Behavior-based Intrusion Detection Approach deployed on a Naval Testbed

1st Estelle Hotellier
Naval Cyber Laboratory, Naval Group
Ollioules, France
Univ. Grenoble Alpes, Inria, Grenoble INP
Grenoble, France
estelle.hotellier@inria.fr

2nd Nahi Boukhobza
Naval Cyber Laboratory, Naval Group
Ollioules, France
nahi.boukhobza@student-cs.fr

3rd Franck Sicard
Naval Cyber Laboratory, Naval Group
Ollioules, France
franck.sicard@naval-group.com

4th Julien Francq
Naval Cyber Laboratory, Naval Group
Ollioules, France
julien.francq@naval-group.com

5th Stéphane Mocanu
Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG
Grenoble, France
stephane.mocanu@inria.fr

2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA) | 979-8-3503-6123-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/ETFA61755.2024.10710831

***Abstract*—This paper presents an application of an intrusion detection approach onto a naval physical testbed. The deployed approach is tailored for complex Industrial Control Systems (ICSs). Such systems play a critical role in managing complex industrial processes and ensuring their security against cyber threats is a major concern. Our work concerns Process-Aware Attacks (PAAs) which are sophisticated attacks aiming at disrupting ICS physical processes. The methodology instantiates a specification-based and process-aware Network Intrusion Detection System (NIDS). The specifications are systematically extracted from international and industry standards. In order to be monitored, such specifications are translated into security requirements which are verified during the execution of the system. Our IDS relies on network traffic capture on fieldbuses as well as Ethernet networks. In addition to our previous work, deploying our approach on a realistic naval testbed allows us to demonstrate its extensibility to different environments. Furthermore, the evaluation of our approach shows both its good detection capabilities and scalability.**

***Index Terms*—Behavioral, Intrusion Detection System (IDS), Industrial Control System (ICS), Naval Testbed**

I. Introduction

Industrial Control Systems (ICSs) consist of combinations of control components (e.g., electrical, mechanical, hydraulic, pneumatic) acting together to achieve an industrial objective [1]. They have to preserve the integrity of the physical process while also achieving a control objective like tracking a trajectory or reaching a set point. ICSs serve a wide range of activity sectors like manufacturing, energy, or transportation systems. As ICSs concern highly critical processes, any industrial error (intentionally) induced into the industrial process can lead to potentially disastrous consequences. Some famous events like Stuxnet in 2010 [2] or Industroyer/CrashOverride [3], [4] in 2016 have shown that cyber-attacks may, effectively, damage the underlying physical process. As the number of attacks targeting industrial systems is continuously rising [5], the study of their cybersecurity became an important topic in the last decade.

A. ICS Architecture

The typical architecture of an industrial information system was synthesized for the first time around the 80s with a Reference Model for Computer Integrated Manufacturing (CIM) [6] commonly known as Purdue Model.

Purdue Model is hierarchical and shows distributed control functions. It is structured in 6 levels: (i) Level 5: Corporate Strategy (strategy, direction), (ii) Level 4: Operational Management (planning), (iii) Level 3: Process Management (scheduling, inter-unit management), (iv) Level 2: Supervisory Control (unit management), (v) Level 1: Basic Control and (vi) Level 0: Operative Part. The reference architecture for enterprise is illustrated in Figure 1.

While Purdue Model describes the full enterprise architecture, we mainly focus on the ICS in the present work which represents Levels 0 to 2 of Purdue Model (see Figure 2). Communication at the ICS levels is submitted to real-time constraints and uses specific technologies. The involved components are designated as OT (Operational Technology) in contrast with classical IT (Information Technology) systems.

The elementary building block of the ICS is the *local loop*, which is an autonomous sub-process together with its local controller. This local controller is directly interacting with the physical process via sensors and actuators. Typically, sensors and actuators are directly wired to the controller although some networked solutions may exist. The controller periodically acquires data from sensors and applies computed controls on actuators.

B. ICS Security Controls

ICSs were not designed to be directly exposed on Internet and present a large attack surface that can be targeted by

