SCADA Cybersecurity Awareness and Teaching with Hardware-In-The-Loop Platforms

Maxime Puys[1]*, Pierre-Henri Thevenon[1], Stéphane Mocanu[2], Mathieu Gallissot[1], and Camille Sivelle[1]

[1]Univ. Grenoble Alpes, CEA, LETI, DSYS, F-38000, Grenoble, France
Firstname.Name@cea.fr

[2]Laboratoire d'Informatique de Grenoble,
Univ. Grenoble Alpes, CNRS, Inria, Grenoble-INP, Grenoble, France
Stephane.Mocanu@imag.fr

Abstract

This article deals with SCADA cybersecurity awareness and teaching. We present two twin demonstrators based on the same technology: (i) WonderICS, an Advanced Persistent Threat (APT) demonstrator used for awareness demonstrations and (ii) G-ICS, a flexible lab used for students training and pentesting. Both are based on a common Hardware-In-the-Loop (HIL) technology which combines simulation, emulation and real devices to reproduce realistic industrial environments. Our solution simulates the physical process alongside real sensors and actuators, which are then connected with real industrial control devices using open-source electronic interface boards. Moreover, an innovative firmware emulation platform allows to run real devices' firmwares taken from vendors' websites without the need of the actual physical devices. After describing the architecture and implementation inner workings of our HIL platform, we explain what are the attack scenarios implemented on both platforms. These attacks scenarios allow us to conduct both demonstrations and teaching. We show how they are carried out and the feedback they get.

Keywords: SCADA, Cybersecurity, Simulation, Emulation, Testbeds

1 Introduction

Industrial Control Systems (ICS) are often used to monitor and control a physical process such as energy production and distribution, manufacturing or transport systems. They are often loosely referred as Supervisory Control And Data Acquisition (SCADA) systems. Historically deployed on dedicated and isolated proprietary networks, they used to be considered protected from networks threats by design. However, due to both their recent interconnection need and the discovery of the Stuxnet malware attacking Iranian plants in Natanz [27], they increasingly face cyberattacks caused by various intruders, including terrorists or enemy governments. From this point, cyber-incidents concerning ICS are regularly discovered and the number of attacks against industrial facilities is continuously growing. Indeed, the deployment of industry 4.0, connecting ICS to Internet exposed them to remote threats and interconnection with general IT make them vulnerable to general purpose malware such as ransomwares. As the frequency of such attacks is increasing, the security of ICS becomes a priority for governmental agencies and, consequently, security controls, is mandatory for some critical industrial activities (for instance

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 5, number: 4 (December), pp. 1-28

*Corresponding author: Maxime PUYS, CEA-Leti, 17 rue des Martyrs, 38054 GRENOBLE CEDEX 9, France. This work is supported by the French National Research Agency in the framework of the "Programme d'Investissement d'Avenir IRT Nanoelec" (ANR-10-AIRT-05) and "Investissements d'avenir IDEX UGA" (ANR-15-IDEX-02).

the operators of essential services as defined by the European Directive on Security of Network and Information Systems [18]). In particular, is recurrently mentioned one key security element, that directly motivated the present work: Awareness and Training. Indeed, a large proportion of cyber-incidents will exploit the lack of user awareness of computer and Internet risks. A statistical study released in 2019 by the German Federal Office for Information Security (BSI) teaches us that among the top ten threats used by attackers to penetrate ICS (see Table 1 from [10]), at least six can be addressed with an adequate user awareness and training in complement to added security equipments (Infiltration via Removable Media, Infection via Internet and Intranet, Human Error and Sabotage, Social Engineering and Phishing, Intrusion via Remote Access, Compromising of Smartphones in the Production Environment).

Infiltration via Removable Media and External Hardware
Malware Infection via Internet and Intranet
Human Error and Sabotage
Compromising of Extranet and Cloud Components
Social Engineering and Phishing
(D)Dos Attacks
Control Components Connected to the Internet
Intrusion via Remote Access
Technical Malfunctions and Force Majeure
Compromising of Smartphones in the Production Environment

Table 1: Top 10 threats against ICS in 2019 [10].

