instances that communicate with Modbus-Servers.

[10]https://www.docker.com/

To launch cyberattacks against the testbed, we connect an attacker node to the ICS network. It is assumed that the attacker has access to internal network traffic because many ICS systems now offer remote monitoring for industrial equipment, allowing attackers to access the ICS internal network through infected remote HMIs. However, the attacker does not have prior knowledge of the control system or controlling variables. Therefore, the attacker must sniff the network and gather the necessary information for complex attacks.

After setting up the testbeds, we captured network data to create the ICS-Flow dataset, which includes the communication between ICS components. We captured network packets by running TCPdump on the switch, and this data can be employed to train IDSs on the switch or as a standalone component within the control network. Implementing the intrusion detection module within the control network facilitates the detection of any dubious control activities and allows for the monitoring of traffic among ICS components. This is particularly useful in identifying attacks that launched internally or attacks that have already penetrated the network perimeter.

### B. Attacks

Protocols used in ICSs are vulnerable due to their lack of authentication, communication encryption, and integrity checks [16]. These weaknesses allow attackers to eavesdrop on or alter network transmissions, potentially disrupting ICS operations. To exploit these vulnerabilities, we considered four attack types to discover the network, sniff or modify packets and disrupt ICS operation. We selected the attacks regarded as common attack types in previous studies [16], [26], [32]. Moreover, the *'MITRE ATT&CK'*[11], which is a globally-accessible knowledge base of adversary techniques based on real-world observations, and European Union Agency for Cybersecurity (ENISA) reports[12] [36] consider these attacks to be common cyberattacks on ICSs.

Our team has developed a Python script that automates the execution of attacks on the control system. Using this script, the attacker node can quickly launch attacks based on a predefined scenario. We have also designed the script to allow the control system sufficient time to recover from any destabilization caused by the attacks, ensuring that the system remains stable and secure. The details of implemented attacks are listed below.

*1) Reconnaissance Attack:* Intruders often begin with a reconnaissance attack, which involves gathering valuable system information as a preliminary step towards launching further attacks [37]. A reconnaissance attack is passive since the attacker merely captures information rather than disrupting the victim's functionalities. This attack does not impact industrial operations, although its trace on the network traffic can still reveal it. In our implementation, the intruder discovers the network by collecting information such as IP addresses, media access control addresses (MAC), or open ports.

We have implemented the Reconnaissance attack with two different scenarios: IP-Scanning and Port-Scanning. In the first scenario, we utilize Ettercap[13], a free and open-source network security tool, and a Python script using the Scapy[14], a packet manipulation tool for networks, to broadcast Address Resolution Messages (ARP) to discover the alive network nodes. In the second scenario, we use NMap[15], a free and open-source utility for network discovery, to gather information about the hosts and ports on them in the network. NMap uses a port scanning technique to find vulnerabilities on victim hosts. Each scenario could reveal information about the target network with different traces on the dataset.

*2) Replay Attack:* A commonly used attack in control systems is the replay attack, which involves exploiting valid network packets captured during normal system operation by maliciously retransmitting them [16]. In this type of attack, the attacker sniffs the network passively to collect valid packets and then actively sends the recorded packets frequently to other nodes, disrupting the system's normal behavior. Since replaying valid packets at inappropriate times can lead to unexpected results, this attack does not require in-depth knowledge of network traffic packets. It is essential to note that this type of attack can be highly damaging, even with minimal information about the system.

We have implemented the replay attack using Scapy by developing a Python script that executes the attack in two phases. In the first phase, the attacker uses ARP poisoning and the MitM technique to sniff network packets for 15 seconds. These packets are later replayed three times (45 Seconds) by the attacker to disrupt the control system's regular operation. However, replaying the exact same messages is not possible since every TCP connection depends on two 32-bit random sequence numbers generated by the client/server, and packets with duplicate sequence numbers will be rejected. As a solution, a replay attack on a TCP connection reuses only TCP payloads. Therefore, the attacker uses intercepted IP addresses, ports, Modbus commands, and arguments to create a new TCP-Connection to perform a replay attack.

*3) Distributed Denial of Service (DDoS) Attack:* In order to disrupt the industrial operation, attackers can flood the network or service with a large number of packets or service requests, resulting in a denial of service on the victim component [38]. Meanwhile, using multiple attackers makes the attack more effective. The DDoS target can be limited bandwidth, storage, or computing power.

We leverage the network addresses collected during the reconnaissance attack and the Modbus addresses gathered by sniffing the network to conduct a DDoS attack. In this attack, we use the 'DDoSAgent' class from ICSSIM to generate a massive flood of reading requests to the PLCs. Our implementation involves a script that spawns 800 instances of the

[11]https://attack.mitre.org/techniques/ics/

[12]https://www.enisa.europa.eu/publications/good-practices-for-security-of-iot

[13]https://www.ettercap-project.org/

[14]https://scapy.net/

[15]https://nmap.org/

'DDoSAgent' class, which relentlessly bombards the PLCs with read requests for 60 seconds. This results in significant delays in the ICS network communication during the attack.

*4) MitM Attack:* We use a MitM attack to inject false data into the controlling system. The false data injection attack is one of the most critical malicious cyberattack [39] that involves injecting incorrect data into the controlling system, compromising the communication in ICSs. Using the MitM technique, an attacker intercepts or manipulates communications between two ICS components while they believe they are interacting directly with each other [40]. As part of our testbed, we employ MitM techniques to inject false data into the ICS system. Therefore, we refer to this attack as the false data injection attack and the MitM attack.

To execute this attack, the attacker first performs ARP poisoning on the ICS component, redirecting network packets to their node. Next, the attacker intercepts the packets and modifies the Modbus write requests and read responses by multiplying their values with a specific factor. The attacker then sends the manipulated packets to the intended destination. Finally, ARP messages are sent to the ICS component to clear routing tables and erase any evidence of the attack. A custom Python script based on the Scapy library was used to carry out the attack. The attack was repeated several times for a duration of 30 seconds each time, resulting in a 10% error in the Modbus read response data.

