X-IIoTID: A Connectivity-Agnostic and Device-Agnostic Intrusion Data Set for Industrial Internet of Things

Muna Al-Hawawreh, Elena Sitnikova, *Member, IEEE*, and Neda Aboutorab, *Senior Member, IEEE*

***Abstract*—Industrial Internet of Things (IIoT) is a high-value cyber target due to the nature of the devices and connectivity protocols they deploy. They are easy to compromise and, as they are connected on a large scale with high-value data content, the compromise of any single device can extend to the whole system and disrupt critical functions. There are various security solutions that detect and mitigate intrusions. However, as they lack the capability to deal with an IIoT's co-existing heterogeneity and interoperability, developing new universal security solutions to fit its requirements is critical. This is challenging due to the scarcity of accurate data about IIoT systems' activities, connectivities, and attack behaviors. In addition, owing to their multiplatform connectivity protocols and multivendor devices, collecting and creating such data are also challenging. To tackle these issues, we propose a holistic approach for generating an appropriate intrusion data set for an IIoT called X-IIoTID, a connectivity-agnostic and device-agnostic intrusion data set for fitting the heterogeneity and interoperability of IIoT systems. It includes the behaviors of new IIoT connectivity protocols, activities of recent devices, diverse attack types and scenarios, and various attack protocols. It defines an attack taxonomy and consists of multiview features, such as network traffic, host resources, logs and alerts. X-IIoTID is evaluated using popular machine and deep learning algorithms and compared with 18 intrusion data sets to verify its novelty.**

***Index Terms*—Cybersecurity, data set, Industrial Internet of Things (IIoT), intrusion detection.**

I. Introduction

THE PROLIFERATION of smart devices and digitization has extended to vital infrastructure and industrial sectors. Devices with wired and wireless connectivities enable healthcare, water, energy, and other parts of critical systems to operate more efficiently, productively, and reliably [1]. Such implementations are known as the Industrial Internet of Things (IIoT) whereby industrial assets are increasingly becoming capable of intelligently reacting to and changing their actions based on the information received through the large-scale cyber–physical control loop extending from the edge and cloud to the enterprise tiers [2]. However, this great convergence between information technology (IT) and operational technology (OT) systems, which is the key essence of IIoT technologies, has widened the threat landscape and increased the potential risks of cyberattacks being conducted against such critical systems [3]. A more significant concern relates to legacy OT systems (i.e., brownfield IIoTs), which are usually isolated but are becoming more connected with new IT technologies. Sophisticated attackers can easily gain access to an entire IIoT system and damage its functionality and production for a lengthy period [1], [3].

Recent threat reports provide details of several hacking campaigns that targeted critical industrial control systems (ICSs) and devices in an IIoT [4], [5]. For example, Ekans and LockerGoga ransomware attacks were specifically designed for compromising multiple Windows machines operating industrial applications [4]. These attacks encrypted files related to data historians and Web applications, leading to the shutting down of systems and interruptions to production. More recent attacks were Distributed Denial of Service (DDoS) over Constrained Application Protocol (CoAP) ones, and COVID-19 pandemic-based malware, which injected several smart healthcare systems to breach the intellectual property of vaccines [5]. Also, millions of IIoT/IoT devices were affected by two groups of TCP/IP stack's vulnerabilities called Ripple-20 and Urgent-11 [6] which attackers can exploit to control critical devices and affect their functionalities remotely. Therefore, due to the fact that, in such a system, any vulnerability can immediately become a threat to public safety, it is of particular importance that the security postures of such systems (both IT and OT) are modified and improved to fit IIoT ones.

A comprehensive and robust security posture can be achieved using multiple security countermeasures and tools that can be extended from reactive to proactive, preventative to remedial, as well as forensic and even intelligence techniques [2], [3]. This security approach is known as Defense in Depth (DiD) and includes a spread of several technological barriers, a multilevel or layered approach and implementation of mechanisms that reveal intrusions [7]. The best parts of this strategy are the observation, monitoring, and detection of attacks during their life cycles using intrusion detection systems (IDSs) [1], [8]. IDSs can handle highly skilled attackers and sophisticated attacks which are capable of overtaking a perimeter's security (e.g., firewall and encryption). They typically reveal zero-day attacks and contribute significantly to detecting nonessential traffic in an IIoT environment, thereby

Manuscript received May 26, 2021; accepted July 24, 2021. Date of publication August 3, 2021; date of current version February 21, 2022. *(Corresponding author: Muna Al-Hawawreh.)*

The authors are with the School of Engineering and Information Technology, University of New South Wales–Australian Defence Force Academy, Canberra, ACT 2612, Australia (e-mail: m.al-hawawreh@student.adfa.edu.au).

Digital Object Identifier 10.1109/JIOT.2021.3102056

improving the efficiency of critical industrial systems and their security management systems [9], [10]. In this regard, researchers have increased their interest in developing new IDS techniques for revealing attacks in a timely fashion and performing the appropriate remediation technique.

However, developing IDSs and security solutions tailored for IIoT systems and adopting them in a real-world environment has been hampered because these solutions require significant amounts of evaluation, testing, and tuning using labeled data sets that fit the IIoT systems' requirements [9]. Unfortunately, for evaluating the accuracy and efficiency of proposed IDSs and other security solutions, there is a lack of available labeled real-world data sets that fit the heterogeneity and interoperability nature of IIoT systems and have accurate data about their new connectivity protocols' patterns, system activities, and potential cyber attacks [8], [9]. This is mainly because of the difficulty of obtaining data from real-world projects for security and privacy reasons [11]. Therefore, this unavailability of an appropriate IIoT data set severely weakens the evaluations of proposed security solutions, as was pointed out in many studies [2], [9], [12]. Given that, the development of a realistic labeled IIoT intrusion data set is an important research topic that must be addressed to help researchers validate their security hypotheses, test proposed solutions, particularly those based on data mining and machine learning techniques, and increase the applicability of their results [9].

Although there are many intrusion data sets [12]–[22], most do not represent realistic IIoT systems' activities (e.g., machines and sensor-generated data, edge, mobile, and cloud traffic and activities), the behaviors of their new connectivity protocols and services (e.g., MQTT, CoAP, and WebSocket), the diverse communication patterns [i.e., machine-to-machine (M2M), machine-to-human (M2H) and human-to-machine (H2M)], high speed and large volume network traffic and systems' events, and specific IIoT attacks. Furthermore, they do not have multiview information about attacks from different sources (e.g., the network, host logs, resources and commercial IDS alerts) to provide a holistic solution. Also, they do not fit IIoT systems' requirements, such as interoperability, as most depend on features restricted to specific connectivity protocols' parameters and system activities (i.e., nonagnostic). As a consequence, they cannot be adequately used to design and develop anomaly-IDS (as IIoT bengin/normal behavior is different) or universal and plug-in security solutions that can protect any IIoT system [23]. IDSs and security solutions generated from such unrepresentative data sets are likely to prove ineffective in a real environment [9]. Therefore, a realistic intrusion data set tailored to IIoT systems is needed. This data set should be connectivity agnostic and device agnostic, which means it should be compatible with the IIoT system irrespective of the used connectivity protocols and their platforms, configurations, and the deployed hardware and software. Our research contributions are as follows.

1) We propose a new framework for creating an appropriate IIoT intrusion data set. Our proposed framework defines a standard means for establishing, developing, and evaluating this data set to satisfy the key requirements of IIoT systems.
2) We generate a first-of-its-kind IIoT intrusion data set that reflects the changes and heterogeneity of network traffic and systems' activities generated from various IIoT devices, connectivity protocols, and communication patterns.
3) We present various attack scenarios and new attacks related to IIoT connectivity protocols. Also, we provide an exact and clear attack taxonomy to classify attack types.
4) We propose new features obtained from different sources, including network traffic, logs, alerts, and system resources. These features are connectivity-agnostic and device-agnostic features for supporting an IIoT system's interoperability. As this data set has features extracted from any IIoT system regardless of the used devices vendors and protocol platforms, it is called X-IIoTID (X denotes any IIoT system).
5) We present a first-hand exploratory data analysis and evaluation of the performances of eight common machine and deep learning algorithms on the proposed data set as a baseline for further research.
6) We conduct a comprehensive analysis and comparison of existing data sets with X-IIoTID. The results prove the efficiency of our proposed framework for generating an IIoT intrusion data set and the superiority of the X-IIoTID data set compared with the existing ones.

The remainder of this article is organized as follows: related work is presented in Section II; a new framework for the generation of a real-life IIoT intrusion data set is proposed in Section III; a comprehensive evaluation and comparison of X-IIoTID and existing data sets is provided in Section IV; and, finally, Section V concludes this article and presents some future directions.

II. Related Work

Examples of common data sets used extensively by researchers to evaluate IDSs and other security solutions are the KDD CUP 99 [13], NSL-KDD [14], CAIDA [15], UNSW-NB15 [16], CICIDS [17], ISCX [18], NGIDS-DS [24], and TUIDS [25]. They provide the network traffic characteristics of attacks against traditional IT services, such as file transfer and Web access. However, they suffer from the fact that they do not sufficiently represent IIoT systems while most are obsolete and do not fit the new technologies. The massive changes occurring in traffic behaviors as well as new connectivity protocols and attacker tactics, techniques and procedures are not considered in these data sets. Another issue is their attack taxonomies which do not offer full support for understanding different types of attacks and their classes. For example, probing attacks in KDD CUP 99 [13], NSL-KDD [14], and TUIDS [25] can only be considered attacks if their numbers of iterations exceed a defined threshold [15]. In UNSW-NB15 [16] and NGIDS-DS [24], fuzzing, analysis, and reconnaissance are related because they are used to achieve the same objective but with different techniques and all can be categorized as reconnaissance. Also, exploit, shellcode, and backdoor all fit in the exploitation category.

Few researchers have developed intrusion data sets for ICS networks and their physical processes. Morris *et al.* [26] provided a data set for attacks, including reconnaissance, response injection, command injection, and DoS, that affect the Modbus protocol in gas pipeline systems. Similarly, Pan *et al.* [27] produced a data set for attacks on a Modbus power system network that covered multiple attacks, such as replay, masquerading and injection, and common faults and errors in industrial systems. Another available data set, Secure Water Treatment (SWaT) [28], was designed to represent hijacking attacks against the Modbus protocol. The main objective of these attacks is to change sensor measurements falsely. In more recent studies, Rodofile *et al.* [29] presented an intrusion data set for attacks generated against the DNP3 protocol in an ICS network, such as replay, DoS, injection and Man-in-the-Middle (MitM), while Myers *et al.* [30] generated one for injection and flood attacks against the S7comm protocol in an ICS testbed. These data sets depended highly on features related to sensor measurements, an actuator's status and specific parameters of industrial packets, which limited their use for diverse industrial systems. They were focused on only the malicious activities related to Stuxnet behaviors (an old malware). Unfortunately, such data sets do not encompass the new horizontal information and control flows in IIoT systems (i.e., edge to a physical asset, edge to cloud and enterprise to edge) or new attack patterns.

