IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 17 November 2022, accepted 14 January 2023, date of publication 23 January 2023, date of current version 26 January 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3238874

RESEARCH ARTICLE

A Network Intrusion Detection System for Building Automation and Control Systems

VITOR GRAVETO, TIAGO CRUZ, (Senior Member, IEEE), AND PAULO SIMÕES, (Senior Member, IEEE)

University of Coimbra, Centre for Informatics and Systems of the University of Coimbra, Department of Informatics Engineering, 3030-290 Coimbra, Portugal

Corresponding author: Tiago Cruz (tjcruz@dei.uc.pt)

This work was supported in part by the Fundo Europeu de Desenvolvimento Regional (FEDER)-Competitiveness and Internationalization Operational Program (COMPETE 2020), Portugal 2020 Framework, in the scope of the Smart5Grid Project, under Grant POCI-01-0247-FEDER-047226; and in part by the FCT–Foundation for Science and Technology, Instituto Público (I.P.)/MCTES through National Funds [Programa de Investimentos e Despesas de Desenvolvimento da Administração Central (PIDDAC)], within the scope of Centre for Informatics and Systems of the University of Coimbra (CISUC) Research and Development Unit, under Grant UIDB/00326/2020 and Project UIDP/00326/2020.

ABSTRACT Building Automation and Control Systems (BACS) are traditionally based on specialized communications protocols, such as KNX or BACnet, and dedicated sensing and actuating devices. Despite the increased awareness about the security risks associated with BACS, there is a lack of security tools for protecting this special breed of cyber-physical systems. This is further aggravated by the fact that general-purpose security tools are typically not able to cope with the specific requirements and technologies associated with BACS, making it necessary to devise domain-specific approaches – as shown, for instance, by the KNX Secure initiative led by the KNX Association. Nevertheless, despite the advances brought by KNX Secure and similar initiatives, there is still a considerable gap between the security needs of BACS and the solutions available. In this paper, we address this gap by proposing a Network Intrusion Detection System (NIDS) specifically designed for BACS. This NIDS is protocol-agnostic and can potentially support different BACS protocols and technologies, such as KNX, BACnet, Modbus or mixed ecosystems, without loss of generality. We also present a specific proof-of-concept implementation of this NIDS concept for KNX – one of the more widespread BACS protocols. To this purpose, a real-world KNX deployment was used to showcase and evaluate the proposed approach.

INDEX TERMS Home automation, building automation and control systems, BACS, NIDS, smart buildings, security, safety, KNX.

I. INTRODUCTION

Over the past few years, there has been a progressive mindset shift in the automation domain towards considering security as much of a critical requirement as reliability or safety. From this perspective, Building Automation and Control Systems (BACS) constitute no exception, as both the need for monitoring the proper operation of physical devices and the security of the whole building operation should be considered as key requirements. This is especially important if one considers the increasing permeability between building automation and traditional IT systems, which increases the security challenges faced by BACS, since most of the current BACS implementations were originally designed with isolation as an acquired safety guarantee.

The growing awareness of security problems led to various improvements to the standards used in building automation (e.g., KNX [1], BacNet [2] and ZigBee [3]), for instance incorporating authentication and encryption mechanisms. However, in most deployments it will not be easy or even possible to retrofit these improvements, since existing devices lack memory and/or computational power to implement those novel security features. Moreover, even buildings where these improvements are retrofitted are still vulnerable to a wide range of attacks.

In this paper we explore the concept of a domain-specific Network Intrusion Detection System (NIDS) for BACS as a way of mitigating this situation. The proposed NIDS is a

The associate editor coordinating the review of this manuscript and approving it for publication was Diana Gratiela Berbecaru.

monitoring probe that can intercept all the fieldbus traffic of the building automation network (control network) and also observe the local area network (LAN). In this way, all the messages can be observed (e.g. firmware updates, command messages, sensors' outputs, actuators' inputs and status).

The feasibility of this BACS-specific NIDS concept depends on two key capabilities. First, it must incorporate anomaly detection mechanisms able to ingest the data obtained from both the legacy BACS control network and the general-purpose building LAN, to detect system anomalies and potential cyberattacks. Second, this domain-specific NIDS must target a suitable cost-efficiency balance, ideally within the magnitude of a single device retail price.

This proposition provides a viable (albeit not perfect) alternative to implement a security layer for existing deployments by adding a new device, instead of undergoing mass replacement of existing equipment. Moreover, this approach is complementary even for modern deployments, that already incorporate security-oriented features. In this paper both challenges are addressed.

The rest of this paper is organized as follows. First, we provide an overview of KNX-based BACS (Section II), process safety and BACS security mechanisms (Section III). Next, we detail the proposed concept and the associated requirements in the scope of BACS (Section IV). Section V details the proposed architecture, and Section VI presents the developed Proof-of-Concept (PoC). Validation is addressed in Section VII and, finally, Section VIII concludes the paper.

II. KNX-BASED BACS

This section provides a basic description of the KNX protocol and related technology ecosystem. More detailed information can be found on the KNX Association repositories [4].

The KNX standard appeared in the early 1990s, driven by the European Installation Bus Association (EIBA [1]) as a way to enable the connection, configuration and communication between multiple building automation devices (e.g. sensors, actuators, buttons and other user interfaces), using a common language and a standard communications protocol. It is widely used for home and building automation, for instance to control lighting, shutters, security systems, energy management, heating, ventilation, air-conditioning systems, signalling and monitoring systems, remote control, and audio/video control. All these functions are managed via the KNX protocol set.

Opposite to traditional electric installations, KNX installations have no dedicated hard-wired connections between control devices and actuating devices. For example, a light switch is not directly connected with the controlled lights. Instead, all devices are connected via a shared bus that runs on 29 Volt. All bus devices can be programmed with a common tool, enabling easy and flexible deployment. Moreover, subsequent changes require no rewiring.

A KNX system requires the following components (cf. Figure 1):

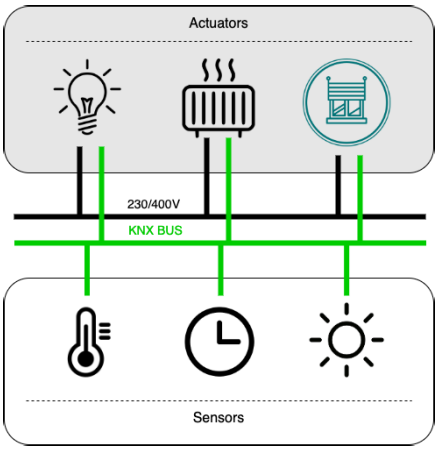

FIGURE 1. Basic KNX elements.

- Power supplies that feed the bus and KNX devices.
- Sensors (push buttons, thermostats, air velocity meters, etc.) that generate commands as *telegrams*.
- Actuators (switch relays for lights, blinds, etc.) that receive the *telegrams* and perform predefined actions.
- The bus that connects all sensors and actuators.

KNX is designed to be independent of any particular hardware platform – simple control functionalities are often implemented using basic 8-bit micro controllers, while more complex functionalities may require more powerful hardware platforms. The most common transmission medium in KNX is twisted pair (TP), but KNX also supports other medium, such as powerline networking (PL), radio frequency (RF), infrared, and Ethernet/IP – even though some of these medium are rarely used in production scenarios.

The smallest entity within the KNX network topology is a line. A line contains a maximum of 64 devices, which is enough for most small scale projects. Larger projects may use up to 15 lines, combined within one area, all connected via a main line. Different lines may be connected to the main line with line couplers. Furthermore, it is also possible to connect up to 15 areas to a backbone. Single areas are connected to the backbone line via backbone couplers (cf. Figure 2).

KNX *end devices* may be connected anywhere in this topology. Up to 255 end devices may be addressed in any sub-network. Those devices may be numbered from 1 to 255. Each device (backbone coupler, line coupler, end device) must have an Individual Address (IA), which is unique throughout the complete topology.

To standardize the configuration and commissioning, the Engineering Tool Software (ETS) [5], created by the KNX Association, should be used to define, program, configure and commission the entire KNX network. The most recent versions of ETS support the use of device catalogues, available online, to streamline the solution design. The catalogues are supported by the KNX Association with contributions from their manufacturer members (around 500 at the present date, with around 8,000 products).

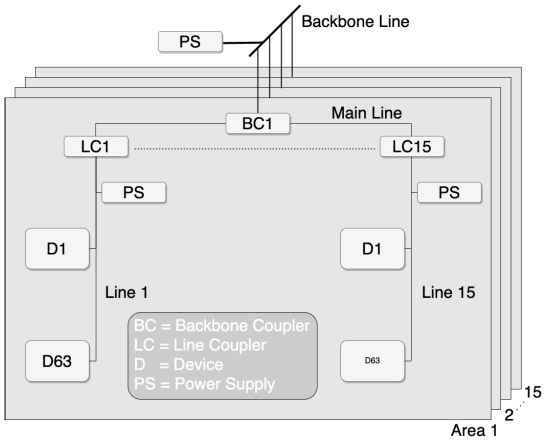

FIGURE 2. KNX logical topology (adapted from [4]).

In ETS terminology each installation is known as a project. These projects incorporate a local copy of all the installation details – like devices, topology and commissioning state. The application allows exporting the project or some of its parts in a well-documented XML format [6].

The development of KNX-certified devices can follow three different models: partial, OEM (Original Equipment Manufacturer) or full development. *Partial* development devices are based on available and already certified system components, communication stacks and modules, including the (KNX-certified) Application Program. *OEM devices* are straightforward relabels of already certified KNX devices (typically developed by other KNX members), with this option the development effort is reduced to nearly zero, and only the Application Programs need to be registered in the name of the reselling manufacturer. Lastly, *fully developed* devices require several steps: definition of the characteristics, selection of the profile, selection of the communication medium, implementation of the stack and, finally, the development of the *Application Program*. Then, the *Application Program* will be signed by the KNX association and certified using tests by certified entities.