### C. Feature extraction

Existing literature in the network security domain captures the network either in packet-based or flow-based format [21]. Although packet-based datasets provide detailed information regarding network anomalies, AI analysis of these voluminous datasets is time-consuming, making it challenging to develop an IDS. An alternative is using the network flows format, which aggregates packets in a time interval with certain properties. The typical shared properties are network protocol, Source IP, Destination IP, Source Port, and Destination Port [41]. The flow definition varies depending on the testbed configuration. We also use a customized definition of flow since our testbed uses the Modbus protocol, which always uses port number 502 as the default port. Therefore, we define a flow as an aggregated record of network packets within a time interval with a similar source address, a destination address, and network protocol. Based on the time setting of our testbed, we considered 500 ms as an interval for flows in this experiment.

To generate a network flow dataset, we developed the ICS-FlowGenerator tool using the Scapy library, which is available in our public GitHub repository[16]. The ICSFlowGenerator tool follows the procedure presented in Algorithm 1. It iterates through network packets in a dumped PCAP file and generates flow features based on the interval option. The output is a flow file in CSV format, containing flow-based features. For network flows that are not TCP-based, the TCP payload features columns are empty. In addition to TCP traffic, our dataset includes ARP messages to detect MitM attacks. To account for the lack of IP addresses in ARP messages, we generate ARP flows based on their MAC addresses.

[16]https://github.com/AlirezaDehlaghi/ICSFlow

Table III
FLOW FEATURES

| # | Feature | Description | Type |
|---|---|---|---|
| 1, 2 | (s/r)Address | Sender/Receiver IP address or MAC address of flow | str |
| 3 | protocol | Packet type and Network Protocol (ARP, IPV4-TCP, IPV4-UDP, ...) | str |

**Algorithm 1** - ICSFlowGenerator

**Inputs:** *PcapFile, Interval*
**Output:** *FlowFile*
**procedure** GENERATEFLOWS
  $flows \leftarrow \varnothing$
  **foreach** *packet*, *time* **in** *PcapFile*
    // discard LCC frames without protocol
    **if** '*type*' **not in** *packet.fields* **then**
      **Conitnue**
    $src \leftarrow$ ***Min***$(packet.src, packet.dst)$
    $dst \leftarrow Max(packet.src, packet.dst)$
    $protocol \leftarrow packet.protocol$
    **if** $flows[src, dst, protocol]$ **then**
      $flow \leftarrow flows[src, dst, protocol]$
    **else**
      // a flow is a collection of packets
      $flow \leftarrow new\ flow(src, dst, protocol)$
    // flush flows that was open for windows lenght
    **if** $time - flow[0].time > Interval$ **then**
      **Print(**$flow$**) in** $FlowFile$
      $flow \leftarrow \varnothing$
    $flow.add(packet, time)$
    $flow.UpdateFeatures()$
  **foreach** *flow* **in** *flows*
    // flush rest of flows
    **Print(**$flow$**) in** $FlowFile$

The flow dataset contains 54 columns, including 50 features and 4 label columns. Feature columns are categorized into three categories: flow features, general features, and TCP features. Table III presents the flow features, including the source address, destination address, and network protocol. Table IV introduces the general features category with 16 features that are shared for all network flows regardless of the protocol type. This dataset also contains 20 TCP header features outlined in Table V. These features are extracted from TCP headers and contain statistics about Flags, time to live, TCP window, and delays.

### D. Labelling

The goal of labeling is to provide information about the context of data. Although the quality of labelling data is not precisely defined, several articles have stressed the importance

Table IV
GENERAL FEATURES

| # | Feature | Description | Type |
|---|---|---|---|
| 4 | start | Time stamp of the first packet in the flow | float |
| 5 | end | Time stamp of the last packet in the flow | float |
| 6 | startOffset | Time offset of first packet in the flow relative to first packet in the dataset | float |
| 7 | endOffset | Time offset of last packet in the flow relative to first packet in the dataset | float |
| 8 | duration | Flow interval, the difference between end and start time stamps | float |
| 9, 10 | (s/r)Packets | Sent or received packets count within the flow | int |
| 11, 12 | (s/r)BytesMax | Maximum bytes sent or received in one packet within the flow | int |
| 13, 14 | (s/r)BytesMin | Minimum bytes sent or received in one packet within the flow | int |
| 15, 16 | (s/r)BytesAvg | Average sent or received bytes per packet within the flow | float |
| 17, 18 | (s/r)Load | Sent or received bits per second | float |
| 19, 20 | (s/r)PayloadMax | Maximum bytes sent or received as a payload in one packet within the flow | int |
| 21, 22 | (s/r)PayloadMin | Minimum bytes sent or received as a payload in one packet within the flow | int |
| 23, 24 | (s/r)PayloadAvg | Average payload bytes of sent or received packets | float |
| 25, 26 | (s/r)InterPacket | Average send or receive inter-packet arrival time | float |