Many frameworks for generating intrusion data sets for IoT systems have been presented in the literature. Meidan *et al.* [19] relied on a video surveillance network to create a network intrusion data set known as N-BaIoT. They concentrated on simulating the behaviors of the Mirai and Bashlite botnets over traditional IT communication protocols. Similarly, Hyunjae *et al.* [20] introduced an IoT intrusion data set for a smart home network focused on the Mirai botnet's behavior. The normal traffic was collected by interacting with wireless smart camera devices. The attack behavior was also generated on a laptop and manipulated to appear as if it originated from IoT devices. In a related study, Bezerra *et al.* [22] presented a botnet attack on an IoT data set in which they reimplemented real botnet samples. In contrast to the previous data sets, Koroniotis *et al.* [21] relied on virtual machines, traditional IT protocols, and simulated MQTT traffic to generate botnet ones (Bot-IoT). Bot-IoT data set's normal traffic was synthetic as it was generated using the Ostinato traffic generating tool and MQTT scripts. The most recent IoT-DDoS data set generated by Al-Hadhrami and Hussian [12] was developed using the Cooja simulator to represent a multihop wireless sensor network. The stated goal of these data sets was to evaluate the effectiveness of IoT security solutions, such as IDSs, against DDoS attacks. While they were substantial, their lack of data related to new IIoT/IoT connectivity and industrial protocols and diverse attacks made them inadequate to understand new attack patterns and build new IIoT security solutions.

By reviewing the existing intrusion data sets presented in the literature, we can state that the element most critical for developing a successful new IIoT one is a holistic approach or plan. Unfortunately, most relevant studies in the literature lacked such an approach or used an incomplete one (i.e., only an attack profile). This limited provenance and unclear approach for creating data sets is pointed out in a recent survey article [9] and can also be noticed in the data sets mentioned above. Most of which lack a clear and exact attack taxonomy, new technologies' traffic patterns (e.g., cloud, mobile, and edge) and environmental structure and configuration. They did not investigate recent sophisticated/multistage attacks and attacks related to new IIoT connectivity protocols and services, making it difficult to build more effective and robust IDSs. Most did not represent IIoT systems and consider the critical nature of IIoT systems, which are heterogeneous, fragmented, and have special requirements such as interoperability, and their data characteristics. The existing data sets' normal data cannot represent the exact IIoT systems' normal behavior. Therefore, considering the limitations of existing intrusion data sets, a framework for producing and generating a new IIoT one is proposed. It defines a standard means of establishing, developing, and evaluating an intrusion data set that satisfies the key requirements of IIoT systems. Then, a new and first-of-its-kind IIoT intrusion data set (i.e., X-IIoTID) is created.

III. Proposed Framework for Generation of Real-Life IIoT Intrusion Data Set

The structure of our proposed framework shown in Fig. 1 consists of three tiers. The first (on the left-hand side) represents the key requirements for generating a data set tailored to IIoT systems and consists of an IIoT testbed, threat modeling, attack generation models, and the main characteristics and features of the IIoT data set. The middle tier contains the operations for constructing the data set, including collecting normal and attack data, extracting features, and preprocessing data (i.e., labeling, enrichment, and correlation). The third tier presents the final data set, analyzes it using an exploratory data analysis tool, and evaluates it using shallow and deep learning algorithms.

Overall, the ideal process for generating an accurate data set is to have an actual IIoT system operating in real time to characterize its behavior. Since the implementation of an IIoT data set is still in its early stages, with most existing implementations being special projects and unavailable publicly, a realistic security testbed can be used [31]. Such a security testbed structure should maintain the main characteristics of a real IIoT system with high fidelity. Utilizing a realistic IIoT security testbed will handle the shortcomings of existing data sets generated using simulators and traditional IT network structures. Therefore, we use our realistic IIoT security testbed called the Brown-IIoTbed [11], which considers the key requirements of IIoT systems and provides a generic platform for security testing. Details of this testbed and its functionalities are explained in Section III-A.

As the distributed and heterogeneous natures of IIoT systems widen their threat landscapes and surfaces, it is difficult to map them for an entire environment [2]. Nevertheless, as attackers may take advantage of a weakness that comes with adopting new technologies and devices, it is critical to address the security of such devices. Most recent threat-modeling approaches prioritize an edge layer, particularly an

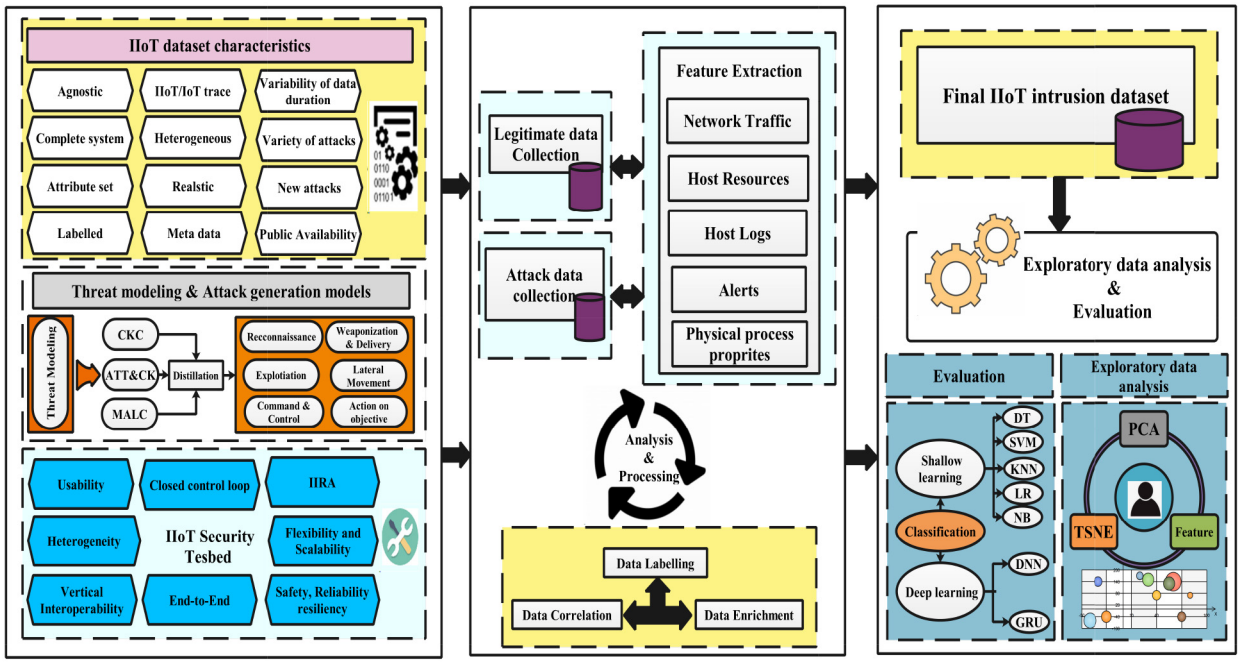

Fig. 1. Proposed data set generation framework utilizing three tiers.

edge gateway, which is the part of an IIoT system most targeted by advanced attackers [2], [8], [32]. This is because of their positions in IIoT systems as bridges between IT and OT ones, and they are critical parts of the closed-control loops and connect directly to data sources [11], [32]. On the other hand, they are the best positions to integrate security solutions to protect the legacy networks in brownfield IIoT systems, and other resource-constrained IIoT devices [1]. Therefore, we consider an edge gateway, the first and primary target for an attacker who can then target other system components, such as physical field devices, the cloud and local servers.

A predefined approach should be followed to model attacks and provide various scenarios. In our work, we intend to use the most common threat-driven frameworks, that is, the cyber kill chain (CKC) [33], mandiants attack life cycle (MALC) [34], and MITREs adversarial tactics, techniques, and common knowledge (ATT&CK) [35]. However, as these frameworks were released separately for IT and OT systems, and a specific one for IIoT systems has not yet been provided, we distil them and introduce a generic attack life-cycle framework to develop an appropriate IIoT intrusion data set with a variety of attacks. This handles the key challenges of the existing data sets by considering the new generation of multistage attacks and precisely describing and classifying them.

As IIoT systems connect a wide variety of legacy and new IIoT devices using various connectivity and networking protocols, configurations, data formats, and platforms, they require solutions that fit their interoperability requirements. Therefore, in this framework, we propose a connectivity-agnostic and device-agnostic intrusion data set to provide universal and plug-in security solutions to protect IIoT systems irrespective of their protocol platforms and device vendors and their configurations. Also, due to the complexity of IIoT systems and attacks that hinder single-view data capabilities, we propose multiview data that can adequately represent the information of all observations holistically. Moreover, after reviewing the main features of the existing data sets and those required for an IIoT intrusion one, we define the latter's unique features and properties (their full descriptions are provided in Section V). We consider the applicability of such features when generating our data set and then evaluating it compared to peers. The details of this proposed framework and the data set generation are explained in the following sections.

A. IIoT Testbed Architecture and Descriptions of Functions

To generate an IIoT data set, we use our testbed, Brown-IIoTbed [11], which was implemented in the IoT lab at the University of New South Wales (UNSW) in Canberra. Fig. 2 shows the Brown-IIoTbed architecture. It is a holistic and end-to-end IIoT security testbed developed based on an industrial Internet reference architecture (IIRA) model. It focuses on representing brownfield IIoT systems in which the legacy industrial devices [e.g., the programmable logic controller (PLC) and input/output (I/O)] are integrated with new IoT technologies since ensuring these systems' security is the most challenging aspect. The Brown-IIoTbed consists of a variety of M2M, M2H, and H2M connectivity protocols, sensors, actuators, various mobile and IT devices, access media, APIs, and states that are deployed on the three tiers of an IIoT system, including its edge tier (i.e., physical devices and edge computing), platform tier (i.e., cloud storage and analytics), and enterprise tier (i.e., service and application devices). It also contains an opensource host IDS in its edge gateway: opensource security (OSSEC) and a firewall/router device for securing its edge layer's devices and systems from the external world (i.e., Internet) and providing high-fidelity security testing. It includes attackers' machines (e.g., Kali Linux) both

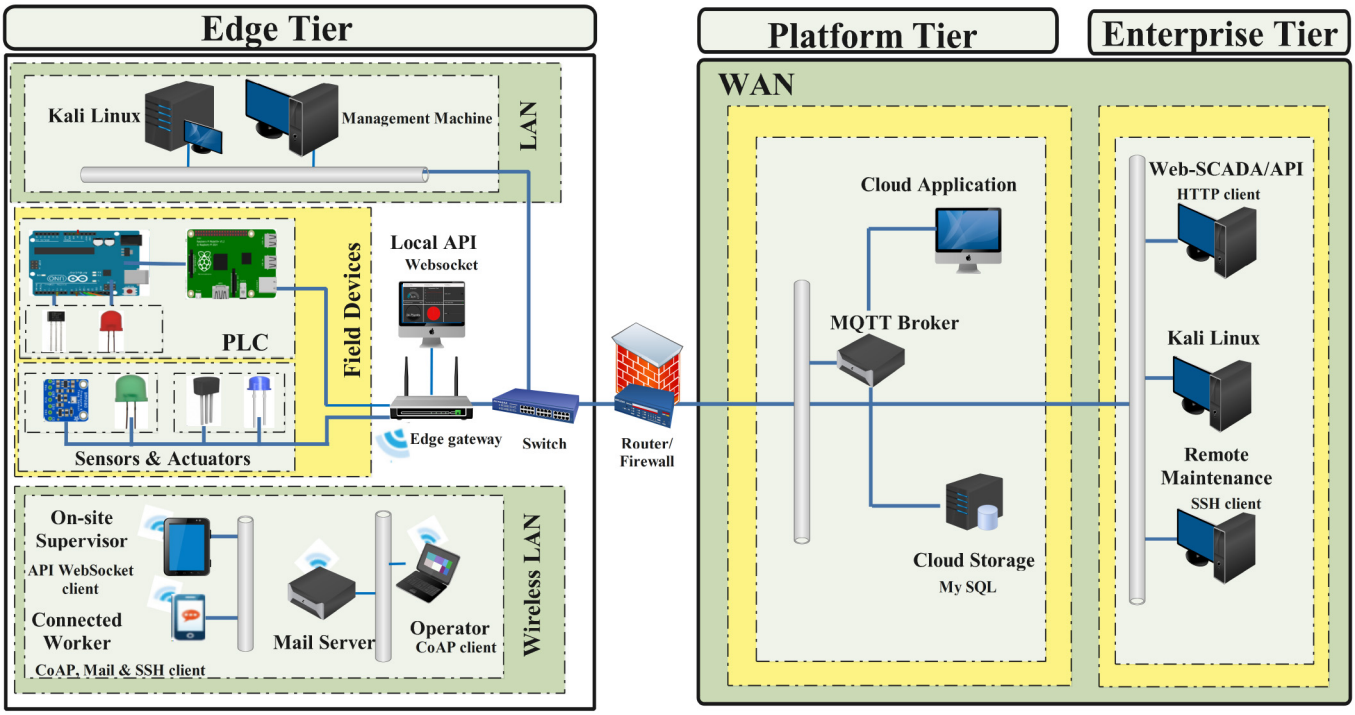

Fig. 2. Brown-IIoTbed architecture.

inside and outside networks to perform internal and external attacks.