Proper awareness, training and teaching is generally the very first recommendation stated in industrial systems cybersecurity deployment guides and normative frameworks. See for instance the on-line training resources of ICS-CERT[1] or the French agency ANSSI ICS training guide[2]. It was especially proven that hands-on training and attack scenario demonstrators are the most effective tools [43, 6] for industrial cybersecurity awareness. Yet, the main challenge is to be able to train on realistic environments. It is clearly not viable to launch attacks on a real facility, due to the risks for human and environment safety, and for the possible financial losses. Moreover, reproducing a physical process of realistic size in university labs is very costly, challenging or can simply be impossible in case of a dangerous physical process (e.g., chemical factory, dam, nuclear plant, etc). For these reasons, several cyber-range initiatives emerged to allow building of proper research and training environments. The Idaho National Laboratories (INL) is certainly the most famous as it was featured in the Aurora project [19]. It reproduces a real electrical substation dedicated to smart-grid cybersecurity assessment and training [25]. Other initiatives come from industrial vendors such as CRIAB[3] cyber-range proposed by Boeing or the European Airbus[4]. In 2015, Holm [24] et al. proposed a survey on industrial control system testbeds. They distinguish between three categories of testbeds depending on the virtualization degree:

- *Simulation:* Consists in building a new platform that mimics the desired behavior.

[1]`https://us-cert.cisa.gov/ics/Training-Available-Through-ICS-CERT`

[2]https://www.ssi.gouv.fr/entreprise/guide/guide-pour-une-formation-sur-la-cybersecurite-des-systemes-industriels/

[3]`https://www.boeing.com/defense/cybersecurity-information-management/`

[4]`https://airbus-cyber-security.com/products-and-services/prevent/cyberrange/`

- *Virtualization:* Those approaches will virtualize/emulate the process and/or the control equipment to some degree (virtualized hardware, firmware, etc) in an heterogeneous fashion.
- *Hardware-In-The-Loop:* Consists in mixing simulation with real off-the-shelf industrial components.

Obviously, the more a platforms tends towards simulation, the most flexible and less difficult to maintain it gets, in comparison to a lab reproduction of a real industrial system. Such platforms also can be easily extended while this will not imply the modification of a real physical process. On the other hand, while they simulate and/or virtualize the process and the control devices, they cannot reproduce the behavior of real field devices like Programmable Logic Controllers (PLC) or or dedicated industrial equipments working a real-time and built on dedicated hardware. Therefore it is not possible to reproduce threats that exploit vulnerabilities of such real devices.

Contributions: This article is an extended version of a conference paper presented as part of the workshop on Education, Training and Awareness in Cybersecurity (ETACS'21) held in conjunction with the 16th International Conference on Availability, Reliability and Security (ARES'21) [38]. In the conference paper, we introduced two twin demonstrators based on the same technology: (i) an Advanced Persistent Threat (APT) demonstrator used for awareness training and (ii) a flexible lab used for students training and pentesting. Both are based on a common Hardware-In-the-Loop (HIL) technology which combines the advantages of virtualization and real cyber-ranges. We previously adopted a solution that only simulated the physical process alongside real sensors and actuators. These simulated components were connected with real industrial control devices in this HIL setup using open source electronic interface boards. In this extended version, we build upon our previous works to allow more flexibility and realism in the platform. We thus propose: (i) an innovative firmware emulation platform allowing to run real devices' firmwares taken from vendors' websites without the need of the actual physical devices, and (ii) two different kinds of Man-In-The-Middle attack on the WonderICS platform for awareness and teaching purpose. The first one is fully software based and targets network flow between SCADA and PLCs, while the second one relies on a custom hardware device and targets fieldbus protocols.

Outline: In Section 2, we briefly present the HIL system shared by both testbeds. Then, we present the APT training scenarios in Section 3, followed by the pen-testing, protocol fuzzing and reverse-engineering labs in Section 4. We conclude the paper in Section 5 with a description of the feedback we obtained from demonstrations and trainings and detail our future development plans.