Table V
TCP FEATURES

| # | Feature | Description | Type |
|---|---|---|---|
| 27, 28 | (s/r)ttl | Average sending or receiving time-to-live | float |
| 29, 30 | (s/r)AckDelayMax | Maximum interval between packets and their acknowledge in sender or receiver node | float |
| 31, 32 | (s/r)AckDelayMin | Minimum interval between packets and their acknowledge in sender or receiver node | float |
| 33, 34 | (s/r)AckDelayAvg | Average interval between packets and their acknowledge in sender or receiver node | float |
| 35, 36 | (s/r)AckRate | Rate of sent or received packets contains ácknowledgeflag | float |
| 37, 38 | (s/r)FinRate | Rate of sent or received packets contains 'Finish' flag | float |
| 39, 40 | (s/r)PshRate | Rate of sent or received packets contains 'Push' flag | float |
| 41, 42 | (s/r)RstRate | Rate of sent or received packets contains 'Reset' flag | float |
| 43, 44 | (s/r)UrgRate | Rate of sent or received packets contains 'Urgent' flag | float |
| 45, 46 | (s/r)AckRate | Rate of sent or received packets contains 'Synchronisation' flag | float |
| 47, 48 | (s/r)WinTCP | sender or receiver average TCP window advertisement | float |
| 49, 50 | (s/r)FragmentRate | sender or receiver Fragmentation rate of TCP packets | float |

of accurate labeling as a crucial component of producing high-quality network traffic datasets [42]. Human-guided or automatic labeling are two possible strategies to label datasets. In this study, automatic labeling was employed since it is fast, requires little expertise, and is easy to adapt for all types of attacks [17]. The data required for training ML models are different depending on how we define ML-based intrusion detection. In this article, we aim to support two types of definitions, anomalous record detection, and intrusion detection.

In anomalous record detection, the network trace dataset is labeled as normal or malicious, and the ultimate goal is to classify dataset records into different categories. Several approaches exist to do this classification, including binary classifications for attack detection or multi-class classifications for attack identification. However, the labeled dataset is always required as a training set in supervised approaches or an evaluation set for unsupervised ML methods, as record labels are available in some datasets such as [11], [16], [25], [27]. Automatically recognizing anomalous network traces from normal traffic is challenging [17]. To overcome this, we use two different strategies for labeling; namely, Injection Timing (IT) [43] and Network Security Tools (NST) [17]. In the former strategy, we consider all network traffic during an attack anomalous, while in the latter strategy, we only consider network traffic coming from or going toward the attacker node as an anomaly.

In intrusion detection, the ultimate objective is to find attack occurrences by analyzing unlabeled datasets and system logs. In other words, we do not look for anomalous flows but rather for anomalous system behavior in a time period. A log data of attack occurrence, along with unlabeled network and system data, is required for this task to verify the intrusion detection results. As pointed out in [29], a combination of unlabeled network traces and the attack log file forms an unsupervised or semi-supervised dataset for attack detection.

By providing attack occurrence log files and labeled network traces using IT and NST strategies, we provide a dataset with the flexibility to perform all described anomaly detections. The labelling features are listed in table VI. However, we believe that unlabeled datasets are more realistic for industrial implementation since the final mission of IDSs is attack detection, and labeled flows are not available in operational ICSs.

Finally, as a complement to the class labels using the IT and NST labeling strategy in binary and multi-class values, we present the attack log file in Table VII to provide a ground for further analysis.

## IV. DATASET ANALYSIS

This section will provide a detailed account of how we implemented our experiment and conduct a thorough analysis of the ICS-Flow dataset. The analysis will include statistical information about the dataset records, an explanation of the final format of the dataset files, and using 2D representations to visualize the dataset.

Table VI
LABEL FEATURES

| # | Feature | Description | Type |
|---|---|---|---|
| 51 | IT-B-Label | 0 if record is normal, 1 if record is malicious (using IT labeling methodology) | int |
| 52 | IT-M-Label | 'normal' if record is normal, 'attack-name' if record is malicious (using IT labeling methodology) | str |
| 53 | NST-B-Label | 0 if record is normal, 1 if record is malious (using NST labeling methodology) | int |
| 54 | NST-M-Label | 'normal' if record is normal, 'attack-name' if record is malicious (using NST labeling methodology) | str |

Table VII
ATTACK LOG FILE

| Field Name | Description |
|---|---|
| **Attack** | Shows the attack that applied in the dataset |
| **Start time** | Attack start time stamp in Unix time format |
| **End time** | Attack end time stamp in Unix time format |
| **Attacker IP** | IP address of the attacker |
| **Attacker MAC** | IP address of the attacker |
| **Extra Info** | Contains extra information about the applied attack |

The network configuration was optimized to accommodate the burst of network packets caused by a DDoS attack. To capture all packets, we increased the buffer size of the Switch to 400MB and utilized TCPdump to capture traffic. The testbed was operational for three hours, with the first hour being attack-free. This period without attacks provided normal samples for anomaly detection techniques that use semi-supervised AI. Over the following two hours, attacks were conducted randomly and intermittently. We included gaps between attacks to prevent attack overlaps, considering the time required for system recovery after each attack.

We captured a total of 2GB of raw network traffic, comprising over 25 million packets. Subsequently, we analyzed this traffic using ICSFlowGenerator and generated a final dataset of 45719 network flows. The detailed statistics on the labeled flows using IT and NST labeling strategies are presented in Table VIII and Figure 3. These statistics reveal that the number of attack flows identified using the NST approach is lower than that of the IT approach, as the NST method has stricter criteria for classifying a flow as an attack. Furthermore, as part of the experiment, we logged process variables using the PLCs' logger. These log files are available in CSV format and can aid researchers in identifying anomalies based on process variable analysis. Lastly, the dataset comprises four files, namely, the raw Pcap file, the labeled network flow dataset, the attack log file, and process state variables, all of which are publicly accessible in our repository[17].