As the IIoT implementation is designed for industrial automation, most of its traffic is conducted on M2M, H2M and M2H communications and continuously generated between physical field devices and other system components without human intervention. For example, the PLC has deterministic scan cycles for polling and sending the measurements of physical processes. Furthermore, an edge gateway sends an e-mail notification to a worker's mobile devices when a specified action occurs. In H2M communication, the worker connects with edge devices using Web-SCADA/API. Also, Python scripts act as users that interact with the data historian and perform remote access actions. Real human mobile users are included in this testbed to act as CoAP and mobile Web-SCADA clients.

To ensure the high fidelity and accurate data generated in the testbed, we validated Brown-IIoTbed using a conceptual model and statistical data validation methods [36]. This is because there is no publicly available IIoT system or reliable data to perform a comparison. Typically, the conceptual model is used to ensure that the model or testbed design (structure) has been done with sufficient accuracy by including the key aspects of the real-world system. In our testbed, we utilized the IIRA model, a standard international architectural model for IIoT, published by the industrial Internet consortium (IIC), to define the main requirements of real IIoT systems. IIRA emphasizes the key characteristics and requirements of any IIoT system, such as local and larger closed control loops, three-tiers (i.e., edge, platform, and enterprise), interoperability, and the appropriate information and control command flows. Our testbed adopted these requirements and characteristics as described in Fig. 2. Furthermore, the statistical data analysis was conducted to ensure that the testbed's data are as accurate as possible. The system and network performance factors were analyzed and compared with standard recommendations, and the results showed sufficient confidence on the testbed and its data. For example, CPU and Memory utilization work well(less than 50%), the testbed's network jitter does not exceed the tolerable value (30 ms based on Cisco), average Modbus TCP response time is around 10.49 ms which fits the Modbus requirement recommendation ($<$ 20 ms), and the same conclusion for MQTT, HTTP, WebSocket, DNS, CoAP, and SMTP. Other factors were also demonstrated, and complete details were provided in our previous work [11].

B. Attack Generation

A new-generation attack can be emanated from the most common threat-driven frameworks CKC, MALC, and ATT&CK. The CKC framework consists of seven stages: 1) reconnaissance; 2) weaponization; 3) delivery; 4) exploitation; 5) installation; 6) command and control (C&C); and 7) action on objectives [33]. Although it is the most common attack life-cycle framework and extensively used by organizations and researchers, it only describes its linear and one-off form stages. It does not include its internal reconnaissance and lateral movement. Inspired by the CKC, the cybersecurity company Mandiant released the MALC as its attack life-cycle framework. It has eight stages: 1) initial reconnaissance; 2) initial compromise; 3) establishment of a foothold; 4) escalation of privileges; 5) internal reconnaissance; 6) lateral movement; 7) persistence; and 8) completion of a mission. Some of these stages related to lateral movement are described in a cycle

as they can be performed multiple times on different targets. The MITRE Corporation developed ATT&CK, and it includes 12 tactics: 1) initial access; 2) execution; 3) persistence; 4) escalation of privilege; 5) evasion of defense; 6) access to credentials; 7) discovery; 8) lateral movement; 9) collection; 10) C&C; 11) exfiltration; and 12) impact [35]. It focuses on describing the objective of an attacker for performing any action (i.e., tactic), the action is followed to achieve this objective (i.e., technique) and the details of its implementation (i.e., procedure). It also provides post-intrusion or compromised information and attack intelligence based on real incidents.

The above frameworks are IT specific, although some (e.g., CKC and ATT&CK) have new versions with a few modifications to fit attacks on ICSs and their architectural designs. However, they cannot be applied directly to IIoT systems and networks because an IIoT combines both OT and new IT technologies, each of which has a different security perspective and a new architectural design, devices, and protocols. Furthermore, since an IIoT is a new technology that has not yet encountered several incidents, it cannot be considered to be based on the ATT&CK framework [2], [37]. Therefore, we distil from these frameworks a generic attack life-cycle one for achieving our objective of designing an IIoT intrusion data set. We focus on using this life cycle to present, describe, and classify various types and stages of cyberattacks rather than following it fully when an attack is performed. Details of generated attacks are explained in the following sections.

1) Reconnaissance: This stage involves an attacker searching the desired target and identifying and selecting an attack methodology [10]. Its main objective is to collect information about the target, which can be achieved using various techniques and procedures.

1) *Generic Scanning:* It refers to the process of defining a target machine's listening ports, its operating system (OS), and available services [25]. In this stage, we attempt to gain information about the target machine using the Nmap tool [38] which sends multiple TCP and UDP packets to a specific port and then analyzes the response and compares it with its database to return the extracted information.
2) *Scanning Vulnerabilities:* It refers to the process of determining known vulnerabilities and misconfigurations in a target machine [35]. In this scenario, we use Namp-vulners and Open-VAS [39] tools to scan the target machine and look up the common vulnerabilities and exposures (CVE) database to find appropriate vulnerabilities and misconfigurations.
3) *Fuzzing*: It aims to find system or software errors and exceptions by sending random or semivalid data into the system/software and then analyzing the output to discover faults. In our testbed, we perform fuzzing for the WebSocket API interface built based on a well-known Web programming framework, that is, "Node.js," using the Zap tool [40].
4) *Discovering Resources:* It refers to the process of dumping the list of available CoAP resources at the endpoint. These resources are related to physical processes, such as reading sensor values or controlling actuators, and are usually identified using a uniform resource identifier (URI) [5]. The resource list can be discovered by sending a "GET" request to a specific URI "./well-known/core." In our experiments, we use a Nmap-NSE script [38] to send a GET request to the CoAP server and dump the resource list.

2) Weaponization: This stage, also known as the initial compromise in MLAC and initial access in ATT&CK frameworks, includes the delivery stage in the CKC framework. It represents activities whereby an attacker gains a foothold in the environment and conveys malware to the victim machine. We focus on implementing the following most popular techniques and procedures for performing weaponization in IoT, and ICS systems [8], [17].

1) *Brute Force Attack:* It is one of the most popular attacks against network and Web applications and has recently been defined as one of the top initial-access vectors in an IBM threat intelligence report [41]. It tends to break into a user's account with weak credentials (i.e., user name and password) using a trial-and-error strategy [25]. It is designed to obtain an HTTP session against our Web server in the edge gateway. We use the Hydra tool [42] to execute it for 30 min and end up with a successful HTTP session.
2) *Dictionary Attack:* It is considered one of the most common attacks against remote access services and is used to explore a password from a dictionary, or word list [17]. Our experiments perform this attack against SSH using the Hydra tool and customized word list, which contains thousands of alphanumeric entries of different lengths. We execute it for more than an hour, during which superuser credentials are returned by the user, and a new session opened to deliver the malware payload.
3) *Malicious Insider:* It can be any trusted employee with malicious intentions who has legitimate access to a system. 80% of cyber incidents in ICSs stem from those [43] that create a cyber vulnerability or deploy a weapon. This attack is designed to deliver a malicious code to a target machine by creating an outbound connection, similar to legitimate ones, with a malicious website to download a malicious code.

3) Exploitation: This stage represents exploitation and installation in the CKC framework, establishing a foothold and escalating privilege in the MALC and executing, scripting, persistence, and escalating privileges in the ATT&CK framework. Since these stages occur immediately when an attacker exploits a vulnerability, all these phases can be combined in the exploitation [7]. In our experiments, we perform the following two scenarios.

1) *Reverse Shell:* An attacker can initiate a connection from the edge gateway back to its machine [7] and, in our implementation, exploit a malware payload delivered in previous stages to create a persistent reverse TCP shell/backdoor in the edge gateway.
2) *MitM Attack:* According to IBM X-Force's threat intelligence report [41], more than 35% of exploitation activities include MitM attacks whereby an attacker

exploits the communication between two endpoints and inserts itself between them. In our implementation, an attacker can exploit communications between the edge gateway and router and send packets with a false source address to fool the edge gateway into thinking that the attacker's machine is the router [11].

4) Lateral Movement: MALC and ATT&CK frameworks describe this stage as discovery, internal reconnaissance, and movement through a system and network. The key objective is to explore a victim's environment, to move ever deeper, and to compromise more systems and networks [44]. In an IIoT system, this lateral movement can be performed using M2M and M2H protocols and, in different directions, through the system's tiers [i.e., edge-to-cloud, edge-to-physical device and edge-to-IT local area network (LAN)]. We introduce the following three scenarios.

1) *MQTT Cloud Broker-Subscription:* An attacker can use its gained privileges to connect to the cloud broker to discover and obtain all the information collected from physical devices [11]. It can create an MQTT subscriber by sending a subscriber message for wildcard topics # to the cloud broker and receiving the most recent published messages containing information about the names of control systems and details of the collected measurements.
2) *Modbus-Register Reading:* As the Modbus/TCP protocol does not support authentication mechanisms, it is easy to perform lateral movement [43]. An attacker can connect with a PLC device using this protocol to read and discover connected register addresses and ranges of supported addresses of Modbus/TCP inputs and coils.
3) *TCP Relay Attack:* It is one of the most popular techniques for pivoting and moving around a network [45], particularly a network segment that does not directly connect with the attacker's local machine. We perform listener-to-listener relays to open the channel between an attacker's device and the mail server through the edge gateway. Therefore, the attacker can easily access the mail server set up in a different network.

5) Command and Control: Both the CKC and ATT&CK frameworks describe this as an essential stage for advanced persistent threat (APT) attackers as they require manual interactions with a compromised device rather than automatic activities. Therefore, a C&C channel can be established to connect their servers with compromised devices to perform multiple activities, such as sending commands and instructions to the malware and telling it the next step [46]. We perform DNS tunneling in a stealth mode (based on a common port and protocol) since it is one of the most common techniques and procedures used by APT.

6) Exfiltration: Refers to the process of leaking private and sensitive information related to IIoT devices, such as a PLC configuration, sensor data, credentials and devices' software versions. Advanced attackers usually use multiple techniques and procedures to achieve this objective, such as compression, obfuscation, and an alternative protocol for the C&C channel to avoid detection [2]. We follow the same technique by compressing files and infiltrating them via different protocols instead of using the C&C channel.

7) Tampering: It is defined as the deliberate destroying, manipulating or editing of data-in-transit or data-in-rest [47]. In our implementation, we introduce the poisoning of cloud data (i.e., false data injections) and fake notifications. In an IIoT, the sensors' data are collected and sent to the cloud for training data analytics models, which are then used to make predictions and decisions about the machine's maintenance. Attackers can fabricate and inject false data to affect the accuracy of a cloud data analysis. In another scenario, they can send fake e-mail notifications or alarms to connected operators [11].