KNX is a fully distributed network, which accommodates up to 65,536 devices in a 16 bit Individual Address space (see Figure 2). The IA of each device is composed by its area, line and device numbers, in the format *area.line.device*.

There are two addressing modes, corresponding to specific communication profiles and purposes:

- **Broadcast and Unicast**, used for network and resource management. As a first step, broadcast (optionally using a device's unique serial number) is used to assign a unique IA to the device. Then, point-to-point communications are used to upload the *applet* binary image of the device *Aplication Program* (firmware) and to directly communicate with the device.
- **Multicast** Group Addresses (GA) are used for full multicast (group) addressing, for runtime efficiency. The various devices have the ability to expose multiple Datapoints, which can be independently grouped as network-shared variables. These shared variables, with read and write capability, use a 16-bit address space that provides the system with a total of 64K GA space.

octet 0	1	2	3	4	5	6	7		8	..	N - 1	N ≤ 22
Control Field	Source Address		Destination Address		Address Type; NPCI; length	TPCI	APCI	data/ APCI	data			FrameCheck

FIGURE 3. KNX LPDU standard frame structure.

Figure 3 presents the Logical Protocol Data Unit (LPDU) structure of the KNX frame [7]. KNX messages are serially encoded in frames or telegrams which are sent over the bus. These messages include the following fields:

- Control Field – determines the frame priority and distinguishes between Standard and Extended frames (where $N < 255$).
- Source Address – the IA of the source.
- Destination Address – either an IA for point to point communication or a GA for multicast (group) communication.
- Address type – specifies the type of the destination address (IA or GA).
- Hop Count – is decremented by routers to avoid looping messages; when it becomes zero, the frame is discarded from the network.
- Length – the frame length.
- Transport Layer Protocol Control Information (TPCI) – controls the Transport Layer communication relationships (for instance, build up and maintain a point-to-point connection).
- Application Layer Protocol Control Information (APCI) – can tap into the full toolkit of Application Layer services (Read, Write, Response, ...) which are available for the relevant addressing scheme and communication relationship.
- Data – the message payload. Depending on the addressing scheme and APCI, the standard frame can carry up to 14 octets of data (even more on extended frames). Segmentation for bulk transfer, like the download of an entire application program, is handled by the management client (e.g. ETS tool). The standard frame ensures direct upward compatibility with EIB.
- Frame Checksum.

Considering the nature of the information conveyed over the KNX bus, it becomes obvious why its protection is a relevant matter – moreover if one considers the implications in terms of BACS security risks. The potential impact of such risks scales according with the dimension of the application scenario, which can range from a single house to a multi-story smart building supported by a complex automation infrastructure encompassing functionalities such as access/gate control, alarms, lightning or climate control.

If left unprotected, BACS can be exploited for malicious purposes, such as unlocking access to building premises or deactivating alarms, as well as allowing for intruders to eavesdrop unprotected data from presence detectors, energy consumption and administration programs. The manipulation of lighting control systems, heating control systems and other processes in building technology is also a potential risk. In response to these potential risks, KNX Secure [8] has been developed to improve cyber security in building automation, enabling protection at the message data level (*KNX Data Secure*) and at the level of communication over IP (*KNX IP Secure*). However, the inertia derived from the large number of existing legacy systems is a serious obstacle to the widespread deployment of KNX Secure. Existing devices would require upgrading and that is not possible by simple upgrade of their *Application programs*, due to insufficient memory and computing resources.

III. CYBERSECURITY AND BACS

Historically, BACS have been designed to work in an isolated fashion, without any connection to ICT networks. Security was supposedly achieved by isolation (the ''airgap principle'') and by the use of obscure proprietary solutions with poor or non-accessible documentation. Meanwhile, the popularity of ICT commodity technologies – driven by simplicity, lower total cost of ownership and improved operation, configuration and management – has led to the introduction of network bridging mechanisms. Suddenly, several underlying security assumptions were broken, and BACS became connected to globally accessible networks such as the Internet, without taking into account the potential implications in terms of security and safety.

As pointed out by Graveto et al. [9], BACS security breaches are often considered to be a consequence of using systems, protocols and standards that were originally conceived to operate in isolated environments, without any connection to ICT networks or the Internet. This is aggravated by the fact that many legacy devices cannot be patched, often meaning that only isolation or complete replacement might ensure adequate security [10]. In general, most attack categories that are characteristic of Industrial Automation and Control Systems (IACS) may be somehow transposed to BACS scenarios [11]. However, even though some the protection strategies used in IACS might somehow provide hints on how to keep BACS secure, there are considerable context differences that require domain-specific approaches. Similarly, the protections typically used in IACS at network level also need to be adapted to BACS.

Wendzel et al. [10] performed a comparative analysis of the security issues for some of the most widespread BACS communication protocols (KNX/EIB, BACnet, ZigBee and EnOcean). Attacks are identified as belonging to two different levels: network (management and automation levels of BACS architecture) and device (field level of BACS architecture). At network level, attacks are split into four different forms: traffic interception (network sniffing), malicious packet creation, network packet change (man in the middle attacks) and outage or reduction of network service quality (denial of service). On device level, attacks are grouped into three patterns: physical tampering, side channel analysis (e.g. monitoring for obtaining cryptographic keys) and software attacks such as code injection.

Graveto et al. [9] analysed several incidents involving real-world BACS, such as the attacks against the St. Regis Shenzhen luxury hotel [12], the Google's Wharf 7 building workspace [13], the Target Corporation (which was used to access the company's BACS systems [14]), and the Singapore Fragrance Hotel [15]. Overall, most of those attacks could have been easily detected and/or prevented with adequate monitoring and intrusion detection mechanisms.

There are also reports about a recent incident in Germany, in which a building automation engineering company lost contact with roughly three quarters of the BACS devices in an office building system network, after being locked out of the system by a cyberattack [16]. The company was forced to revert to manual operation of central circuit breakers to control the lights in the building, until a security consultant was able to retrieve the Bus Coupling Unit key from a bricked device's memory, in order to regain control. Once again, adequate monitoring mechanisms could have mitigated the impact of this attack – even basic system logs were unavailable to support forensics analysis, making system recovery much more difficult.

IV. INTRUSION DETECTION FOR BACS SCENARIOS

As already discussed, several successful attacks have occurred and existing mechanisms were unable to detect them, pointing to the need of better intrusion detection systems for BACS. Actually, there is already extensive work in anomaly and intrusion detection in related areas, such as cyber-physical systems and IoT [17], [18], [19], [20], but there are much less proposals in the specific topic of building automation.

Pedro and Silva [21] proposed the development of generic monitoring and actuation of home automation facilities for use with different technologies. This solution is based on the DomoBus technology, whose device abstraction model and communications service supposedly enable the development of configurable applications from XML (Extensible Markup Language) files – allowing for monitoring and control of device networks based on several technologies.

Jones et al. [22] used a Single Board Computer (SBC) to deploy an unsupervised artificial neural network to monitor building automation systems. The proof-of-concept used BACnet and all the network packets were stored and analyzed using an on-board Adaptive Resonance Theory neural network. When anomalies are found, the source and destination addresses are added to an access control list and those communications are blocked. In our opinion, this type of automated reactions may become a problem for the overall performance of the building system, and even constitute a

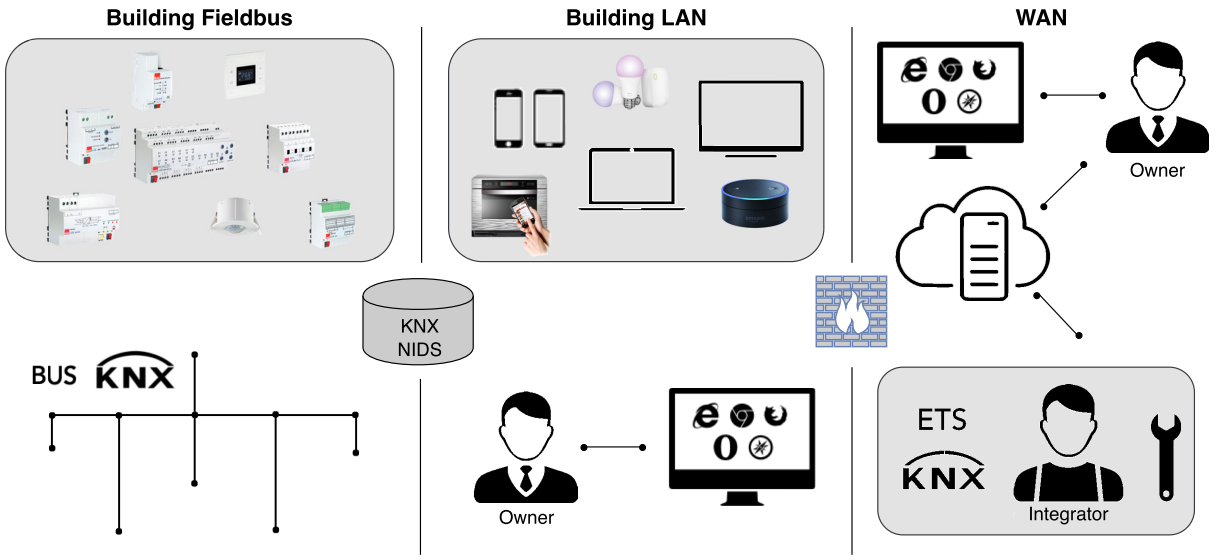

FIGURE 4. BACS Security Architecture.

new attack vector to be exploited by malware – as already known in similar domains such as IACS. This could be solved by means of human intervention in the reaction process, for improved safety.

A multi-modeling based approach, using a mix of modeling, simulation and analysis tools, was used by the CPS Association to design the INTO-CPS cyber-physical platform [23], which has been used to assess the security of smart buildings, using a *man-in-the-middle* attack for validation purposes [24].