Visualizing dataset records help us identify data patterns. However, a large number of normal network flows in this

[17]https://www.kaggle.com/datasets/alirezadehlaghi/icssim

Table VIII
FLOW STATISTICS USING IT AND NST LABELING STRATEGIES IN ICS-FLOW DATASET

| Attack type | # of IT Flows | # of NST Flows |
|---|---|---|
| Normal | 30236 | 36706 |
| IP-Scan | 712 | 192 |
| Port-Scan | 3235 | 1944 |
| Replay | 4300 | 2358 |
| DDoS | 4221 | 1934 |
| MitM | 3014 | 2584 |
| Total | 45719 | 45719 |

unbalanced dataset obscure the attack network flows. To alleviate this problem, we chose to visualize only the second half of the dataset, which contains a mix of normal and attack records. We applied two techniques to show multi-dimensional data in the 2D diagrams. Firstly, we applied Principal Component Analysis (PCA) [44], a popular technique for analyzing large datasets with many features. Although PCA is primarily used for reducing data dimensions, extracting the first two or three features enables us to demonstrate data in 2D or 3D presentation. Figure 4-A demonstrates that while PCA is helpful for visualizing data, it does not clearly show data clusters. Secondly, to visualize the high-dimensional data, we employed t-Distributed Stochastic Neighbor Embedding (t-SNE) [45], a statistical method known for its effectiveness in dimensionality reduction. We conducted a grid search over a range of perplexity values [30, 50, 100, 250, 500, 1000], and chose a perplexity of 250 based on its optimal performance. Figure 4-B illustrates the 2D presentation of the ICS-Flow dataset using t-SNE with perplexity 250. Our analysis revealed that DDoS, Port-Scanning, and MitM attacks form distinct clusters. However, distinguishing between replay attacks and normal attacks is challenging since replay attacks are designed to mimic the behavior of a normal system. Furthermore, IP-Scan attack records are merged with other groups, which we will analyze and explain in the results and discussion section.

## V. ML-BASED ANOMALY DETECTION MODELS

This section illustrates how ML techniques can be applied to detect and identify intrusions using our dataset. We conducted two separate experiments with the dataset: intrusion detection and identification. The objective of the intrusion detection experiment was to identify attack flows irrespective of their attack type. To achieve this, we classified records into normal or attack flows. In the intrusion identification experiment,

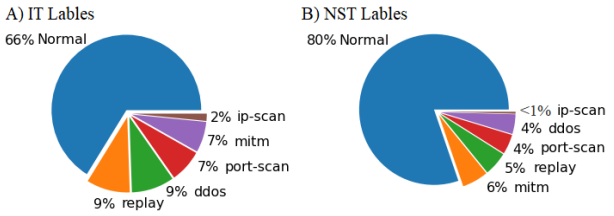


Figure 3. IT and NST Class labels distribution over ICS-Flow dataset.

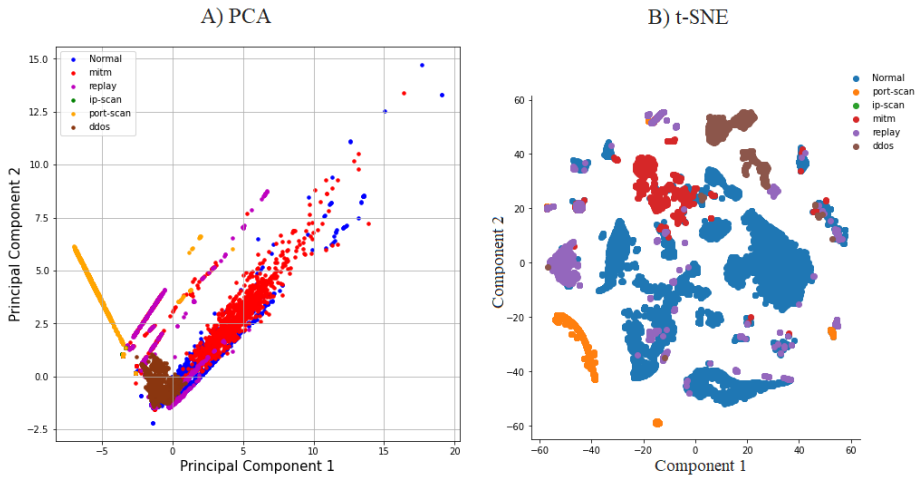


Figure 4. 2D presentation of ICS-Flow dataset records using PCA and t-SNE techniques

we aimed to identify the specific attack type, which could help devise an effective mitigation strategy. The performance evaluation of the ML methods on the 'ICS-Flow' dataset was conducted using a four-phased methodology, depicted in Figure 5. In the rest of this section, we will talk about actions performed in these phases in detail.

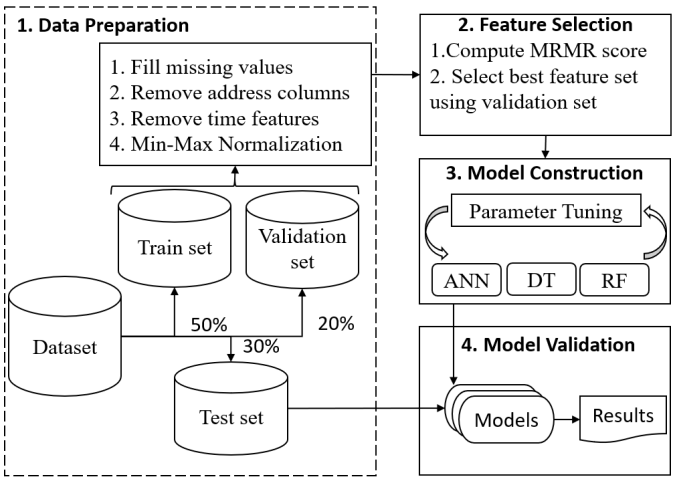


Figure 5. Methodology to assess ML-methods

### A. Data Preparation

During the dataset preprocessing phase, the following steps were conducted:

1) In the dataset, certain rows have missing column values due to the inability to compute TCP features for non-TCP protocol flows like UDP flows. To address this, we have replaced the missing values with a placeholder value of '0'.
2) For the attack detection and identification experiments, the 'NST-B-Label' and 'NST-B-Label' columns were chosen as output variables, respectively.