8) Crypto-Ransomware: In this attack, an attacker injects malware to deny access to the data or system and force a victim to pay a fee in the form of cryptocurrency [7]. In our experiments, the main objective of the ransomware scenario is to encrypt the configuration and setting files of physical devices and use them to threaten to reboot these devices if the ransom is not paid within a specific time. Due to the unavailability of ransomware-executable files for the raspberry pi OS, we use our customized ransomware developed in our previous work [2] and a publicly available Python crypto-ransomware source code [48].

9) Ransom Denial of Service: This attack has recently increased as one of the new ways of extortion that threaten to knock a system's availability offline. An attacker sends a message threatening to launch massive DDoS traffic unless a ransom is paid by a specific deadline [49]. In our scenario, we focus on providing such volumetric attacks over IIoT application protocols (e.g., CoAP), affecting the physical process.

C. Data Collection

We collect data related to the end-to-end network traffic (i.e., from physical field devices to the edge gateway and from the edge gateway to the cloud and enterprise devices), host device logs and the host device's resources, physical properties, and alert logs. The main task for monitoring network traffic is to capture a lossless packet and timestamp it accurately. Therefore, software and hardware that can guarantee that all the traffic with full packet sizes is captured are required. This can be achieved by installing a dumpcap tool [50] in the edge gateway (raspberry pi B+ with a 64-GB memory card) and capturing the traffic periodically for a short time (a maximum of 2 h of continuous collecting). The dumpcap tool captures network traffic from a live network and stores it in a pcapng file format. We configure the tool to capture all the packet sizes from all the edge gateway interfaces. We also collect the data related to the edge gateway's resources, such as the CPU time, I/O activities, memory, context switch, and load average using the system activity reporter (SAR) tool [51]. The SAR collects system performance data during experiments and stores it in text files. Also, we gather logs of OSSEC (running in online mode) alerts and the edge gateway, such as the system, authorization, applications, and access logs, after

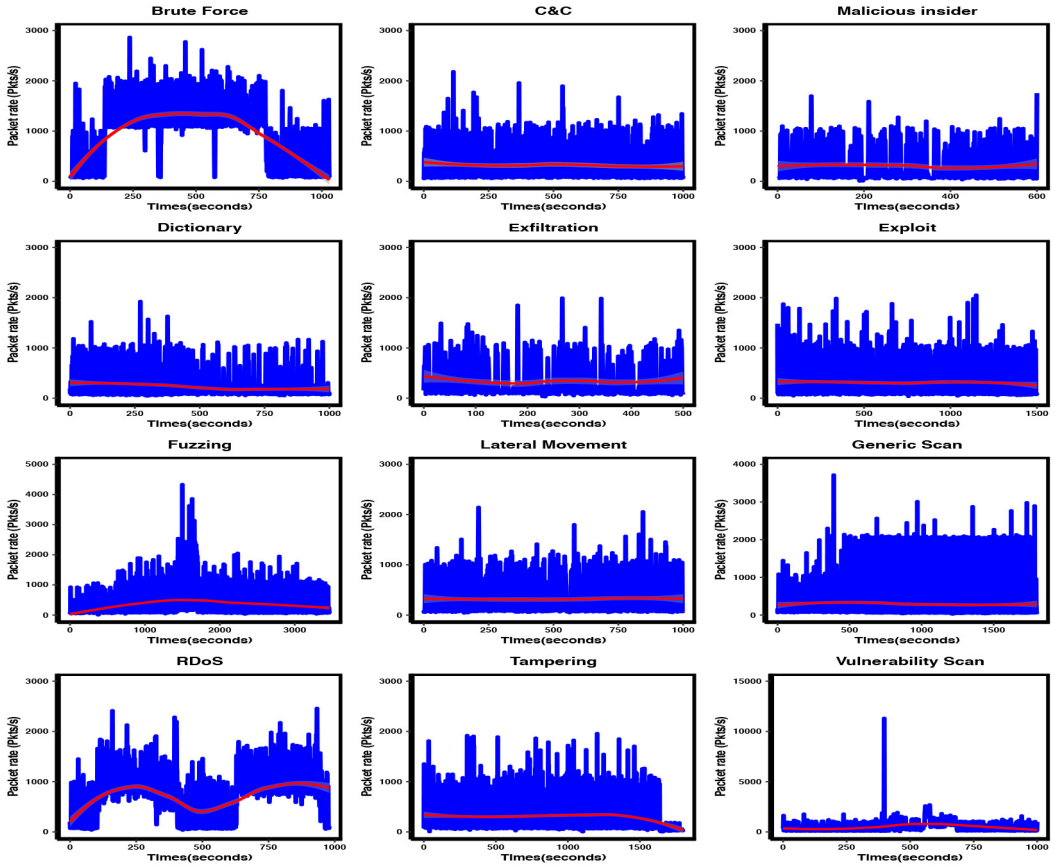

Fig. 3. Sample of quantitative pattern of network packets (mixed normal with attack) captured during attacks.

each experiment. All the desired data are collected over different lengthy periods to obtain the complete characteristics of the IIoT systems' activities and network traffic, with the whole process lasting for approximately four months.

The period for capturing normal data began on December 5, 2019, ran for many hours each day, and ended on March 23, 2020 (not continuous). The data included pure normal (without attack) and background normal traffic during the execution of an attack. Involving both types of traffic in the data set is an essential step in building security solutions that can function properly in real-time deployments. Furthermore, the experiments on the collected attack data took place over different times and days from January 7, 2020 to March 27, 2020, with each attack experiment repeated multiple times to collect more data. To observe the effects of real threats on the network's performance and understand the quantitative pattern of network packets during attacks, we capture network packets from the experiments. For the sake of brevity, Fig. 3 shows the packet rates over certain times for different attacks. We use the red curves to enable network traffic patterns to be clearly seen, which shows that the patterns of network packets vary. The most distinct ones are brute force and Ransom Denial-of-Service (RDoS) attacks because of the number of packets sent and caused by them.

D. Feature Extraction

For network traffic data, we focus on extracting connectivity-agnostic features to fit IIoT systems' interoperability requirements. This means that the extracted features should not be related to the content or parameters of application or connectivity protocols, which have specific-use implementations and configurations. In this regard, we use the Zeek network security monitor (formerly Bro) [52] to extract information about network connections by producing connection logs. Following the extraction of these connections, we develop Batch and Python scripts to parse the collected data and import it into Excel files for further processing. We then use the basic features to generate new ones (at packet and flow levels), with the final set of extracted network traffic features listed in Table I. Also, we extract features from the OSSEC and Zeek alert logs. As these signatures-IDSs do not have any signatures or rules for new attacks, particularly for new IIoT application protocols, they cannot be individually used to protect such systems [8], [53]. Given that, we combine the features extracted from them with other data to improve the capabilities of security solutions. We develop a Python script to convert and parse the JSON-OSSEC alert logs and a Zeek anomaly one (i.e., unexpected network activity) to simplify the feature extraction process (descriptions of the preprocessing and correlation of these features are provided in Section III-C).

Table II lists the features related to the edge gateway's resources, such as its CPU and memory loads, I/O activities, and the system's load, process, and context switch. After collecting these data, we develop a Python script to calculate the average and standard deviation for each parameter in a window

TABLE I
NETWORK TRAFFIC EXTRACTED FEATURES

Feature Name	Type	Description
Basic Network Traffic Features		
Ts	Discrete	Timestamp
Scr_IP	Discrete	Source endpoint's IP address
Des_IP	Discrete	Destination endpoints' IP address
Scr_Port	Discrete	Source endpoint's TCP/UDP port, or ICMP code
Des_Port	Discrete	Destination endpoint's TCP/UDP, or ICMP code
Protocol	Discrete	Protocols (e.g, TCP, UDP, ICMP, or other)
Service	Discrete	Application protocol running in Destination port
Duration	Continuous	The time difference between the last packet and the first packet seen
Scr_bytes	Continuous	Number of bytes from source to destination
Des_bytes	Continuous	Number of bytes from destination to source
Missed_byte	Continuous	Number of missing bytes in content gaps
Scr_pkts	Continuous	Number of sending packet
Des_pkts	Continuous	Number of received packet
Scr_IP_bytes	Continuous	Number of sending bytes in the IP header total length field
Des_IP_bytes	Continuous	Number of received bytes in the IP header total length field
Generated Network Traffic Features		
Conn_state	Discrete	Connection status (1 complete, 2 Rest, 3 partial)
Total_bytes	Continuous	Total number of bytes exchanged between source and destination
Byte_rate	Continuous	Total number of bytes per second
Total_Pkts	Continuous	Total number of packet exchanged between source and destination
Pkts_rate	Continuous	Total number of packet per second
orig_bytes_ratio	Continuous	Percentage of sending bytes to the total bytes
resp_bytes_ratio	Continuous	Percentage of receiving bytes to the total bytes
orig_packts_ratio	Continuous	Percentage of sending packets to the total packets
resp_pkts_ratio	Continuous	Percentage of receiving packets to source IP packets
SYN	Discrete	If connection has packet with SYN flag (1 or 0)
SYN-ACK	Discrete	If connection has packet with SYN-ACK flag (1 or 0)
Pure ACK	Discrete	If connection has packet with pure ACK flag (1 or 0)
Packet with payload	Discrete	If connection has packet with payload (1 or 0)
FIN or RST	Discrete	If connection has packet with FIN flag or RST (1 or 0)
Bad checksum	Discrete	If connection has packet with bad checksum (1 or 0)
SYN with RST	Discrete	If connection has packet with both SYN and RST flags (yes or No)

Con_state values present are 0 and 1

TABLE II
HOST EXTRACTED FEATURES

eature name	Type	Description
Generated host Resources Features		
Avg_user_time	Continuous	Average of user time (the time of process runs programs/codes) in the last 10 seconds
Std_user_time	Continuous	Standard deviation of user time in the last 10 seconds˜
Avg_nice_time	Continuous	Average of nice time (the time used for defining the priority of process) in the last 10 seconds
Std_nice_time	Continuous	Standard deviation of nice time in the last 10 seconds
Avg_system_time	Continuous	Average of system time (time the processor works at operating system functions) in the last 10 seconds
Std_system_time	Continuous	Standard deviation of system time in the last 10 seconds
Avg_IO_wait_time	Continuous	Average of I/O wait time (total time of the CPU is idle waiting for I/O operation) in the last 10 seconds
Std_IO_wait_time	Continuous	Standard deviation of I/O wait time in the last 10 seconds
Avg_idle_time	Continuous	Average of idle time (total time of the CPU is not busy and does ˜not have an outstanding disk I/O requests in the last 10 seconds
Std_idle_time	Continuous	Standard deviation of idle time in the last 10 seconds
Avg_tps	Continuous	Average of the number of transfer requests per second issued to ˜the device in last 10 seconds
Std_tps	Continuous	Standard deviation of the number of transfer transaction per second issued to the device in last 10 seconds
Avg_rtps	Continuous	Average of the number of read transaction per second issued to the device in last 10 seconds
Std_rtps	Continuous	Standard deviation of the number of read transaction per second issued to the˜device in last 10 seconds
Avg_wtps	Continuous	Average of the number of write transaction per second issued to the device in last 10 seconds
Std_wtps	Continuous	Standard deviation of the number of write transaction per second issued to the device in last 10 seconds
Avg_ldavg_1	Continuous	Average of average system load during the last minute in window time size 10 seconds
Std_ldavg_1	Continuous	Standard deviation of average system load during the last minute in window time size 10 seconds
Avg_Kbmemused	Continuous	Average of used memory in kilobytes in last 10 seconds
Std_Kbmemused	Continuous	Standard deviation of used memory in kilobytes in last 10 seconds
Avg_num_proc/s	Continuous	Average of number of tasks created per second in last 10 seconds
Std_num_proc/s	Continuous	Standard deviation of number of tasks created per second in last 10 seconds
Avg_num_swch/s	Continuous	Average of number of context switches per second in last 10 seconds
Std_num_swch/s	Continuous	Standard deviation of number of context switches per second in last 10 seconds
Generated Host Logs Features		
Anomaly_Alert	Discrete	1 if the connection has alert from Zeek; 0 otherwise
OSSEC_alert	Discrete	1 if the connection has alert from OSSEC; 0 otherwise
Alert_level	Discrete	OSSEC alert severity level
R_W_physical	Discrete	i if there is a read or write activity to the physical process; 0 otherwise
File_act	Discrete	1 if there is a file activities (read, write,delete, copy,create and download); 0 otherwise
Proc_act	Discrete	1 if there is a new process is executed or started; 0 otherwise
Is_privileged	Discrete	1 if the performed activity (login, process or file activity) is privileged ; 0 otherwise
Login_attmp	Discrete	1 if there is attempt to login; 0 otherwise
Succ_login	Discrete	1 if a successful login ; 0 otherwise

time of 10 s, which is chosen as it was reported in earlier studies [54] for detecting malware and attacks. For host-based logs (Table II), we develop a Python script to parse and analyze them to extract features related to an attacker's downloading tools, exert C&C channel, file modification, and others. The whole extracted host's features are device-agnostic features irrespective of the OS, programs, and hardware.