2 Hardware-In-The-Loop Cyber-ranges (WonderICS, G-ICS)

In this section, we present the Hardware-In-the-Loop technology shared by both cyber-ranges. These testbeds allow the reproduction of a complete and realistic industrial control system. The global architecture of these platforms was first introduced in [30] and is represented in Figure 1. To correctly replicate the behavior of industrial systems, we aim to include all different layers of the Purdue model [47]. Thus, both platforms include real industrial devices and commercial SCADA software mixed with simulation for the physical process, its sensors and actuators. Because industrial interfaces from PLCs are not compatible with those of a computer, we developed interface electronic boards able to connect industrials devices to a computer running the simulations. We also intend to reproduce industrial systems of a size close to real industrial cases. Our symbolic target is one hundred industrial equipments and one thousand sensors and actuators. Thus, the interface board needs to scale up to the targeted number of inputs and

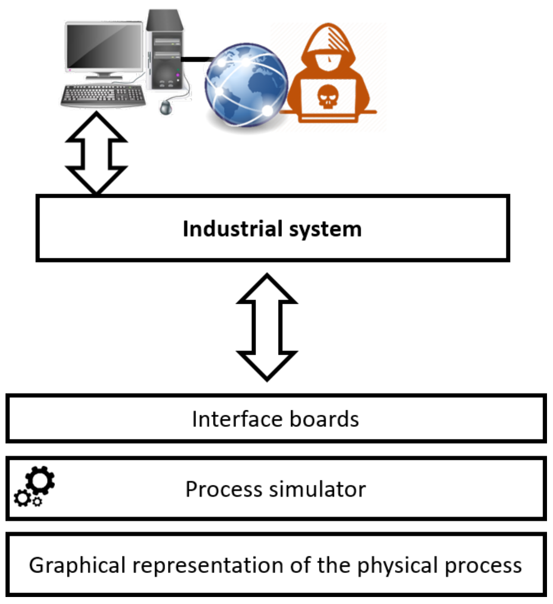


Figure 1: Overview of the presented platforms

outputs. Connected to these interface boards, the simulator of physical process is an open python-based software able to reproduce the behavior of different use-cases such as the management of hazardous gazes or the chemical process of Tennessee-Eastman [29]. An optional software is able to communicate with the simulator through a shared database to animate in real-time a graphical representation of the physical process. The rest of this section presents the common elements between the two testbeds: interface boards and the simulator.

2.1 Interface Boards

As explained in the above, our interface boards allow communication between the physical process simulator and off-the-shelf industrial hardware components from manufacturing and smart-grids applications fields. Figure 2 displays a synoptic of these boards, detailing their inputs and outputs.