A Hidden Markov model is proposed by Ramapatruni et al. [25] to identify anomalous activities. Sensor data from a smart home environment was used to train this model.

Chavis et al. [26] proposed a system that helps reducing the cognitive load on a user in keeping his smart home network secure. Machine Learning (ML) is used to achieve that goal, with data being collected and stored in pcap format [27].

Since the amount of previous work related with intrusion detection in BACS is so scarce, one might also look at works addressing intrusion detection in other domains.

The open source projects Snort [28], Suricata [29] and Zeek [30] are among the most widely used NIDS in TCP/IP networks in general. Bhosale and Mane [31] review these three solutions, concluding that Snort and Suricata are mostly signature-based IDS, while Zeek describes itself as a passive open source network traffic analyser for security monitoring purposes – e.g. capturing traffic and forwarding it to Security, Information and Event Management (SIEM) systems.

Dupont et al. [32] surveyed NIDS for Controller Area Network (CAN) systems. The CAN bus, mostly used for in-vehicle control systems, is an automation technology that also poses a challenge to intrusion detection systems – with a large number of messages that, without context, carry very little information. In some way, this is similar to BACS, where the majority of messages carry only binary information such as ON/OFF or UP/DOWN.

Considering the foreseen requirements for BACS NIDS in terms of embedded security probe capabilities and deployment, other works were also considered in this analysis. Al-Maksousy et al. [33] presented a real time monitoring system with very low CPU usage that is capable of detecting and classifying malware based on network behaviour, using split deep neural networks. Ficke et al. [34] analysed the differences between flow-based and packet-based NIDS in terms of detection effectiveness. Hinting towards possible optimisation strategies, Gouveia et al. [35] evaluated the performance of NIDSs with feature set tuning and reduction. Robinson and Kim [36] validated their intrusion detection framework by using a ModbusTCP control system, based on a SBC that allows the simulation of cyber-attacks and illustrates a mitigation measure with the added feature of Modbus monitoring using Snort. Also in this line, Graveto et al. [37] proposed the Shadow Security Unit, a monitoring probe designed to be attached in parallel to IACS control devices, being able to passively monitor the network communication flows and the physical process interfaces, in order to detect anomalies with potential impact on system safety and security.

V. PROPOSED BACS NIDS

The discussion presented in the previous section clearly identified a gap of security mechanisms for BACS environments. This motivated us to develop the BACS NIDS, an intrusion and anomaly detector that operates mainly at the control network level (fieldbus). The concept and main requirements are outlined in this section, with the architecture being presented in Section VI.

The proposed NIDS fits into the BACS security architecture, as shown in Figure 4, supporting local monitoring of:

- The building fieldbus where all the control devices are connected (e.g. actuators, blind control, sensors, light dimming, heating control systems).

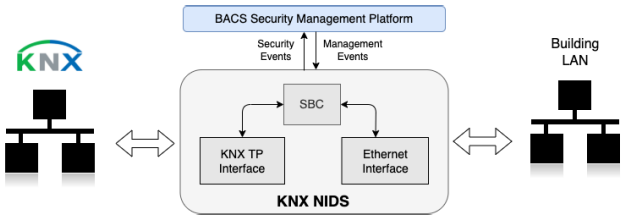

FIGURE 5. BACS KNX NIDS Integration Model.

- And the local area network, where commodity devices such as computers, phones and other devices (e.g. IoT devices) may be present.

The BACS NIDS also features a Web-based dashboard for (optional) local management, together with a secure interface for management and event feeds. The later is intended to provide integration into larger security frameworks, as well as support for off-site management by owners and building automation integrators.

A. THE CONCEPT

The proposed integration model for the BACS NIDS is illustrated in Figure 5. It is built around an SBC with two interfaces, one connected to the building fieldbus (KNX twisted pair or other support medium) and an Ethernet interface connected to the local building LAN. The proposed approach is also compatible with other building automation standards (e.g. BACnet, ZigBee, EnOcean), despite KNX being the target protocol for the proof-of-concept implementation hereby presented.

The BACS Security Management Platform constitutes a web application that can be either locally served by the KNX NIDS host or remotely exposed by a web server, using an out-of-band ethernet connection. The *security events* generated by the system are forwarded to the platform, processed and presented in a friendly user interface (UI). This same UI may also provide management and configuration capabilities, supported by KNX NIDS *management events*.

B. REQUIREMENTS

The design of the BACS NIDS considers the following key principles:

- Seamless and transparent operation – by design, most of the required evidence for processing should be obtained by passively monitoring the protocol message flow, without any interference in the normal operation of the BACS.
- Cost-effectiveness – the NIDS must be cost-effective when compared to regular automation devices, ideally being within the same price range.
- Protocol readiness – The device should be fully compatible with the target BACS standards (e.g. KNX), for stealth operation and compatibility with existing BACS devices.

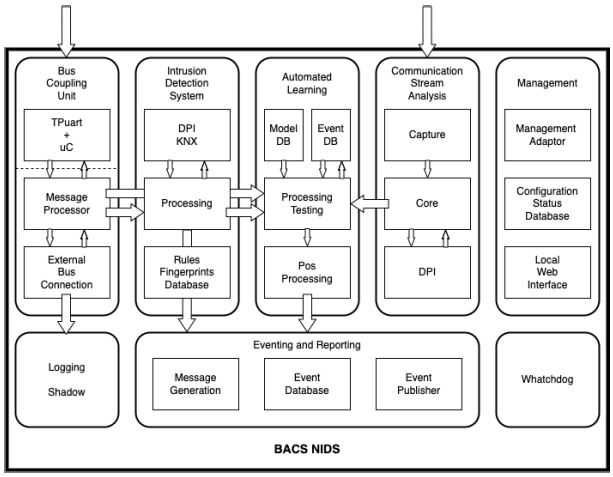

FIGURE 6. BACS NIDS Architecture.

In addition to those design principles, the NIDS is expected to address the following key requirements:

- Semantic command stream processing – the NIDS should be able to transparently capture and decode inflight protocol messages (e.g. KNX TP or KNX/IP).
- Reliability – when evaluated under stress testing, the NIDS should obtain consistent results for the same given use cases and should clearly identify ongoing anomalies or attacks.
- Trustworthiness – the NIDS must provide trusted results under different test use cases, encompassing validation of accuracy and false positive rates for the system.
- Stealthiness – the system should be invisible to outside attackers. In addition to the aforementioned passive monitoring abilities, this implies the support of (optional) out-of-band communication channels with the BACS security management platforms, to avoid exposure in the local LAN or in the KNX TP network.
- Auditing support – information and event records (both for alerts and metrics) must be preserved for auditing purposes.

Next, we present a PoC implementation of the proposed BACS NIDS.

VI. PROOF-OF-CONCEPT IMPLEMENTATION

The architecture of the PoC NIDS is presented in Figure 6. It is based on a neutral concept that is compatible with the majority of BACS protocols (e.g. KNX, BACnet, Modbus), communications technologies (e.g. KNX TP, KNX RF, Ethernet, Zigbee, RS-485) and deployment scenarios. Nevertheless, for sake of readability, some of the technical details discussed in the following subsections directly relate with the PoC prototype developed for BACS, which based on a KNX TP bus.

The remainder of this section discusses in more detail each of the main building blocks of the KNX NIDS architecture: bus coupling unit; intrusion detection system;

automated learning; communications stream analysis; management; shadow logging module; eventing and reporting; and watchdog.

A. BUS COUPLING UNIT

The Bus Coupling Unit (BCU) is responsible for the connection with the fieldbus (control network), passively capturing all in-flight traffic. While this module could also be able to actively interact with the control network (e.g. for active device fingerprinting), such capability is not leveraged in the PoC implementation. The BCU encompasses three main blocks:

- The bus coupler is an hardware device that connects to the bus (e.g. KNX TP, Zigbee, Ethernet). In the case of KNX TP, a shield was developed using a Twisted Pair – Universal Asynchronous Receive Transmit (TP-UART) and a microcontroller (ATmega 2560) that handles real time needs of the protocol and message flows to/from the host single board computer (SBC).
- The message processor handles the telegrams collected from the fieldbus, using the well-known pcap format. This block also enriches those messages with a timestamp, unifying the communication with other system modules.
- The External Bus Connection interfaces with the log system.

Naturally, the adaptation of this generic architecture for different BACS scenarios will impose the use of a specialized BCU modules for each type of system (e.g. KNX, BacNet, ZigBee). In some cases, when Ethernet transport is used, the Bus coupler may simply consist of a conventional network interface card.

B. INTRUSION DETECTION SYSTEM

The Intrusion Detection System (IDS) module provides the anomaly and/or attack detection functionalities. Depending on the capabilities of the host SBC, the IDS may implement several detection techniques, such as signature-based (recognising bad patterns, such as malware), anomaly-based (detecting deviations from known *good* working model), or reputation-based detection (scoring the reputation of potential threats and raising an alert when a predefined threshold is reached). This module includes the following building blocks:

- Deep packet inspection (DPI) – this block is used to decode the byte array of the messages from the building automation protocol. This block also depends on the underlying BACS technology (KNX, BacNet, ZigBee).
- Processing – this block constitutes the core processing unit of the IDS, using raw messages from the BCU *message processor* and *automated learning* blocks as input for decision making, supported by a pattern database. It can also use the DPI module, if needed.
- Database – persists and stores the information about the patterns used to process and identify the anomalies or attacks (e.g. fingerprints, rules).

The aforementioned modules provide the basic protocol data acquisition and threat detection capabilities, which are to be complemented by the *automated learning* module described next.

