Cybersecurity

RESEARCH

Open Access

Enabling data-driven anomaly detection by design in cyber-physical production systems

Rui Pinto[1*], Gil Gonçalves[1], Jerker Delsing[2] and Eduardo Tovar[3]

Abstract

Designing and developing distributed cyber-physical production systems (CPPS) is a time-consuming, complex, and error-prone process. These systems are typically heterogeneous, i.e., they consist of multiple components implemented with different languages and development tools. One of the main problems nowadays in CPPS implementation is enabling security mechanisms by design while reducing the complexity and increasing the system's maintainability. Adopting the IEC 61499 standard is an excellent approach to tackle these challenges by enabling the design, deployment, and management of CPPS in a model-based engineering methodology. We propose a method for CPPS design based on the IEC 61499 standard. The method allows designers to embed a bio-inspired anomaly-based host intrusion detection system (A-HIDS) in Edge devices. This A-HIDS is based on the incremental Dendritic Cell Algorithm (iDCA) and can analyze OPC UA network data exchanged between the Edge devices and detect attacks that target the CPPS' Edge layer. This study's findings have practical implications on the industrial security community by making novel contributions to the intrusion detection problem in CPPS considering immune-inspired solutions, and cost-effective security by design system implementation. According to the experimental data, the proposed solution can dramatically reduce design and code complexity while improving application maintainability and successfully detecting network attacks without negatively impacting the performance of the CPPS Edge devices.

Keywords: Artificial immune systems, Cyber-physical production systems, IEC 61499, Function blocks, Industrial Internet of Things, Anomaly-based host intrusion detection system, Model-based engineering

Introduction

The journey to the 4th Industrial Revolution, or also known as Industry 4.0 (I4.0), is marked by significant and high-impact technological advances in the industrial context. I4.0 (Lasi et al. 2014) corresponds to the new industrial paradigm where traditional methods of control are being transformed into new mechanisms that allow remote and distributed digital control. I4.0 makes networking and distributed computing an essential characteristic of modern automation systems. Besides, Edge devices or Internet of Things (IoT) enabled devices , i.e., lightweight controllers with limited memory and CPU resources, are becoming increasingly common within industrial settings since they enhance overall efficiency in task performance. The application of Industrial Internet of Things (IIoT) principles is possible with the development of cyber-physical production systems (CPPS), where the physical and virtual worlds are related and mutually dependent. On the one hand, actuation decisions in the virtual world would impact the physical world. On the other hand, physical process data will influence the virtual world for an updated decision-making process.

As industrial systems continue to grow in complexity, their attack surface also increases, making it more challenging to guarantee protection and correct operation at all times. CPPS are based on distributed industrial control systems (ICS) that use IIoT technologies, and there is usually an intrinsic integration with legacy components.

*Correspondence: rpinto@fe.up.pt
[1] Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal
Full list of author information is available at the end of the article

Thus, enabling security features in such systems is a complex problem and a hot research topic. Unfortunately, the Information and Communication Technology (ICT) security community has ignored the industrial domain in the past years (Loukas 2015). ICS design focuses mainly on efficiency and safety rather than on security. Like many other novel technologies that emerged in the past, developers think about the inherent security properties after the technology is developed and ready to be released. Security in CPPS is a significant research challenge to speed IIoT development towards a robust and trustworthy technology, resulting in further acceptance by the community side.

Naturally , efforts have been made to keep ICS secure, for instance, by using classical defense strategies and monitoring systems, such as firewalls, cryptography, access control, intrusion detection systems (IDS) (Sekar and Bowen 1999), among others. Regarding CPPS and IIoT security, more recent efforts focus on mapping solutions from existing related domains, such as supervisory control and data acquisition (SCADA) systems and wireless sensor and actuator networks (WSAN). Although detection and prevention techniques were studied to be applied to SCADA systems and WSAN, these solutions were usually not designed for CPPS and may not cover all the security requirements of IIoT. Even if they were suitable, no system could be full-prove to all attack vectors, especially considering zero-day attacks (Costin et al. 2014; Cui and Stolfo 2010).

In the literature, there are mentioned several successful manufacturing-related cyber-physical attacks driven by monetary and political interests (Loukas 2015). Well-known examples are: (1) the Maroochy Shire Water Services attack in 2000, which resulted in dumping around waterways in Queensland, Australia, large amounts of raw sewerage (Slay and Miller 2008; (2) the *Stuxnet* worm, which damaged an Iranian nuclear facility of Natanz back in 2010 (Baezner and Robin 2017); and (3) the German Steel Mill attack at the end of 2014, which blast a furnace by preventing it from correctly shutting down (Lee et al. 2014).

So, with the increasing adoption of intelligent and autonomous components in CPPS, the implementation of security features can be very complex. Security features should be included in the CPPS already in the design stage. Also, considering system components' heterogeneity and networking capabilities, CPPS development requires flexible automation that provides modifications with less engineering effort. This raises a need to support the design and verification of the CPPS by including security components as default. A solution can be achieved using model-based engineering (MBE) development approaches, which uses models to design software and perform component testing, accommodating the complexity and the dynamics of such systems.

The MBE approach enables component-oriented design, where components of the CPPS are modeled separately, clearly defining their roles, functionality, and purpose. These components can be combined and reused on-demand in different cost-effective implementations. This process should automate the demanding and error-prone tasks in the design phase so that no defects are introduced, and the models' final implementation behaves as intended. In manufacturing scenarios, the IEC 61499 standard, compliant with function blocks (FBs), is already being used as a general framework to develop distributed ICS by following the MBE approach.

This work aims to research and contribute in the creation of an IDS architecture for industrial Edge devices that adhere to the MBE approach for system design, such as the industrial standard IEC 61499. This IDS should employ a bio-inspired approach , making use of techniques in the artificial immune systems (AIS) family, for online intrusion detection . The main objective is to study the IEC 61499 standard and understand the feasibility of designing a FB-based immune IDS for overall CPPS by design. Thus, based on this, the research questions addressed are:

RQ T.1 *How effective is IEC 61499 in supporting the implementation of a bio-inspired intrusion detection solution for CPPS security by design, in terms of low complexity and high maintainability?*

RQ T.2 *What impact the iDCA algorithm has on the computational resources of Edge devices while enabling intrusion detection at host level?*

Given the previous study objectives and the research questions to be pursued, the main contributions presented in this work are:

- Application of a bio-inspired anomaly-based host IDS (A-HIDS), based on the incremental Dendritic Cell Algorithm (iDCA) technique, in a CPPS testbed. The A-HIDS detects network attacks that explore vulnerabilities in the OPC UA protocol;
- Distributed and real-time management, deployment, and monitoring of the iDCA, by following the FB oriented IEC 61499 standard (considering an MBE approach);
- Development and management of a CPPS testbed based on IEC 61499 FBs, making use of the Dynamic INtelligent Architecture for Software MOdular REconfiguration (DINASORE) framework;
- Validation of the iDCA suitability while deployed and executing in Edge devices. The iDCA FB Pipeline

validation is twofold. On the one hand, the solution's complexity and maintainability are validated. On the other hand, the iDCA is validated in terms of performance overhead, i.e., both detection performance of incoming/outgoing OPC UA network attacks and computational resource consumption.

The remainder of this paper is organized as follows. "Related work" section provides a review of the IEC 61499 standard, while providing a state of the art about MBE architectures and design approaches for embedded security capabilities in CPPS. "Proposed approach" section provides a detailed characterization of the proposed approach, an A-HIDS Pipeline for enabling the iDCA as CPPS network intrusion detection already in the design phase of the system. "Test and validation" section describes the experimental setup and methodology for testing and validating the proposed solution and discusses the results achieved. Finally, the "Conclusion" section concludes the paper, stating final remarks about the work presented and providing orientations for future work.

Related work

Until recently, considering the industrial context, automation systems development relied on the legacy IEC 61131-3 family of languages, such as ladder logic, structured text, and sequential function charts (Otto and Hellmann 2009). These languages are no longer suitable for the current development requirements of such complex systems since they rely on primitive abstractions for hardware and control flow. Within the I4.0 context, the IEC 61499 standard is emerging as a popular design standard (Vyatkin 2013, 2011), since it allows for MBE, where components and their behaviors can be defined and encapsulated into FBs (Commission et al. 2005).

The IEC 61499 standard, compliant with distributed FBs, can facilitate the development of distributed ICS. This standard provides a component-oriented design methodology for developing CPPS, supporting several object-oriented programming techniques and allowing reusable components that encode state and behavior. With IEC 61499, the encapsulation of software components is improved for increased re-usability, providing a vendor-independent format, and simplifying support for Machine-to-Machine (M2M) communication. Its distributed functionality and the inherent support for dynamic reconfiguration offer the required infrastructure for I4.0, enabling the design and management of large automation systems or IIoT applications (Jazdi 2014).

Considering FBs compliant with the IEC 61499 standard, they are functional software units that execute software code when triggered by events. After execution, they can generate and pass on new events. The software code typically runs algorithms that process input data to update internal variables and output results. These results are associated with output events when the algorithm finishes execution, which will trigger another FB for execution (Querol et al. 2016). Figure 1 represents a generic interface of an IEC 61499 FB.

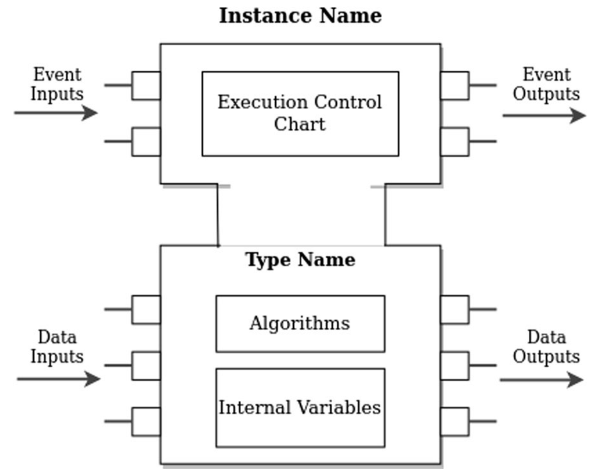

Fig. 1 Generic interface of an IEC 61499 function block

FBs are linked by events and data connections into FB Pipelines, which can be executed using a runtime environment (RTE) (Prenzel et al. 2020). A RTE enables the design and development of distributed architectures using the IEC 61499 by implementing the execution model defined by the standard. Also, along with the IEC 61499 standard, several integrated development environments (IDE) emerged for orchestration, mapping, and deployment of IEC 61499 enabled applications based on different RTEs.