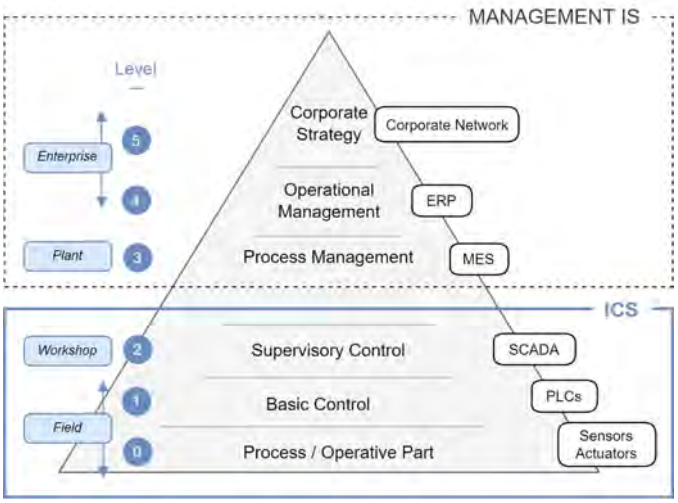

Fig. 1. Illustration of Purdue Model

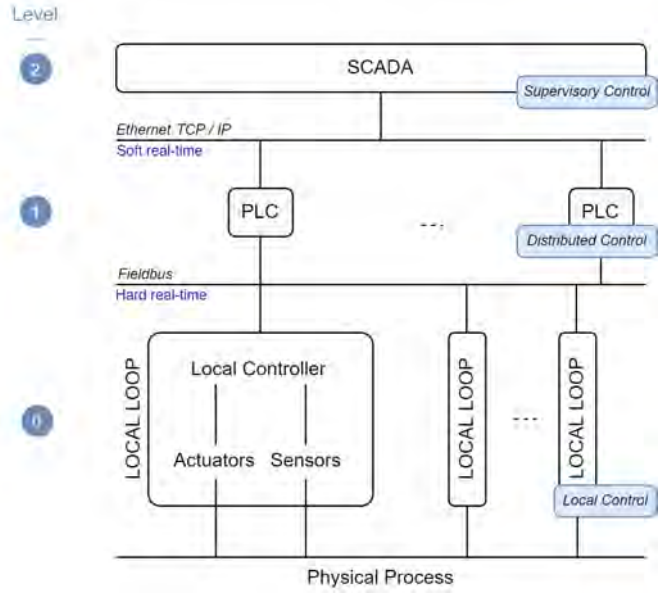

Fig. 2. Typical architecture of an ICS

a wide diversity of attacks. Due to the response time constraints needed by control functions and scarce computing resources, classical host security controls like anti-viruses or firewalls are difficult to deploy on ICSs devices. Therefore, network security controls are, generally, more adequate to ICSs. Among various network security controls, this paper focuses on Network Intrusion Detection Systems (NIDSs). The most common classification of IDSs identifies two categories: signature or misuse-based IDSs and behavioral or anomaly-based IDSs. While signature-based IDSs are the most common in on-the-shelf products, they may detect only known attacks and usually checks violations of syntax or semantics of the communication protocols. On the other hand, behavior-based IDSs will look for deviation from a "normal" behavior of the system, therefore they are able to detect new attacks but, in exchange, are more difficult to set up as "normal behavior" may be difficult to describe and produce an important number of false positives if the description is not complete.

In this paper, we study the detection of attacks which specifically target the controlled physical process. They are sometimes called *Process-Aware Attacks* (PAAs) [7], [8] and will usually carry the threat in the application payload of the frame without violating the specifications of the protocols. Therefore, we adopt a behavior-based detection approach with normal behavior extracted from system specifications (specification-based IDS).

This paper can be seen as a continuation of our previous work described in [9]. Security properties, which are cybersecurity related requirements, were obtained with the systematic analysis of international and industry standards specifications concerning local safety, global safety and networks of the industrial process. These security properties were translated into security patterns in order to be runtime monitored by our NIDS. We experimentally showed on a real ICS that our distributed NIDS was able to detect a broad range of attacks, and that our approach was scalable since the detection response time of our NIDS can be modeled as a linear function of the number of monitored security patterns. Our new work confirms the benefits of the approach described in [9] in terms of intrusion detection performance, low false positive rate, and scalability. Moreover, it validates our methodology on another real ICS platform, giving a high level of confidence on its extensibility throughout different industrial environments and platforms.

The rest of the paper is organized as follows. In Section II, we describe our approach: threat model, normal behavior extraction and detection technique. Section III presents our naval experimental testbed followed by the description of the use case and the experimental validation of our IDS in Section IV. In Section V, we evaluate our experimental results, and in Section VI, we describe the related work on intrusion detection approaches for ICSs and maritime testbeds. We then conclude with a review of our contributions and a short description of our future work.

II. Attack Detection Approach

A. Threat Model

As in [9], we consider the whole ICS as a potential target. Attackers can target any components from Level 0 to 2 (see Figure 2) or any network link between components. In our study, we consider that the threat has already gained access to the system and has control over one or several high-level controllers. Whatever the target is, we aim at detecting effects of an attack on the system i.e. we look for the system's incorrect behaviors. Therefore, we focus on PAAs which are attacks that disrupt the physical process and induce incorrect behaviors of the system. PAAs do not violate the syntax of the communication protocol. Instead, these attacks concern the manipulation of data related to the physical process of the system in order to disrupt it in a stealthy way. Some

examples of PAAs are: commands using legitimate orders sent out of their context (e.g., stopping a movement before a target is reached), forcing out-of-range values for process variables (e.g., sending a speed setting point out of safe range), inverting valid commands (e.g., inverting the order of products in a tank filling), targeting control logic (e.g., forcing transitions or changing internal states of controllers), etc.

We define the class of attacks with respect to the MITRE ATT&CK for ICS framework [10]. This latter provides the relevant tactics for our studies: "Execution", "Collection", "Command & Control", "Inhibit Response Function", "Impair Process Control" and "Impact". The other tactics of the MITRE Matrix are not relevant for our work since they are mostly concerning initial access and deployment of the threat. Our assumption is that the threat is already inside the system. However, it is important to note that relying on the MITRE framework can present some limitations. A technique coverage is rarely complete as a technique can be broken down into several expressions. Ensuring a technique total coverage assumes knowing the exact number of its expressions which is an information not provided by the framework.

B. Overview of our Approach

Our intrusion detection approach is fully detailed in [9]. It has the following characteristics:

- It is a specification–based IDS. Specifications are extracted from international and industry standards in order to obtain a set of security properties. The assumption we rely on is the fact that ICSs' devices and networks comply with standards which ensures a specific level of safety and expected behavior.
- The set of security properties are translated into temporal specification patterns which are generalized descriptions of the permissible state or event sequences in a finite-state system. We use qualitative and quantitative temporal specification patterns in order to describe the variety of temporal behaviors of hybrid ICSs. The temporal specification patterns are generalized sets of temporal patterns that facilitate system's specifications. In particular, we use Dwyer patterns [11], Konrad patterns [12] and Maler [13] patterns.
- A methodology from Runtime Verification (RV) [14] is performed, which allows to keep the performance of the detection task close to real-time. The core object of RV is a *monitor* which is a mathematical object that uses both observations from the monitored ICS and the set of previously obtained security properties. In order to be evaluated, the latter are expressed onto adequate Temporal Logic (TL) formalisms. TL is the most used family of specification language for RV. From TL formulas, monitors are automatically generated and evaluated against the monitored system's execution.
- From the practical side, we deploy a NIDS and network traffic capture is performed at fieldbus level, as well as TCP/IP level. The detection process is carried out by monitors that are activated or deactivated based on operational contexts. The module responsible for this task is referred to as the *Scope Recognizer*.