E. Data Labeling, Enrichment, and Correlation

As previously reported, each feature group is extracted separately. Therefore, it is essential to label, enrich and correlate all of them to create the final data set. In the first stage, the collected raw network traffic is converted into connections and then new features extracted (as described in Section III-B). Knowing the IPs of malicious attacks and their time intervals enable us to differentiate between normal and attacks behaviors (including their types) and label each network traffic connection. Each connection can be described as $A = (Uid, T, Scr\_IP, Scr\_port, Des\_IP, Des\_port, Serv, \{S\}, L)$, where ($Uid$) is its unique identity number, (T) is its timestamp, ($Scr\_IP, Scr\_port, Des\_IP,$ *and* Des_port) is its source and destination IP addresses and ports, respectively, ($Serv$) is its service, $\{S\}$ is its set of features, and (L) is its label. Similarly, the Zeek anomaly logs data are converted into records, each of which is described as $B = (Uid, Scr\_IP, Scr\_port, Des\_IP, Des\_port, Descrp)$, where ($Descrp$) describes the network's abnormal activity. All these extracted records are passed to the correlation process, the resulting procedure shown in Algorithm 1. They are used as a starting point for correlating with the Zeek anomaly records to avoid duplication and retain all the extracted features of the (A) records in the merging process. In more detail, both sets of records are matched based on the (Uid) field. If there is a match, "1" is assigned to this record; otherwise, "0," and this new extracted feature is merged with the other network traffic features in the set ($D1$).

After this stage, the algorithm matches the (A) set of records and the OSSEC alert ones to enrich them with more information. As a first step, each record generated from the OSSEC alert logs is defined as C= $(T, Scr\_IP, rule\_levl, rule\_comm, de\_name)$, where ($T$) is a timestamp, ($rule\_levl$) the severity level of the rule, ($rule\_comm$) a description of the bypassed rule and ($de\_ name$) the decoder name which acts similarly to the service (e.g., SSH and Web). We perform a deep analysis of the log files to reveal the appropriate attack information and construct the final alert records. To correlate an OSSEC alert with network traffic connections, we have to find each network connection corresponding to an OSSEC alert record. The matching process is conducted on multiple attributes, namely, the timestamp, Scr_IP, and decoder_name, because the timing information extracted from the OSSEC alert logs may not be precisely aligned with network connections as the log files generally do not provide millisecond precision. This may result in one alert corresponding to

Algorithm 1 Enrichment and Correlation Procedure 1

Input: Feature groups, F={A,B,C}
Output: Correlated and labeled Dataset, D

Initialize $D1$={}
Initialize $D2$={}
Initialize D={}
Call A={(*Uid, T, Scr_IP, Scr_port, Des_IP, Des_port, Serv, {S}, L*)}
Call B={(*Uid, Scr_IP, Scr_port, Des_IP, Des_port, Descrp*)}
Call C={(*T, Scr_IP, rule_lev, rule_comm, dec_name*)}
for $i = l$ to N_A **do**
  $D1.Add$ {A_i}
  for $j = l$ to N_B **do**
    if ($Uid_A == Uid_B$) **then**
      $D1.Add$ { *anomaly_alert* = "1" }
      Break
    end if
  end for
  $D1.Add$ { *anomaly_alert* = "0"}
end for
for $i = l$ to N_A **do**
  $D2.Add$ {A_i}
  for $j = l$ to N_C **do**
    if ($T_A \wedge Scr\_IP_A \wedge Serv == T_C \wedge Scr\_IP_C \wedge dec\_name$) **then**
      $D2.Add$[*OSSEC_alert* = "1", *alert_level* = *rule_levl* }
      Break
    end if
  end for
  $D2.Add$ { *OSSEC_alert* = "0", *alert_level* = "0" }
end for
D=$D1.merge$ ($D2, on = [A], how =$ "*left*")
return D

Algorithm 2 Enrichment and Correlation Procedure 2

Input: Previous dataset and logs, F={D,E,G}
Output: Final dataset, D_f

Initialize $D3$={}
Initialize $D4$={}
Initialize D_f={}
Call D={(*Uid, T, Scr_IP, Scr_port, Des_IP, Des_port, Serv, {S}, L*)}
Call E={(*T, Scr_IP, Serv,{event}*)}
Call G={(*T, {Resc_Fea}*)}
for $i = l$ to N_D **do**
  $D3.Add$ {D_i}
  for $m = l$ to N_E **do**
    if ($T_D == T_E$) **then**
      $D3.Add${$Event\_feat = \{T, Scr\_IP, Serv, \{event\}, Act =$ "*Proceed*"}
      Break
    end if
  end for
  $D4.Add${$Event\_feat = 0, Act =$ "*Analysis*" }
end for
for $i = l$ to N_D **do**
  $D4.Add$ {D_i}
  for $j = l$ to N_G **do**
    if ($T_A == T_G$) **then**
      $D4.Add${$Resc\_Feat = Resc\_Fea$ }
      Break
    end if
  end for
  $D4.Add$ {$Resc\_Feat = \{NAN\}$}
end for
D_f=$D3.merge$ ($D4, on = [D], how =$ "*left*")
return D_f

multiple different connections. Furthermore, we also need to match alerts with only network connections (based on source IP addresses). Most notably, OSSEC logs provide only the source IP addresses of incoming connections to the edge gateway. If there is a matching value of "1" assigned to this record, an alert level is added; otherwise, "0" is assigned to both parameters. To construct the final data set (D) as the outcome of this procedure, the extracted one ($D2$) is merged with ($D1$) based on the latter's parameters. The procedure for this matching is described in Algorithm 1.

We perform the same process for host-based logs and their extracted information by filtering and organizing this information and then converting it into event logs as $E = (T, Scr\_IP, Serv, \{event\})$. In this stage, as described in Algorithm 2, we rely on the timing information and our experience in analyzing and matching the records produced from the previous step and event logs because (Scr_IP) and ($Serv$) are not always available in logs data. Therefore, we add the parameter *(Act)* to ensure that input records are correlated with the log features to help us with the matching process. This parameter takes two values (*"proceed"*) or more (*"analysis"*). Also, we conduct a matching process between the final records produced and resource information extracted $G = (T, \{Resc\_Fea\})$ based on the timing information. It should be noted that, in this process, it is not necessary to have millisecond precision for the timing information as we extract resource-based features according to the size of the time window (i.e., 10 s). This forces the extracted records within this time window to have the same information regarding resource consumption. The data set of the extracted logs ($D3$) is merged with ($D4$) based on the former's parameters and then the final data set (D_f) is created. We also performed more proceeding for logs' data to ensure the correct correlation process. Later, we construct the final labeled data set and choose randomly 421 417 observations/instances for normal and 399 417 for attacks with 67 features (including three label levels, i.e., normal and attack, normal and subcategory attack, and normal and sub-sub-category attack). Table III shows the numbers of instances or observations for each type in the data set.

TABLE III
STATISTIC OF OBSERVATIONS INCLUDED IN THE X-IIOTID DATA SET

Attack class #1	Attack class #2	Number of instances	Total number of instances
Reconnaissance	Generic Scanning	50277	127590
	Scanning vulnerability	52852	
	Discovering resources	23148	
	Fuzzing	1313	
Weaponization	Brute-force	47241	67260
	Dictionary	2572	
	Insider malicious	17447	
Exploitation	Reverse shell	1016	1133
	MitM	117	
Lateral Movement	Modbus-register reading	5953	31596
	MQTT-cloud broker subscription	23524	
	TCP Relay	2119	
Command &Control	Command &Control	2863	2863
Exfiltration	Exfiltration	22134	22134
Tampering	False data injection	5094	5122
	Fake notification	28	
Crypto Ransomware	Crypto Ransomware	458	458
RDoS	RDoS	141261	141261
Normal	Normal	4211417	421417

F. Exploratory Data Analysis and Evaluation

Preparing the data set is an essential step for enhancing its data efficiency and making it ready for use by analysis tools, such as machine and deep learning algorithms. This process involves the followings [1], [8], [55]: first, cleaning the data set of unnecessary features, such as IP addresses and ports, and replacing missing values; second, encoding features by mapping nonnumeric ones to numeric values, for example, the service feature's categorical discrete values "WebSocket," "CoAP," and "MQTT" are converted to "1," "2," and "3," respectively; third, scaling the features by adjusting the values of the data set to within the range [0, 1] to prevent any feature

TABLE IV
PERFORMANCE RESULTS (%)

Algorithm	Binary-class(Normal and Attack)				Multi-class (9 Attack, 1 Normal)				Multi-class (18 Attack, 1 Normal)			
	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
DT	**99.54**	**99.54**	**99.54**	**99.54**	**99.49**	**97.33**	**97.20**	**97.27**	**99.45**	94.16	**93.54**	**93.80**
NB	84.84	84.46	84.80	84.82	35.52	55.87	75.20	47.96	47.08	61.65	86.86	60.50
KNN	98.31	98.32	98.30	98.31	98.20	95.70	91.25	93.24	98.21	94.68	86.84	89.89
SVM	97.89	97.99	97.84	97.88	98.12	97.33	94.06	95.61	98.14	**98.27**	86.49	90.72
LR	91.47	91.57	91.40	91.45	95.72	91.67	82.84	86.34	96.61	82.79	72.67	76.05
DNN	98.96	98.97	98.94	98.95	97.88	97.69	93.13	95.24	98.39	94.11	90.05	91.73
GRU	99.46	99.46	99.45	99.45	99.47	97.11	96.69	96.89	99.46	96.47	91.58	93.54

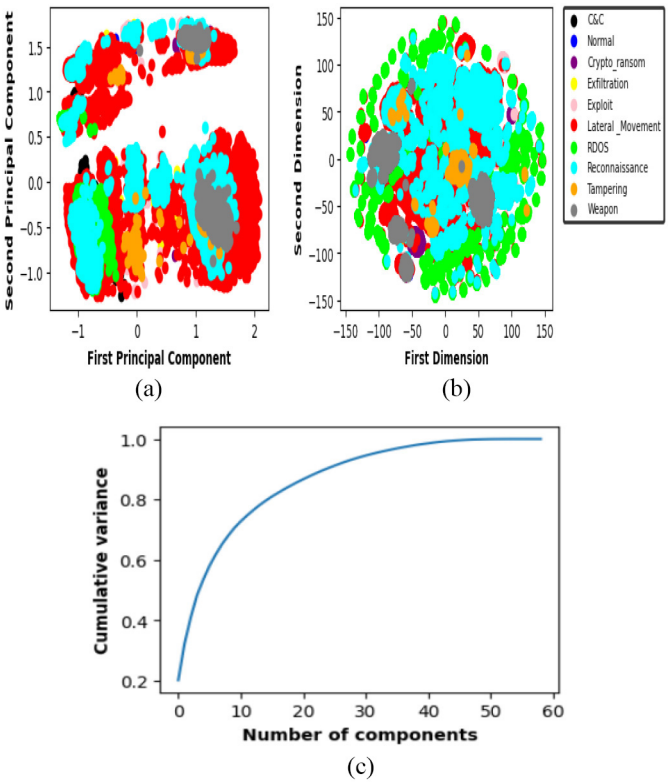

Fig. 4. Data set visualization. (a) 2D-PCA. (b) 2D-T-SNE. (c) PCA components number versus variance.

having a bias and ensure that they contribute proportionately; and, finally, splitting the data set for training and testing. In this last step, we use K-fold cross-validation whereby the data set is divided into k subsets. Each time one subset is used as a testing set, the $k-1$ subset is used as a training one, with the holdout process repeated K times. This, in return, reduces the bias in training and variance in the validation.