Lindgren et al. (2014) addressed the outsets for real-time execution of FB-based designs onto IoT devices. On the other hand, Muthukumar et al. (2019) proposed an MBE approach for an IIoT architecture design, verification, and auto-code generation of control applications in process industries. In this case, the MBE approach was based on multiple system views and used to perform design and verification of an IIoT enabled control within the benchmark problem 'quadruple tank process'.

Focusing on security solutions deployed in manufacturing systems that follow the IEC 61499 standard, Sierla et al. (2014) proposed a security risk analysis methodology consisting of vulnerability and impact analysis. The analysis is applied to the communications network topology of an IEC 61499-based electric grid automation system, simulated under a co-simulation environment. In this case, the authors demonstrated the methodology with a case study of fault location, isolation and service restoration (FLISR) smart grid automation.

In work related to design-level support for security in CPPS, and with a focus on the communications between programmable logic controllers (PLC), Tanveer et al. (2018) propose embedded encryption in the communication between PLCs, using the IEC 61499 standard. The proposed solution is designated confidentiality layer for function blocks (CL4FB). It consists of a security layer that enables encrypted data communications using advanced encryption standard (AES) by encoding IEC 61499 compatible FBs. The applicability of the CL4FB was validated using an IEC 61499 based solution for protection and control functions in electric power distribution within a Smart Grid test case scenario. Feasibility analysis consisted of assessing the solution's impact on the overall system's performance, characterized mainly by the introduced latency. Results show that, although the CL4FB does introduce latency, most real-time constraints of the PLC can be met.

On a different work, Tanveer et al. (2019) investigate the case for providing security protection at the application level of PLCs. The approach consists of adding an IDS at the PLC application level, using IEC 61499 FBs. The IDS is added to the application at compilation time and is used for preventing typical PLC attacks on the application and device levels by analyzing network data between the network interface and the logic components of the system. This analysis is based on the Snort tool (Koziol 2003), and the authors assessed the approach in terms of Snort packets dropped over a time interval, running in an IEC 61499 environment and deployed in Wago PFC200 PLCs. Experiments show that the IDS-like functionality introduced by the FB can successfully log and prevent attacks at the application level, by providing active security protection from unknown (zero-day) attacks. By measuring the performance analysis of Snort when increasing the intensity of attacks, results show that the IDS drops more packets and loses essential data for analysis purposes. The PLC device breaks down at higher attack intensities, resulting in total denial of service (DoS).

Dowdeswell et al. (2020a, 2020b) also used the MBE approach for integrating the design and creation of fault identification and diagnostic capabilities. The proposed solution, designated fault diagnostic engine (FDE), was designed to recognize and diagnose faults in IEC 61499 FB Pipelines, not detect attacks nor protect communication channels like in previous work. The solution can monitor the system's behavior using appropriate fault detection strategies based on a diagnostic multi-agent system (MAS) that interact with the IEC 61499 FBs. The feasibility of the FDE was assessed when operating with several agent instances in a heating, ventilation, and air-conditioning (HVAC) test case scenario. Results show no actual performance issues in the HVAC.

More recently, Tanveer et al. (2021) proposed abstract design extensions to the IEC 61499 development standard, designated *Secure Links*, which implement both lightweight and traditional security mechanisms into ICS applications. *Secure Links* help include different communication security mechanisms into IEC 61499 applications in a consistent and reusable way, depending on specific security requirements. These requirements are defined in IEC 62443-4-2 and can be managed using the Traceability of Requirements using Splices (TORUS) framework. TORUS enables application development with security by design features, avoiding manual security mechanism coding. These security mechanisms can be transport layer security (TLS)-based and authenticated encryption with associated data (AEAD) security. Later, the developed applications can be deployed into target Edge devices, such as PLCs, since they are fully compliant with the IEC 61499 standard.

Experimental results show that *Secure Links* significantly reduce design and code complexity and improve application maintainability and requirements traceability. On the one hand, the latency of the encryption process and the key exchange are used as metrics to compare lightweight and TLS security mechanisms. On the other hand, the system design complexity ($C*$) is measured regarding structural ($S*$) and data ($DC*$) complexity , as well as the maintainability (MI) of the proposed solution. A higher MI value indicates a more maintainable program, while a higher $C*$ means more overall system complexity.

Table 1 presents a summarized overview, for comparison purposes, of the proposed solution main features (more detail in "Proposed approach" section) and similar approaches found in the literature. These features are related with: (1) *Methods* - the security methods implemented; (2) *Application Scenario* - the context of the application of the methods implemented; (3) *Data Analyzed* - type and origin of the data collected and analyzed by the security method; (4) *Tools* - technologies used to develop and deploy the proposed solution; (5) *Attack/ Fault Spectrum* - types of cyber-attacks or anomalies supported by the security method methodology; (6) *Assessment* - evaluation methodology and main results achieved while validating the proposed approach.

Table 1 Summary of IEC 61499-based security solutions

Work	Methods	Application scenario	Data analyzed	Tools	Attack/fault spectrum	Assessment
Sierla et al. (2014)	Security risk analysis	Electric grid FLISR (simulated)	Network (MODBUS and DNP3)	Matlab; simulink.	DoS; attempted break-ins; message spoofing.	N.A.
Tanveer et al. (2018)	Encryption of data communications	PLC in ICS lab testbed (raspberry Pi)	Network (TCP and UDP)	AES; KE.	N.A.	2–10 ms latency of encryption process
Tanveer et al. (2019)	Rule-based IDS	PLC in ICS Lab testbed (Wago PFC200 PLCs)	Network (TCP and UDP)	Snort; hping3.	DoS/DDoS; Masquerade; flooding.	Packets drop (N.A. results)
Dowdeswell et al. (2020a, 2020b)	MAS-based fault diagnostic	CPPS	Sensor (Temperature)	GORITE; SysML.	Sensor fault; software fault; actuator electrical and mechanical faults.	Performance and timeliness (N.A. results)
Tanveer et al. (2021)	Secure links (protection of communications)	PLC in ICS lab testbed (Wago PFC200 PLCs)	Network (TCP and UDP)	TLS; AEAD; KE; TORUS.	N.A.	2–5 ms latency of AEAD; 370–500 ms latency of KE; $MI = 75.68$; $S* = 23.8$; $DC* = 20.6$; $C* = 44.4$.
Proposed approach	A-HIDS	CPPS lab testbed (raspberry Pi)	Network (OPC UA)	iDCA; DINASORE; pyshark; river.	DoS; MITM; message spoofing.	30% CPU; 50% RAM; 300 KB/s network I/O; 0.1% packet drop; $MI = 100.12$; $S* = 16$; $DC* = 0.65$; $C* = 16.65$.

Proposed approach

A-HIDS is a software application that enables malicious activity monitoring at the host level, based on the collection and analysis of system data (Butun et al. 2014). The main goal is to infer intrusions/attacks in the system by identifying deviant behavior compared to a system baseline/normal behavior. An A-HIDS can achieve this while the attack is in progress or afterward. According to (Butun et al. 2014), and taking into account the nature of the processing involved in the behavioral model, an A-HIDS may be further characterized as statistical-based, knowledge-based, machine learning (ML)-based and *Soft Computing*-based. We are interested in studying *Soft Computing*-based approaches, such as AIS (Misra et al. 2014; Kim et al. 2007). This work focuses on developing and deploying an A-HIDS, based on the iDCA technique, in the Edge layer of a CPPS. This IDS analyses OPC UA network data to detect cyber attacks, which explore this communication protocol to be deployed in the system. Considering the MBE paradigm, specific FBs compliant with the IEC 61499 standard are created by using the DINASORE open-source technology (DIGI2-FEUP 2021; Pereira et al. 2020). These FBs can be reused for embedded intrusion detection in Edge devices in different application scenarios.

iDCA

Relevant AIS algorithms abstract biological immune processes related to Negative Selection, Immune Network Models, and Clonal Selection theories (Dasgupta et al. 2011). More recently, the Danger Theory (Aickelin et al. 2002) became the immune model that inspired techniques such as the Dendritic Cell Algorithm (DCA) (Greensmith 2007) and related variants. The DCA is a binary classification algorithm based on the natural functionality of dendritic cells (*dc*) present in the human immune system (HIS). Considering the network intrusion detection problem, Pinto et al. (2020) applied the DCA to detect network attacks in an OPC UA dataset. More recently, the authors proposed the iDCA, a modified version of the original DCA, suitable for real-time network intrusion detection, coping with the online nature of the anomaly detection problem (Pinto et al. 2021). Figure 2 illustrates the activity diagram of an iDCA-based network IDS.