III. Naval Testbed

The authors in [15] give the following definition for a ICS *testbed*: it is a "testing infrastructure, consisting of a scaled-down version of a real ICS, created *ad hoc* to reproduce real-world systems but in a controlled environment". Using a *physical testbed* allows to conduct experiments under realistic conditions since real hardware and software are used. The naval physical testbed on which we deployed our approach represents a surface warship. A complete description of this testbed is provided in [16]. Figure 3 shows the parts constituting the testbed: (i) HMIs and physical processes and (ii) industrial control devices.

This setup reflects typical warship equipment brands (Siemens and Schneider Electric) and general architecture, though it differs in complexity and quantity of equipment compared to real warships.

The architecture of this testbed encompasses four main functions of the ship: Direction, Energy, Artillery and Propulsion. We call *subsystem loop* the system comprising the local network of a specific maritime function. Four subsystem loops are implemented:

- Direction subsystem: controlling the direction of the ship via two rudders.
- Energy subsystem: controlling fuel supply by filling and emptying the fuel tank.
- Artillery subsystem: controlling a 76mm gun turret model.
- Propulsion subsystem: controlling the propulsion of the ship via propellers.

Each subsystem loop is driven by a PLC. Each subsystem loop is controlled by authorized human operators, either manually on the equipment, locally from a local HMI or remotely from the Supervisory Control HMI. Communication between low-level components relies on fieldbus networks using Modbus RTU and Profibus, whereas higher level components implement Modbus TCP communication protocol.

IV. Experimentation

A. Use case

The part of the system we consider in this paper is the Direction subsystem of the warship. It is composed of two servo drives (SD328) each controlling a rudder via a stepper motor (BRS368). Each servo drive communicates with a PLC (M340-20) via a Modbus RTU fieldbus. Additionally, an HMI allows to control the subsystem. The architecture of the Direction subsystem is represented in Figure 4. As represented in the Figure, network traffic capture is achieved at fieldbus level as well as between the PLC and the HMI.

B. Security Properties

This section details the security properties synthesis. Most of the security properties are extracted from standards directly or from the implementation of the standards. For instance,

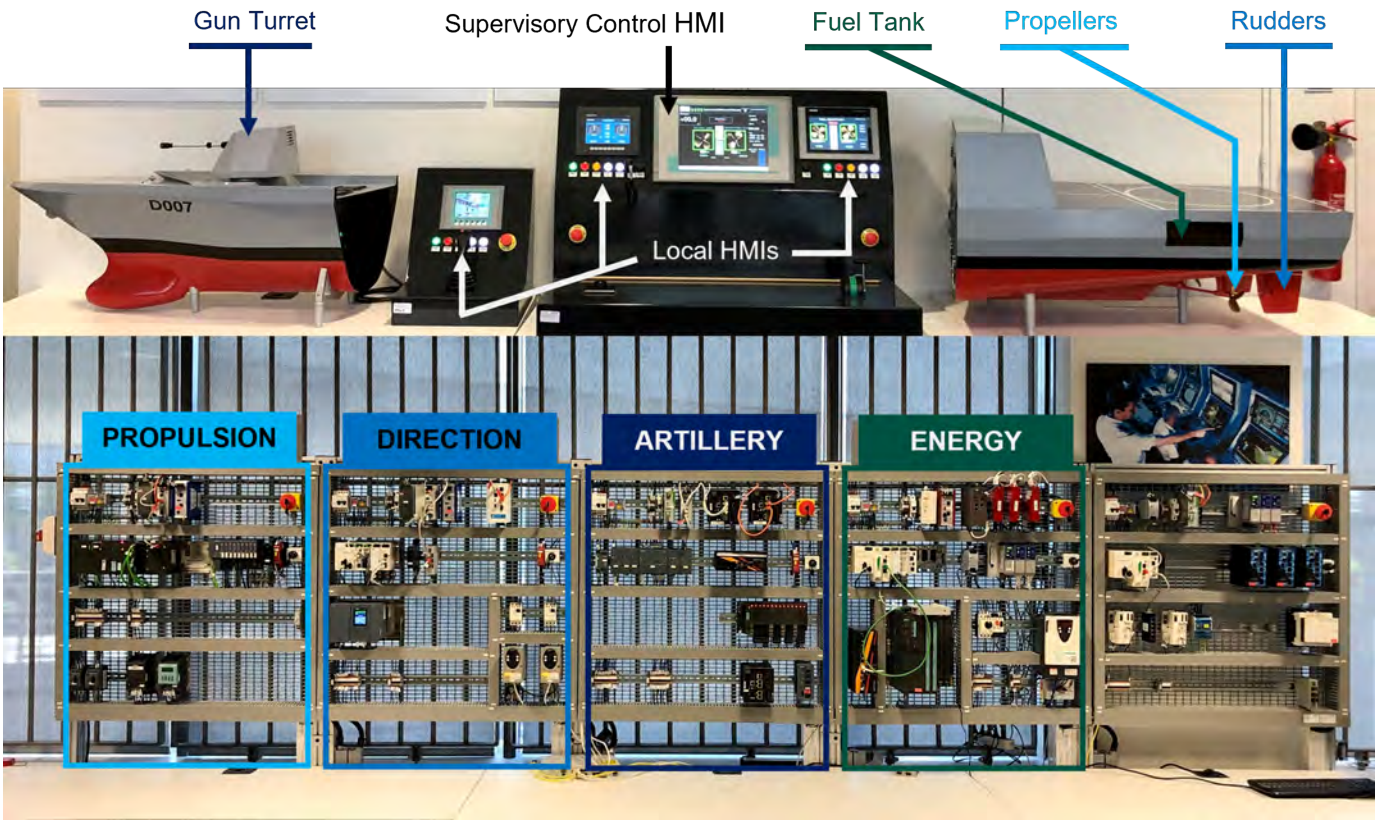

Fig. 3. Overview of the naval testbed – HMIs and Physical processes (top) and industrial control devices (bottom)