As the final version of the X-IIoTID data set is high-dimensional and consists of 820 834 instances and 59 features (without labels), it is challenging to understand its structure. However, by describing labels via a collection of features and visualizing them in two dimensions, such a data set is easy to understand. Therefore, we use two of the most unsupervised dimensionality reduction techniques [56], namely, principal component analysis (PCA) and T-distributed stochastic neighbor embedding (T-SNE). PCA attempts to find the direction of maximum variance by the eigenvectors of the covariance (i.e., principal components). T-SNE works in a probabilistic way to determine the optimal way to cluster or close low-dimensional observations similar to high-dimensional ones. We randomly choose 300 000 instances without labels from the data set to fit them to PCA and T-SNE. Fig. 4(a) and (b) shows samples of the lower dimensions in the data set of PCA and T-SNE (perplexity $=$ 300 and iterations $=$ 5000), respectively. As can be seen, the observations are not well clustered and overlap each other, indicating that two principal components/dimensions are not sufficient to describe the variance of the data. Fig. 4(c) shows that 93% of variations in the data can be explained if we consider 30 components in the PCA.

Developing IDSs using machine and deep learning algorithms is becoming prevalent due to their excellent capabilities for discovering hidden patterns and the main differences between normal and attack behaviors [54], [55]. Therefore, we implement the most commonly used machine learning algorithms, that is, the decision tree (DT), naive Bayes (NB), K-nearest neighbor (KNN), support vector machine (SVM), and logistic regression (LR) [1], [8], as well as the most popular deep learning ones, that is, the deep neural network (DNN) [parameters: three hidden layers (200 neurons), activation function (relu, sigmoid(binary), softmax (multiclass)), batch $=$ 250, epoch $=$ 10, optimizer $=$ RMSprop] and gated recurrent unit (GRU), [parameters: 3 GRU layers (200 unit) and activation function (tanh, sigmoid (binary and recurrent), and softmax (multiclass))] [53], [57], to identify an attack and classify its type. We perform fivefold cross-validation to evaluate these algorithms' performances based on Accuracy (Acc) which represents the fraction of predictions an algorithm correctly identifies, Precision (P) that describes the fraction of positive predictions that are actually correct, Recall (R) which represents the fraction of actual positives identified correctly, and the $F1$-score (F1) which is a weighted average of precision and recall. It should be noted that we use the default parameters of the algorithms and, in cases of multiple classes, Macro because it is reliable and less sensitive to unbalanced data [55]. Our experiments are conducted using the Python language (Scikit-learn and Keras libraries).

Table IV shows the experimental results for a binary class (normal and attack records) and two multiple classes, one with nine attack and normal records and the other with 18 attack and normal records). As can be seen, most machine and deep learning algorithms achieve good performances for all cases. However, the NB is the worst for both the multiple classes. For nine attacks, it achieves 35.52%, 55.87%, 75.20%, and 47.96% for accuracy, precision, recall, and the F1-score, respectively, and for 18, 47.08%, 61.65%, 86.86%, and 60.50% because it

TABLE V
DETECTION RATE (%) FOR NINE ATTACKS

Algorithm	C&C	Crypto_ransom	Exfiltration	Exploitation	Lateral _Movement	RDoS	Reconnaissance	Tampering	Weaponization
DT	89.66	99.86	89.76	**98.52**	99.48	**99.99**	**99.22**	**99.47**	**99.97**
NB	**100**	99.62	98.23	52.67	8.34	99.06	1.5	99.12	98.67
KNN	80.96	99.60	71.40	91.28	98.62	99.99	96.88	78.68	99.85
SVM	81.94	**99.92**	83.94	89.87	**99.83**	99.96	91.70	98.91	99.96
LR	58.01	99.86	46.34	78.42	98.09	99.83	86.66	74.07	99.07
DNN	77.16	81.88	99.91	81.29	97.69	99.96	95.85	98.15	99.94
GRU	88.26	96.94	**99.93**	86.23	97.89	99.98	98.86	99.18	99.97

TABLE VI
DETECTION RATE (%) FOR 18 ATTACKS

Algorithm	Bruteforce	C&C	Crypto_ransom	Dictionary	Exfiltration	Generic_Scan	Vul_Scan	RDOS	Data_inject	Discov_resources	Shell	MitM	TCP_Relay	Fake_not	Fuzz	Insider_malicious	Modbus_read	MQTT_Sub
DT	99.98	86.62	97.38	95.41	99.61	99.86	85.71	76.77	99.95	**99.84**	67.52	**99.83**	99.44	**99.89**	**99.99**	90.16	79.94	**99.80**
NB	99.43	**92.11**	**98.06**	94.76	99.73	99.56	96.43	77.68	99.95	93.12	**99.15**	99.65	7.01	96.42	99.06	74.02	92.02	32.20
KNN	**99.99**	80.86	87.09	96.29	81.78	99.58	92.86	48.44	99.81	95.23	51.28	83.87	98.60	99.76	**99.99**	72.05	64.98	99.69
SVM	**99.99**	81.87	59.40	94.76	98.96	**99.92**	92.86	34.73	**99.99**	95.40	58.12	79.59	**99.84**	99.65	99.95	88.29	61.16	99.73
LR	99.95	52.50	55.76	93.67	76.48	99.91	0.0	22.62	99.80	91.03	0.0	78.15	98.82	96.01	99.82	63.39	62.58	97.11
DNN	99.94	84.81	94.76	99.57	99.91	99.72	**99.82**	99.95	98.43	83.22	87.80	45.30	70.46	82.14	67.56	**99.96**	99.76	99.54
GRU	**99.99**	89.03	93.89	**99.65**	**99.92**	99.39	**99.82**	**99.96**	98.68	95.52	89.47	45.30	72.30	82.14	75.78	99.83	**99.80**	99.78

treats each feature independently and fails to identify the hidden patterns that can occur by combining sets of features. In contrast, the DT achieves the best performances of the algorithms for all cases. For binary classes, it obtains 99.54% for accuracy, recall, precision, and the $F1$-score, and 99.49%, 97.20%, and 97.27% for accuracy, recall, and the $F1$-score, respectively, for nine attacks. It also obtains 99.45%, 93.54%, and 93.80% for 18 attacks. It can identify the best sets of features and then use them at different stages of classification.

Tables V and VI present the detection rates for each attack. Table V shows the results for nine attacks and the different performances of the algorithms; for example, NB achieves the best detection rate for C&C but is very poor in detecting exploitation, lateral movement and reconnaissance. GRU performs best for detecting exfiltration and SVM for detecting crypto-ransomware and lateral movement. However, overall, the DT obtains the best detection rates for all attacks, as confirmed in Table V. Despite the diversity in the performance of each algorithm for detecting 18 attacks, the DT is still the best overall. This could be a good starting point for using the DT algorithm to explore the important features and select the most relevant ones for improving the performances of IDSs using fewer features. To give an idea about these features, Fig. 5 shows the best ten features based on DT.

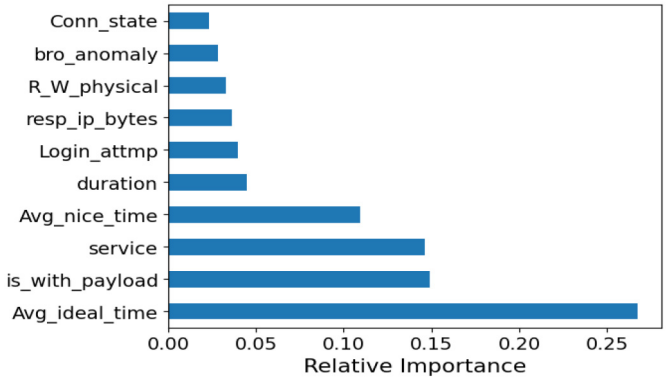

Fig. 5. Best ten features based on DT.

IV. COMPARISON AND DISCUSSION

To compare the X-IIoTID data set with existing ones, we use the following characteristics and traits defined in previous studies [10], [12], [17], [18] and new traits related to the IIoT system's requirements.

1) *Complete System and Network Configuration:* This is the foundation for generating a data set to represent a real-world system. For IT systems, it should include a network with all the required components, such as PCs, servers, and switches, combined with other physical industrial devices (i.e., OT) in an ICS. For IIoT/IoT systems, their configurations should represent holistic end-to-end ones (from their edges and cloud to their enterprise layers).
2) *Heterogeneous Data Sources:* To efficiently detect advanced attacks, it is necessary to collect information about these attacks from different sources, such as network traffic, devices logs, alerts, and system resources.
3) *Complete Capture:* To create efficient and robust security solutions, the data set should capture all the traffic in the network (with a payload) and not remove non-functional and odd-looking traffic which plays a critical role in evaluating the false alarms of proposed security solutions.
4) *Realistic Normal Network Traffic:* This should be generated from the real processes and communications among the system's components in the same environment in which an attack occurs. This trait takes a "NO" value if the normal traffic is generated using a tool or in a different environment from the attack.
5) *Divers Attack Types:* A generated data set should cover many attack types and stages as they are essential for performing complete evaluations of security solutions.

6) *Diverse Data Collection Duration:* A data set should cover a system's activities and traffic over a long time to capture periodical effects. As we consider more than one week as the basic duration, the value will be "YES" for a data set generated over more than one week.
7) *Feature Set:* Usability is a main requirement for any intrusion data set so that it can be used by security researchers to repeat and test proposed solutions which can be achieved by providing an attribute or feature set. Therefore, we assign a "YES" value to a data set with a predefined attribute or feature set and "NO" otherwise.
8) *Recent IIoT Application Protocols:* Many of these are used extensively in IIoT systems due to their good capabilities to provide low latency, efficient bandwidth usage, and high-quality services, such as MQTT, WebSocket (WS), and CoAP. An IIoT intrusion data set should contain these new protocols' traffic patterns.
9) *Recent Attacks:* The data set should contain modern-day Internet and IIoT traffic attacks, including those related to new IIoT application protocols.
10) *Agnostic Features:* As a key requirement for IIoT systems is interoperability, the attributes or features in the generated data set should be connectivity-agnostic and device-agnostic to help the creation of universal security solutions.
11) *IIoT Traces:* The data set should include real IIoT traffic traces from real connected sensors, actuators, PLCs, and industrial protocols.
12) *Labeled Data Set:* The data set can be unlabeled, partially labeled (i.e., normal and attack records), or fully labeled (i.e., normal and attack types). As, to increase its usability and federation, it should be fully labeled, a "YES" value is assigned if it is and "NO" otherwise.
13) *Meta Data:* The data set should have complete documentation and sufficient information about the environment and testbed's structure, software, hardware, and attack scenarios and protocols.
14) *Public Availability:* The data set should be publicly available to facilitate ongoing research by serving as a basis for evaluating and comparing different IDSs and security solutions. We assign "YES" if it is available online because there is no need to obtain approval to access it.