Considering the theoretical foundations of the DCA and iDCA, these algorithms explore the functionalities

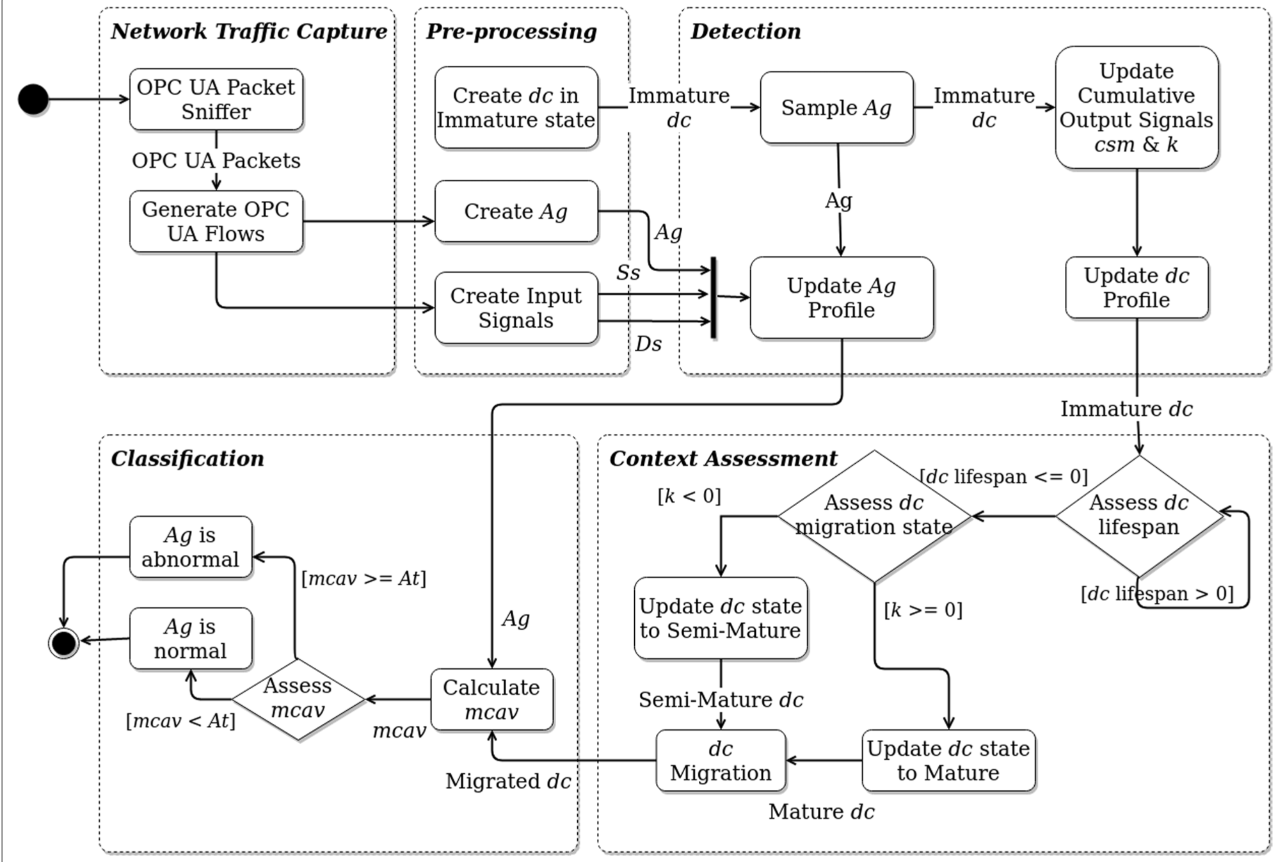

Fig. 2 Activity diagram of the iDCA-based network IDS

of *dc*, which are a type of antigen-presenting cell in the HIS. These cells have the job of catching, processing, and revealing antigens (*Ag*) to T-cells (to suppress or activate immune responses). *Ag* is an outside molecule representing a pathogen invading the body, such as bacteria or viruses. Its presence triggers typically an immune response, which starts with the scouting task of *dc*. The state of each *dc* in the system greatly depends on the signals sensed from the surrounding environment. These signals can be the pathogenic associated molecular patterns (*PAMP*), safe signals (*Ss*), and danger signals (*Ds*). Thus, if a *dc* receive more quantity of *Ss*, this indicates that the *Ag* collected by that *dc* was found in a normal context. On the other hand, if *Ds* and *PAMP* are produced in more quantity, this indicates abnormality and the need for immune activation. The reader can find more detail about the DCA functionality in previous work (Pinto et al. 2020, 2021).

Back to an engineering scenario, the iDCA is characterized by four main stages, namely *Pre-processing*, *Detection*, *Context Assessment* and *Classification*. The *Pre-processing* stage focus on mapping selected attributes/features to *Ss*, *Ds* and/or *PAMP* signals, while generating *Ag* or patterns to be classified, according to the available input data. The input data is available in a streaming fashion instead of a batch format, and it is collected by the *Network Traffic Capture* component. In this case, the data represents OPC UA network flows captured in real-time within a given Edge device and is made available in a streaming fashion for anomaly detection.

The *Detection* stage focus on the fusion of input signals (*Ss*, *Ds* and/or *PAMP*) with *Ag* collected. The detection is possible using a population of artificial *dc*, which will sample different *Ag* multiple times and combine those with the input signals. This process will enable the *dc* profile to be updated, by calculating the cumulative output signals. They are represented by the co-stimulatory signal (*csm*) and the context output value (*k*), calculated to assess the *dc* migration state.

Based on the process resulting in the *Detection* stage, the *Context Assessment* stage focus on the assessment of the *Ag* sampling context, by differentiating migrated *dc* into mature or semi-mature. If the *dc* is presented in a mature state, then the migration process will react to the *Ag* processed by that *dc*. Otherwise, if the *dc* is presented

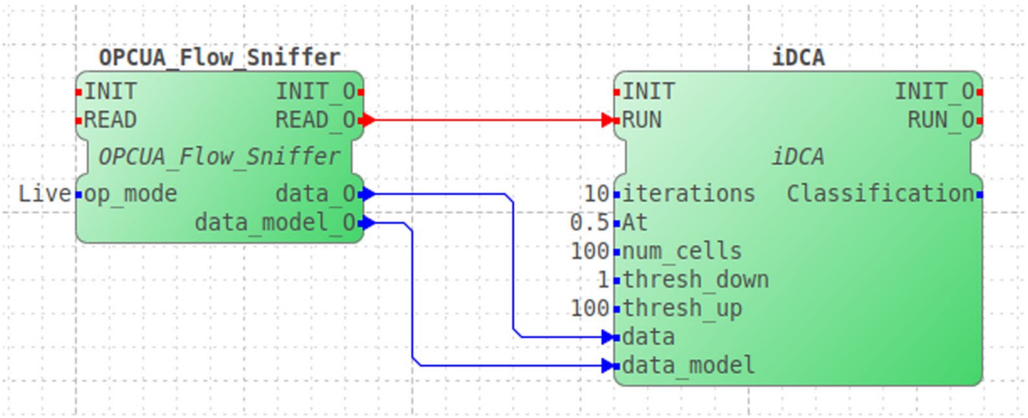

Fig. 3 A-HIDS FB pipeline

in a semi-mature state, then the migration process will tolerate the respective *Ag*.

Finally, the *Classification* stage uses the *dc* context resulting from the previous step to derive the nature of the response, by measuring the number of sampled *Ag* by *dc* that are fully matured. It is used the mature context antigen value (*mcav*) to assess the degree of an anomaly of a given *Ag* (with the help of the anomaly threshold (*At*) for binary classification).

Function block-based implementation

DINASORE adopts the 4DIAC-IDE (Foundation 2021b) as IDE for the FB Pipeline creation and distributed deployment in the Edge devices. DINASORE executes in each Edge device, based on the FORTE RTE (Foundation 2021a). The main advantage is the support of FB development based on Python coding language. Python enables the latest advances in ML to be integrated at the Edge level. Moreover, M2M communication at the Edge level is enabled by default thanks to the OPC UA communication protocol for 3rd-party integration.

In this section, detail is provided on the developing and deploying process of the iDCA technique in an actual Edge layer of an IIoT application, following the IEC 61499 standard. By implementing the iDCA in an FB compliant format with the IEC 61499 standard, the authors explore a new approach to enable data-driven anomaly detection by design, according to the MBE approach, for system design and implementation. The main outcome of this work is the A-HIDS FB Pipeline, which is represented in Fig. 3.

Based on this architecture, there are 2 main FBs, the *OPCUA_Flow_Sniffer* and the *iDCA*. The first FB sniffs the network for OPC UA packets. On the other hand, the second FB performs the data analysis and intrusion detection, based on the iDCA technique. Building FBs

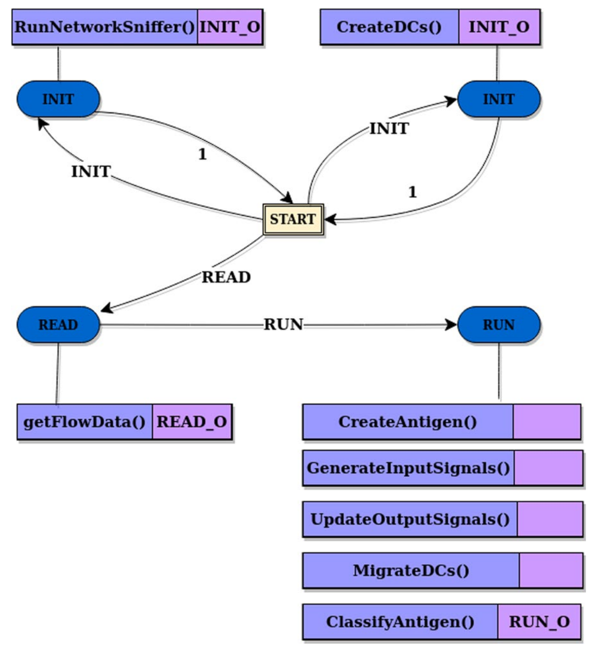

Fig. 4 ECC of the A-HIDS FB pipeline

using DINASORE requires Python version 3.6 or higher, along with several Python packages, such as `psutil`, `cryptography`, `opcua`, `numpy`, `scikit-learn` and `pandas`. Also, to operate specifically with *OPCUA_ Flow_Sniffer* and the *iDCA* FBs, there are extra Python dependencies, besides the ones already mentioned, namely `pyshark` (KimiNewt 2021) and `river` (Montiel et al. 2021). `pyshark` is a Python wrapper for the *Tshark* library, required for the network packet sniffing in the *OPCUA Flow Sniffer* FB. On the other hand, `river` is a Python library for ML application in streaming data or online ML. This is used in the *iDCA* FB, for the incremental analyses and real-time intrusion detection. Figure 4, represents the execution control chart (ECC) of the FB Pipeline proposed.

OPCUA flow sniffer

The *OPCUA_Flow_Sniffer* FB can operate in three different modes, which can be defined in the FB input `op_mode`:

- Live: The live packet capture mode corresponds to the Edge device's real-time OPC UA packet collection. All the OPC UA traffic is sniffed using the `pyshark` package from a live network interface.
- File: The file packet capture mode corresponds to the local OPC UA packet collection from a .pcap file, containing historical OPC UA traffic from an Edge device. So, all the OPC UA traffic is read from the file using, again, `pyshark`.
- Dataset: The dataset mode is a backdoor mode. The user can test the implementation using an existing dataset in a .csv file format, containing OPC UA traffic data and network features defined. In this specific mode, the FB was optimized to receive as input the *M2M using OPC UA* dataset (Pinto 2020). This dataset was already created by capturing OPC UA traffic in a distributed control system, as shown by Pinto et al. (2020).