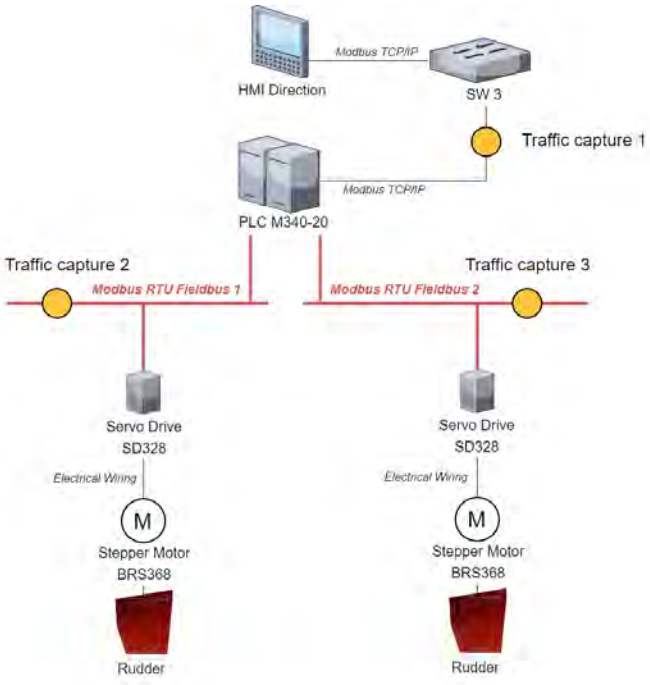

Fig. 4. Architecture of Direction subsystem and traffic capture points

manufacturer documentation provides details for specific device configurations conforming with standards. If we consider a servo drive, the IEC 61800 standards [17] specify its functioning states and operating modes as well as the corresponding mapping to communication protocols. The manufacturer documentation specifies the allocation of standard parameters and variables to memory addresses. For example, in Figure 5, the Finite State Machine (FSM) of a standard servo drive is represented. This FSM specifies the allowed transitions and events for each state. In our use case, there are two servo drives that comply with IEC 61800 drive state machine standards [17], thus the allowed transitions and events can be translated as security properties. In this use case, the servo drives are limited to two functioning modes only which provide new security properties concerning the restrictions of modes in this specific use case. The two functioning modes used in this use case are: "Homing" which provides a reference position to the system, and "Profile Position" which consists in an absolute or relative positioning.

Additionally, some security properties concern the way the PLC is programmed. PLC programs of our use case contain motion control tasks such as rotating an axis in certain directions. Especially, Function Blocks (FBs) from the library PLCopen Motion Control (MC) are used in the PLC programs of our use case. Such FBs comply with the specifications of PLCopen standards [18] from which we are then able to extract

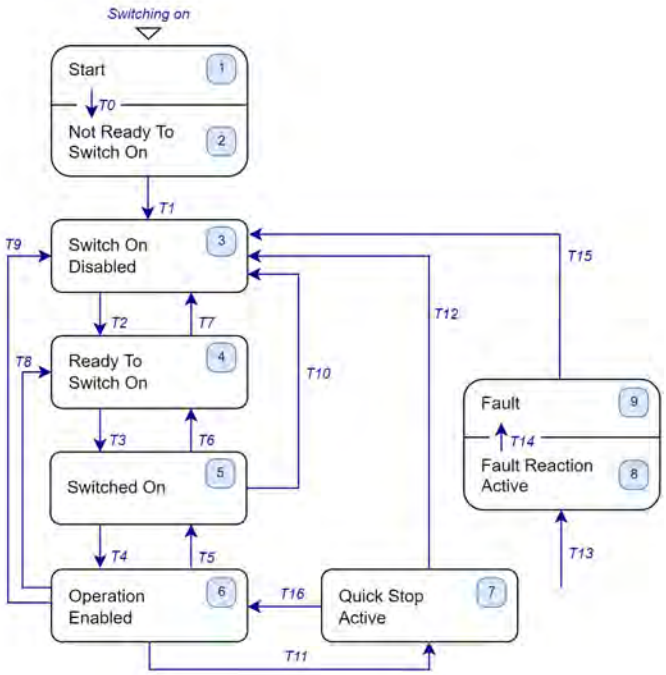

Fig. 5. Servo drive FSM from IEC 61800 standard

security properties.

Note that many manufacturers adhere to PLCopen standards. For example, they provide standard FB libraries such as Motion Function Blocks (MFBs) for the majority of automation platforms. Therefore, additional security properties can be obtained.

Finally, some security properties are use case dependant and they are observed from network traffic. For example, in our use case, a standard PLCopen FB is used to issue a "Homing" command to the two servo drives controlling rudders. The network traffic will reveal two consecutive commands: one for the left rudders and one for the right rudder. The specific order in which these commands are executed (such as Homing the right rudder before the left one) is left to the discretion of the programmer, as it is not explicitly defined by the standard. We extract this type of information from the network traffic which gives additional security properties.