Table VII shows the comparative results obtained from an analysis of the weaknesses and strengths of 18 existing data sets and our proposed one (i.e., X-IIoTID). It can be seen that the existing ones encompass, at most, 13 traits of an appropriate intrusion data set. However, only the X-IIoTID exhibits all of the features (i.e., 20) highlighted in the literature and those related to IIoT systems, proving the effectiveness of our proposed framework for generating a data set and its implementation. X-IIoTID fulfils the requirements of a complete network and system configuration with an architectural trait. It is the only IIoT/IoT intrusion data set that obtains this trait because most IoT ones were generated using partial systems, such as edge ones consisting of simulated sensors and virtual machines. Therefore, they do not have all the information and control flows from the edge and cloud to the enterprise level and *vice versa*. In turn, this affects the quality of the normal traffic generated and system activities. Moreover, as obtaining the behaviors of recent traffic patterns and new connectivity protocols is essential for reflecting real-world systems, our data set is the only one that can achieve this. It provides legacy industrial and recent connectivity protocols' traffic patterns and system activities related to new devices and applications, such as the edge gateway, cloud broker, mobile operators, and Web-SCADA/API.

Unlike all current data sets, the X-IIoTID data set defines a clear and exact attack taxonomy, provides a new generation of attacks, and contains new attacks, such as WebSocket fuzzing, CoAP resource discovery, and MQTT malicious subscription, cloud data poisoning, fake notification, crypto-ransomware and RDoS. It also covers diverse attacks' protocols, such as Modbus, WebSocket, CoAP, MQTT, TCP, ARP, HTTP, SSH, DNS, ICMP, SMTP, and UDP. Its data are collected over a long time which helps the capture of periodical effects in the environment. Also, it provides holistic and multiview features, such as network traffic, host resources, system and physical processing logs, and alert-based features, to increase the efficiency of IDSs and other security solutions for addressing advanced threats. It is the only data set with these features that makes it more comprehensive and valuable than its peers.

Moreover, as it is connectivity agnostic and device agnostic, it supports the interoperability requirements of IIoT systems and assists the development of universal and plug-in security solutions. X-IIoTID is a labeled data set with three labeling levels: normal and attack, normal and subcategory attack (i.e., nine attack types) and normal and sub-sub-category attack (i.e., 18 attack types). It also has metadata and documentation that provides complete and sufficient information about a system's architecture, devices, OSs, installed programs, running protocols, attack types, tools used, and times of attacks. It is presented with complete features, clear definitions, and publicly available.[1]

Overall, X-IIoTID is the first-of-its-kind intrusion data set specifically for IIoT systems. It is a well-described and thoughtfully designed data set that reflects the changes and heterogeneity in the system's activities and composition of network traffic generated from multiple and diverse IIoT devices (e.g., physical control, edge, mobile, and cloud), connectivity protocols and communication patterns (i.e., M2M, M2H, and H2M). X-IIoTID offers unique insight into IIoT network's and system's threats and attacks, and it presents new and comprehensive features. However, although the current version of the X-IIoTID data set represents an important contribution to research on IIoT security, it has some limitations that must be recognized. For example, it does not cover cyber–physical attacks that directly affect the physical properties of PLCs (e.g., false command or data). As previously stated, we leave the exploration of such attacks to the future. However, to reduce this limitation, existing data sets, such as those of Morris *et al.* [26], and SWaT [28], can be used with ours to represent a complete IIoT system. Moreover, although the X-IIoTID data set is sufficiently large and its

[1] http://ieee-dataport.org/4532

TABLE VII
COMPARISON WITH EXISTING DATA SETS

Dataset	CNSC	HDS						RNT	DAT	DDD	FS	IIoT-CP			RA	AF	IIoT-T	LD	MD	PA
		NT	HR	L	PP	AL	CC					MQTT	CoAP	WS						
KDD CUP 99 [13]	**YES**	**YES**	NO	**YES**	NO	NO	**YES**	NO	**YES**	N/A	**YES**	NO	NO	NO	NO	**YES**	NO	**YES**	NO	**YES**
CAIDA [15]	**YES**	**YES**	NO	NO	NO	NO	NO	**YES**	NO	**YES**	NO	NO	NO	NO	NO	NO	NO	NO	NO	NO
NSL-KDD [14]	**YES**	**YES**	NO	**YES**	NO	NO	**YES**	NO	**YES**	N/A	**YES**	NO	NO	NO	NO	**YES**	NO	**YES**	NO	**YES**
ISCX [18]	**YES**	**YES**	NO	NO	NO	NO	**YES**	**YES**	NO	NO	NO	NO	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
Morris et al. [26]	**YES**	**YES**	NO	NO	**YES**	NO	**YES**	**YES**	**YES**	**YES**	**YES**	NO	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
UNSW-NB15 [16]	**YES**	**YES**	NO	NO	NO	NO	NO	NO	**YES**	NO	**YES**	NO	NO	NO	NO	NO	**NO**	**YES**	**YES**	**YES**
TUIDS [25]	**YES**	**YES**	NO	NO	NO	NO	**YES**	**YES**	NO	**YES**	**YES**	NO	NO	NO	NO	YES	NO	**YES**	**YES**	NO
Pan et al [27]	**YES**	**YES**	NO	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	NO	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
SWaT [28]	**YES**	**YES**	NO	**YES**	NO	NO	NO	**YES**	**YES**	NO	**YES**	NO	NO	NO	NO	NO	NO	**YES**	**YES**	NO
NGIDS-DS [24]	**YES**	**YES**	NO	**YES**	NO	NO	NO	NO	**YES**	NO	**YES**	NO	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
Rodofile et al [29]	**YES**	**YES**	NO	NO	NO	NO	**YES**	**YES**	**YES**	N/A	NO	NO	NO	NO	NO	NO	NO	NO	**YES**	NO
Myers et al [30]	**YES**	**YES**	NO	**YES**	NO	NO	**YES**	**YES**	N/A	NO	NO	NO	NO	NO	NO	NO	NO	NO	**YES**	NO
CICIDS [17]	**YES**	**YES**	**YES**	NO	NO	NO	**YES**	**YES**	**YES**	NO	**YES**	NO	NO	NO	NO	NO	NO	**YES**	**YES**	NO
N-BaIoT [19]	**NO**	**YES**	NO	NO	NO	NO	**YES**	**YES**	NO	**YES**	**YES**	NO	NO	NO	**YES**	**YES**	**NO**	**YES**	**YES**	**YES**
Bezerra et al [22]	NO	**YES**	**YES**	NO	NO	NO	NO	**YES**	NO	NO	**YES**	NO	NO	NO	**YES**	**YES**	**NO**	**YES**	**YES**	NO
BoT-IoT [21]	NO	**YES**	NO	NO	NO	NO	**YES**	NO	NO	N/A	**YES**	**YES**	NO	NO	**YES**	**YES**	**NO**	**YES**	**YES**	**YES**
Kang et al [20]	NO	**YES**	NO	NO	NO	NO	**YES**	NO	NO	N/A	NO	NO	NO	NO	**YES**	NO	NO	NO	NO	NO
Al-Hadhrami and Hussian [12]	NO	**YES**	NO	NO	NO	NO	**YES**	NO	NO	NO	**YES**	NO	NO	NO	NO	NO	NO	**YES**	**YES**	**YES**
X-IIoTID	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**	**YES**

CNSC: Complete Network and System Configuration **NT:**Network Traffic **L:**Logs **AL:** Alerts **RNT:** Realistic Network Traffic
DDD: Diverse Data Duration **IIoT-CP:** IIoT Connectivity Protocols **AF:** Agnostic-Features **LD:** Labeled Dataset **PA:** Public Availability
HDS: Heterogeneous Data Sources **HR:** Host Resources **PP:** Physical Process **CC:** Complete Capture **MD:** Metadata
DAT: Divers Attacks Scenario and Types **FS:** Feature Set **RA:** Recent Attacks **IIoT-T:** IIoT Traces

training and testing procedures are appropriate for it, it is not divided into training and testing data sets. A predefined data split is essential to increase federation capabilities and help researchers perform fair comparisons. Albeit this limitation is partly resolved by adopting K-fold cross-validation in the evaluation process as in Section III-F, an empirical study using different data-split techniques, such as CADEX, DUPLEX, and a heuristic approach [55], will be conducted in the future. The X-IIoTD data set has minority attack classes, such as fake notifications and MitM attacks. Although we confirm that they occur in real data, this obstacle can be exposed using various data preprocessing techniques such as data augmentation [57].

V. CONCLUSION AND FUTURE WORK

In this article, we proposed a new framework for generating a first-of-its-kind IIoT intrusion data set, i.e., X-IIoTID, that researchers can use to evaluate their proposed IDSs and security solutions. It handles the limitations of the existing data sets and is suitable for IIoT system's requirements. It consists of a labeled network and host data that collectively reflect the realistic and complete representation of an IIoT network and system activities in both normal and attack scenarios. The data are collected from network traffic, host resources, logs, and security mechanism alerts and cover the new IIoT connectivity protocols and recent attack tactics, techniques, and procedures. This data set is analyzed using an exploratory data analysis tool and evaluated by the most common machine and deep learning algorithms. Furthermore, a comprehensive analysis and comparison with existing intrusion data sets are performed. The overall results demonstrate that the X-IIoTID data set obtains the 20 traits required for an appropriate IIoT intrusion data set and is superior to 18 others.

For future work, we plan to explore the faults, cyber and physical attacks against physical field devices and extend the X-IIoTID data set by adding more attack scenarios related to the API, data historian, cloud, and mobile devices. Also, we plan to conduct an empirical study of the current version of this data set using different data-splitting techniques to choose the most appropriate one for splitting it into training and testing sets. Other interesting directions for future work are a comprehensive feature analysis, selection, and comparative analysis using online machine learning.

REFERENCES

[1] M. Al-Hawawreh, E. Sitnikova, and F. den Hartog, "An efficient intrusion detection model for edge system in brownfield Industrial Internet of Things," in *Proc. 3rd Int. Conf. Big Data Internet Things*, 2019, pp. 83–87.

[2] M. Al-Hawawreh, F. den Hartog, and E. Sitnikova, "Targeted ransomware: A new cyber threat to edge system of brownfield Industrial Internet of Things," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 7137–7151, Aug. 2019.

[3] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of Industrial Internet of Things security: Requirements and fog computing opportunities," *IEEE Commun. Surveys Tuts.*, vol. 22, no. 4, pp. 2489–2520, 4th Quart., 2020.

[4] A. Rege and R. Bleiman, "Ransomware attacks against critical infrastructure," in *Proc. 20th Eur. Conf. Cyber Warfare Security*, 2020, p. 324.

[5] A. T. Vasques and J. J. Gondim, "Amplified reflection DDoS attacks over IoT reflector running coap," in *Proc. 15th Iberian Conf. Inf. Syst. Technol. (CISTI)*, 2020, pp. 1–6.