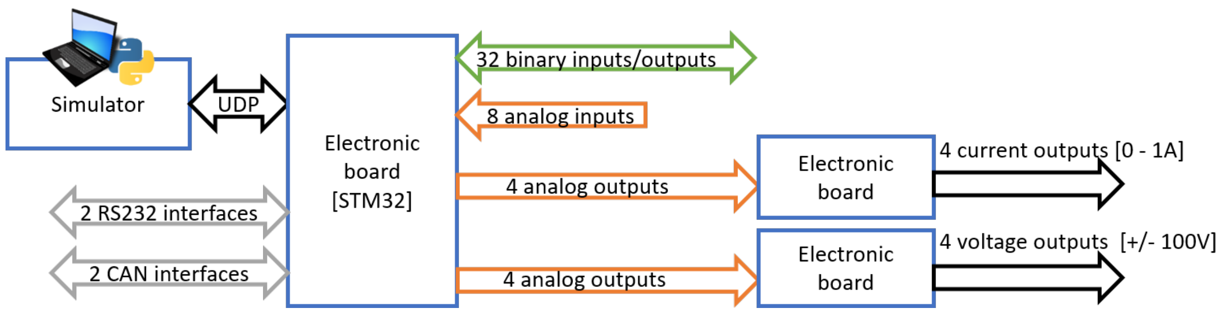


Figure 2: Interface board synoptic

These industrial devices can be Programmable Logic Controllers, Remote Terminal Units (RTU), Human Machine Interfaces (HMI) or embedded regulators and they interact with the physical process via a variety of I/O modules and interfaces. The electronic interface board integrates a quite large set of digital and analog inputs/outputs. As the electrical characteristics of the input and output signals vary a lot in industrial devices, we focused on the most common electrical signals (i.e. 0/24 V digital signals and -10/+10 V analogs). The current version of the interface board does not support specialized signals (pulse train or Pulse Width Modulation) or 4/20 mA current loops. The board also integrates industrial serial interfaces such as Modbus RTU and CAN. These legacy communication interfaces are still very used to drive actuators and sensors. Smart-grid hardware controllers are mostly Intelligent Electronic Devices (IED) preprogrammed with electrical protection functions. To measure current and voltage in an electric network, these devices use digital and specialized analog inputs. Two normalized current levels are mainly used: 0/1 A and 0/5 A and voltage sensors are issuing signals in the range 0/100 V. For instance, protection relays are a type of IED commonly used in power management that measures and analyses the three phases of the electrical signals to detect and localize electrical faults such as over-current and control and electrical circuit breaker that will isolate the faulty circuit. Two other electronic boards are dedicated for current and voltage conversions and allow to emulate electrical signals compatible with IED. These boards can be used to demonstrate real attacks on electric networks. Figure 3 presents the two use cases of the interface cards.

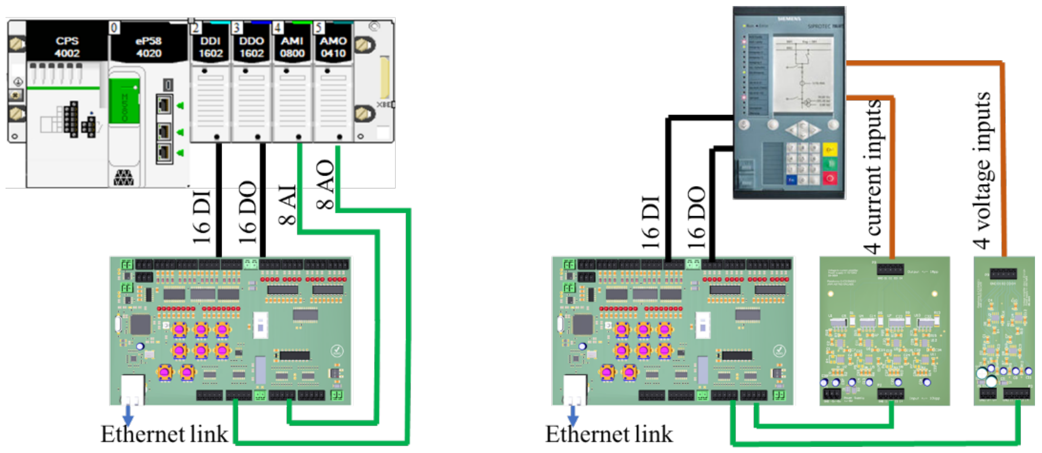


Figure 3: Interface board connection : PLC (left) and protection relay (right)

In terms of scalability, the use of an Ethernet network to connect interface boards to the simulator facilitates the integration of a new interface board allowing the implementation of real size industrial systems simulations. The number of connected interface board is only limited by the bandwidth of the Ethernet network. In order to made the boards accessible to university labs we try to keep the price as low as possible. There is an obvious trade-off between price and performance. A professional high-precision testing hardware for IED may cost as high as 40.000€ for a single measurement point signal generator. In order to keep the price affordable, we fixed a target objective cost of 400€ per interface card and another 400€ for a couple of power cards (current transformer and voltage transformer).