C. AUTOMATED LEARNING

The information from the BCU, the IDS and the communication stream analysis feeds this module, that locally implements a correlator and a classifier based on Artificial Intelligence (AI) techniques. The correlator may perform event reduction and aggregation within preferred time windows, as well as checking if commands arrive from a legitimate source, if they are coherent with the expected control interface flow, and if I/O information is in line with expected values. The classifier is supported by models that, due to potential limitations of computational resources of the NIDS device, may be built outside the NIDS device and imported to the *Model DB*. To build the required models, a learning phase may be executed during an initial monitoring period, before entering into detection mode. Alternatively, an external dataset may be used to skip this stage. The correlation rules, the module database and the learning processes are controlled via the *Management* module (cf. Section VI-E).

The *Automated learning* module provides a generic framework where different models can be uploaded and classification algorithms can be deployed. It is supported by four functional blocks:

- Model DB – stores the preloaded modules to be used by the AI processor, as well as minor updates resulting from model improvements in real time.
- Event DB – this database stores the inputs or aggregated raw events, allowing for time window-based and basic batch processing capabilities.
- Processing/Testing – this is the core block where correlation and classification take place, producing the output of the *Automated Learning* module.
- Post-Processing – this block formats the module outputs according to the established data model, for subsequent handling by user interfaces or external security systems.

D. COMMUNICATIONS STREAM ANALYSIS

This module connects to the Ethernet LAN. It is responsible for capturing network traffic and forwarding it to the corresponding modules, for processing and analysis. The use of a passive TAP allows to intercept traffic in a seamless way, thus hiding its presence from potential attackers.

The *Communications stream analysis* is functionally similar to the bus coupler module, but is connected to the building LAN instead of a fieldbus. It is composed by three functional blocks:

- Capture – this block gets the byte array messages from the link, queues them and feeds the remaining blocks, also performing timestamping.

- Core – processing block that controls encoding, integrity checking and the connection to the *IDS* and *Automated learning* modules.
- DPI – this block is invoked as needed for deep pack inspection and decoding.

E. MANAGEMENT

The *Management* module provides the means to configure and persist settings for the device, thus constituting the main interface to the KNX NIDS, both for local and remote operations. It encompasses three key functional blocks:

- An adapter that exposes a secure TLS connection for interaction with broader security management platforms and/or for remote access by integrators and building owners.
- A database service that provides access to the configurations of the various BACS NIDS modules, that are persisted in local storage – thus allowing for device reconfiguration.
- A lightweight local web server that (optionally) allows an operator to query basic functional indicators and/or to configure the system.

F. LOGGING SHADOW

This module provides a database to log incoming fieldbus protocol messages. It can be used either for auditing purposes or to assist debugging (possible) operational faults, by comparing the differences between intercepted messages and the ones actually sent.

G. EVENTING AND REPORTING

The *Eventing and Reporting* module processes all the output events from the KNX NIDS, from both the *IDS* and the *Automated Learning* subsystems. This module takes care of the BACS NIDS security event stream, also persisting messages/events to provide support for disconnected operation, in case of communication interruptions. It is composed by the following functional blocks:

- Message Generation – aggregates and processes all the events from other modules and generates the output events, aligned with an established data model.
- Event Database – persists the output events for retrieval by local or remote consumers.
- Event Publisher – implements the interface between the system and the security event consumers.

All the KNX NIDS outputs are processed via this module, thus ensuring integrity and consistent temporal sequencing for future analysis.

H. WATCHDOG

A watchdog module takes care of self-monitoring, for both component and service operation. This watchdog leverages the Docker framework to provide isolation, implementing a series of software routines that periodically check component operation and attempt recovery or restart in case of stalled operation. Moreover, it also provides system-level checks through a Linux kernel module working together with a watchdog service that provides regular feedback to the hardware watchdog timer of the SBC. Using the hardware watchdog makes it possible to reboot the entire BACS NIDS platform in case of failure, after a predefined number of missed timer events.

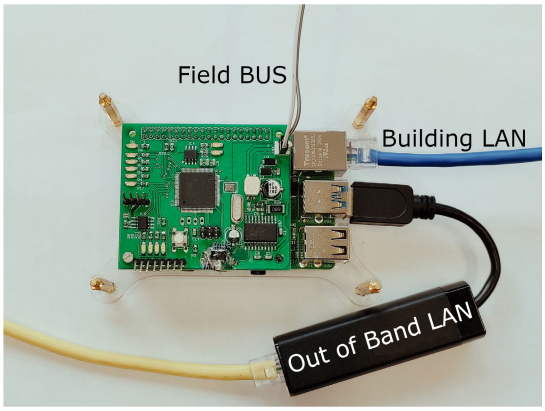

FIGURE 7. External view of the KNX BACS NIDS.

I. OVERVIEW OF THE PROOF-OF-CONCEPT

Figure 7 presents the external view of the proof-of-concept BACS NIDS that has been built for demonstration and evaluation purposes, addressing the specific scenario of KNX TP BACS systems.

A Raspberry Pi 4 was adopted for the PoC host SBC, due to its availability, expandability and price/performance ratio, as well as the considerable amount of available documentation and related information sources. Moreover, a hardware watchdog is already supported, thanks to the native CPU watchdog driver for the RPi [38]. Overall, the PoC cost was kept below 200€, which is deemed acceptable for a prototype, moreover considering that a mass produced system could easily cost a fraction of this value (probably less than 100€ for the most common types of fieldbus technologies), due to savings obtained via bulk component acquisition and increased component integration.

The connection to the KNX bus was achieved by developing a shield, connected to SBC GPIO interface. This shield, depicted in Figure 8, contains a TP-UART (optically isolated from the SBC host using a ILD213T optocoupler) that provides fieldbus connectivity, as well as an ATmega 2560 micro controller that establishes the interface between one of the SBC serial ports and the TP-UART serial line. An interface to a PCD8544 LCD was added to locally display device information, and a 24C02C EEPPROM was used to persist data.

The KNX NIDS Operating System is based on a Debian Linux distribution (Raspbian). Docker was used to deploy each functional module in a separate container, for sake of improved isolation, stability and reliability of the overall system.

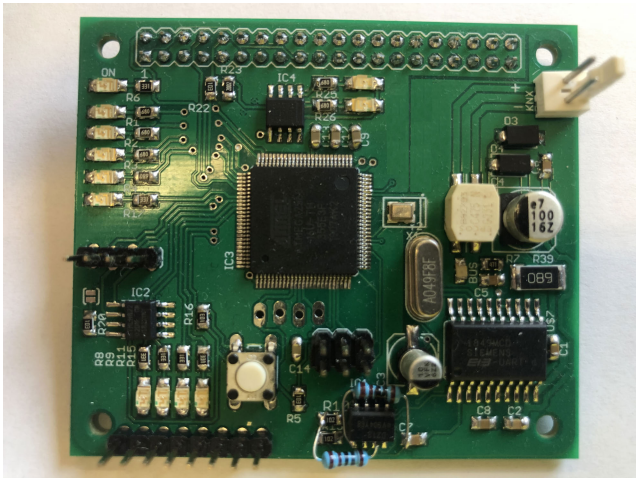

FIGURE 8. KNX TP shield for BACS NIDS.

Most modules were developed using the GO language and taking advantage of the open source GoPacket [39] library for decoding and encoding messages. Since this library provided no support for KNX, we extended it with several functionalities for manipulating KNX protocol flows. The communication between different modules uses the ZeroMQ library [40].

Apart from the KNX-TP shield (which was integrated into a single board), the whole prototype was built using commercial of-the-shelf hardware components, with no special optimization. The ATmega micro-controller could implement the required KNX stack for commercial certification as a KNX device, and the SBC could be stripped from the unnecessary components, eventually being replaced by a Raspberry Pi Compute module. This means that a mass-produced KNX NIDS would take a fraction of the prototype footprint.

Overall, the presented PoC fulfils all the requirements deemed crucial for the implementation of a BACS NIDS (cf. Section V). Next, we discuss some of the capabilities of the current implementation.

VII. DEMONSTRATION OF THE NIDS CAPABILITIES

This section describes how the proposed NIDS is able to detect BACS anomalies and security attacks, based on rules, heuristics and AI techniques. For this purpose, we deployed the NIDS in a real KNX environment and staged various types of attacks.

A. TESTBED

The validation of the proposed architecture was carried out using a single family house as reference scenario. This three-floor house has a building automation system based on KNX. It includes four bedrooms, kitchen, two living rooms, office, laundry, and a machines room.

For validation purposes, we deployed the KNX NIDS in house. The BCU was connected to the KNX bus, and the Communication Stream Analysis module was connected to the house LAN.

The BACS consists of around eighty control devices, automating the operation of lighting systems, shutters, central heating, security, interior and exterior monitoring, etc. The system is based on KNX and has devices such as actuators, switches, motion and luminance sensors, temperature sensors, roller shutter actuators, solenoid controllers, power supplies, a touch panel for information and action, a weather station, and security and alarm devices.

The KNX network uses a twisted pair line (1.1.X) for the connection of all devices. A wide range of three-tier group addresses is used for the automation system operation. The main group distinguishes the various floors, the middle group sectorizes the various services (lighting, scenarios, blinds, air conditioning, alarm...) and the address specifies each of the spaces or zones (for instance, the group address 2/0/8 corresponds to the second floor – 2, light on/off – 0, and the kitchen room – 8).

B. REFERENCE SCENARIOS

The two scenarios that were considered allow, when combined, the validation of a wide array of potential attacks.

1) SCENARIO 1 – COMPROMISED DEVICE

The first approach for validating the tool is to assume that somehow the attacker managed to access at least one of the BACS devices and compromised its behavior. The establishment of such a bridgehead is typically one of the first steps in the cyber kill chain.

The attacker's objective is to understand the building's control network and, later, to collect detailed information about the various devices and their programming. Executing the necessary commands from a valid device will allow these tasks to be somehow hidden and carried out with little interference in the normal system operation.