The *OPCUA_Flow_Sniffer* FB has 2 different data outputs, namely the `data_O` and the `data_model_O`. Both these outputs correspond to data inputs in the *iDCA* FB, which are associated with the input event *RUN*, when it triggers the *iDCA* FB execution. The `data_O` corresponds to the OPC UA bidirectional flow to be analyzed by the iDCA technique. The *OPCUA_Flow_Sniffer* FB sends OPC UA flows as soon as they are available in a sequential order, to be analyzed by the iDCA. On the other hand, the `data_model_O` corresponds to the network features extracted from this respective setup, which will be used in the iDCA technique for the processing of the (*Ds* and *Ss* input signals).

iDCA

Regarding the *iDCA* FB, besides the dynamic data input from the *OPCUA_Flow_Sniffer* FB, such as the `data` and the `data_model`, there are other important data inputs, related to the iDCA initial parametrization. In this case they are:

- `iterations`: Number of iterations of the iDCA.
- *At* : Anomaly threshold used in the *Classification* phase of the iDCA.
- `num_cells`: Population of *dc*.
- `thresh_down`: Lower limit of the *dc* lifespan (expected time to live of the *dc*).
- `thresh_up`: Higher limit of the *dc* lifespan.

The *iDCA* can analyze incrementally over the data since the input data (network flows) to be analyzed become available gradually over time (stream data). Intrusion detection is possible using the iDCA technique. Finally, as data output, the *iDCA FB* makes available the classification result of each OPC UA flow (received in the `data` input) in the output `classification`. In this case, a `classification` of 0 means that the OPC UA flow analyzed is a normal network instance. On the other hand, if it is 1, the OPC UA flow is an abnormal network instance, which is evidence of a possible attack on the Edge device deployed in the network. Finally, suppose `classification` contains any other value other than 0 or 1. In that case, this means that the iDCA couldn't classify all OPC UA flows due to insufficient information to be used by the iDCA, such as no migrated *dc* or no *Ag* sampled by *dc*.

Test and validation

Vargas Martinez and Vogel-Heuser (2021) defend that, considering limited computational resources, the integration of an A-HIDS in embedded industrial devices can be pretty challenging, especially under strict operational requirements. So, the authors abstracted a set of requirements while considering both capabilities from ICT A-HIDS and considerations regarding industrial devices. Industrial devices should follow industry environments and security standards to assess the suitability of an IDS approach applied in such context. According to the authors, essential requirements that a general A-HIDS should possess are:

- *Configurability:* It should be possible to configure the A-HIDS according to the requirements of the system to be protected.
- *Configuration and knowledge security*: The A-HIDS configuration and collected information for analysis should be protected.
- *Resiliency:* The A-HIDS should be available all the time, and so itself should be protected against attacks.
- *Low-performance overhead:* While executing in the host being protected, the A-HIDS should not negatively impact the system's performance.
- *Low detection time:* Quick detection and response to intrusions to avoid extra damage in the system being attacked.

- *Interoperability with other IT:* Interaction with other security tools in the system for an overall security evaluation.
- *Centralized A-HIDS configuration and management*: This approach can be used to manage distributed systems, such as CPPS.
- *Log collection:* Enable log data for further processing and analysis.
- *System audit:* Collect host-related performance metrics.
- *Host network analysis*: Analyze network traffic in the device (input and output network packets).
- *Event-triggered or monitoring analysis:* Intrusion detection by monitoring host-related data.
- *Passive intrusion detection actions:* Logging and trigger alarms when facing important IDS events.

Considering the A-HIDS FB Pipeline, the proposed solution enables an immune-based A-HIDS to detect real-time network attacks in CPPS. Analysis of network data consists of incoming/outgoing OPC UA network packets in the Edge device. So, it is intended to validate the suitability of the proposed approach in resource-constrained Edge devices. Despite analyzing network packages, the solution can be classified as an A-HIDS since it is executed within Edge devices while analyzing only local network packets. This means that integrating the A-HIDS into the IIoT application can be challenging. Also, while running, certain Edge device operation constraints may not hold.

In this scenario, the assessment of the proposed solution may consider multiple metrics in different system properties, such as (1) *Detection rate* , (2) *Resource consumption* , and (3) *Performance overhead*. Regarding the *Detection rate*, since the A-HIDS analyzes network-related data, there is the advantage of carrying offline assessment in datasets. Typical system properties assessed are the detection accuracy, detection time, and network packets processing capabilities. In this case, the detection effectiveness of the iDCA as an anomaly detection algorithm for network data has been studied in previous work (Pinto et al. 2021). Hence, using the *Detection rate* system property for the evaluation of the A-HIDS is out of the scope of this contribution.

On the other hand, both *Resource consumption* and *Performance overhead* properties are suitable for assessment purposes. *Resource consumption* refers to the evaluation of resource utilization of the A-HIDS itself. In contrast, the *Performance overhead* refers to assessing the overhead added by the A-HIDS to the Edge device where it is deployed. This means that, for testing purposes, the A-HIDS Pipeline is deployed in a physical test case scenario to be evaluated. This test case scenario includes Edge devices in a M2M scenario (more detail in Experimental Setup). However, to further compare achieved results with related work in the literature, this approach presents several challenges. First, the physical test case isn't a benchmark, so it is impossible to repeatably assess and compare the proposed solution with other similar solutions. Even if one replicates exactly a physical test case similar to an existing one used previously in the literature, different Edge devices might not have the same baseline conditions. Consequently, the evaluation results of the same solution may be completely different when assessed in different host devices.

This means that direct comparison of results is not possible using this testing methodology. So, despite being considered for assessing the feasibility of deploying the proposed solution in an industrial Edge layer (more detail is provided in "Evaluation methodology and results" section), a more appropriate approach is used for result comparison with similar previous work. In this case, we consider the impact of a given IEC 61499 FB Pipeline on design complexity and maintainability of the system, i.e., the effort required to develop, deploy, and refine secure CPPS/ICS applications. According to the MBE and IEC 61499 paradigms, advantages for building and maintaining CPPS/ICS systems, design complexity, and maintainability of such solutions are critical measures to quantify the actual impact in the overall industrial system (Zhabelova and Vyatkin 2015). More detail can be found in "FB pipeline measures" section.

Next, it is described in detail the CPPS testbed as an experimental setup used to assess the proposed approach. Then, more detail is provided regarding the actual evaluation methodology, such as performance overhead and FB Pipeline metrics (complexity and maintainability). Finally, the results of the tests performed are described and compared with similar state-of-the-art work, along with a critical discussion of achieved results.

Experimental setup

We deployed the solution in a lab CPPS testbed to assess the feasibility of this approach. The testbed consists of a typical M2M scenario between multiple Edge devices in a CPPS, as represented in Fig. 5. Each Edge device consists of a Raspberry Pi 4 Model B, with 2GB RAM, connected to a cabled Local Area Network (LAN), created by a typical network switch. Each Edge device executes a DINASORE instance with all its respective dependencies. Finally, there is a malicious node (`pi@192.168.0.10`), which gets access to the network to deploy several attacks, such as DoS, Message Spoofing, and Man-in-the-middle (MITM).

Each Edge device implements an OPC UA server (default of DINASORE) to publish in OPC UA variables

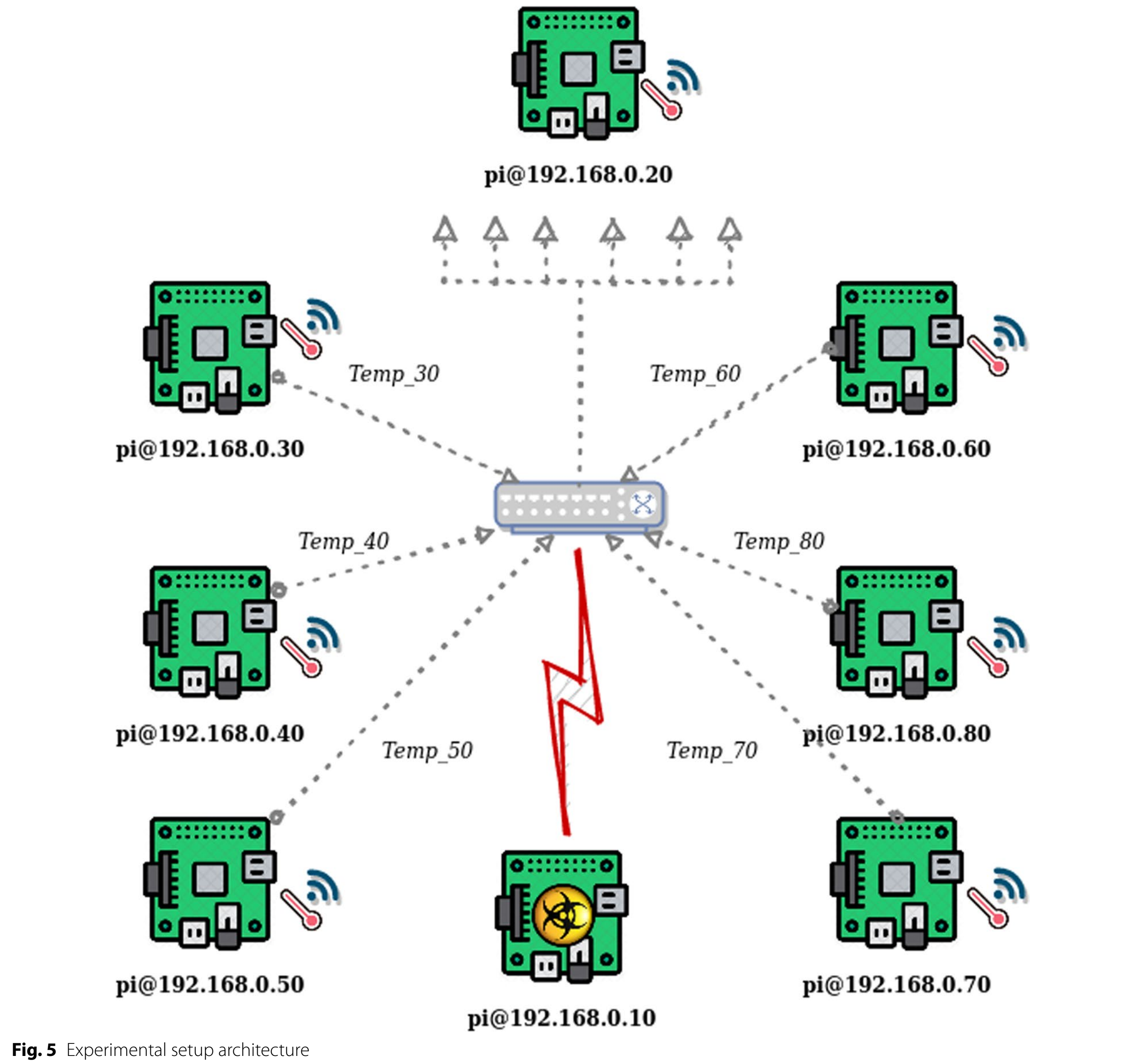

Fig. 5 Experimental setup architecture

updates regarding different sensor readings. In this case, each device is simulating dummy temperature sensor readings. Also, one device (`pi@192.168.0.20`) implements an OPC UA client and subscribes to all OPC UA variables from all other devices in the same network. For testing purposes, the A-HIDS is deployed and executed only in this device (`pi@192.168.0.20`) since it has more processing overhead and resource consumption. Most of the OPC UA traffic generated in the network can be found in this node. Figure 6 represents the FB Pipeline used to design and deploy the described functionalities among the Edge devices in the network.

