OpenPLC Based Control System Testbed for PLC Whitelisting System

Shintaro Fujita[1], Kousuke Hata[1], Akinori Mochizuki[1], Kenji Sawada[1], Seiichi Shin[1] and Shu Hosokawa[2]

[1]The University of Electro-Communications, Japan

[2]Control System Security Center, Japan

(Tel: 81-42-443-5253, Fax: 81-42-443-5253)

[1]knj.sawada@uec.ac.jp

Abstract: This paper proposes a security testbed system for industrial control systems. In control systems, controllers are final fortresses to continue the operation of field systems. Then, we need countermeasures of controllers. The whitelisting function is efficient in controller security. The whitelisting function registers normal operations in a list and detects unregistered operations as abnormal. We need a testbed system to check whether the whitelist function does not affect other functions of the controller. The industrial controller and its engineering tool are relatively expensive, and are customized with respect to controller vendors. To enhance the whitelist development, this study proposes a testbed system using OpenPLC which is an open source software. This system is independent of controller vendors and is applicable for controller programming languages. We implement a whitelist based anomaly detection method for the testbed system and validate that the anomaly detection method operates correctly.

Keywords: PLC, Security, Testbed, Whitelist.

1 INTRODUCTION

Control systems face a lot of cyber-attacks [1], such as Stuxnet, WannaCry, CrashOverride, Bad Rabbit. The typical control system consists of SCADA (Supervisory Control And Data Acquisition), network switches, controllers, and field devices. Initially, it is supposed that malicious attackers target SCADA and penetrate its vulnerabilities because Windows OS is introduced to SCADA, and its version often remains old. However, recent malware directly targets controllers. Controllers are the final fortresses of control systems. Even if SCADAs stop suddenly, controllers themselves continue the operation of field devise. If controllers stop, control systems cannot be operated by SCADAs. Therefore, we need countermeasures focusing on controllers [2].

The main functions of the controller are operating field devices and communicating with other devices. System resources for the security function are not high, and then we cannot apply common antivirus software to controllers directly. Standard antivirus software is based on the blacklisting system in which anomaly behaviors caused by malware/worms are listed, and actions of application commands are always checked. This system load of blacklist checking is very high for controllers. Further, the backlisting system requires frequent updates of pattern files to maintain its defensive strength. Therefore, it is supposed that the whitelisting system is familiar with control systems rather than the blacklisting system [3]. The whitelisting system registers the normal application/network commands and usual network information (IP, MAC address) and accepts/approves only commands/information on the list. Its system load is lower than that of the blacklisting system. The list updates timing is restricted to the system maintenance changing the control system operation.

Motivated by the above, the current authors study a whitelisting system of the controller, mainly, PLC (Programmable Logic Controller). PLCs are commonly used in ICSs (Industrial Control Systems), so enhancing the security functions of PLC leads to improving those of ICSs. Further, our proposed whitelist of PLC [4] is expressed by LD (Ladder Diagram), which is one of the most presently available PLC programming languages. Using LD, our whitelisting function does not need to change the firmware of PLC and then is applicable for various PLCs.

On the other hand, to develop a whitelisting function of PLC, we need a testbed system using PLC to check whether the whitelist function does not affect other functions of PLC. PLC and its engineering tool are relatively expensive are customized with respect to PLC vendors. To enhance the whitelist development, this study proposes a testbed system using OpenPLC which an open source software [5][6]. The proposed testbed is independent of PLC vendors and is applicable for PLC programming languages including LD. The proposed testbed provides a space-saving and low-cost environment for ICS security research. We implement a whitelist based anomaly detection method for the testbed system. This research validates that the anomaly detection method for PLC operates correctly. This paper reports that the testbed system is useful for control system security.

2 CONTROL SYSTEM

Fig. 1 shows a typical control system architecture in critical infrastructures and industrial systems. Control systems are composed of computers such as HMI (Human Machine Interface) and Engineering Computer, network switches, controllers such as PLC, and DCS (Distributed Control System), and field devices such as sensor and actuator. Also, Remote Computer connecting with the internet manages and monitors the control system.