C. Software Deployment

From a practical aspect, capturing network traffic is required in order to implement our intrusion detection approach on the testbed. We decided to rely on Zeek[1], an existing open source NIDS, since this type of solution is usually supported by a large community. Consequently, it is expected to be maintained and tested in different environments, making it safer than a home-made one.

Traffic capture at high levels network (Traffic capture 1 on Figure 4) is more straightforward than fieldbus traffic capture (Traffic capture 2 and 3 on Figure 4). In the first case, we rely on switch port mirroring. On the second, we use RS-485 to USB converters[2] together with a stand-alone Modbus RTU worker[3] [19] and Zeek. Figure 6 shows the set up we rely on for network traffic capture on one fieldbus. The first RS-485 to USB converter is used to capture every network packet that flows on the fieldbus, while the second converter is used for sending attacks on the bus. The stand-alone Modbus RTU Worker was developed using C and C++ languages.

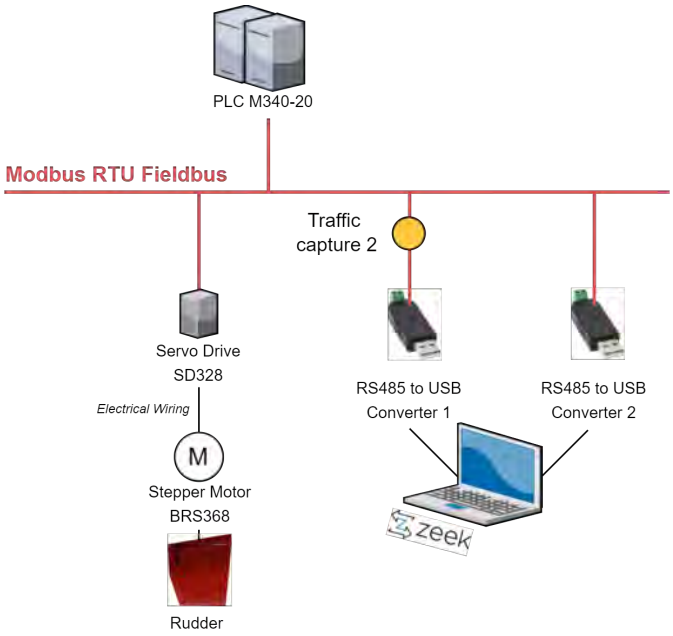

Fig. 6. Set up for fieldbus traffic capture

As previously introduced, the Scope Recognizer identifies state changes by monitoring status messages in the network traffic. This allows to activate and deactivate monitors according to operation contexts (each monitor evaluates a security property). Concerning the implementation of the framework, the Scope Recognizer encompasses 6 state machines:

- A version of IEC 61800 standards drive FSM (see Figure 5) mapped to Modbus RTU protocol. There is a FSM for each servo drive (two FSMs in total).
- Servo drive operating machine modes inherited from IEC 61800 standard and Supervisory Control specifications. In our use case, as previously stated, only Homing and Profile Position modes are allowed. Here again, there is a FSM for each servo drive (two FSMs in total).
- A FSM with two states: "Movement" or "Motionless" to state either the rudders are in movement or not (two FSMs in total).

In total, 6 FSMs (3 for each local loop) identify the current behavior of the two servo drives from the aspect of internal state, network, operating modes and movement. The FSM

[1]https://zeek.org (last accessed: June 2024)

[2]https://joy-it.net/en/products/SBC-TTL-RS485 (last accessed: June 2024)

[3]Codes can be found in the GitHub https://github.com/V3Hr7LnNRu7T/SSIDS (last accessed: June 2024).

regroups every status information available in the network traffic.

V. Evaluation

Our evaluation methodology is essentially the same as the one described in our previous work [9].

In total, 21 monitors are deployed including 2 monitors concerning synchronisation of the two servo drives. In this specific use case, only 3 types of Dwyer security patterns are used: *Precedence*, *Response* and *Absence*. The monitors are implemented using Reelay[4] library [20]. This library relies on Past-MTL with the temporal logic operators: *previously* (analogous to *next* in the past), *once* (*sometime* in the past), *historically* (*always* in the past) and *since* (*until* in the past).

As in [9], all the attacks are detected as expected given that they are targeted to violate the monitored security properties. But contrary to [9], where some false positive alerts were observed due to a misconfiguration of some Controller Area Network (CAN) nodes, no false positive are recorded, which validates the correctness of control programs (i.e. the normal behavior is safe).

Concerning the evaluation, we compute the execution time of the detection loop. The idea is to measure the computational time of a full monitoring step. Due to the few security patterns used in the implementation (21 security patterns), the evaluation was conducted on each of them (i.e. on a monitor evaluating the pattern). The results are shown in Figure 7, 8, and 9. For each type of pattern, the curves show that the execution time of the detection loop depending on the number of active monitors is linear. In order to have a statistically relevant estimation of the Worst Case Execution Time (WCET), we gradually increased the number of active monitors up to 800 by duplicating the active monitors. In order to decide the scalability, we compare the WCET with PLC cycle time. As a data exchange with the PLC (either via TCP/IP or Modbus RTU) takes at most two cycle times (one for sending the request and one for the reception of the answer) and the cycle time of the PLC in this use case equals 10ms, we consider that the detection is achieved "inline" if the WCET is less than 20 ms. Inspecting the values of WCET in Figure 7, 8, 9, one can notice that we can run up to 800 monitors, whatever the monitored pattern is, without backlogging events. Therefore, we can notice that our approach has a good scalability, which confirms the good results of [9] on another ICS platform.