[6] K. D. Rosso. *Ripple20: Critical Vulnerabilities Might Be Putting Your IoT/OT Devices at Risk*. Accessed: Jul. 30, 2020. [Online]. Available: https://cisco.com

[7] A. Alshamrani, S. Myneni, A. Chowdhary, and D. Huang, "A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1851–1877, 2nd Quart., 2019.

[8] M. Eskandari, Z. H. Janjua, M. Vecchio, and F. Antonelli, "Passban IDS: An intelligent anomaly based intrusion detection system for IoT edge devices," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6882–6897, Aug. 2020.

[9] A. Kenyon, L. Deka, and D. Elizondo, "Are public intrusion datasets fit for purpose characterising the state of the art in intrusion event datasets," *Comput. Security*, vol. 99, pp. 102–122, Dec. 2020.

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

[11] M. Al-Hawawreh and E. Sitnikova, "Developing a security testbed for Industrial Internet of Things," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5558–5573, Apr. 2021.

[12] Y. Al-Hadhrami and F. K. Hussain, "Real time dataset generation framework for intrusion detection systems in IoT," *Future Gener. Comput. Syst.*, vol. 108, pp. 414–423, Jul. 2020.

[13] (1999). *KDD Cup 1999 Data*. [Online]. Available: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[14] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD cup 99 data set," in *Proc. IEEE Symp. Comput. Intell. Security Defense Appl.*, 2009, pp. 1–6.

[15] P. Hick, E. Aben, K. Claffy, and J. Polterock. (2007). *The CAIDA 'DDoS Attack 2007' Dataset*. [Online]. Available: https://www.caida.org/catalog/datasets/ddos-20070804_dataset/

[16] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. Military Commun. Inf. Syst. Conf. (MilCIS)*, 2015, pp. 1–6.

[17] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a reliable intrusion detection benchmark dataset," *Softw. Netw.*, vol. 2017, no. 1, pp. 177–200, 2018.

[18] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Security*, vol. 31, no. 3, pp. 357–374, 2012.

[19] Y. Meidan *et al.*, "N-BaIoT—Network-based detection of IoT Botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul.–Aug. 2018.

[20] K. Hyunjae, H. A. Dong, M. L. Gyung, D. Y. Jeong, H. P. Kyung, and H. K. Kim, (2019). *IoT Network Intrusion Dataset*. doi: http://dx.doi.org/10.21227/q70p-q449.

[21] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull, "Towards the development of realistic BotNet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset," *Future Gener. Comput. Syst.*, vol. 100, pp. 779–796, Nov. 2019.

[22] V. H. Bezerra, V. G. T. da Costa, R. A. Martins, S. B. Junior, R. S. Miani, and B. B. Zarpelao, "Providing IoT host-based datasets for intrusion detection research," in *Proc. Anais Principais XVIII Simpósio Brasileiro Segurança Informação Sistemas Computacionais*, 2018, pp. 15–28.

[23] M. Noura, M. Atiquzzaman, and M. Gaedke, "Interoperability in Internet of Things: Taxonomies and open challenges," *Mobile Netw. Appl.*, vol. 24, no. 3, pp. 796–809, 2019.

[24] W. Haider, J. Hu, J. Slay, B. P. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017.

[25] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita, "Towards generating real-life datasets for network intrusion detection," *Int. J. Netw. Security,*, vol. 17, no. 6, pp. 683–701, 2015.

[26] T. H. Morris, Z. Thornton, and I. P. Turnipseed, "Industrial control system simulation and data logging for intrusion detection system research," in *Proc. 7th Annu. Southeastern Cyber Security Summit*, 2015, pp. 3–4,

[27] 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.

[28] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in *Proc. Int. Conf. Crit. Inf. Infrastruct. Security*, 2016, pp. 88–99.

[29] N. R. Rodofile, K. Radke, and E. Foo, "Framework for SCADA cyber-attack dataset creation," in *Proc. Aust. Comput. Sci. Week Multiconf.*, 2017, pp. 1–10.

[30] D. Myers, S. Suriadi, K. Radke, and E. Foo, "Anomaly detection for industrial control systems using process mining," *Comput. Security*, vol. 78, pp. 103–125, Sep. 2018.

[31] U. P. D. Ani, J. M. Watson, B. Green, B. Craggs, and J. R. C. Nurse, "Design considerations for building credible security testbeds: Perspectives from industrial control system use cases," *J. Cyber Security Technol.*, vol. 5, no. 2, pp. 71–119, 2020.

[32] S. Xu, Y. Qian, and R. Q. Hu, "Edge intelligence assisted gateway defense in cyber security," *IEEE Netw.*, vol. 34, no. 4, pp. 14–19, Jul./Aug. 2020.

[33] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues Inf. Warfare Security Res.*, vol. 1, no. 1, p. 80, 2011.

[34] M. I. Center, *APT1: Exposing one of China's Cyber Espionage Units*, Mandian, Alexandria, Virginia, 2013.

[35] B. E. Strom *et al.*, *Finding Cyber Threats with ATT&CK-Based Analytics*. MITRE Corp., Bedford, MA, USA, Rep. MTR170202, 2017.

[36] P. J. Durst, D. T. Anderson, and C. L. Bethel, "A historical review of the development of verification and validation theories for simulation models," *Int. J. Model. Simulat. Sci. Comput.*, vol. 8, no. 2, 2017, Art. no. 1730001.

[37] A. Hassanzadeh and R. Burkett, "SAMIIT: Spiral attack model in IIoT mapping security alerts to attack life cycle phases," in *Proc. 5th Int. Symp. ICS SCADA Cyber Security Res.*, 2018, pp. 11–20.

[38] G. Lyon. (2014). *NMAP Security Scanner*. [Online]. Available: http://nmap.org/

[39] S. Rahalkar, "Openvas," in *Quick Start Guide to Penetration Testing*. Berkeley, CA, USA: Springer, 2019, pp. 47–71.

[40] *ZAP*. Accessed: Apr. 30, 2020. [Online]. Available: https://www.zaproxy.org/docs/desktop/addons/fuzzer/

[41] IBM. *IBM X-Force's Threat Intelligence Index*. Accessed: May 1, 2020. [Online]. Available: https://www.ibm.com/security

[42] *Hydra Package Description*. Accessed: Jan. 7, 2020. [Online]. Available: https://tools.kali.org/password-attacks/hydra/

[43] M. Kalech, "Cyber-attack detection in SCADA systems using temporal pattern recognition techniques," *Comput. Security*, vol. 84, pp. 225–238, Jul. 2019.

[44] F. Wilkens, S. Haas, D. Kaaser, P. Kling, and M. Fischer, "Towards efficient reconstruction of attacker lateral movement," in *Proc. 14th Int. Conf. Avail. Rel. Security*, 2019, pp. 1–9.

[45] U. Ahmad, H. Song, A. Bilal, M. Alazab, and A. Jolfaei, "Securing smart vehicles from relay attacks using machine learning," *J. Supercomput.*, vol. 76, no. 2, pp. 2665–2682, 2020.

[46] S. Kim, G. Heo, E. Zio, J. Shin, and J.-G. Song, "Cyber attack taxonomy for digital environment in nuclear power plants," *Nucl. Eng. Technol.*, vol. 52, no. 5, pp. 995–1001, 2020.

[47] P. Zhao, H. Huang, X. Zhao, and D. Huang, "P^3: Privacy-preserving scheme against poisoning attacks in mobile-edge computing," *IEEE Trans. Comput. Social Syst.*, vol. 7, no. 3, pp. 818–826, Jun. 2020.

[48] Mark. *Markransom*. Accessed: Mar. 1, 2020. [Online]. Available: https://github.com/r3nt0n/markransom

[49] S. Newman, "Surviving ransom driven DDoS extortion campaigns," *Cyber Security Peer-Reviewed J.*, vol. 3, no. 1, pp. 37–43, 2019.

[50] *Wireshark Foundation, Wireshark: Go Dee*. Accessed: Jan. 12, 2020. [Online]. Available: http://www.wireshark.org/

[51] *System Activity Reporter (SAR)*. Accessed: Jan. 12, 2020. [Online]. Available: http://www.softpanorama.org/admin/monitoring/sar.shtml

[52] *Zeek:An Open Source Network Security Monitoring Tool*. Accessed: Apr. 15, 2020. [Online]. Available: https://zeek.org/

[53] M. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in Industrial Internet of Things based on deep learning models," *J. Inf. Security Appl.*, vol. 41, pp. 1–11, Aug. 2018.

[54] A. N. Jahromi, S. Hashemi, A. Dehghantanha, R. M. Parizi, and K.-K. R. Choo, "An enhanced stacked LSTM method with no random initialization for malware threat hunting in safety and time-critical systems," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 4, no. 5, pp. 630–640, Oct. 2020.

[55] I. Kotenko, I. Saenko, and A. Branitskiy, "Machine learning and big data processing for cybersecurity data," in *Data Science in Cybersecurity and Cyberthreat Intelligence*, vol. 177. Cham, Switzerland: Springer, 2020, p. 62.

[56] J. Pareek and J. Jacob, "Data compression and visualization using PCA and T-SNE," in *Advances in Information Communication Technology and Computing*. Bikaner, India: Springer, 2020, pp. 327–337.

[57] M. Al-Hawawreh and E. Sitnikova, "Industrial Internet of Things based ransomware detection using stacked variational neural network," in *Proc. 3rd Int. Conf. Big Data Internet Things*, 2019, pp. 126–130.

Muna Al-Hawawreh received the M.E. degree in computer science from Mutah University, Mu'tah, Jordan, in 2017. She is currently pursuing the Ph.D. degree from the University of New South Wales (UNSW), Canberra, ACT, Australia.

She also works as a Research Assistant with UNSW. During her Ph.D., she developed the world's first ransomware framework targeting IIoT edge gateway in the critical infrastructure. Her research interests include cloud computing, industrial control systems, Internet of Things, cybersecurity, and deep learning.

Ms. Al-Hawawreh was awarded the first prize for High Impact Publications in the School of Engineering and Information Technology (SEIT), UNSW Canberra, in 2019, and Dr. KW Wang Best Paper Award from 2018 to 2020. She is also a program committee member and reviewer for several cybersecurity conferences. She is a Reviewer of high-impact factor journals, such as the IEEE INTERNET OF THINGS JOURNAL, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.

Elena Sitnikova (Member, IEEE) received the Bachelor of Electrical Engineering degree (Hons.) in information control systems and the Ph.D. degree in computer technology (specialization in communication control systems) from Georgian Technical University (formerly, Polytechnic Institute), Tbilisi, Georgia, in 1983 and 1992, receptively, issued by USSR Academy of Science, Moscow, Russia. Thesis Title: Mathematical Modelling of Space Communication Systems.

She is an Award-Winning Academic and Researcher with the University of New South Wales (UNSW), Australian Defence Force Academy, Canberra, ACT, Australia. Her current research interests are in the critical infrastructure protection area, carrying out research projects in intrusion detection for control systems cybersecurity, cyber–physical systems, and Industrial IoT.

Dr. Elena is holding a Senior Fellowship of the Higher Education Academy and the Australian Office for Learning and Teaching Team Citation Award for Outstanding Contributions to Student Learning. She is one of the first Australians to be certified in Certified Secure Software Lifecycle Professional.

Neda Aboutorab (Senior Member, IEEE) received the Ph.D. degree in electrical engineering from the University of Sydney, Sydney, NSW, Australia, in 2012.

She is currently a Senior Lecturer with the School of Engineering and Information Technology, University of New South Wales, Canberra, ACT, Australia. From 2012 to 2015, she was a Postdoctoral Research Fellow with the Research School of Engineering, Australian National University, Canberra. Her research interests include network coding, wireless communications, data caching and storage systems, Internet of Things, and signal processing.