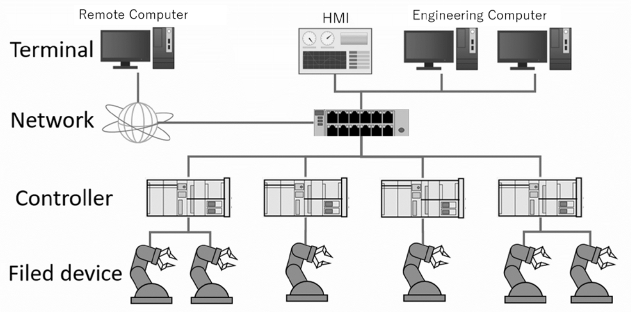

Fig. 1. Industrial control system

Cyber-attacks on control systems include the following connection of unauthorized terminals to the network, cyber-virus infections, attacker intrusions. A cyber-attack is the equivalent of sending unauthorized packets or instructions over a network switch. Therefore, at least, the testbed for PLC security measures needs a PLC and a field device, as well as an HMI and a network switch.

3 OpenPLC

OpenPLC [5][6] is an open source software developed by Thiago Rodrigues Alves. The computer with OpenPLC and I/O device simulates PLC functions.

OpenPLC supports Microsoft Windows and Linux as computer OS. OpenPLC operates on computers as a WEB application using Node.js [7]. The implementation of PLC control programming is realized by uploading ST (Structured Text) through the OpenPLC.

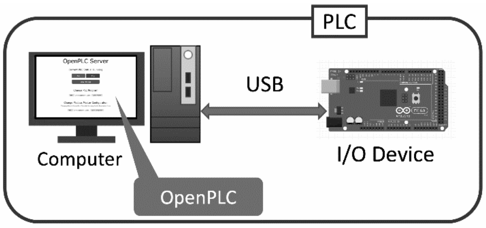

Fig. 2. OpenPLC system

OpenPLC recommends PLC Editor as a developing environment, PLC Editor supports five PLC languages such as LD, ST, FBD, IL and SFC. The languages except for ST are translated to ST on OpenPLC.

I/O devices control field devices according to the computer commands. OpenPLC supports the following I/O devices: Raspberry Pi [8], Arduino and compatible boards [9], UniPi Industrial Platform [10], Modbus Slave Devices [11], ESP8226 [12] and PiXtend [13].

4 WHITELISTING FUNCTION

The whitelisting system registers normal operations on the list and rejects the operations which are not registered on the list. In control systems, normal operations of communication commands and execution orders of actuator and sensor depend on their blueprints. Then, the whitelisting system detects zero-day attacks that deviate from the blueprints. Also, its system load is lower than that of the blacklisting system [3].

The previous research of the current authors implements this whitelisting function to PLC, as a PLC anomaly detection method [4]. This method registers execution orders of actuators and sensors on the list. The list is modeled by Petri Net [14] and its state transition and state constraint rules of the model are converted to LD. If the rules are not satisfied, the method detects anomaly executive orders. Almost PLCs can implement LD, so we can apply the whitelisting function to almost PLCs without updating firmware and improving hardware.

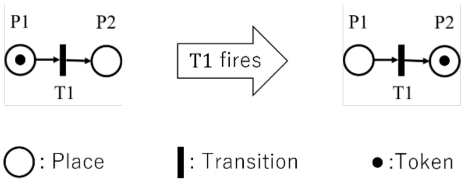

Fig. 3. Petri Net operation example

The following is an overview of Petri Net based PLC anomaly detection method. Petri Net is a directed bipartite graph with a Place and a Transition. Places may contain tokens, a number of Tokens in Places, and transitions of Tokes express states and state changes of Petri Net. The operation of the Transition causes the Token to transition. This action is called fire. Fig. 3 shows an example of Petri Net operation. When T1 is fired, the Token moves from P1 to P2.

First, we model the normal sequence of operation of the control system using Petri Net. We take the case of Petri Net with n Places and m Transitions. The model is represented by the state equation

$$\boldsymbol{x}[k+1] = \boldsymbol{x}[k] + \boldsymbol{B}\boldsymbol{u}[k] \qquad (1)$$

where $\boldsymbol{x}[k] \in \mathbb{N}^n$ is the state vector, $\boldsymbol{u}[k] \in \mathbb{N}^m$ is the input vector and $\boldsymbol{B} \in \mathbb{Z}^{n\times m}$ is the connection matrix. Also, for each element $x_i \in \mathbb{N}$ of the state vector $\boldsymbol{x}[k]$, the possible values to be taken are restricted by

$$0 \le x_i \le N_i. \qquad (2)$$

This N_i is a natural number that is determined for each element depending on the target to be modeled. The whitelist detects the behavior that does not satisfy Eq. (2) as an anomaly. Consider Fig. 3, $n = 2,\ m = 1,\ \boldsymbol{B} = [-1 \quad 1]^T$.