VI. Related Work

Intrusion detection approaches based on specifications were already studied in the context of Cyber-Physical Systems [21]–[24] as they are able to detect new attacks in an industrial system although extracting specifications may be challenging. Most commonly, they will be extracted manually by an expert from system documents or embedded programs specification [22], [25], [26] but this method is prone to human error. An automatic way to obtain the security patterns is to apply specification mining from normal network traffic [27]–[29]. Although exhaustive, this approach will generate a large number of redundant or overlapping specifications and, generally, a second optimization step is required in order to reduce the number of deduced patterns [29].

[4]https://github.com/doganulus/reelay (last accessed: June 2024)

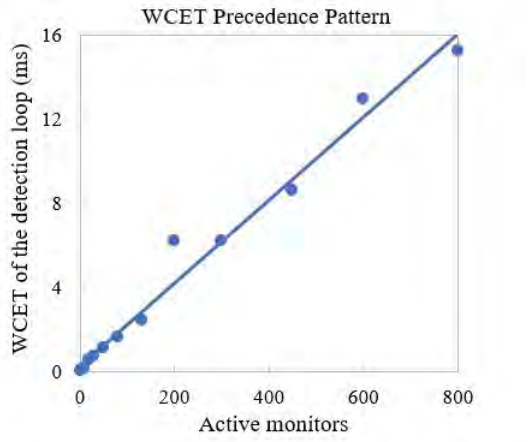

Fig. 7. Execution time of the detection loop depending on the number of active monitors – Precedence pattern

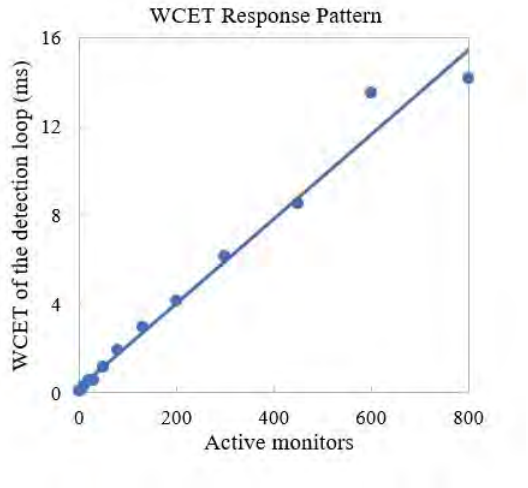

Fig. 8. Execution time of the detection loop depending on the number of active monitors – Response pattern

Complementary approaches in PAA detection are those based on the mathematical model of process dynamics [30]–[33]. These approaches are mostly based on system diagnosis (in the control system sense) and, therefore, they generally require access to the internal variables of controllers that are not available into the network traffic. Finally, some approaches are based purely on network traffic analysis and do not consider physical process characteristics [34]–[39]. Therefore, they are out of the scope of the present work.

Due to the lack of tools allowing traffic monitoring at

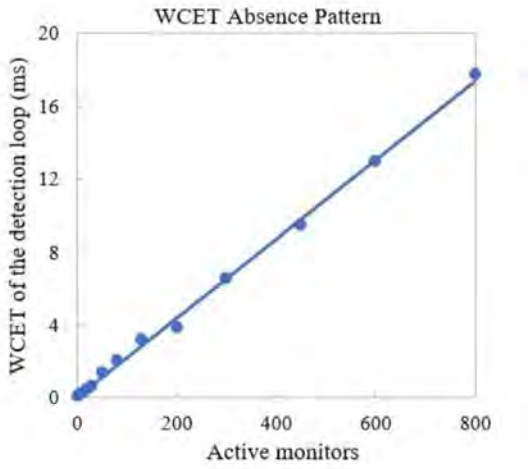

Fig. 9. Execution time of the detection loop depending on the number of active monitors – Absence pattern

fieldbus level, almost all approaches rely on the observation of network traffic between Supervisory Control and PLCs. They will, typically, survey the memory variables of a PLC and reconstruct the image of process variables. They generally do not take into account functioning states and modes of lower level controllers and they are unable to detect attacks at fieldbus level.

With respect with the previously cited works, our approach does not generate redundant security patterns and is based on a systematic extraction of properties from the standard. As we are able to monitor the traffic at several network levels, we are able to detect attacks at fieldbus level and we do not rely on the survey of the internal PLC variables.

On the experimental part, several maritime testbeds are described in the literature. In [40], a roadmap is proposed for the creation of a specialized maritime-cyber lab, which combines maritime technology and traditional cybersecurity labs. The proposed Cyber-SHIP Lab (Software, Hardware, Information and Protections) hosts a range of real, non-simulated, maritime systems. In [41], a conceptual and generic analysis on how Cyber Ranges (CRs) can be used in the maritime context is presented. The work in [42] proposes a virtual port logistics and supply chain cybersecurity training platform. This CR is built to organize small cyber exercises on specific topics. It enables the simulation of cyberattacks and defences, identification of threats and vulnerabilities, traffic monitoring and in-depth analysis. The authors in [43] present a maritime CR environment, combining navigational, information and telecommunications systems, networks, and SCADA systems. It supports maritime vulnerability, penetration and exploitation scenarios, traffic eavesdropping, positional systems spoofing, navigation takeover, signal intelligence scenarios, and others. It combines simulated and emulated systems to be as realistic as possible. Finally, the work presented in [44] is about a testbed for cyber security awareness in the maritime domain. Two subsystems are implemented: the propulsion system with the engine control, and the navigation system with a rudder. Some sensors and actuators (e.g., temperature sensing) are simulated with Arduinos.