The KNX *Line Scan* operation consists of the repeated use of an attempt to establish a one-to-one communication link with all possible destination addresses of a given line. This procedure allows identifying valid and existing Individual Addresses in the system.

After obtaining those valid IA's, the simple use of the KNX *Device info* function allows querying these devices, obtaining information such as manufacturer, data, configuration parameters. This extraction of information is possible due to the KNX specifications, since according to those specifications all devices must report their data and characteristics whenever queried.

These commands are not typical during normal BACS operation, since they are used mostly for debugging and system development. Therefore, their detection outside of this scope should trigger a security alert.

2) SCENARIO 2 – NETWORK ACCESS

The second approach used in validation to assume the attacker somehow got access to the BACS field network, being able to read and inject KNX messages.

KNX devices ignore malformed messages, therefore our tests use valid messages with malicious purposes – even

though the tool could also be used to identify or remove invalid KNX messages, as mentioned in Section VII-D1.

In this scenario several attacks can be performed, such as:

- creating malicious messages from an unknown sender, with the purpose of hacking or controlling other devices;
- injecting a large number of messages in a short period of time, with the aim of causing a denial of service situation;
- simulating human action by sending messages that are valid but not appropriate for the context (such as the faked activation of a physical switch in rooms where presence detectors report no human presence).

These cases were used as the basis for the experiments described next.

C. EXPERIMENTS

The experiments were carried out in two steps. First, we monitored the testbed system to capture fieldbus traffic in regular operation, in order to create a dataset of *normal operation*. Next, we staged several different attacks to assess the detection capabilities of the BACS NIDS.

The BACS system was monitored during thirty-seven days, to create the initial dataset. This dataset, which has been made publicly available [41], was archived using the pcap format [27]. It contains 379,875 timestamped messages, with a total size of 16.9 MiB. The dataset was processed to extract features such as:

- arrival time between messages;
- number of messsages per second;
- cumulative average of messages per second;
- message size;
- and message checksum.

Next, we describe each of the attacks we carried out in the scope of these experiments.

1) LINE SCAN ATTACK

The *line scan attack* consists of sending a sequence of messages of type *TConnect* (a connection request for one-to-one communication). Requests are sent in sequence to all possible individual addresses (IA). The used sender address (SA) was a valid address IA 1.1.102 (a switch located in one of the rooms, which was previously hacked for the purpose of this attack). A total of 1,649 malicious messages were sent.

Whenever a valid device is present on the destination IA, it replies with a *TDisconnect* message. Then, the attacker sends a new *TConnect* and a *DeviceDescription* request, followed by *TACK* and *DeviceDescription* messages from the target device. Finally the attacker sends a *TACK* and *TDisconnect* messages ending that conversation.

2) DEVICE INFO ATTACK

The *device info attack* consisted of an information request to two known devices (a light and motion sensor and a switch in one of the rooms). Again, the attacker uses a valid but previously compromised light switch.

For each *device info* request the attacker sends *TConnect* and *DeviceDescription* messages. The device replies with *TACK* and *DeviceDescription* messages. Then, the attacker sends *TACK* and *MemoryRead* messages, which are replied with *TACK* and *MemoryResponse* messages. The attacker sends *TACK* and *ADCRead* messages, replied with *TACK* and *ADCResponse* messages. The last two exchanges are repeated a few times to gather the device information and, finally, the attacker ends the conversation by sending the last *TACK* and *TDisconnect* messages.

3) MESSAGE INJECTION ATTACK

The *message injection attack* consists of sending valid KNX messages with different rates and contents, with the purpose of disturbing the BACS. Two variants were considered, according to the message rate.

The *slow rate message injection attack* uses KNX messages with valid fields but with invalid SA or DA, injected at random intervals. Three different datasets were created, with different attack message ratios (1%, 5% and 10 % of the total bus messages during the attack periods, respectively).

The *high rate message injection attack* consists of a Denial of Service (DoS) attack where a large number of messages is injected in a short period (one thousand messages per second during a 15s period). A valid KNX message is injected, with valid SA and DA, with a command payload to open the living room blinds. A message of type *GroupValueWrite* is sent to GA 2/4/3 with value 0×00.

4) INVALID CONTEXT ATTACK

Invalid context attacks explore the possibility of an apparently valid message, with apparently valid addresses, being issued out of context. For instance, pressing a switch will turn on the lights in the bedroom. However, it will not be possible for a human to physically access the switch without being flagged by the bedroom presence detector. Thus, a light activation on behalf of the switch, without presence detection, will likely be a message improperly inserted in the system and out of context. In real-world scenarios, this principle can be used to create more complex context rules.

For instance, in our testbed a motion detector (with IA 1.1.120) emits a message of type *GroupValueWrite* to turn on the room light, whenever movement is detected in room. When pressed, the light switch also emits a *GroupValueWrite* message with a value of 0 or 1 (requesting a suite bedroom light to turn off or on).

The analysis of the original network trace with a relatively short time window allows, in normal situations, to verify that whenever the switch is pressed, there is also a motion detection. Thus, the existence of switch messages not matched by motion detection messages is abnormal and interpreted as a potential sign of attacks.

D. DETECTION TECHNIQUES

The concept proposed in this paper supports two different types of approaches to detect anomalies and potential cyber

TABLE 1. Field Values of IDS Rules.

action	protocol	source address (SA)	source port (SPort)	operator	destination address (DA)	destination port (DPort)
pass alert log	any knxtp KNXTP	any IA	any	$->$ <>	any IA GA	any

attacks: rule-based detection and AI-based anomaly detection. Next, we describe some of the specific techniques we explored in the scope of the validation work.

1) RULE-BASED DETECTION

As described in Section VI-B, the IDS module provides the anomaly and/or attack detection capabilities, supported by patterns or fingerprints that are stored as rules on the existing local database. To the best of our knowledge, there is no previous work regarding the development of rule-based detection mechanisms for KNX. For this reason, we created a domain-specific syntax that resembles (with specific adaptations to KNX) the well-known SNORT syntax [42]. The syntax has the following structure:

```
action protocol SA SPort operator DA DPort (options)
```

The valid values for each field are provided in Table 1.

The keyword `any` specifies all the messages (packets) that satisfy a condition. The `pass` action ignores the message that satisfies the rule, `alert` action issues an *alert* when it identifies a message that satisfies the rule, while the `log` action exports that message to a PCAP file. The `protocol` filters the search for the specified protocol (e.g., KNX). The operators are used to apply the rule in one or both directions.

Table 2 presents the available options, which can be combined in the same rule (`content` can actually be used multiple times to build the intended pattern or fingerprint). All option values can use the *!* (not) operator, thus facilitating the construction of rules. In the following example, all contents that do not include the word *light* will trigger an *alert*:

```
alert any any any -> any any ( msg:"exclude
light msg"; content: !"light"; rid:200; rev:1; )
```

For each packet, the rule engine goes sequentially through the rules until it finds a match. After this trigger, the engine will: (i) jump to the next message in case of a `pass` action; (ii) write to the output file in the case of a `log`; and (iii) trigger an *alert* in case of an `alert` rule. Rules can be combined to express more complex cases.

This process allows a multitude of uses for the tool. For instance, to create a file containing only valid KNX messages (excluding all others), the following rule could be used:

```
log knxtp any any -> any any ( msg:"valid KNX msg";
pcre:".+"; rid:300; rev:1; )
```

The DPI KNX functional block of the IDS is used to decode KNX messages whenever the protocol is specified or when the source or destination addresses are used on any rule. Moreover, this module also implements basic feature extraction, computing statistics such as: time between two consecutive messages; number of messages per second; moving average of messages per second; and message size.

TABLE 2. Available options for IDS Rules.

Option Keyword	Mandatory	Usage	Example
msg	No	simple text to identify the rule	msg: "this is a rule";
tpci	No	used to search a transport layer function	tpci: "T_Connect-PDU";
apci	No	used to search an aplication layer function	apci: "A_GroupValue_Write_PDU";
content	No	text or hexadecimal bytes (allowed multiple times on same rule)	content: "light"; or content: "IBCl";
pcre	No	use of regular expression to search the message	pcre: ".+";
rid	No	integer to identify the trigged rule	rid: 5;
rev	No	integer that represents the revision of the rule	rev:3;

2) AI-BASED ANOMALY DETECTION

Since rule-based detection is only able to handle previously known attacks, AI-based anomaly detection was also explored. Anomaly detection using AI usually resorts to classifiers, which can require resources not in line with the limited hardware capabilities of devices such as the proposed BACS IDS. For this reason, in the proposed architecture AI models are built and trained in external equipment (based on previously collected traffic captures), and then imported to the IDS, to allow real-time traffic analysis and anomaly detection.

The use of AI is already widespread in many cyber-security application domains. However, a search for its application in the specific scope of BACS cyber-security revealed few examples. Patil et al. [43] present a machine learning (ML) algorithm to distinguish between normal operation, malfunctions and attacks on a BACS. The scenario consists of a building with HVAC control, fire alarm, access control and lighting, in which a bi-linear classifier is used to distinguish between the three situations, using various previously trained bi-linear classifiers. Chavis at al. [26] developed the Connected Home Automation Security Monitor (CHASM), which uses a Multiclass Decision Forest classifier to identify present IoT devices, and intend to use Multiclass Neural Networks to leverage the time series nature of IoT data in order to characterize normal and abnormal behavior of an IoT-based BACS. Finally, Ramapatruni et al. [25] use Hidden Markov Models to detect operating anomalies in a smart home.