5 TESTBED STRUCTURE

Fig. 4 shows the proposed testbed system. We set two laptops. One is for OpenPLC, and the other is for HMI and Cracker PC. The two are connected by Modbus/TCP, which is an open industrial protocol. Fig. 5 shows an I/O device made by Arduino MEGA 2560. Fig. 6 shows the robot arm.

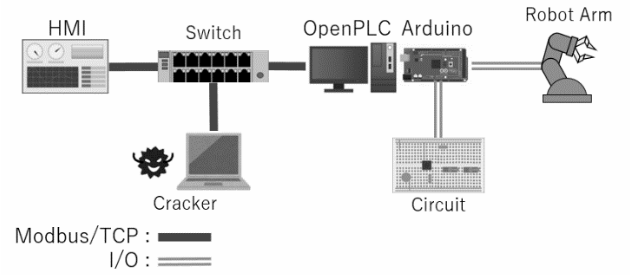

Fig. 4. Testbed system

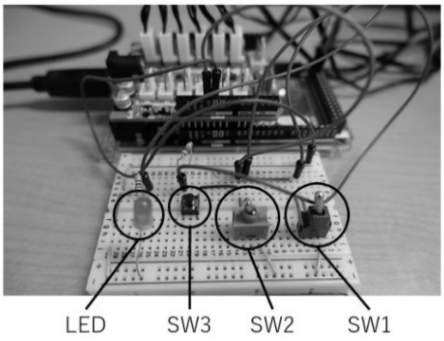

Fig. 5. Arduino MEGA 2560 and circuit

Fig. 6. Robot arm

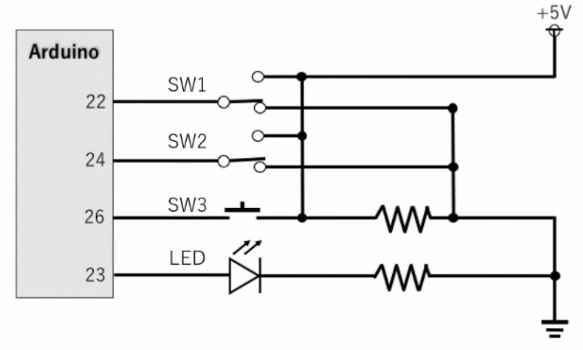

Fig. 7. Control panel circuit

Arduino MEGA 2560 connects with the robot arm and the control panel. As shown in Fig. 7, the panel has three switches (SW1, SW2 and SW3) and one LED. SW1 is the power button of the robot arm, SW2 is the stop-start switch of the arm operation, and SW3 is the reset switch of the arm operation.

This paper supposes that the cracker PC attacks the PLC (the OpenPLC computer) through the network. The system simulates the component conveyer system in factory automation, and the robot arm coveys block type components. This system has four functions: the start-up of the arm, the block conveyer, the stop of the arm, the reset to the initial arm position. The attacking target is the block conveyer, and the whitelisting function is designed for the block conveyer in the testbed. The conveyer procedure is as follows:

1. The arm moves to the front of the block.
2. The arm moves to the grasping position.
3. The arm grasps the block.
4. The arm uplifts the block.
5. The arm conveys the block to the releasing position.
6. The arm releases the block.

Also, the Robot arm drives via 6 servo motors: base, shoulder, elbow, wrist_ver, wrist_rot, gripper. Table 1 shows the target angles of each motor according to the above conveyer procedure.

Table 1. Target angles for each conveyer step

Step	base	shoulder	elbow	wrist _ver	wrist _rot	gripper
1	30	90	180	180	0	10
2	0	90	180	180	0	10
3	0	90	180	180	0	60
4	90	30	150	135	45	60
5	180	120	180	60	90	60
6	180	120	180	60	90	10

Fig. 8 shows the LD which controls the target angle of the base motor. The MUX block controls the angle of the robot arm in the block conveyer function. The MUX selects inputs IN n $(n = 0, 1, 2, 3, 4, 5)$ and outputs the corresponding angle value depending on variable K of each motor according to the above conveyer procedure. When K is 0, MUX selects IN0 and outputs 30, and then the target angle of the base motor is set to 30 degrees. When the six motors reach the target angles, the value of K is incremented. When K is 5, the next value of K is 0. For the six servo motors, the change of K is equal to the change of target angle and then is repeated. That is, we suppose that the normal operation of the robot arm is the repeat of this convey procedure. The parts other than the MUX block in Fig. 8 are the LD program for changing functions or initializing the servo motors by operating the switches (SW1, SW2 and SW3).