Table IX
MRMR SCORE OF SELECTED FEATURE

| Rank | Feature | MRMR Score |
|---|---|---|
| 1 & 2 | rBytesAvg & sBytesAvg | 0.4309 - 0.0559 |
| 3 & 4 | sFinRate & rFinRate | 0.4058 - 0.1479 |
| 5 & 6 | sSynRate & rSynRate | 0.3886 - 0.1731 |
| 7 & 8 | sRstRate & rRstRate | 0.2323 - 0.2072 |
| 9 & 10 | rttl & sttl | 0.1738 - 0.1075 |
| 11 & 12 | sAckRate & rAckRate | 0.1366 - 0.0690 |
| 13 & 14 | sMaxAckDelay & rMaxAckDelay | 0.1241 - 0.0831 |
| 15 & 16 | sPackets & rPackets | 0.0825 - 0.0587 |
| 17 | Protocol | 0.821 |
| 18 & 19 | sWinTCP & rWinTCP | 0.0718 - 0.0688 |
| 20 & 21 | rPayloadAvg & sPayloadAvg | 0.0716 - 0.0671 |
| 22 & 23 | rInterpacket & sInterpacket | 0.0701 - 0.0553 |

3) The sender address ('sAddress') and receiver address ('rAddress') columns were excluded from the experiment to prevent the attack detection task from becoming trivial due to the use of a fixed IP for the attacker.
4) The start time ('start' and 'startOffset') and end time ('end' and 'endOffset') columns were eliminated, as they solely entail temporal data and do not offer any valuable understanding of the cyberattacks.
5) The dataset was partitioned into three subsets: training, validation, and test sets, which comprised 50%, 20%, and 30% of the data, respectively.
6) We utilized the 'Min-Max' normalization method described in Eq. 1 to normalize the numerical features. Here, Min and Max represent the minimum and maximum values of a feature in the training set, respectively.

$$V_{Norm} = \frac{V - Min}{Max - Min} \quad (1)$$

### B. Feature Selection

During the feature selection phase, we utilize the Maximum Relevance, and Minimum Redundancy (MRMR) technique [46] to reduce the dimensionality of our data. MRMR assigns

Table X
HYPER-PARAMETERS SEARCH SPACE OF ML ALGORITHMS

| Method | Search space parameters |
|---|---|
| DT | Split criterion: [Gini's diversity index, Twoing rule, and Maximum deviance reduction] |
| RF | Number of learners: [10:500], Number of predictors to sample: [1:40] |
| ANN | Layers: [1, 2, 3] Layer size: [1 : 300] Activation: [ReLU, Tanh, None, and Sigmoid] |

a score to each feature based on its relevance and redundancy with other feature columns. To constrain the size of our feature set, we only select features with scores above 0.07. This threshold was determined by iteratively testing various thresholds and measuring the model's accuracy on the validation set. Note that many of our parameters, such as 'sBytesAvg' and 'rByteAvg', are dual counterparts; Therefore, selecting one requires selecting its corresponding dual parameter. Table IX presents the 23 features that have been selected for our experiment, along with their corresponding MRMR scores.

### C. Model Construction

In the third phase, we apply supervised ML models to perform the flow classification. During this phase, we use three off-the-shelf classification techniques, which have shown better performance in previous studies [47], [48]: Decision Tree (DT), Random Forest (RF), and Artificial Neural Networks (ANN). We also optimize the accuracy of these algorithms by giving effective values to their hyper-parameter settings and measuring their performance on the validation set. Table X summarizes the examined hyper-parameters for each approach.

### D. Model Validation

We chose not to balance the classes in the dataset that we generated, as we were unable to identify a realistic distribution of attacks in operating ICSs. Nevertheless, to handle the unbalanced data, we employed performance metrics that could effectively measure the model's precision on such data. Hence, in addition to accuracy, we employed metrics such as *accuracy*(Eq. 2) , *recall* (Eq. 3), *precision* (Eq. 4), and *F1-score* (Eq. 5), and confusion matrix to showcase how well the ML methods performed on unbalanced data.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$F1 - Score = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \tag{5}$$

Where $FP$=False Positive, $FN$=False Negative, $TP$=True Positive, and $TN$=True Negative.

Table XI
HYPER-PARAMETERS OF ML ALGORITHMS

| Model | Mode | Hyper Parameters |
|---|---|---|
| DT | Binary | Split criterion: Maximum deviance reduction, Maximum number of Splits: 314 |
| | Multi-class | Split criterion: Twoing rule, Maximum number of Splits: 1000 |
| RF | Binary | Number of learners: 10, Maximum number of splits 850, Number of predictors to sample: 17 |
| | Multi-class | Number of learners: 54, Maximum number of splits:1680, Number of predictors to sample: 8 |
| ANN | Binary | Fully connected layers: 1, Activation: Sigmoid, First layer size: 79 |
| | Multi-class | Fully connected layers:1, Activation: Sigmoid, First layer size 257 |

Table XII
TEST RESULTS OF THREE ML-METHODS FOR ATTACK DETECTION

| | | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|
| DT | Normal | 99.4% | 99.8 | 99.5 | **0.9965** |
| | Attack | | 97.9 | 99.1 | **0.9850** |
| RF | Normal | 99.5% | 99.9 | 99.6 | **0.9965** |
| | Attack | | 98.2 | 99.5 | **0.9885** |
| ANN | Normal | 99.5% | 99.5 | 99.8 | **0.9965** |
| | Attack | | 99.2 | 98.0 | **0.9860** |

## VI. RESULTS AND DISCUSSION

The results for ML-based intrusion detection using the optimal hyper-parameter setup (Table XI) are shown in Table XII. While all algorithms identify attack flows with greater than 99.4% accuracy, the RF technique outperforms the others with 99.5% accuracy and a higher F1-score. Despite the 98.2% precision of the RF algorithm in detecting attacks, even a low number of false alarms can impose high disruption costs in system operation.