Our testbed is a representative model of a surface warship using real industrial control devices controlling a reduced size ship model. Although not all the characteristics, complexity and dynamics of a real warship are available, the security properties that can be studied are the same as for a real ship.

VII. Conclusions and Future work

In this paper, we presented the experimental validation of a specification-based IDS instantiated in a realistic naval testbed. We validated our approach and evaluated the response time properties of the system. Among other interesting results, this article shows the extensibility of our previous work to another real ICS platform.

From a practical point of view, the IDS is deployed over three different networks including two fieldbuses. We are able to detect violation of security properties that involves data from several capture points. The IDS proves to be very fast, while WCET statistical evaluation shows that we would be able to deploy around 800 monitors and evaluate them in less than a network communication cycle (or equivalently two PLC program cycles). Therefore, our IDS is adequate for inline detection of attacks.

Last but not least, we extended the detection capabilities of an open source IDS to Modbus RTU (RS-485 fieldbus) and made the code available online.

The future work will consist mainly in applying our intrusion detection approach to the other subsystems of our naval testbed and consider distributed security properties. For the moment, our IDS is a centralised detection system using a distributed data source. That means that the monitoring system aggregates the traffic from the three network sensors and performs detection on locally (i.e. at subsystem level) observable security properties. We want to extend the set up to distributed security properties. Consider for instance a property like *"do not modify navigation speed if the gun is acquiring the target"*: such a security property will need either a fusion of the traffic of the two subsystems and perform a centralized detection (which in practice will not scale up due to bandwidth limitations), or, a distributed detection framework. For instance, one IDS will monitor the artillery systems and raise the event "target acquiring" to the IDS monitoring the propulsion allowing detection of the security property violation. In the short term, we will study the automatic decomposition of global properties in local events to be monitored, then we will study the optimal implementation of the detection framework.

We will also continue to develop IDS extensions to fieldbuses for the open source Zeek IDS.

References

[1] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, and S. Lightman, "Guide to Operational Technology (OT) Security," NIST Special Publication 800-82 Rev. 3, Technical Report, 2022, [Online] https://doi.org/10.6028/NIST.SP.800-82r3.ipd, last accessed Feb. 2023.

[2] N. Falliere, L. O. Murchu, and E. Chien, "W32.stuxnet dossier," Symantec Security Response, Technical Report, 2011, [Online] https://docs.broadcom.com/doc/security-response-w32-stuxnet-dossier-11-en, last accessed June 2024.

[3] US-CERT, "Crashoverride," https://www.us-cert.gov/ncas/alerts/TA17-163A, 2017, last accessed June 2024.

[4] Dragos, "Crashoverride: Analysis of the threat to electric grid operations," https://dragos.com/blog/crashoverride/, 2017, last accessed June 2024.

[5] Kaspersky ICS CERT, "Threat landscape for industrial automation systems," Kaspersky, Technical Report, 2022, [Online] https://ics-cert.kaspersky.com/media/H1_2019_kaspersky_ICS_REPORT_EN.pdf, last accessed June 2024.

[6] T. J. Williams, "A Reference Model for Computer Integrated Manufacturing from The Viewpoint of Industrial Automation," *IFAC Proceedings Volumes*, vol. 23, no. 8, pp. 281–291, 1990.

[7] A. Carcano, A. Coletta, M. Guglielmi, M. Masera, I. Nai Fovino, and A. Trombetta, "A Multidimensional Critical State Analysis for Detecting Intrusions in SCADA Systems," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 2, pp. 179–186, 2011.

[8] J. Nivethan and M. Papa, "A SCADA Intrusion Detection Framework that Incorporates Process Semantics," in *Proceedings of the 11th Annual Cyber and Information Security Research Conference*, 2016, pp. 1–5.

[9] E. Hotellier, F. Sicard, J. Francq, and S. Mocanu, "Standard Specification-based Intrusion Detection for Hierarchical Industrial Control Systems," *Information Sciences*, 2024, article 120102.

[10] MITRE — ATT&CK® for Industrial Control Systems, 2021, [Online] https://attack.mitre.org/matrices/ics/, last accessed June 2024.

[11] M. B. Dwyer, G. S. Avrunin, and J. C. Corbett, "Patterns in property specifications for finite-state verification," in *Proceedings of the 21st international conference on Software engineering*, 1999, pp. 411–420.

[12] S. Konrad and B. Cheng, "Real-time specification patterns," in *Proceedings of the 27th international conference on Software engineering*. ACM, 2005, pp. 372–381.

[13] O. Maler and D. Ničković, "Monitoring Temporal Properties of Continuous Signals," in *International Symposium on Formal Techniques in Real-Time and Fault-Tolerant Systems*. Springer, 2004, pp. 152–166.

[14] E. Bartocci, J. Deshmukh, A. Donzé, G. Fainekos, O. Maler, D. Ničković, and S. Sankaranarayanan, "Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications," *Lectures on Runtime Verification: Introductory and Advanced Topics*, vol. 10457, pp. 135–175, 2018.

[15] M. Conti, D. Donadel, and F. Turrin, "A survey on Industrial Control System Testbeds and Datasets for Security Research," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2248–2294, 2021.

[16] F. Sicard, E. Hotellier, and J. Francq, "An industrial control system physical testbed for naval defense cybersecurity research," in *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. IEEE, 2022, pp. 413–422.

[17] IEC 61800, "Adjustable speed electrical power drive systems," Industrial Electrotechnical Commission, International Standard, 2021.

[18] PLCopen, "PLCopen - for efficiency in automation," PLCopen, Technical Specification, 2011, [Online] http://www.plcopen.org/, last accessed Apr. 2024.