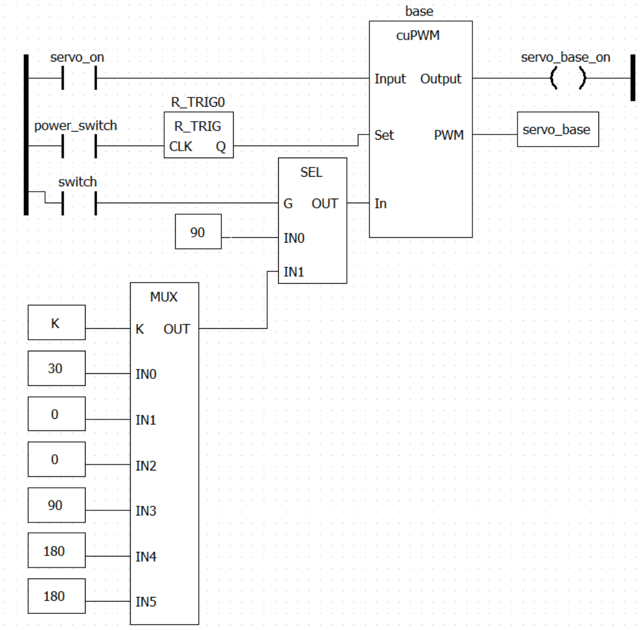

Fig. 8. LD of the base servo motor.

6 EXPERIMENTAL VERIFICATION

6.1 Implementation of whitelist

The Petri Net model of the normal operation of the testbed is shown in Fig. 9. This model is based on the anomaly detection method proposed in [4]. The number of Tokens in Places is constrained by Eq. (2). This constraint is different for each structure of the Petri Net model and each Place. In the model shown in Fig. 8, the initial state is P1 and the constraint is given by

$$0 \le x_i \le 1 \quad (i = 1, 2, 3,4,5,6). \tag{3}$$

Steps 1 to 6 in Table 1 correspond to Places P1 to P6. Each Transition Tn $(n = 1, 2, 3, 4, 5, 6)$ fires and K is incremented when the angle of each motor of the robot arm matches the target angle.

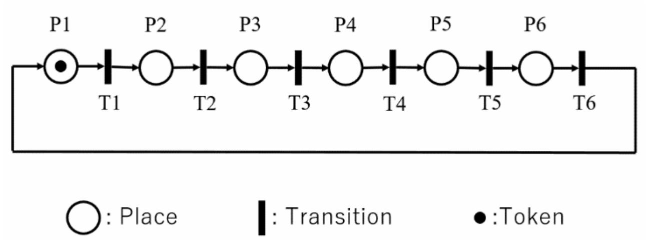

Fig. 9. Petri Net model of the robot arm

Following the method in Reference [4], we transform the Petri Net model in Fig. 9 into an LD. This transformation generates two LDs, one for token movement and one for anomaly detection. We explain the behavior of each LD, taking as an example, the change focused on P1 in Fig. 9. We use FB (Function Block) on the LD to implement whitelist functions into the PLC. The operations for each FB on the LD are shown in Table 2.

Table 2. Operation of each FB

FB	Operation
EQ	Outputs True when IN1 and IN2 are equal.
R_TRIG	Detects the rising edge of CLK.
ADD	Outputs “IN1 + IN2” when EN is True.
SUB	Outputs “IN1 – IN2” when EN is True.
LT	Outputs True when “IN1 < IN2”.
GT	Outputs True when “IN1 > IN2”.