In another experiment, we assessed the performance of ML methods for attack identification, and the results are presented in Table XIII. The RF method outperformed the others with an accuracy of 98.4%. However, the ML methods exhibited a decrease in accuracy due to their inability to recognize certain attack types. Although all three methods remained above 98.1% accuracy, there was a significant decline in F1-score, which is a more reliable assessment metric when dealing with imbalanced classes. Despite this, the ML methods effectively identified normal flows, as evidenced by their F1-score exceeding 0.99. The high F1-score for DDoS attack identification was expected since DDoS flows are easily detectable by monitoring massive packets and connections during the attack. Conversely, the F1-scores for IP-scan, Port-Scan, MitM, and Replay attack indicate that accurately classifying flows into the correct attack types is a non-trivial task, especially for IP-Scan with an F1-score of only 0.52.

Figure 6 depicts the RF classifier's confusion matrix to analyze misclassification samples. The majority of mistakes are caused by either: 1) attack records that were mistakenly classified as IP-Scan attacks, and 2) confusion between records of replay and MitM attacks. After conducting an in-depth

Table XIII
TEST RESULTS OF THREE ML-METHODS FOR ATTACK IDENTIFICATION

| | | Accuracy | Precision | Recall | F1-score |
|---|---|---|---|---|---|
| DT | Normal | 98.1% | 99.8 | 99.5 | **0.9965** |
| | DDoS | | 100 | 99.9 | **0.9995** |
| | IP-Scan | | 35.9 | 94.8 | **0.5208** |
| | MitM | | 90.9 | 88.2 | **0.8953** |
| | Port-Scan | | 94.9 | 93.2 | **0.9404** |
| | Replay | | 93.6 | 90.5 | **0.9240** |
| RF | Normal | 98.4% | 99.9 | 99.5 | **0.9970** |
| | DDoS | | 100 | 99.9 | **0.9995** |
| | IP-Scan | | 36.8 | 98.4 | **0.5357** |
| | MitM | | 91.1 | 90.4 | **0.9075** |
| | Port-Scan | | 94.9 | 93.3 | **0.9419** |
| | Replay | | 95.4 | 89.9 | **0.9257** |
| ANN | Normal | 98.2% | 99.5 | 99.8 | **0.9964** |
| | DDoS | | 99.9 | 99.7 | **0.9980** |
| | IP-Scan | | 36.5 | 65.6 | **0.4690** |
| | MitM | | 90.7 | 92.3 | **0.9149** |
| | Port-Scan | | 93.3 | 89.6 | **0.9141** |
| | Replay | | 98.9 | 88.6 | **0.9347** |

technical analysis of the Port-Scan, Replay, and MitM attacks, it was discovered that they employ the ARP poisoning technique as part of their process, which is remarkably similar to the IP-Scan process. This similarity in process results in the misclassification of some flows. Additionally, both the Replay and MitM attacks follow a same strategy for directing packets to the attacker node. Although the MitM attacker node alters routed packets, while the replay attacker node simply records packets temporarily, both attacks have the same routine for route redirection, causing similar impacts on network traffic.

Finally, the findings in this section show that ML methods can detect cyberattacks with high accuracy. However, determining attack types is not always straightforward. Individual flow analysis cannot differentiate between flows of identical processes used in different attack procedures. Although the impacts of various cyberattacks on network parameters are sometimes extremely similar, assessing the network status before and after individual records can provide a more accurate approximation of the type of prospective attack on the system. To tackle the mentioned problem, analyzing the predecessor and successor is a promising technique, which is the primary goal of sequence anomaly detection techniques.

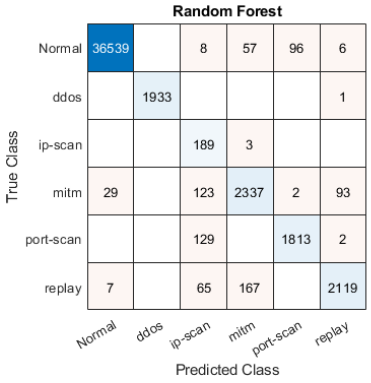


Figure 6. Confusion matrix of RF model for attack identification

## VII. THREATS TO VALIDITY

To enhance the validity of our research against potential challenges, we have adopted the well-established guidelines outlined in [49]. As part of this process, we have identified potential validity threats and implemented appropriate mitigation strategies to address them.

### A. Threats to construct validity

In order to simulate cyber threats, we had to select a limited number of attacks. We chose to focus on attacks commonly referenced in previous studies, such as those cited in [16], [26], [32]. These attacks are also recognized as common cyberattacks on ICSs by globally-accessible knowledge bases, such as the *'MITRE ATT&CK'*[18] and reports from the European Union Agency for Cybersecurity (ENISA) [36]. Moreover, while there is no standard approach or tool for implementing cyber attacks, we simulate attacks using common and open-source tools like Scapy and NMap. This decision was made to reduce the possibility of attack simulation errors, which could compromise the construct validity of our results.

Attack implementation in our analysis assumes that the attacker has already gained access to the control zone network. This type of access can be obtained using various methods, including physical access to network equipment, hijacking ICS wireless communications, or exploiting infected ICS components with malware, as described in [18]. For example, malicious software, such as trojans and viruses, can enable unauthorized operations on the ICS. In addition, attackers may establish backdoors or use remote access software to access the control zone network remotely.

### B. Threats to internal validity

To mitigate any potential bias towards fake results, we took measures to ensure the integrity of our findings. Expressly, we set aside 30% of the data as a test dataset and allocated 50% of the data for training the model and 20% for validation. This rigorous approach allowed us to avoid any potential bias introduced by hyper-parameter tuning, and the results presented in Section VI can be considered reliable.

Given the lack of a realistic distribution of carried attacks in operating ICSs, we deliberately decided not to balance the classes in our generated dataset between normal and under-attack samples. Instead, we utilized various performance metrics such as F1-score, precision, recall, and confusion matrix to assess the efficacy of our ML methods on unbalanced data. This approach allowed us to accurately evaluate the performance of our model without artificially manipulating the data distribution.