Besides the already mentioned *OPCUA_Flow_Sniffer* and the *iDCA* FBs, responsible for the A-HIDS functionalities, there are other FBs to be considered in this experimental setup:

- `SENSOR_SIMULATOR`—This FB introduces the functionality of simulating sensor readings. There is one `SENSOR_SIMULATOR` for each Edge device, which contains 3 main data inputs: (1) `UA_NAME`—Name of the OPC UA variable that will contain the sensor reading updates; (2) `RANGE`—Range of possible values to be simulated, according to a Gaussian distribution; (3) `RATE`—Frequency of the sensor

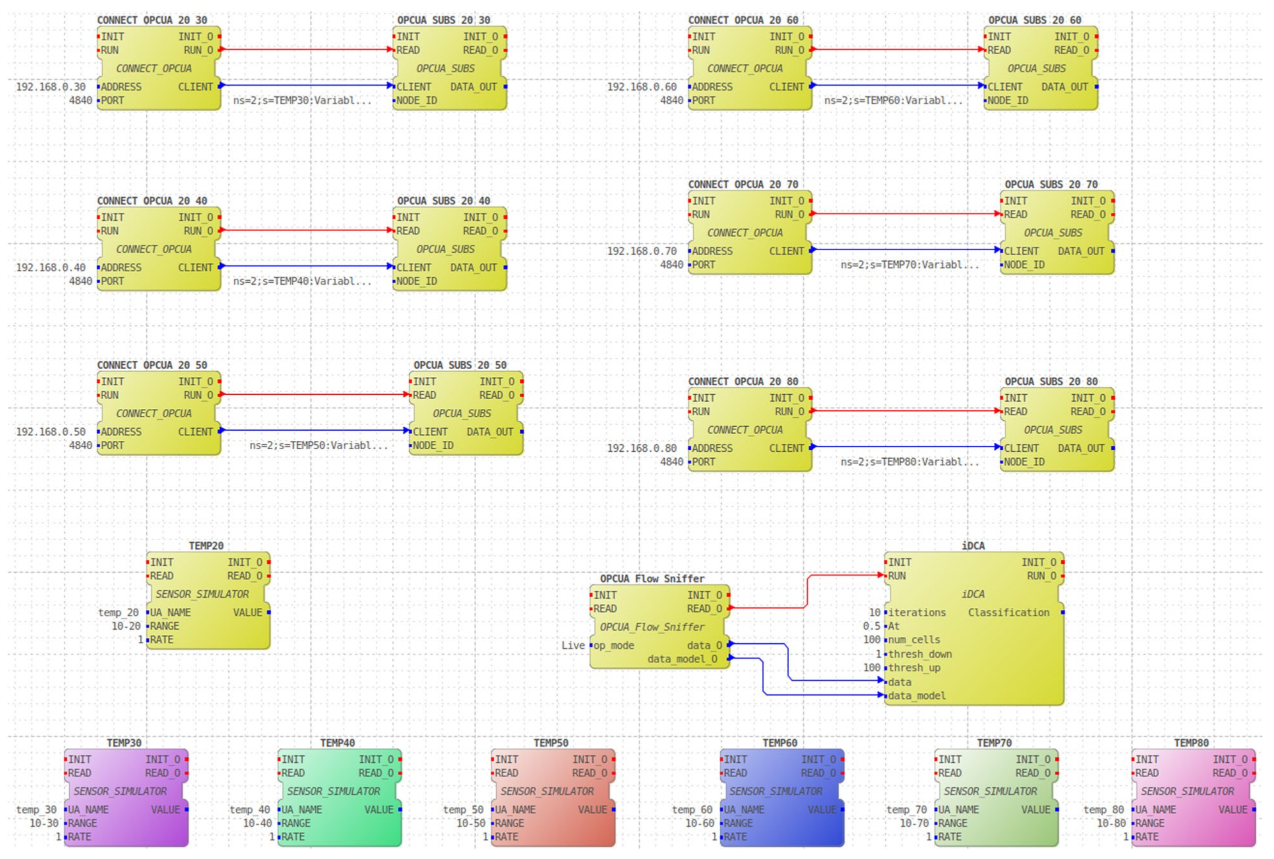

Fig. 6 Experimental setup—function block pipeline

reading update/generation; Also, there is a data output (`VALUE`), which makes available the sensor readings.

- `CONNECT_OPCUA`—This FB is responsible to create an OPC UA client. This client connects to an existing OPC UA server, so data input can be the IP address (`ADDRESS`) and port (`PORT`), to create and connect to the OPC UA server device. As data output, the `CONNECT_OPCUA` makes available the connection object (`CLIENT`), which will be passed as input to the `OPCUA_SUBS`.
- `OPCUA_SUBS`—This FB enables the subscription of one OPC UA variable. As data inputs, `Node_ID`, and the `CLIENT` are used to specify the path for the OPC UA variable to be subscribed, considering the client object. `DATA_OUT` makes available the subscribed sensor readings.

There are 7 Edge devices in this experimental testbed, all of them simulating temperature sensor readings. One of them is subscribing to all the sensor readings. The FBs mapped to the subscribing device (`pi@192.168.0.20`) are represented on the top of the figure in yellow, as represented in Fig. 6. In contrast, the FBs that execute in other devices are represented on the bottom of the figure with different colors for each device mapping. So, in each Edge device is deployed a `SENSOR_SIMULATOR` FB, where the FB instances are designated *TEMP20* to *TEMP80*, according to the respective device (`pi@192.168.0.20` to `pi@192.168.0.80`). In the subscribing device, there are deployed six pairs of `CONNECT_OPCUA` and `OPCUA_SUBS` FBs, each pair for the connection and subscribing to one specific publishing device. Finally, as mentioned before, the respective FBs for the A-HIDS functionalities (*OPCUA_Flow_Sniffer* and *iDCA*) are also deployed in the subscribing device.

Evaluation methodology and results

The evaluation of the proposed solution is twofold. On the one hand, it considers the performance overhead of the iDCA technique when deployed in the industrial Edge layer. On the other hand, critical FB Pipeline measures, such as design complexity and maintainability, are assessed.

Performance overhead

The evaluation of the proposed solution considers multiple system properties, according to the HIDS requirements mentioned previously. In this case, the assessment considers host and network-related metrics. On the one hand, host-related properties are used to evaluate the performance overhead of executing the A-HIDS in the Edge device by assessing CPU and RAM consumption. On the other hand, network-based properties assess the solution's suitability to process network data by evaluating network overhead and missing or non-classified network flows. As mentioned before, the iDCA classification performance is out of the scope of this study. It is intended in this work to test the feasibility of embedding intrusion detection capabilities in CPPS Edge devices by design and not the actual classification performance of the immune technique. One can find more information about the classification performance of the DCA and iDCA, as intrusion detection techniques in previous work (Pinto et al. 2021, 2020).

This evaluation methodology considers the execution of the A-HIDS (see "Proposed approach" section) in one Edge device (`pi@192.168.0.20`), tackling the experimental setup described in the "Experimental setup" section. Based on this methodology, the goal is to evaluate the performance overhead, such as CPU-intensive, memory-intensive, network-intensive, and the rate of missing classifications of processed network flows. The overhead performance metrics are collected using the *nmon* tool (Griffiths 2020), a well-known monitor tool for computer performance in Linux-based systems. On the other hand, the A-HIDS is self-aware of the number of non-classifications while executing in the Edge device under normal operation. So, the test scenarios consider three different monitoring moments:

(I) At system ramp-up and while no operation is being performed, which means little workload (processes, network connections, etc.);
(II) System in operation, by executing the underlying tasks, such as sensor simulation, publishing and subscribing sensor data, as mentioned in "Experimental setup" section, but without executing the A-HIDS;
(III) Executing the A-HIDS while the Edge device is under regular operation, including attack injection.

The monitoring process in all test scenarios is performed during 60min of system operation. Table 2 summarizes the monitoring results achieved, considering the evaluation methodology described previously.

By inspection of the results, one can extrapolate some conclusions on the suitability of the M2M scenario designed for this specific experimental setup—test scenario (II), and the application of the A-HIDS in this distributed CPPS—test scenario (III). First, in test scenario (II), it is evident that deploying the M2M scenario would increase CPU consumption and network traffic while reducing the free system's memory. However, the impact is residual, i.e., only 3.2% of CPU is allocated to the underlying tasks, leaving almost 1GB of free memory. Considering the 2GB of memory of the Raspberry Pi 4 Model B, this represents nearly 77% of free memory. Also, the overall network traffic (read and write) in all the interfaces (local, wireless and wired) is under 14 KB/s.