Looking at the somehow related domain of IACS, it is possible to find a wider range of works [44], [45], [46]. Rosa et al. [47], for instance, propose a flexible intrusion detection platform able to support multiple AI-based anomaly detection tools. Anton et al. [48] used Support Vector Machines and Random Forests for intrusion detection. Philips et al. [49] used a SCADA system based on the Modbus protocol to evaluate the use of SVM, Decision Trees, K-Nearest Neighbors

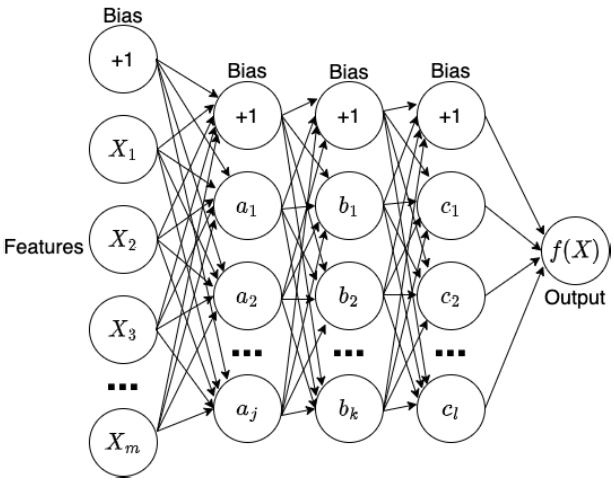

FIGURE 9. Multi-layer perceptron network.

and K-Means Clustering techniques for anomaly detection. Keliris et al. [50] also use SVM-based algorithms as a defense for process-aware attacks in IACS.

To assess the suitability of AI-based anomaly detection in the scope of the proposed BACS IDS, we explored three techniques: neural networks, support vector machines, and logistic regression.

For the **neural network** (NN), we used a sklearn's multi-layer perceptron [51] which produces, through supervised learning with backpropagation, a non-linear function $f(X)$: $R^m - > R^o$ that represents the data. The training process was accomplished using the datasets collected in our reference scenario. The resulting model was created and then imported into the KNX NIDS.

Figure 9 represents the used neural network, with three hidden layers (a, b, c) and, respectively, 100, 20 and 100 neurons. Each neuron in the hidden layers transforms the values received from the previous layer through a weighted sum $(w_1x_1 + w_2x_2 + \ldots + w_mx_m)$, followed by an activation function. The *ReLu* rectified linear unit activation function is used. The *Adam* solver, a stochastic gradient-based optimizer, is used for weight optimization. Finally, the output layer receives the values from the last hidden layer and computes them into the output values.

Fine tuning of this network is beyond the scope of the present work. Nevertheless, as this type of model is quite sensitive to feature scaling, the data values were previously standardized to a mean of zero and a variance of one.

The **support vector machine** (SVM) proposed by [52] was adopted as the second AI approach for our experiments. For binary classification purposes, the created dataset entries were labeled as 1 (true) or 0 (false), to signal if they belong to an attack or normal operation.

The input vectors are non-linearly mapped into a high-order multidimensional space, where a linear decision surface is constructed for allowing the separation of input data in two distinct groups – by maximizing the margin between instances of different classes.

We used cost and regression loss epsilon with values of 1.00 and 0.10, respectively, where cost represents the penalty in terms of loss and epsilon the distance between true values within which no penalty is associated with the predicted values. The *RBF* kernel function $exp(-g|x-y|^2)$ was used to transform the attribute space into a new feature space, where $g = 1/k$, with k being the number of attributes.

Before creating the model, the following preprocessing tasks are sequentially undertaken:

- removal of instances with unknown target values;
- turning of categorical variables into continuous variables;
- removal of empty columns;
- imputation of missing values with mean of surrounding values.

As with the neural network, the model was built using an external computer and later imported into the KNX NIDS. The IDS supports the coexistence of several models, allowing them to be used separately or in parallel, in the scope of ensemble approaches.

Logistic regression (LR) consists of estimating parameters for a logistic model, which are the coefficients necessary for the linear combination of one or more variables. It uses the natural logarithm to create a continuous criterion that maps probability to log odds using a linear combination of one or more independent variables.

We used Ridge regression [53] as the regularisation type. This is a method to estimate the parameters in scenarios where linearly independent variables are highly correlated. Thus, using a dataset with N points, each one represented by a set m of variables $(x_1, x_2, \ldots, x_m)$, called independent variables, and a dependent variable, the output y (in our dataset named *target*), the logistic regression allows creating a predictive model for that output.

Again, the models were produced in an external computer and later imported to the KNX NIDS, to enable real-time classification of received messages.

The **approach followed for the training and testing phases**, for AI-based detection, is described next.

The datasets were split using 70 per cent for training and 30 per cent for testing. The following features were considered:

- InterMessageTime
- NumMessagesSec
- AvgMessagesSec
- MessageSize
- MessageChecksum
- SA_int
- DA_int
- KNX_info_int
- KNX_Code

First, the three methods (*neural network*, *support vector machine* and *logistic regression*) were explored, using ten-fold cross-validation, to identify which ones best suited each attack scenario. Next, the models were trained using the entire

training subset, being saved for later use. Afterwards, the produced models were loaded and tested in the scope of each attack scenario (using the testing subset).

Observed results are presented using confusion matrices. Moreover, we used the following metrics:

- Area Under the Curve (AUC) – area under Receiver Operation Characteristic (ROC) curve, which is created plotting the True Positive Rate (Recall) against the False Positive Rate (FPR)

$$\frac{FP}{FP + TN} \tag{1}$$

- Accuracy (CA)

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

- F1

$$2 * \frac{Recall * Precision}{Recall + Precision} \tag{3}$$

- Precision

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

- Recall

$$\frac{TP}{TP + FN} \tag{5}$$

E. OBTAINED RESULTS

In this section we analyse the results obtained for each type of attack previously presented. For rule-based detection, we present mostly the rules that have been defined and discuss observed detection capabilities. Regarding AI techniques, we resort to the aforementioned confusion matrices and metrics.

1) MESSAGE INJECTION ATTACK

In order to assess the IDS capabilities, we performed an attack consisting of sending a thousand valid messages per second during a period of 15s, which corresponds to an abuse of the system.

The statistical features were extracted from the corresponding capture and compared to those of the initial data (without the attack). This attack was easily detected both by AI techniques and a basic ruleset defining a threshold for network traffic rates. Patient attackers could try to reduce the rate of the attack, to bypass detection, but at some point the attack would become useless since it would put no stress on the system.

2) LINE SCAN ATTACK

Line scan operations provide the means for an attacker to undertake scouting operations in a KNX environment. However, this is an operation which is not commonly used in production environments, raising the suspicion of security issues. The compromised device (with IA 1.1.102) was used to send a sequence of messages of type *TConnect*, representing a request to establish a one-to-one connection. To detect this situation, a specific rule was added into the IDS signature testing module database:

```
alert knxtp any any -> any any ( msg:
"Transport Layer Connect {-} Line Scan detection";
tpci:"T_Connect-PDU"; rid:20; rev:1; )
```

This rule allows to flag *knxtp* protocol messages from *any* source to *any* destination, and a Transport Layer Control Field (tpci) equal to *T_Connect-PDU*.

The rule-based mechanisms detected connection attempts to all possible Individual Addresses (DA 0 to 255), coming from 1.1.102, as shown in Listing 1.

```
2020/04/29 16:00:22 detection: Engine: [**] [20:1]
Alert - "Transport Layer Connect - Line Scan
detection" on packet 15403[**] - SA: 1.1.102
DA:0.0.0 - T_Connect-PDU

2020/04/29 16:00:22 detection: Engine: [**] [20:1]
Alert - "Transport Layer Connect - Line Scan
detection" on packet 15404[**] - SA: 1.1.102
DA:0.0.1 - T_Connect-PDU

2020/04/29 16:00:22 detection: Engine: [**] [20:1]
Alert - "Transport Layer Connect - Line Scan
detection" on packet 15405[**] - SA: 1.1.102
DA:0.0.2 - T_Connect-PDU

(...)

2020/04/29 16:00:22 detection: Engine: [**] [20:1]
Alert - "Transport Layer Connect - Line Scan
detection" on packet 15657[**] - SA: 1.1.102
DA:0.0.253 - T_Connect-PDU

2020/04/29 16:00:22 detection: Engine: [**] [20:1]
Alert - "Transport Layer Connect - Line Scan
detection" on packet 15658[**] - SA: 1.1.102
DA:0.0.254 - T_Connect-PDU

2020/04/29 16:00:22 detection: Engine: [**] [20:1]
Alert - "Transport Layer Connect - Line Scan
detection" on packet 15659[**] - SA: 1.1.102
DA:0.0.255 - T_Connect-PDU
```

Listing 1. KNX NIDS Line Scan output.

3) DEVICE INFO ATTACK

In the *device info* attack scenario (cf. Section VII-C2), the compromised switch (IA 1.1.102) unduly questions a luminosity/presence detector (IA 1.1.142) and a switch (IA 1.1.136), to gather information about their settings and capabilities – using the KNX functions *Device Description* and *Memory Read*, besides the necessary handshakes to establish one-to-one communication with the two target devices. For this attack, we evaluated both AI and rule-based detection, as discussed next.

Regarding AI techniques, all methods achieved almost perfect scores, as expected (due to the nature of the attack). SVM had one false negative (cf. Table 3) and the neural network had one false positive (cf. Table 4). As for logistic regression, all the 2 595 fraudulent messages were detected, and no regular messages were wrongly flagged as fraudulent (cf. Table 5).

Regarding **rule-based detection**, it is known in advance that the attack will need to use the following specific sequence of messages:

TABLE 3. Device Info Attack - SVM Confusion Matrix.

		Predicted		
		0	1	Total
Actual	0	114013	0	114013
	1	1	2594	2595
	Total	114014	2594	116608

TABLE 4. Device Info Attack - NN Confusion Matrix.

		Predicted		
		0	1	Total
Actual	0	114012	1	114013
	1	0	2595	2595
	Total	114012	2596	116608

TABLE 5. Device Info Attack - LR Confusion Matrix.