### C. Threats to internal validity

We have made significant contributions to the ICSSIM Framework [18] by enhancing it to create a network dataset that other researchers can utilize to develop IDSs. Our ICS-Flow dataset was created by identifying relevant network properties

[18]https://attack.mitre.org/techniques/ics/

that could be employed in other ICSs to detect intrusions. To facilitate the creation of this dataset, we developed an open-source ICSFlowGenerator tool that can calculate network flow features from raw network traffic data. These features can be extended or utilized in similar research to convert network traffic into a network flow dataset. Furthermore, the intrusion detection ML models we have described can be retrained with new environment network data for different ICSs. This opens up opportunities for other researchers to build upon our work and adapt it to their use cases.

## VIII. Conclusion

The ICS-Flow dataset is introduced in this paper as a benchmark for validating ML-based network intrusion detection techniques in ICSs. The dataset was created using ICSSIM Tools to set up a virtual ICS testbed for a sample bottle filling factory. To simulate realistic attacks on ICSs, we employed various common attack types, drawing from observations in 'ENISA' and 'MITRE ATT&CK'. During both normal and attack scenarios, we recorded the ICS's network packets and physical process state variables. To handle the computational complexity of analyzing individual network packets, we developed the ICSFlowGenerator tools as a free and open-source solution for processing captured raw network data into a network flow dataset. This tool can analyze network packets and produce network flows that include 50 different network features, such as flow features, general features, and TCP features. We also labeled the network flow records using various strategies to facilitate supervised learning studies and provide a foundation for testing unsupervised approaches. We have made the raw network traffic, the flow dataset, and log attack files publicly available for research in this field. Finally, we evaluated the dataset's applicability for intrusion detection validation using several supervised ML techniques, including ANN, DT, and RF.

Implementing IDS in industrial systems poses a number of challenges, with false alarms being the most significant obstacle. These false alarms can lead to costly interruptions in regular system operations. To mitigate this issue, a promising direction for future research is to explore the use of extended monitoring periods or sequence anomaly detection techniques, which have the potential to reduce the incidence of false alarms. Another challenge associated with IDS implementation is the unpredictability of attacker techniques. Attackers may use novel techniques that cause different effects on network packets, making it impossible to detect them using supervised ML. For this reason, investigating unsupervised binary and multiclass classification is another direction for future work.

Moreover, it is worth noting that network attacks not only impact network traffic but can also modify physical processes. As the ICS-Flow dataset includes both types of data, a potential direction for future research would be to integrate network monitoring with physical process monitoring to identify cyberattacks.

## Acknowledgment

This work was supported by InSecTT (www.insectt.eu), which received funding from the KDT Joint Undertaking (JU) under grant agreement No 876038. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey, Belgium, Germany, Czech Republic, Denmark, Norway.

The document reflects only the authors' views and the Commission is not responsible for any use that may be made of the information it contains.

## References

[1] K. Stouffer, J. Falco, K. Scarfone *et al.*, "Guide to industrial control systems (ics) security," *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

[2] M. Conti, D. Donadel, and F. Turrin, "A survey on industrial control system testbeds and datasets for security research," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2248–2294, 2021.

[3] J. Cheng, W. Chen, F. Tao, and C.-L. Lin, "Industrial iot in 5g environment towards smart manufacturing," *Journal of Industrial Information Integration*, vol. 10, pp. 10–19, 2018.

[4] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[5] T. Alladi, V. Chamola, and S. Zeadally, "Industrial control systems: Cyberattack trends and countermeasures," *Computer Communications*, vol. 155, pp. 1–8, 2020.

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

[7] A. Di Pinto, Y. Dragoni, and A. Carcano, "Triton: The first ics cyber attack on safety instrument systems," in *Proc. Black Hat USA*, vol. 2018, 2018, pp. 1–26.

[8] K. I. CERT, "Threat landscape for industrial automation systems. statistics for h1 2021," https://ics-cert.kaspersky.com/cards/, October 1, 2021, accessed:.

[9] W. Schwab and M. Poujol, "The state of industrial cybersecurity 2018," *Trend Study Kaspersky Reports*, vol. 33, 2018.

[10] B. Filkins, D. Wylie, and J. Dely, "Sans 2019 state of ot/ics cybersecurity survey," *SANS™ Institute*, 2019.

[11] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization." *ICISSp*, vol. 1, pp. 108–116, 2018.

[12] A. Ahmim, M. Derdour, and M. A. Ferrag, "An intrusion detection system based on combining probability predictions of a tree of classifiers," *International Journal of Communication Systems*, vol. 31, no. 9, p. e3547, 2018.

[13] M. A. Umer, K. N. Junejo, M. T. Jilani, and A. P. Mathur, "Machine learning for intrusion detection in industrial control systems: Applications, challenges, and recommendations," *International Journal of Critical Infrastructure Protection*, p. 100516, 2022.

[14] Y. Luo, Y. Xiao, L. Cheng, G. Peng, and D. Yao, "Deep learning-based anomaly detection in cyber-physical systems: Progress and opportunities," *ACM Computing Surveys (CSUR)*, vol. 54, no. 5, pp. 1–36, 2021.

[15] Y. Hamid, V. R. Balasaraswathi, L. Journaux, and M. Sugumaran, "Benchmark datasets for network intrusion detection: A review." *Int. J. Netw. Secur.*, vol. 20, no. 4, pp. 645–654, 2018.

[16] Á. L. P. Gómez, L. F. Maimó, A. H. Celdrán, F. J. G. Clemente, C. C. Sarmiento, C. J. D. C. Masa, and R. M. Nistal, "On the generation of anomaly detection datasets in industrial control systems," *IEEE Access*, vol. 7, pp. 177 460–177 473, 2019.

[17] J. L. Guerra, C. Catania, and E. Veas, "Datasets are not enough: challenges in labeling network traffic," *Computers & Security*, p. 102810, 2022.

[18] A. Dehlaghi-Ghadim, A. Balador, M. H. Moghadam, H. Hansson, and M. Conti, "Icssim—a framework for building industrial control systems security testbeds," *Computers in Industry*, vol. 148, p. 103906, 2023.