On the other side, considering test scenario (III), the impact is more evident when the A-HIDS is used to process the network traffic in the M2M scenario considered. First, the CPU consumption increases by 10× to 31% CPU allocation. Despite this significant CPU consumption increase, there is no apparent negative impact on this system, considering almost 70% CPU free to be allocated to other processes. Also, a small impact is verified in the RAM since the free memory is reduced to 51%. Secondly, the overall network traffic increases significantly, to almost 300 KB/s, mainly due to the increase in traffic in the local interface. This may be explained by the actual data exchange between FBs, since the network traffic on other external interfaces remains stable. Finally, the rate of missing packet classification is relatively low. Less than 0.1% of the overall packets weren't classified.

Table 2 Performance overhead evaluation methodology results

Test scenario	(I)	(II)	(III)
Avg. CPU (%)	0.6	3.2	31.2
Max. CPU (%)	9.1	8.1	34.9
Avg. free memory (MB)	1432.6	953.4	958.5
Min. free memory (MB)	1380.6	887.1	845.5
Avg. network I/O (KB/s)	0.19	13.75	297.6
Max. network I/O (KB/s)	24.7	34.1	425.7
Non classification (%)	–	–	0.07

FB pipeline measures

In this work, we perform validation and analysis of both the FB Pipeline of the A-HIDS and the CPPS testbed. For this, we used adapted measures from software, presented earlier by Zhabelova and Vyatkin (2015) and already used by Tanveer et al. (2021) when assessing an industrial security solution proposal. These measures are:

- High level design complexity metrics:
    - Structural Complexity ($S(i)$): Reflects coupling of the FB i to the rest of the system. The metric con-

siders the number of outgoing connections or output events (f_{out}) in a FB i, and is given by Eq. (1).

$$S(i) = f_{out}^2(i) \tag{1}$$

- Data Complexity ($DC(i)$): Shows efficiency of data utilization and information flow. The metric considers the ratio between the number of data inputs NI and outputs NO in a FB i with the number of output events f_{out}, and is given by Eq. (2).

$$DC(i) = \frac{NI + NO}{f_{out}^2(i) + 1} \tag{2}$$

- System Complexity ($C(i)$): As data and structural complexity increases, the overall system complexity grows and modularity decreases. This metric considers both the structural and data complexity in a unique measure, and is given by Eq. (3).

$$C(i) = S(i) + DC(i) \tag{3}$$

Note that $S*$, $DC*$, and $C*$ represent the aggregated values for the entire FB Pipeline, defined respectively as the sums of all $S(i)$, $DC(i)$, and $C(i)$ for each FB instance in the Pipeline.

- Maintainability index (MI): Represents the degree to which a FB Pipeline is open to change after being deployed in the Edge layer. One of the most popular metrics to measure maintainability takes into account the lines of code (LOC), Halstead's measures of program volume (V) and McCabe's complexity metric M_m (explained next), and is given by Eq. (4). All these metrics consist in the sum of both analysis for the FB internal ECC and the FB algorithms source code.

$$MI = 171 - 5.2ln(V) - 0.23M_m - 16.2ln(LOC) \tag{4}$$

- Halstead's Metrics (M_H): is an indicator of volume and entropy measure, by taking into account the number of operators and operands of both FB's ECC an respective source code. Halstead's Metrics are: program length, program vocabulary, estimated length, purity ratio, program volume, difficulty and program effort. The program volume V is needed for the MI calculus, and is given by Eq. (5), where N is the program length, i.e., total number of operators and operands, while n is the program vocabulary, i.e., total number of distinct operators and operands.

$$V = Nlog_2(n) \tag{5}$$

- McCabe's Metric (M_m): McCabe's cyclomatic complexity indicates the number of paths in the control flow graph, and it is a different way of representing control and data flow complexity using graph based metrics. It is given by Eq. (6).

$$M_m = \sum_{i=1}^{n} M_m(Alg_i) + M_m(ECC) \tag{6}$$

$M_m(Alg_i)$ represents the cyclomatic complexity of the FB i (algorithm source code), by counting the number of *if/else* statements and loops, such as *for* and *while*. On the other hand, $M_m(ECC)$ represents the cyclomatic complexity of the FB i (internal ECC), by counting both the number of edges (transitions in the ECC) and nodes (states in the ECC). They are given by Eq. (7).

$$\begin{aligned} M_m(Alg_i) &= N_{ifs} + N_{loops} + 1 \\ M_m(ECC) &= N_{edges} - N_{nodes} + 2 \end{aligned} \tag{7}$$

This evaluation methodology considers the deployment of the A-HIDS FB Pipeline (see the "Proposed approach" section) in the CPPS testbed described in the "Experimental setup" section. Based on this methodology, the main goal is to consider the design complexity and maintainability of the standalone A-HIDS FB Pipeline, and the FB Pipeline of the experimental setup (CPPS testbed) with and without the A-HIDS FBs, i.e., executing or not the iDCA. Also, it is pretended to compare the design complexity and maintainability metrics achieved in this study with similar work in the literature. In this case, the only work that targets IEC 61499-based security solutions and uses these metrics for assessment is presented by Tanveer et al. (2021). The authors propose several security solutions based on TLS and AEAD mechanisms while deploying those in two different case studies: an industrial mixer control system (IMCS) case study and a large baggage handling system (BHS) application. The test scenarios considered are:

(I) *Security Approaches Design*: Assess the structural, data and system complexity, and maintainability of three different security approaches proposed, namely: (a) A-HIDS (proposed approach in this study); (b) Average results of the *Secure Links* approach proposed by Tanveer et al. (2021), which is a set of security mechanisms; and (c) *Secure Links* best result mechanism proposed in the same work.

(II) *CPPS Tesbed Impact:* Assess the impact (added complexity and reduced maintainability) that the application of the A-HIDS has when deployed in the CPPS Testbed. This is done by comparing the assessment of the structural, data and system complexity, and maintainability of both: (a) CPPS Testbed standalone; and (b) CPPS Testbed while executing the A-HIDS.

(III) *IMCS Impact:* Assess the complexity and maintainability impact of the *Secure Links* application to the IMCS testbed. Comparison is done between: (a) IMCS; and (b) IMCS with *Secure Links*.

(IV) Same as the previous one, but this time the testbed considered is the BHS instead of the IMCS: (a) BHS; and (b) BHS with *Secure Links*.

Based on these test scenarios, Table 3 summarizes and compares the results achieved in this study and the work of Tanveer et al. (2021), considering the evaluation methodology described previously.

Considering the results of test scenario (I), the A-HIDS presents great results regarding data complexity (0.65). When compared with the average results of the *Secure Links* approach, it is clear that the A-HIDS is a better solution in terms of complexity and with very competitive results when compared with the best result of the *Secure Links* approach. Also, when considering the maintainability metric, the A-HIDS presents great results (100.12), surpassing both average and best results of *Secure Links*. Overall, despite not being the same type of security solution, the A-HIDS is a better maintainable FB application solution with low complexity compared with the state of art work.

When the A-HIDS is deployed in the CPPS Testbed (II), it is undeniable the impact that the IDS has on the testbed. For system complexity and maintainability, one can find a little more than 20% increase in complexity (22.16%) and a decrease in maintainability (23.57%). However, using the A-HIDS has a negligible effect on data complexity. When comparing to the test scenarios (III) and (IV), the increase of system complexity by 22.16%, when deploying the A-HIDS in the CPPS Testbed, is shorter in test scenarios (III) and (IV). This means that the negative impact in terms of complexity when deploying the proposed solution is better when compared to the deployment of the *Secure Links* in both IMCS and BHS. However, the impact on the system maintainability is worse (23.57% when *Secure Links* deployment is under 5%).

In summary, when considering the standalone security solutions, the A-HIDS is an overall better maintainable solution and with low complexity when compared with *Secure Links*. Regarding the impact of solution deployment in testbeds, the A-HIDS presents good complexity results but a poorer performance when considering system maintainability. This is mainly due to increased program volume, control and data flow complexity of the A-HIDS.

Discussion

Considering the requirements described at the beginning of the "Test and validation" section, we took most of them into consideration in the design, implementation, and evaluation of the proposed solution. Regarding the *Configurability* of the solution, the parametrization of the proposed A-HIDS is possible by updating the data-input fields of the *iDCA* FB. However, the data model, i.e., the network features considered, are derived internally in the *OPCUA_Flow_Sniffer*. This means that the user can't specify or add new features to be used without updating the FB source code. On the other hand, the A-HIDS *Configuration and knowledge security*, i.e., the data model,

Table 3 Evaluation methodology results using FB pipeline measures

Test scenarios		Complexity						Maintainability	
		S∗	%	*DC∗*	%	*C∗*	%	*MI*	%
(I)	A-HIDS	16	–	0.65	–	16.65	–	100.12	–
	Secure links (average)	23.8	–	20.6	–	44.4	–	75.68	–
	Secure links (best)	8	–	3.8	–	11.8	–	85.76	–
(II)	CPPS testbed	1444	22.16	0.04	0	1444.4	22.16	124.93	23.57
	CPPS testbed with A-HIDS	1764		0.04		1764.04		95.49	
(III)	IMCS	8	0	29.6	40.54	37.6	31.91	93.51	4.49
	IMCS with *secure links*	8		41.6		49.6		89.31	
(IV)	BHS	196	0	37	227.03	233	36.05	42.19	0.28
	BHS with *secure links*	196		121		317		42.07	

data-input fields, and the collected data for analysis, are not protected. This requirement is not considered in the development of the solution.

Considering the *Resiliency* of the solution, all attacks considered in this work, namely DoS, MITM, and Message Spoofing, exploit the OPC UA communication channel in a M2M scenario. Since the A-HIDS is executing at the Edge device level, it should be protected against OPC UA-related attacks. However, there may be other types of attacks that exploit different vulnerabilities to target the Edge device, specifically the A-HIDS. Those are not considered or tested in this work.

Regarding *Low-performance overhead* and *Low detection time*, according to results in Table 2, there is an expected increase in CPU consumption, RAM usage, and network traffic during the execution of the A-HIDS. However, no evident negative impact is noticeable since there is no evidence of dropped network packets, performance delays, or relevant missing packet classification. The A-HIDS doesn't jeopardize the performance of the Edge device's underlying tasks. Also, the A-HIDS implements the iDCA as a detection technique, which processes input data incrementally and provides online detection in typical stream data scenarios. This way, quick detection and response to intrusions are guaranteed.