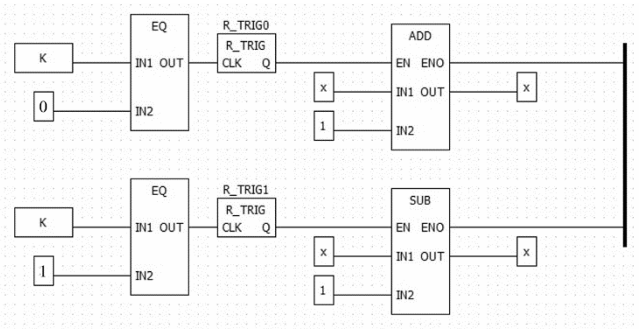

Fig. 10. LD of the Petri Net model

First, Fig. 10 shows an LD for the token movement. This LD stores the number of tokens held by P1 in the variable x. When T6 fires, $\mathrm{K}=0$, the upper LD operates and x is incremented. When T1 fires, $\mathrm{K}=1$, the lower LD works, and x is decremented. In this way, the token movement in the Petri Net is programmed.

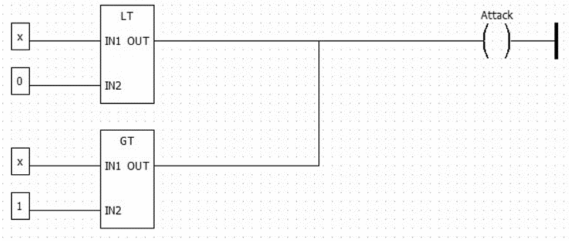

Fig. 11. LD of anomaly detection

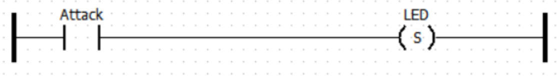

Fig. 12. LD that lights the LED

Next, Fig. 11 shows the LD for anomaly detection. This LD determines whether the number of tokens held by P1 satisfies Eq. (3). If x does not satisfy Eq. (3), the whitelist detects an anomaly. Fig. 12 shows the LD for lighting the LED. The variable LED in Fig. 12 becomes the output of the Arduino pin 23.

6.2 Experimental verification

We verified the validity of the anomaly detection method carrying out the simulated attack on the proposed testbed. We assumed that an unknown device connects with the control network and attacks the PLC. The cracker changes the value of K taking over Modbus/TCP commands and aims to disturb the convey procedure. Fig. 12 shows the screen of the HMI in the experimental result. The screen shows seven graphs. The first six graphs are time responses for six servo motors. The last one is the ON/OFF graph of the LED.

In the verification, we assume the following fail-safe scenario. The simulated attack interferes with the normal operation of the testbed. The operator finds an abnormality by the whitelist and operates SW1 and SW2 to stop the robot arm safely. After stopping, the operator investigates the cause of the abnormality.