[19] M. Ghurab, G. Gaphari, F. Alshami, R. Alshamy, and S. Othman, "A detailed analysis of benchmark datasets for network intrusion detection system," *Asian Journal of Research in Computer Science*, vol. 7, no. 4, pp. 14–33, 2021.

[20] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, 2020.

[21] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, 2019.

[22] R. P. Lippmann, D. J. Fried, I. Graf, J. W. Haines, K. R. Kendall, D. McClung, D. Weber, S. E. Webster, D. Wyschogrod, R. K. Cunningham *et al.*, "Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation," in *Proceedings DARPA Information Survivability Conference and Exposition. DISCEX'00*, vol. 2. IEEE, 2000, pp. 12–26.

[23] D. D. Protić, "Review of kdd cup '99, nsl-kdd and kyoto 2006+ datasets," *Vojnotehnički glasnik/Military Technical Courier*, vol. 66, no. 3, pp. 580–596, 2018.

[24] M. V. Mahoney and P. K. Chan, "An analysis of the 1999 darpa/lincoln laboratory evaluation data for network anomaly detection," in *International Workshop on Recent Advances in Intrusion Detection*. Springer, 2003, pp. 220–237.

[25] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE symposium on computational intelligence for security and defense applications*. Ieee, 2009, pp. 1–6.

[26] T. Morris and W. Gao, "Industrial control system traffic data sets for intrusion detection research," in *Critical Infrastructure Protection VIII: 8th IFIP WG 11.10 International Conference, ICCIP 2014, Arlington, VA, USA, March 17-19, 2014, Revised Selected Papers 8*. Springer, 2014, pp. 65–78.

[27] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[28] A. P. Mathur and N. O. Tippenhauer, "Swat: A water treatment testbed for research and training on ics security," in *2016 international workshop on cyber-physical systems for smart water networks (CySWater)*. IEEE, 2016, pp. 31–36.

[29] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *International conference on critical information infrastructures security*. Springer, 2016, pp. 88–99.

[30] T. H. Morris, Z. Thornton, and I. Turnipseed, "Industrial control system simulation and data logging for intrusion detection system research," *7th annual southeastern cyber security summit*, pp. 3–4, 2015.

[31] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar, "Ton_iot telemetry dataset: A new generation dataset of iot and iiot for data-driven intrusion detection systems," *Ieee Access*, vol. 8, pp. 165 130–165 150, 2020.

[32] L. Faramondi, F. Flammini, S. Guarino, and R. Setola, "A hardware-in-the-loop water distribution testbed dataset for cyber-physical security testing," *IEEE Access*, vol. 9, pp. 122 385–122 396, 2021.

[33] D. Antonioli and N. O. Tippenhauer, "Minicps: A toolkit for security research on cps networks," in *Proceedings of the First ACM workshop on cyber-physical systems-security and/or privacy*, 2015, pp. 91–100.

[34] C. Parian, T. Guldimann, and S. Bhatia, "Fooling the master: Exploiting weaknesses in the modbus protocol," *Procedia Computer Science*, vol. 171, pp. 2453–2458, 2020.

[35] G. Thomas, "Introduction to the modbus protocol," *The Extension*, vol. 9, no. 4, pp. 1–4, 2008.

[36] E. E. U. A. F. Network and I. Security), "Good practices for security of internet of things in the context of smart manufacturing," 2018.

[37] W. Wang, F. Harrou, B. Bouyeddou, S.-M. Senouci, and Y. Sun, "Cyber-attacks detection in industrial systems using artificial intelligence-driven methods," *International Journal of Critical Infrastructure Protection*, vol. 38, p. 100542, 2022.

[38] E. N. Ylmaz, B. Ciylan, S. Gönen, E. Sindiren, and G. Karacayılmaz, "Cyber security in industrial control systems: Analysis of dos attacks against plcs and the insider effect," in *2018 6th international istanbul smart grids and cities congress and fair (icsg)*. IEEE, 2018, pp. 81–85.

[39] H. T. Reda, A. Anwar, and A. Mahmood, "Comprehensive survey and taxonomies of false data injection attacks in smart grids: Attack models, targets, and impacts," *Renewable and Sustainable Energy Reviews*, vol. 163, p. 112423, 2022.

[40] H. Lan, X. Zhu, J. Sun, and S. Li, "Traffic data classification to detect man-in-the-middle attacks in industrial control system," in *2019 6th International Conference on Dependable Systems and Their Applications (DSA)*. IEEE, 2020, pp. 430–434.

[41] B. Claise, "Specification of the ip flow information export (ipfix) protocol for the exchange of ip traffic flow information," Tech. Rep., 2008.

[42] G. Maciá-Fernández, J. Camacho, R. Magán-Carrión, P. García-Teodoro, and R. Therón, "Ugr '16: A new dataset for the evaluation of cyclostationarity-based network idss," *Computers & Security*, vol. 73, pp. 411–424, 2018.

[43] A. Lemay and J. M. Fernandez, "Providing {SCADA} network data sets for intrusion detection research," in *9th Workshop on Cyber Security Experimentation and Test (CSET 16)*, 2016.

[44] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[45] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne." *Journal of machine learning research*, vol. 9, no. 11, 2008.

[46] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[47] G. Singh and N. Khare, "A survey of intrusion detection from the perspective of intrusion datasets and machine learning techniques," *International Journal of Computers and Applications*, vol. 44, no. 7, pp. 659–669, 2022.

[48] S. A. Varghese, A. D. Ghadim, A. Balador, Z. Alimadadi, and P. Papadimitratos, "Digital twin-based intrusion detection for industrial control systems," in *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 2022, pp. 611–617.

[49] R. Feldt and A. Magazinius, "Validity threats in empirical software engineering research-an initial survey." in *Seke*, 2010, pp. 374–379.