The *Centralized A-HIDS configuration and management* can be done using the 4DIAC-IDE, which supports the IEC 61499 standard. By designing and deploying the A-HIDS using DINASORE, it is possible to monitor and manage the distributed system easily. According to results in Table 3, the A-HIDS FB Pipeline is an easily maintainable solution, which presents overall low complexity for configuration and management purposes. On the other hand, regarding *System audit*, despite analyzing network traffic (input and output OPC UA network traffic in the Edge device), host-related data are collected to assess the approach performance on the host device, such as CPU, RAM, and network traffic overhead. However, considering the *monitoring analysis* requirement, that host-related data collected for performance purposes is not being considered as input in the actual A-HIDS for detection purposes. This is out of scope in this work.

Regarding *Passive intrusion detection actions*: The log and alert of the A-HIDS are possible. Whenever an intrusion is detected, the A-HIDS logs the respective network flow as possible intrusion, and the IP address of the source machine is included in a timestamped blocklist. Also, regarding *Log collection*, besides being processed for real-time detection purposes, the collected network data are included in a log file for future offline analysis. In this case, a .csv file type is used to record the network data in a typical ML tabular format dataset by using the data model as features considered in the A-HIDS. This can be very handy since the dataset created can be easily used for validation purposes in other IDSs that use different ML techniques for detection. Finally, the *Interoperability with other security tools* in the system, for an overall security evaluation, is out of scope in this work.

Conclusion

IIoT is a critical enabling technology to transform how industrial automation systems are implemented and controlled. Resources such as open connectivity and new computing environments, such as Edge/Fog Computing, are reshaping how CPPS is designed, deployed, and managed. This brings a new level of complexity to these systems, which means that the traditional development approaches are becoming rapidly unfit (Thramboulidis 2005). All system components are developed independently and then integrated into the final system using conventional development methods. As an alternative, the MBE approach is a good approach for developing CPPS since it uses models to design software and perform component testing, accommodating the complexity and dynamics of CPPS.

With the emergence of the IIoT and the increasing interconnection of CPPS components with the digital world, cyber-attacks are increasingly common to affect real-world processes and devices. This means that CPPS security can't be treated as a classic ICT security problem. This work considers the security of CPPS Edge layer devices by introducing an A-HIDS that analyses OPC UA network data in typical M2M communication. Also, the challenge of providing security by design is addressed, i.e., enabling the intrusion detection capabilities during the CPPS design phase. This is achieved by exploring the IEC 61499 standard in an MBE approach, developing FBs specifically for introducing the A-HIDS functionalities in Edge devices. The FBs are created and deployed using the DINASORE technology.

By analyzing the work in the literature, it is possible to conclude that the proposed approach has many advantages. For starters, it presents a feasible IDS architecture for embedded Edge devices, which considers IDS features from the ICT domain and the industrial domain's specific properties. Secondly, the A-HIDS approach provides a protection scope that may not be covered by classic security solutions in typical information systems. Thirdly, this approach enables a complete anomaly detection approach by implementing complex analysis with bio-inspired techniques, coping with the system's dynamic behaviors over time. Fourth, this approach can act on real-time data by collecting, integrating, analyzing, and detecting attacks while the data are being generated. Data streams processing is a typical scenario in IIoT applications. Thus, the proposed solution can continuously process data streams from Edge devices to get insights without negatively impacting the overall system performance. This is a great advantage since it tackles essential requirements in IIoT applications, according to Vikash et al. (2020). To answer **RQ T.1** *How effective is IEC 61499 in supporting the implementation of a bio-inspired intrusion detection solution for CPPS security by design, in terms of low complexity and high maintainability?*, results show that the IEC 61499 standard is a good approach to support the implementation of security-by-design CPPS. The standard enables security by design, considering the low-complexity and high-maintainable results of different security solutions. When compared with state-of-the-art work (see in Table 3), the A-HIDS Pipeline achieves better maintainability and overall low complexity.

On the other hand, to answer **RQ T.2** *What impact the iDCA algorithm has on the computational resources of Edge devices while enabling intrusion detection at host level?*, there is no evidence of the negative effect of the iDCA when executing at host level in constraint Edge devices. Results show that the iDCA can detect intrusions in an online fashion with an insignificant rate of missing packets classification and a reduced overhead in CPPS consumption and RAM (see in Table 2). As for downsides, testing the approach in resource constraining Edge devices instead of proper industrial PCs, such as PLCs, may neglect important performance characteristics related to resource consumption and process overhead in industrial tasks-related. Also, the proposed solution is not considering the detection performance in a typical distributed ICS since it is only tested in a single Edge device. Finally, the proposed A-HIDS doesn't cope with necessary A-HIDS requirements, such as full configurability, knowledge security, resiliency to other non-OPC UA-driven formats of attack, interoperability with other ICT security tools, and don't consider host-related data for detection purposes.

Future work will consider the distributed nature of the CPPS to enhance detection performance while validating the approach in a real industrial scenario, where Edge devices may have different constraints related to the production process. This will include a truly distributed detection scenario. Also, in future work, host-related data should be considered along with the network data for enhanced detection, considering the close relationship between the physical and cyber worlds in a CPPS. Finally, the proposed solution may include in the future a more active reaction to a detected attack/intrusion, which may consist of a dynamic reconfiguration and deployment of the CPPS, following the IEC 61499 standard.

Abbreviations

AES: Advanced encryption standard; AEAD: Authenticated encryption with associated data; AIS: Artificial immune system; A-HIDS: Anomaly-based host intrusion detection system; BHS: Baggage handling system; CL4FB: Confidentiality layer for function blocks; CPPS: Cyber-physical production system; CPU: Central processing unit; DCA: Dendritic Cell Algorithm; DINASORE: Dynamic Intelligent Architecture for Software Modular Reconfiguration; DoS: Denial of service; ECC: Execution control chart; FB: Function block; FDE: Fault diagnostic engine; FLISR: Fault location, isolation and service restoration; HIS: Human immune system; HVAC: Heating, ventilation and air-conditioning; I4.0: Industry 4.0; ICS: Industrial control system; ICT: Information and Communications Technology; iDCA: Incremental dendritic cell algorithm; IDE: Integrated development environment; IDS: Intrusion detection system; IEC: International Electrotechnical Commission; IIoT: Industrial Internet of Things; IMCS: Industrial mixer control system; IoT: Internet of Things; IP: Internet protocol; LAN: Local area network; M2M: Machine-to-Machine; MBE: Model-based engineering; MAS: Multi-agent system; MITM: Man-in-the-middle; ML: Machine learning; OPC UA: OPC unified architecture; PLC: Programmable logic controller; RAM: Random access memory; RTE: Runtime environment; SCADA: Supervisory control and data acquisition; TLS: Transport layer security; TORUS: Traceability of Requirements using Splices; WSAN: Wireless sensor and actuator network.

Acknowledgements

This research was partially supported by the project "INDTECH 4.0—Novas Tecnologias para Fabricação Inteligente", which embodies a comprehensive response towards the development and implementation of new production systems, embedding advanced production technologies, that will equip the Portuguese manufacturing industry to meet the challenges and opportunities of the 4th industrial revolution. The authors would like to thank all partners for their support and discussions that contributed to these results.

Author information

Rui Pinto

received the M.Sc. degree in Electrical and Computer Engineering at the Faculty of Engineering of the University of Porto (FEUP) in 2013. He is currently pursuing the Ph.D. degree in the Doctoral Program in Informatics Engineering at the same school. From 2013 until 2020, he was a researcher at SYSTEC, and currently,he is a Faculty Research Assistant. also, since 2017, he is an Assistant Lecturer at FEUP. His research interest includes development of intrusion detection techniques in cyber-physical systems applied to industrial contexts. These include a Self-immune approach based on artificial immune systems principles and other bio-inspired techniques. He has an ACM and IEEE Student Membership.

Gil Gonçalves

received the B.S., M.Sc. and Ph.D. degrees in Electrical and Computer Engineering from the Faculty of Engineering of the University of Porto (FEUP), in 1993, 1996 and 2015, respectively. Assistant Professor at FEUP in the Department of Informatics Engineering, he is also the coordinator of the Digital and Intelligent Industry (DIGI2) R&D Lab and coordinator of the Smart Manufacturing research line at SYSTEC. His research interests include cyber-physical systems, IoT and Edge computing, specializing in systems and software engineering, control architectures and design of software for complex systems with adaptive capabilities. He co-authored several publications and supervised M.Sc. and Ph.D. dissertations in Electrical and Computer, and Informatics Engineering. He has been the principal researcher in several national and international research projects.

Jerker Delsing

received the M.Sc. in Engineering Physics in 1982 and the Ph.D. degree in Electrical Measurement in 1988 at the Lund Institute of Technology, Sweden. During 1985 - 1988 he worked part time at Alfa-Laval—SattControl (now ABB) with development of sensors and measurement technology. In 1994 he was promoted to associate professor in Heat and Power Engineering at Lund University. Early 1995 he was appointed full professor in Industrial Electronics at Luleå University of Technology, where he is currently the Scientific Head of EISLAB. His present research profile can be entitled IoT and SoS automation, with applications to automation in large and complex industry and society systems. He holds positions as presidium and board member of ARTEMIS and board member of ProcessIT.EU and ProcessIT Innovations. Within the EISLAB group, he has been coordinator of several large to very large EU projects in the field, e.g., socrades, IMC-AESOP, arrowhead, FAR-EDGE, productive4.0, and arrowhead tools.

Eduardo Tovar
received the B.S., M.Sc. and Ph.D. degrees in Electrical and Computer Engineering from the University of Porto, in 1990, 1995 and 1999, respectively. Currently he his Professor in the Computer Engineering Department at the School of Engineering (ISEP) of Polytechnic Institute of Porto. Also, he heads the CISTER Labs. His research interests are real-time distributed systems, multiprocessor systems, cyber-physical systems and industrial communication systems. He is currently the Vice-chair of ACM SIGBED (ACM Special Interest Group on Embedded Computing Systems) and is member of the Executive Committee of the IEEE Technical Committee on Real-Time Systems (TC-RTS). Notably he has been program Chair/Co-Chair for ECRTS 2005, IEEE RTCSA 2010, IEEE RTAS2013 or IEEE RTCSA 2016, all in the area of real-time computing systems.