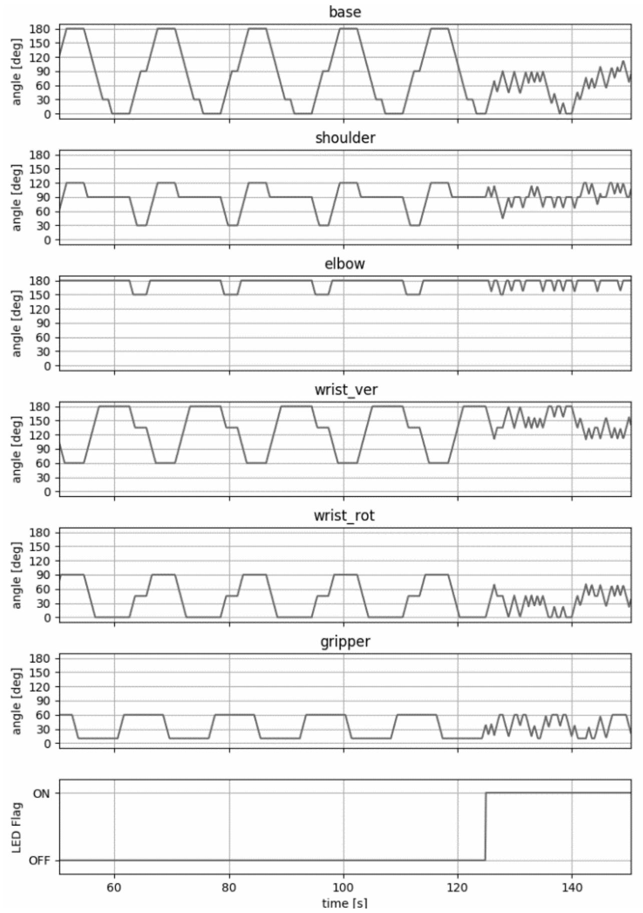

Fig. 12. HMI screen

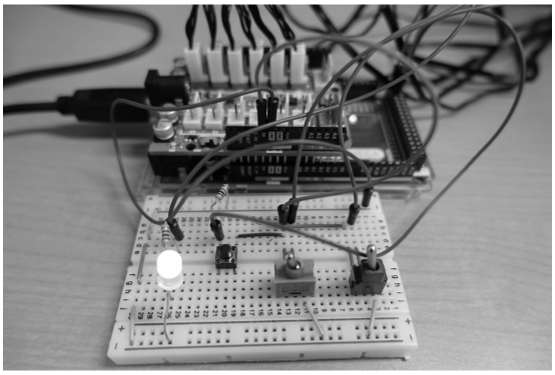

Fig. 13. Anomaly detection

In the normal operation, we see that each servo motor reach target angles repeatedly. When the time is 125 seconds, the cracker attacks the PLC, and all motors fluctuate. At the same time, the LED changes from OFF to ON as shown in Fig. 13. The whitelisting function focuses on the executive orders of field devices. It is difficult to detect the attacks which disturb the executive orders. For example, the proposed method does not detect an attack that changes not the executive orders but the execution time. If the execution time becomes short, servo motors do not reach target angles, and then the robot arm does not convey the blocks. To detect such the attack, we need to apply time conditions for the whitelisting function. This is our future work.

While the simulation-based testbed [15] can only represent the system controller and plant, our testbed is more realistic by considering the HMI and the network switches. Also, compared to testbeds using actual devices [16], the use of open source software makes it possible to build testbeds at a lower cost and to develop vendor-independent security countermeasures. The above two testbeds are famous testbeds for ICS security: chemical plant [15] and a water treatment plant [16].

7 CONCLUSIONS

This paper has proposed a testbed using OpenPLC for control system security. We have verified the validity of the proposed whitelisting function using the testbed. By including the HMI and the network switch in the PLC testbed construction, it was possible to simulate cyber-attacks of illegal commands to the PLC. Also, by using OpenPLC, an open source software, vendor-independent security verification is possible. Our future work is to develop a connection method between the computers with OpenPLC because OpenPLC has no connection with others. By solving this problem, we try to build a large-scale testbed system.

This work was supported by Council for Science, Technology and Innovation (CSTI), Cross-ministerial Strategic Innovation Promotion Program (SIP), "Cyber-Security for Critical Infrastructure" (funding agency: NEDO)

REFERENCES

[1] G. Liang, The 2015 Ukraine Blackout: Implications for False Data Injection Attacks, IEEE Transactions on Power Systems 2017, 3317/3318 (2017)

[2] T. Sasaki, K. Sawada, S. Shin, S. Hosokawa, Model Based Fallback Control for Networked Control System via Switched Lyapunov Function, IECON 2015 - 41st Annual Conference of the IEEE, 2000/2005 (2015)

[3] D. Kim, J. Lee, Blacklist vs. Whitelist-Based Ransomware Solutions, IEEE Consumer Electronics Magazine, Vol. 9, 22/28 (2020)

[4] A. Mochizuki, K. Sawada, S. Shin and S. Hosokawa, On Experimental Verification of Model Based White List for PLC Anomaly Detection, ASCC 2017, 1766/1771 (2017)

[5] T. R. Alves, M. Buratto, F. M. de Souza, T. V. Rodrigues, OpenPLC: An open source alternative to automation, IEEE Global Humanitarian Technology (GHTC 2014), 2014

[6] https://openplcproject.com

[7] https://nodejs.org/en/

[8] https://www.raspberrypi.org/

[9] http://arduino.org/

[10] https://www.unipi.technology/

[11] http://www.modbustools.com/modbus¥_slave.html

[12] https://startiot.telenor.com/learning/esp8226-openrtos-and-managed-iot-cloud/

[13] http://www.pixtend.de/

[14] T. Murata, Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, Vol. 77, No. 4, 541/580 (1989)

[15] C. Martin-Villalba, A. Urquia, G. Shao, Implementations of the Tennessee Eastman Process in Modelica, IFAC PapersOnLine 51-2, 619/624 (2018)

[16] A. P. Mathur, N. O. Tippenhauer, SWaT: A Water Treatment Testbed for Research and Training on ICS Security, 2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater), 31/36 (2016)