[19] S. Mocanu and E. Hotellier, "Modbus RTU network trafic capture." Jan. 2024, last accessed June 2024. [Online]. Available: https://hal.science/hal-04426339

[20] D. Ulus, "Online Monitoring of Metric Temporal Logic using Sequential Networks," *arXiv preprint arXiv:1901.00175*, 2019.

[21] C. McParland, S. Peisert, and A. Scaglione, "Monitoring security of networked control systems: It's the physics," *IEEE Security & Privacy*, vol. 12, no. 6, pp. 32–39, 2014.

[22] R. Mitchell and I.-R. Chen, "Behavior rule specification-based intrusion detection for safety critical medical cyber physical systems," in *IEEE Trans. Dependable Secure Comput.*, 2015, pp. 16–30.

[23] D. Fauri, D. R. Dos Santos, E. Costante, J. den Hartog, S. Etalle, and S. Tonetta, "From System Specification to Anomaly Detection (and Back)," in *Proceedings of the 2017 Workshop on Cyber-Physical Systems Security and Privacy*. ACM, 2017, pp. 13–24.

[24] V. Sharma, I. You, K. Yim, R. Chen, and J.-H. Cho, "Briot: Behavior rule specification-based misbehavior detection for iot-embedded cyber-physical systems," *IEEE Access*, vol. 7, pp. 118 556–118 580, 2019.

[25] N. Walkinshaw and K. Bogdanov, "Inferring finite-state models with temporal constraints," in *2008 23rd IEEE/ACM International Conference on Automated Software Engineering*. IEEE, 2008, pp. 248–257.

[26] H. Lin, A. Slagell, C. Di Martino, Z. Kalbarczyk, and R. K. Iyer, "Adapting Bro into SCADA: building a specification-based intrusion detection system for the DNP3 protocol," in *Proc. CSIIRW '13*, 2013, pp. 1–4.

[27] C. Lemieux, D. Park, and I. Beschastnikh, "General LTL specification mining," in *Proc. ASE'15*, 2015, pp. 81–92.

[28] D. Neider and I. Gavran, "Learning linear temporal properties," in *2018 Formal Methods in Computer Aided Design (FMCAD)*. IEEE, 2018, pp. 1–10.

[29] O. Koucham, S. Mocanu, G. Hiet, J.-M. Thiriet, and F. Majorczyk, "Efficient Mining of Temporal Safety Properties for Intrusion Detection in Industrial Control Systems," in *SAFEPROCESS 2018*, 2018, pp. 1–8.

[30] F. Khorrami, P. Krishnamurthy, and R. Karri, "Cybersecurity for control systems: A process-aware perspective," in *IEEE Design & Test*, vol. 33, 2016, pp. 75–83.

[31] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE Trans. Automat. Contr.*, vol. 58, pp. 2715–2729, 2013.

[32] K. Miao, X. Shi, and W.-A. Zhang, "Attack signal estimation for intrusion detection in industrial control system," *Computers & Security*, vol. 96, p. 101926, 2020.

[33] S. Papa, W. Casper, and S. Nair, "A transfer function based intrusion detection system for SCADA systems," in *IEEE Int. Conf. Technol. Homel. Secur.*, 2012, pp. 93–98.

[34] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes, "Using Model-based Intrusion Detection for SCADA Networks," in *Proc. SCADA Security Scientific Symposium*, 2007, pp. 127–134.

[35] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," in *Int. J. Crit. Infrastruct. Prot.*, 2013, pp. 63–75.

[36] M. Yoon and G. Ciocarlie, "Communication Pattern Monitoring : Improving the Utility of Anomaly Detection for Industrial Control Systems," in *NDSS Workshop SENT '14*, 2014.

[37] M. Caselli, E. Zambon, and F. Kargl, "Sequence-aware Intrusion Detection in Industrial Control Systems," in *Proc. ACM Workshop CPSS*, 2015, pp. 13–24.

[38] O. Yüksel, J. den Hartog, and S. Etalle, "Reading between the fields: Practical, effective intrusion detection for industrial control systems," in *Proc. ACM Symp. Appl. Comput.*, 2016, pp. 2063–2070.

[39] B. Ferling, J. Chromik, M. Caselli, and A. Remke, "Intrusion detection for sequence-based attacks with reduced traffic models," in *Int. Conf. MMB '18*. Springer, 2018, pp. 53–67.

[40] K. Tam, K. Forshaw, and K. Jones, "Cyber-SHIP: Developing next generation maritime cyber research capabilities," in *International Conference on Marine Engineering and Technology (ICMET)*. IMarEST, 2019.

[41] K. Tam, K. Moara-Nkwe, and K. Jones, "The Use of Cyber Ranges in the Maritime Context: Assessing maritime-cyber risks, raising awareness, and providing training," *Maritime Technology and Research*, 2020.

[42] H. Pyykkö, J. Kuusijärvi, S. Noponen, S. Toivonen, and V. Hinkka, "Building a virtual maritime logistics cybersecurity training platform," in *Data Science and Innovation in Supply Chain Management: How Data Transforms the Value Chain. Proceedings of the Hamburg International Conference of Logistics (HICL), Vol. 29*. Berlin: epubli GmbH, 2020, pp. 223–246.

[43] G. Potamos, A. Peratikou, and S. Stavrou, "Towards a maritime cyber range training environment," in *IEEE International Conference on Cyber Security and Resilience (CSR)*. IEEE, 2021, pp. 180–185.

[44] T. Becmeur, X. Boudvin, D. Brosset, G. Héno, T. Merien, O. Jacq, Y. Kermarrec, and B. Sultan, "A platform for raising awareness on cyber security in a maritime context," in *International Conference on Computational Science and Computational Intelligence (CSCI)*. IEEE, 2017, pp. 103–108.