		Predicted		
		0	1	Total
Actual	0	114013	0	114013
	1	0	2595	2595
	Total	114013	2595	116608

```
alert knxtp any any -> any any (
msg:"Application Layer - Device Info 1";
apci:"A_DeviceDescriptor_Read_PDU"; rid:31; rev:1; )

alert knxtp any any -> any any (
msg:"Application Layer - Device Info 2";
apci:"A_Memory_Read_PDU"; rid:32; rev:1; )

alert knxtp any any -> any any (
msg:"Application Layer - Device Info 3";
apci:"A_ADC_Read_PDU"; rid:33; rev:1; )

alert knxtp any any -> any any (
msg:"Application Layer - Device Info 4";
apci:"A_DeviceDescriptor_Response_PDU";
rid:31; rev:1; )

alert knxtp any any -> any any (
msg:"Application Layer - Device Info 5";
apci:"A_Memory_Response_PDU"; rid:32; rev:1; )

alert knxtp any any -> any any (
msg:"Application Layer - Device Info 6";
apci:"A_ADC_Response_PDU"; rid:33; rev:1; )
```

Listing 2. Device Info Attack – Detection Ruleset.

```
A_DeviceDescriptor_Read_PDU
A_DeviceDescriptor_Response_PDU
A_Memory_Read_PDU
A_Memory_Response_PDU
A_ADC_Read_PDU
A_ADC_Response_PDU
```

Based on that, a set of rules was created for detecting this type of attacks, as depicted in Listing 2.

When using this ruleset, attacks were successfully detected, as shown in Listing 3 (in the presented example, the attack started from the node with IA 1.1.142).

It should be noted that, knowing that normal system operation is based mostly on application-level messages such as *A_GroupValue_Write_PDU*, the negation operator ! could be used to produce the following rule:

```
2022/04/29 16:44:56 detection: Engine: [**] [31:1]
Alert - "Application Layer - Device Info 1"
on packet 83292 [**] - 0xB01166118E514300B5 -
SA: 1.1.102 DA:1.1.142 - T_Data_Connected-PDU -
A_DeviceDescriptor_Read_PDU
(...)
2022/04/29 16:44:56 detection: Engine: [**] [31:1]
Alert - "Application Layer - Device Info 4"
on packet 83297 [**] - 0xB0118E11665343400013E4 -
SA: 1.1.142 DA:1.1.102 - T_Data_Connected-PDU -
A_DeviceDescriptor_Response_PDU
2022/04/29 16:44:56 detection: Engine: [**] [32:1]
Alert - "Application Layer - Device Info 2"
on packet 83302 [**] - 0xB01166118E5346010104B6 -
SA: 1.1.102 DA:1.1.142 - T_Data_Connected-PDU -
A_Memory_Read_PDU
(...)
2022/04/29 16:44:56 detection: Engine: [**] [32:1]
Alert - "Application Layer - Device Info 5"
on packet 83307 [**] - 0xB0118E1166544641010402F3 -
SA: 1.1.142 DA:1.1.102 - T_Data_Connected-PDU -
A_Memory_Response_PDU
2022/04/29 16:44:56 detection: Engine: [**] [33:1]
Alert - "Application Layer - Device Info 3"
on packet 83312 [**] - 0xB01166118E5249810835 -
SA: 1.1.102 DA:1.1.142 - T_Data_Connected-PDU -
A_ADC_Read_PDU
(...)
2022/04/29 16:44:56 detection: Engine: [**] [33:1]
Alert - "Application Layer - Device Info 6"
on packet 83317 [**] - 0xB0118E11665449C10805A9DF -
SA: 1.1.142 DA:1.1.102 - T_Data_Connected-PDU -
A_ADC_Response_PDU
(...)
2022/04/29 16:44:56 detection: Engine: [**] [32:1]
Alert - "Application Layer - Device Info 5"
on packet 83377 [**] - 0xB0118E1166546241010900D8 -
SA: 1.1.142 DA:1.1.102 - T_Data_Connected-PDU -
A_Memory_Response_PDU
```

Listing 3. Device Info Attack – output from rule-based IDS.

```
alert knxtp any any -> any any (
msg:"Application Layer {-} Device Info";
apci:!"A_GroupValue_Write_PDU"; rid:34; rev:1; )
```

4) INVALID CONTEXT ATTACK

The implemented *invalid context* attack consisted of sending a false message (containing a valid source address from a real light switch device) to the network. The attack is out of context as it is physically impossible for a human to use this light switch without being signalled by the presence detector present in the same room. Implementing rule-based detection for this case can be easily accomplished by resorting to temporal sliding windows to process multiple messages aggregated in narrow intervals and find infringing patterns. Nevertheless, this approach implies the overhead of manually identifying patterns characteristic of normal operation for a specific deployment, to be later encoded as rules.

As an alternative, the use of AI-based techniques can potentially detect such attacks without requiring the prior definition of all possible invalid context situations.

Table 6 presents the obtained results for each technique. *Logistic Regression* and *neural networks* obtained excellent results, when compared with *SVM*, and therefore only those techniques were explored in more detail.

Table 7 shows that *Neural Network* had 50 false positives and no false negatives. *Logistic regression* (see Table 8) presents worse results, missing the identification of 86 of the

TABLE 6. Invalid Context Attack - Method Analysis.

Model	AUC	CA	F1	Precision	Recall
SVM	0.954	0.893	0.935	0.990	0.893
Neural Network	1.000	1.000	1.000	1.000	1.000
Logistic Regression	0.997	0.998	0.998	0.998	0.998

TABLE 7. Invalid Context Attack - NN Confusion Matrix.

		Predicted		
		0	1	Total
Actual	0	113897	50	113947
	1	0	1154	1154
	Total	113897	1204	115101

TABLE 8. Invalid Context Attack - LR Confusion Matrix.

		Predicted		
		0	1	Total
Actual	0	113755	192	113947
	1	86	1068	1154
	Total	113841	1260	115101

total 1154 messages involved in the attack and, even worse, generating 192 false alerts.

VIII. CONCLUSION

Despite their relevance, from a security and privacy point of view, BACS ecosystems suffer from a lack of domain-specific security tools, since general-purpose tools are unable to fully address the requirements of such environments. To address this gap, in this paper we proposed a domain-specific NIDS designed for BACS environments, able to monitor and analyse fieldbus traffic and able to fit into typical BACS scenarios, in terms of physical deployment, costs and management interfaces.

After introducing the proposed concept, we described a specific PoC implementation for KNX TP fieldbus, discussing its software architecture, hardware details and generic detection and management capabilities. Afterwards, we demonstrated these capabilities in the scope of a real BACS scenario, injecting several types of attacks and using statistical analysis, signature-based rules and artificial intelligence techniques to detect those attacks. Overall, these examples show how the NIDS can cope with a wide range of real-life attacks.

Finally, it should be noted that the dataset created and used for this evaluation work is available in [41]. To the best of our knowledge, this is the first KNX dataset publicly available to the research community.

REFERENCES

[1] KNX Association. (May 1999). *The Legacy of KNX*. [Online]. Available: https://www.knx.org/knx-en/for-professionals/What-is-KNX/KNX-History/index.php
[2] American Society of Heating, Refrigerating and Air-Conditioning Engineers. (2023). *BACnet Website*. [Online]. Available: http://www.bacnet.org
[3] Connectivity Standards Alliance. *ZigBee*. Accessed: Nov. 8, 2022. [Online]. Available: https://csa-iot.org/all-solutions/zigbee/
[4] KNX Association. *KNX ETS5 eCampus*. Accessed: Nov. 8, 2022. [Online]. Available: https://wbt6.knx.org/login/index.php?lang=en
[5] *What is ETS Professional?* Accessed: Nov. 8, 2022. [Online]. Available: https://www.knx.org/knx-en/for-professionals/software/ets-professional/
[6] *KNX System Specifications—XML Data Encoding*, KNX Assoc., Machelen, Belgium, 2004.
[7] *KNX Architecture*, KNX Assoc., Machelen, Belgium, 2009. [Online]. Available: https://www.knx.org
[8] *KNX Secure*. Accessed: Nov. 8, 2022. [Online]. Available: https://www.knx.org/knx-en/for-professionals/benefits/knx-secure/
[9] V. Graveto, T. Cruz, and P. Simões, "Security of building automation and control systems: Survey and future research directions," *Comput. Secur.*, vol. 112, Jan. 2022, Art. no. 102527. https://www.sciencedirect.com/science/article/pii/S0167404821003515
[10] S. Wendzel, J. Tonejc, J. Kaur, and A. Kobekova, *Cyber Security of Smart Buildings*. Hoboken, NJ, USA: Wiley, 2017, pp. 327–351, ch. 16. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119226079.ch16
[11] T. Macaulay and B. Singer, *Cybersecurity for Industrial Control Systems*. Boca Raton, FL, USA: CRC Press, 2011. [Online]. Available: https://www.taylorfrancis.com/books/mono/10.1201/b11352/cybersecurity-industrial-control-systems-tyson-macaulay-bryan-singer
[12] J. Molina. *Learn How to Control Every Room at a Luxury Hotel Remotly: The Dangers of Insecure Home*, pp. 1–13, 2014. [Online]. Available: https://pdfs.semanticscholar.org/047f/7d8626e1e2183c1aed2417b498330c7b033a.pdf
[13] K. Zetter, "Researchers hack building control system at Google Australia office," Wired Mag., May 8, 2013. Accessed: Nov. 8, 2022. [Online]. Available: https://www.wired.com/2013/05/googles-control-system-hacked/
[14] B. Krebs. (2014). *Target Hackers Broke in Via HVAC Company*. [Online]. Available: https://krebsonsecurity.com/2014/02/target-hackers-broke-in-via-hvac-company/
[15] I. Deng. (2018). *Tencent Engineer Slapped With Fine for Hacking Hotel Wi-fi in Singapore*. [Online]. Available: https://www.scmp.com/tech/enterprises/article/2165855/tencent-engineer-slapped-fine-hacking-hotel-wi-fi-singapore
[16] K. Higgins. (2021). Lights Out: Cyberattacks Shut Down Building Automation Systems. Darkreading. [Online]. Available: https://www.darkreading.com/attacks-breaches/lights-out-cyberattacks-shut-down-building-automation-systems
[17] R. Mitchell and I.-R. Chen, "A Survey of intrusion detection techniques for cyber-physical systems," *ACM Comput. Surveys*, vol. 46, no. 4, p. 55, 2013, doi: 10.1145/2542049.
[18] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7307098
[19] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1084804515002891
[20] B. B. Zarpelão, R. S. Miani, C. T. Kawakani, and S. C. de Alvarenga, "A survey of intrusion detection in Internet of Things," *J. Netw. Comput. Appl.*, vol. 84, pp. 25–37, Apr. 2017, doi: 10.1016/j.jnca.2017.02.009.
[21] J. Pedro and S. Silva, "Aplicação de interface com sistema Domótico EIB engenharia informática e de computadores," M.S. thesis, 2007.
[22] C. B. Jones, C. Carter, and Z. Thomas, "Intrusion detection & response using an unsupervised artificial neural network on a single board computer for building control resilience," in *Proc. Resilience Week (RWS)*, Aug. 2018, pp. 31–37.
[23] INTO-CPS Association. *INTO-CPS—Integrated Tool Chain for Model-based Design of Cyber-Physical Systems*. Accessed: Nov. 8, 2022. [Online]. Available: https://into-cps.org/
[24] J. C. Mace, C. Morisset, K. Pierce, C. Gamble, C. Maple, and J. Fitzgerald, "A multi-modelling based approach to assessing the security of smart buildings," in *Proc. Living Internet Things, Cybersecurity IoT*, 2018, pp. 1–10. [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/cp.2018.0031