Authors' contributions
All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
All of the data used in this study is publicly available. The dataset *M2M using OPC UA* used and analysed during the current study is available in GitHub (https://github.com/DIGI2-FEUP/OPCUADataset) and IEEE DataPort (https://ieee-dataport.org/open-access/m2m-using-opc-ua). Also, the software *DINASORE* used during the current study is available in GitHub (https://github.com/DIGI2-FEUP/dinasore).

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Department of Informatics Engineering, Faculty of Engineering, University of Porto, Porto, Portugal. [2]EISLAB, Luleå University of Technology, Luleå, Sweden. [3]Research Centre in Real-Time and Embedded Computing Systems, Polytechnic of Porto - School of Engineering, Porto, Portugal.

Received: 20 July 2021 Accepted: 25 January 2022
Published online: 01 May 2022

References
Aickelin U, Cayzer S, Qz B (2002) The danger theory and its application to artificial immune systems. In: Proceedings of the 1st international conference on artificial immune systems (ICARIS '02), pp 141–148
Baezner M, Robin P (2017) Stuxnet. Technical report, ETH Zurich
Butun I, Morgera SD, Sankar R (2014) A survey of intrusion detection systems in wireless sensor networks. IEEE Commun Surv Tutor 16(1):266–282. https://doi.org/10.1109/SURV.2013.050113.00191
Commission IE et al. (2005) International standard iec61499, function blocks, part 1-part 4. IEC http://www.iec.ch
Costin A, Zaddach J, Francillon A, Balzarotti D (2014) A large-scale analysis of the security of embedded firmwares. In: 23rd USENIX security symposium (USENIX Security 14), pp 95–110. USENIX Association, San Diego, CA. https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/costin
Cui A, Stolfo SJ (2010) A quantitative analysis of the insecurity of embedded network devices: Results of a wide-area scan. In: Proceedings of the 26th annual computer security applications conference. ACSAC '10, pp 97–106. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/1920261.1920276
Dasgupta D, Yu S, Nino F (2011) Recent advances in artificial immune systems: models and applications. Appl Soft Comput 11(2):1574–1587. https://doi.org/10.1016/j.asoc.2010.08.024
DIGI2-FEUP: DINASORE (2021). https://digi2-feup.github.io/dinasore/. Aaccessed Nov 2021
Dowdeswell B, Sinha R, MacDonell SG (2020a) Diagnosable-by-design model-driven development for IEC 61499 industrial cyber-physical systems. In: IECON 2020 the 46th annual conference of the IEEE industrial electronics society, pp 2183–2188. https://doi.org/10.1109/IECON43393.2020.9254620
Dowdeswell B, Sinha R, Jarvis D, Jarvis J, MacDonell SG (2020b) Employing agent beliefs during fault diagnosis for IEC 61499 industrial cyber-physical systems. In: IECON 2020 the 46th annual conference of the IEEE industrial electronics society, pp 2189–2194. https://doi.org/10.1109/IECON43393.2020.9254877
Foundation E (2021a) 4diac FORTE—IEC 61499 Runtime Environment. https://www.eclipse.org/4diac/en_rte.php. Accessed Nov 2021
Foundation E (2021b) Eclipse 4diac. https://www.eclipse.org/4diac/. Accessed Nov 2021
Greensmith J (2007) The dendritic cell algorithm. Ph.D. Thesis, Citeseer
Griffiths N (2020) nmon: a free tool to analyze AIX and Linux performance. https://developer.ibm.com/articles/au-nmon_analyser/. Accessed Nov 2021
Jazdi N (2014) Cyber physical systems in the context of industry 4.0. In: 2014 IEEE international conference on automation, quality and testing, robotics, pp 1–4. https://doi.org/10.1109/AQTR.2014.6857843
Kim J, Bentley PJ, Aickelin U, Greensmith J, Tedesco G, Twycross J (2007) Immune system approaches to intrusion detection—a review. Nat Comput 6(4):413–466. https://doi.org/10.1007/s11047-006-9026-4
KimiNewt: Python packet parser using wireshark's tshark (2021). https://github.com/KimiNewt/pyshark/. Accessed Nov 2021
Koziol J (2003) Intrusion detection with snort. Sams Publishing, Carmel
Lasi H, Fettke P, Kemper H-G, Feld T, Hoffmann M (2014) Industry 4.0. Bus Inf Syst Eng 6(4):239–242. https://doi.org/10.1007/s12599-014-0334-4
Lee RM, Assante MJ, Conway T (2014) German steel mill cyber attack. Ind Control Syst 30:62
Lindgren P, Lindner M, Lindner A, Eriksson J, Vyatkin V (2014) Real-time execution of function blocks for internet of things using the rtfm-kernel. In: Proceedings of the 2014 IEEE emerging technology and factory automation (ETFA), pp 1–6. https://doi.org/10.1109/ETFA.2014.7005232
Loukas G (2015) Cyber-physical attacks: a growing invisible threat. Butterworth-Heinemann, Oxford
Misra AK, Mishra KK, Yang H, Li T, Hu X, Wang F, Zou Y (2014) A survey of artificial immune system based intrusion detection. Sci World J 2014:156790. https://doi.org/10.1155/2014/156790
Montiel J, Halford M, Mastelini SM, Bolmier G, Sourty R, Vaysse R, Zouitine A, Gomes HM, Read J, Abdessalem T, Bifet A (2021) River: machine learning for streaming data in python. J Mach Learn Res 22(110):1–8
Muthukumar N, Srinivasan S, Ramkumar K, Pal D, Vain J, Ramaswamy S (2019) A model-based approach for design and verification of industrial Internet of Things. Fut Gen Comput Syst 95:354–363. https://doi.org/10.1016/j.future.2018.12.012
Otto A, Hellmann K (2009) Iec 61131: a general overview and emerging trends. IEEE Ind Electron Mag 3(4):27–31. https://doi.org/10.1109/MIE.2009.934793
Pereira EM, dos Reis JPC, Gonçalves G (2020) Dinasore: a dynamic intelligent reconfiguration tool for cyber-physical production systems. In: 1st Eclipse international conference on security, artificial intelligence and modeling for the next generation Internet of Things (SAM IoT), pp 63–71. http://ceur-ws.org/Vol-2739/#paper_9
Pinto R (2020) M2M using OPC UA. IEEE Dataport. https://doi.org/10.21227/ychv-6c68
Pinto R, Gonçalves G, Tovar E, Delsing J (2020) Attack detection in cyber-physical production systems using the deterministic dendritic cell algorithm. In: 2020 25th IEEE international conference on emerging technologies and factory automation (ETFA), vol 1, pp 1552–1559. https://doi.org/10.1109/ETFA46521.2020.9212021
Pinto R, Gonçalves G, Delsing J, Tovar E (2021) Incremental dendritic cell algorithm for intrusion detection in cyber-physical production systems. In: Arai K (ed) Intelligent computing. Springer, Cham, pp 664–680

Prenzel L, Zoitl A, Provost J (2020) IEC 61499 runtime environments: a state of the art comparison. In: Moreno-Díaz R, Pichler F, Quesada-Arencibia A (eds) Computer aided systems theory–EUROCAST 2019. Springer, Cham, pp 453–460
Querol E, Romero JA, Serrano J, Sanchis R (2016) Evaluation of closed loop control applications using different event management strategies under IEC 61499. In: 2016 Second international conference on event-based control, communication, and signal processing (EBCCSP), pp 1–8. https://doi.org/10.1109/EBCCSP.2016.7605263
Sekar R, Bowen T (1999) On preventing intrusions by process behavior monitoring. In: 1st Workshop on intrusion detection and network monitoring (ID 99). USENIX Association, Santa Clara, CA. https://www.usenix.org/conference/id-99/preventing-intrusions-process-behavior-monitoring
Sierla S, Hurkala M, Charitoudi K, Yang C-W, Vyatkin V (2014) Security risk analysis for smart grid automation. In: 2014 IEEE 23rd international symposium on industrial electronics (ISIE), pp 1737–1744 . https://doi.org/10.1109/ISIE.2014.6864877
Slay J, Miller M (2008) Lessons learned from the Maroochy water breach. In: Goetz E, Shenoi S (eds) Critical infrastructure protection. Springer, Boston, pp 73–82
Tanveer A, Sinha R, Kuo MMY (2021) Secure links: secure-by-design communications in IEC 61499 industrial control applications. IEEE Trans Ind Inform 17(6):3992–4002. https://doi.org/10.1109/TII.2020.3009133
Tanveer A, Sinha R, MacDonell SG, Leitao P, Vyatkin V (2019) Designing actively secure, highly available industrial automation applications. In: 2019 IEEE 17th international conference on industrial informatics (INDIN), vol 1, pp 374–379. https://doi.org/10.1109/INDIN41052.2019.8972262
Tanveer A, Sinha R, MacDonell SG (2018) On design-time security in IEC 61499 systems: conceptualisation, implementation, and feasibility. In: 2018 IEEE 16th international conference on industrial informatics (INDIN), pp 778–785. https://doi.org/10.1109/INDIN.2018.8472093
Thramboulidis K (2005) Model-integrated mechatronics—toward a new paradigm in the development of manufacturing systems. IEEE Trans Ind Inform 1(1):54–61. https://doi.org/10.1109/TII.2005.844427
Vargas Martinez C, Vogel-Heuser B (2021) A host intrusion detection system architecture for embedded industrial devices. J Frankl Inst 358(1):210–236. https://doi.org/10.1016/j.jfranklin.2019.03.037
Vikash Mishra L, Varma S (2020) Performance evaluation of real-time stream processing systems for internet of things applications. Fut Gen Comput Syst 113:207–217. https://doi.org/10.1016/j.future.2020.07.012
Vyatkin V (2011) IEC 61499 as enabler of distributed and intelligent automation: state-of-the-art review. IEEE Trans Ind Inform 7(4):768–781. https://doi.org/10.1109/TII.2011.2166785
Vyatkin V (2013) Software engineering in industrial automation: state-of-the-art review. IEEE Trans Ind Inform 9(3):1234–1249. https://doi.org/10.1109/TII.2013.2258165
Zhabelova G, Vyatkin V (2015) Towards software metrics for evaluating quality of iec 61499 automation software. In: 2015 IEEE 20th conference on emerging technologies factory automation (ETFA), pp 1–8. https://doi.org/10.1109/ETFA.2015.7301502

Publisher's Note