[25] S. Ramapatruni, S. N. Narayanan, S. Mittal, A. Joshi, and K. Joshi, "Anomaly detection models for smart home security," in *Proc. IEEE IEEE 5th Intl Conf. Big Data Secur. Cloud (BigDataSecurity) Intl Conf. High Perform. Smart Comput., (HPSC) IEEE Intl Conf. Intell. Data Secur. (IDS)*, May 2019, pp. 19–24. [Online]. Available: https://ieeexplore.ieee.org/document/8819458

[26] J. S. Chavis, A. Buczak, A. Rubin, and L. A. Watkins, "Connected home automated security monitor (CHASM): Protecting IoT through application of machine learning," in *Proc. 10th Annu. Comput. Commun. Workshop Conf. (CCWC)*, 2020, pp. 0684–0690. [Online]. Available: https://ieeexplore.ieee.org/document/9031162

[27] The Tcpdump Group. *Tcpdump & Libpcap Offical Projects Website*. Accessed: Nov. 8, 2022. [Online]. Available: https://www.tcpdump.org/

[28] Cisco Systems Inc. *Snort—Network Intrusion Detection & Prevention System*. Accessed: Nov. 8, 2022. [Online]. Available: https://www.snort.org/

[29] Open Information Security Foundation. *Suricata IDS Project Home Page*. Accessed: Jan. 21, 2023. [Online]. Available: https://suricata.io/

[30] V. Paxson. *The Zeek Network Security Monitor*. Accessed: Nov. 8, 2022. [Online]. Available: https://zeek.org/

[31] D. A. Bhosale and V. M. Mane, "Comparative study and analysis of network intrusion detection tools," in *Proc. Int. Conf. Appl. Theor. Comput. Commun. Technol. (iCATccT)*, Oct. 2015, pp. 312–315. https://ieeexplore.ieee.org/document/7456901

[32] G. Dupont, J. D. Hartog, S. Etalle, and A. Lekidis, "A survey of network intrusion detection systems for controller area network," in *Proc. IEEE Int. Conf. Veh. Electron. Safety (ICVES)*, Sep. 2019, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8906465

[33] H. H. Al-Maksousy, M. C. Weigle, and C. Wang, "NIDS: Neural network based intrusion detection system," in *Proc. IEEE Int. Symp. Technol. Homeland Secur. (HST)*, Oct. 2018, pp. 14–19. [Online]. Available: https://ieeexplore.ieee.org/document/8574174

[34] E. Ficke, K. M. Schweitzer, R. M. Bateman, and S. Xu, "Characterizing the effectiveness of network-based intrusion detection systems," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Oct. 2018, pp. 76–81. [Online]. Available: https://ieeexplore.ieee.org/document/8599700

[35] A. Gouveia and M. Correia, "Feature set tuning in statistical learning network intrusion detection," in *Proc. IEEE 15th Int. Symp. Netw. Comput. Appl. (NCA)*, Oct. 2016, pp. 68–75. [Online]. Available: https://ieeexplore.ieee.org/document/7778595

[36] D. Robinson and C. Kim, "A cyber-defensive industrial control system with redundancy and intrusion detection," in *Proc. North Amer. Power Symp. (NAPS)*, Sep. 2017, pp. 1–6. [Online]. Available: https://ieeexplore.ieee.org/document/8107186

[37] V. Graveto, L. Rosa, T. Cruz, and P. Simões, "A stealth monitoring mechanism for cyber-physical systems," *Int. J. Crit. Infrastructure Protection*, vol. 24, pp. 126–143, Mar. 2019, doi: 10.1016/j.ijcip.2018.10.006.

[38] Raspberry PI Foundation. *Raspberry Pi*. [Online]. Available: https://www.raspberrypi.org

[39] Google Inc. *GoPacket Project Repository*. [Online]. Available: https://github.com/google/gopacket

[40] ZeroMQ project team, "ZeroMQ—An open-source universal messaging library," https://zeromq.org

[41] V. Graveto, P. Simões, and T. Cruz. (2022). *A Dataset Bundle for Building Automation and Control Systems Security Analysis*. [Online]. Available: https://dx.doi.org/10.21227/16a5-m134

[42] C. Green and M. Roesch. (2020). *SNORT Users Manual 2.9.16*. p. 269. [Online]. Available: https://www.snort.org/documents

[43] A. Patil, V. Kamuni, A. Sheikh, S. Wagh, and N. Singh, "A machine learning approach to distinguish faults and cyberattacks in smart buildings," in *Proc. 9th Int. Conf. Power Energy Syst. (ICPES)*, Dec. 2019, pp. 1–6.

[44] M. Iturbe, I. Garitano, U. Zurutuza, and R. Uribeetxeberria, "Towards large-scale, heterogeneous anomaly detection systems in industrial networks: A survey of current trends," *Secur. Commun. Netw.*, vol. 2017, Art. no. 9150965, Nov. 2017.

[45] D. Ding, Q.-L. Han, Y. Xiang, C. Ge, and X.-M. Zhang, "A survey on security control and attack detection for industrial cyber-physical systems," *Neurocomputing*, vol. 275, pp. 1674–1683, Jan. 2018.

[46] S. Nazir, S. Patel, and D. Patel, "Assessing and augmenting SCADA cyber security: A survey of techniques," *Comput. Secur.*, vol. 70, pp. 436–454, Sep. 2017.

[47] L. Rosa, T. Cruz, M. B. D. Freitas, P. Quitério, J. Henriques, F. Caldeira, E. Monteiro, and P. Simões, "Intrusion and anomaly detection for the next-generation of industrial automation and control systems," *Future Gener. Comput. Syst.*, vol. 119, pp. 50–67, Jun. 2021.

[48] S. D. D. Anton, S. Sinha, and H. Dieter Schotten, "Anomaly-based intrusion detection in industrial data with SVM and random forests," in *Proc. Int. Conf. Softw., Telecommun. Comput. Netw. (SoftCOM)*, Sep. 2019, pp. 1–6.

[49] B. Phillips, E. Gamess, and S. Krishnaprasad, "An evaluation of machine learning-based anomaly detection in a SCADA system using the modbus protocol," in *Proc. ACM Southeast Conf.*, Apr. 2020, pp. 188–196.

[50] A. Keliris, H. Salehghaffari, B. Cairl, P. Krishnamurthy, M. Maniatakos, and F. Khorrami, "Machine learning-based defense against process-aware attacks on industrial control systems," in *Proc. IEEE Int. Test Conf. (ITC)*, Nov. 2016, pp. 1–10.

[51] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.

[52] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1997, doi: 10.1007/BF00994018.

[53] D. E. Hilt and D. W. Seegrist, "Ridge, a computer program for calculating ridge regression estimates," Northeastern Forest Exp. Station, U.S. Dept. Agricult. Forest Service, Upper Darby, PA, USA, Res. Note NE-236, 1977, vol. 236. [Online]. Available: https://www.biodiversitylibrary.org/item/137258

VITOR GRAVETO received the B.Sc. and master's degrees in civil engineering and the B.Sc. degree in informatics engineering from the University of Coimbra, Coimbra, Portugal, in 1989, 1999, and 2013, respectively, where he is currently pursuing the Ph.D. degree with the Department of Informatics Engineering. His main research interests include building automation and control systems, building management systems, cyber-physical systems security, and cyber-security for critical infrastructures.

TIAGO CRUZ (Senior Member, IEEE) is an Associate Professor with the Department of Informatics Engineering, University of Coimbra. He is the author of more than 100 publications, including chapters in books, journal articles, and conference papers. His research interests include management systems for communications infrastructures and services, critical infrastructure security, broadband access network device and service management, the Internet of Things, software-defined networking, and network function virtualization (among others). He is a member of the IEEE Communications Society.

PAULO SIMÕES (Senior Member, IEEE) is an Associate Professor with the University of Coimbra. He has over 180 journals and conference publications in his research areas. He is regularly involved in several European- and industry-funded research projects, with both technical and management activities. His research interests include security, network management, and critical infrastructure protection.