Controllers in manufacturing application are usually running with cycle times in the range of 10 to 250 ms. The notable exceptions are regulators whose sampling period may be around 1 ms even for simple applications. Smart-grid protection relays on the other hand are more demanding. Power measures in Europe are sampled at 4 KHz, i.e. a 250μs cycle time. Obviously, Ethernet communication is not compatible with the 250μs real-time period. Therefore, we developed two versions of the embedded program of the card: a simple one that copies value between card I/O and network for applications with low real-time demands and a second version which includes the generation of the triphasic measurement point signals on the card. Characteristics of the signals (amplitude, phase and frequency) are sent through the network. Regulation loops with fast sampling rates are not handled for the moment while embedding a process model on the card will be considered in the future.

2.2 Physical Process Simulation

The physical process simulator is a software reproducing an industrial physical process (e.g., a power plant). We designed this simulator with the goal to be easily adaptable to many use cases, thus being able to reproduce various processes. Moreover, we made it able to communicate with both generic IT and real off-the-shelf industrial components. For instance, the platforms we describe below include real industrial devices such as programmable automata. Thus, the simulator will be required to understand uncommon communication media such as electrical inputs or serial buses through interface boards presented in Section 2.1. It can also directly communicate with real components using industrial protocols over TCP/IP such as Modbus or OPC-UA. We propose a physical process simulator written in Python and based around the *simpy*[5] module. It is a process-based discrete-event simulation framework allowing us to model and schedule concurrently physical devices such as valves or coolers.

In our simulator, we chose to model such devices as independent tasks, scheduled on a shared clock. That is, each component is woken up at each – multiple of a – clock tick and can compute its outputs given its inputs. In other words, each component type (e.g., a valve) is modeled as a Python class implementing a special method called *process*. This method is the one called by *simpy* every tick, and processing inputs in order to compute outputs. This way, components can be reused in multiple physical process models as classes part of a library. In a main script, these classes are instantiated with their inputs specified as *lambda* functions (or Python properties). This construction allows the attributes of the modeled device to be physically the same variable as the one in the main script and not a copy passed by value. Then, the *process* method is registered in *simpy*'s scheduler alongside any other component or method needed by the model (e.g., input control or output monitoring). A working toy example is given for a valve in Listing 1 with the main script in Listing 2.

```
class Valve():
  _opened     = None
  _inputFlow  = None
  _outputFlow = None

  def __init__(self, opened, inputFlow):
    self._opened     = opened
    self._inputFlow  = inputFlow

  def process(self, env):
    while True:
      if self._opened():
        self._outputFlow = self._inputFlow()
      else:
        self._outputFlow = 0
      yield env.timeout(1)
```

Listing 1: A valve component

```
import  simpy

opened = True
inputFlow = 50
env = simpy.Environment()
///
def controlValve():
  global opened, inputFlow
  while True:
    # Code controlling the
    # valve during simulation
    yield env.timeout(1)

valve = Valve(
  env,
  opened=lambda: opened,
  inputFlow=lambda: inputFlow
)

env.process(controleValve())
env.process(valve.process(env))
env.run()
```

Listing 2: Main script

Multiple interfaces are built in the simulator to allow communications with other simulated or real components. First, we designed a UDP library handling a custom communication protocol with the interface board described in Section 2.1. This protocol allows to read digital (boolean) and analog (uint16) inputs and to respectively write outputs. Upon need, real industrial protocol servers can also be started

[5]https://simpy.readthedocs.io/

inside the simulator to allow real components or devices to communicate with it. Obviously, as the simulator runs on a computer, a matching physical interface will be needed to handle connections (e.g., an Ethernet port for TCP/IP or a FTDI cable for serial protocols). The simulator also provides a RESTful API allowing to synchronize inputs and outputs with databases for data persistency. Moreover, this REST API potentially allows interaction with other programs such as a visualization of the values of the simulated components or direct modifications of variables with a presentation tablet. The process simulator's code is available on demand on a GIT repository[6].

2.3 Firmware Emulation

Building realistic testbeds is a complicated task as they need to both: (i) be composed of real devices, softwares, etc, and (ii) scale to match real installations in terms of number of devices, variables, network flows, etc. Unsurprisingly, these two goals generally contradict with each other since real devices are needed to allow realistic training. Yet such devices are pricy, require knowledge to be electrically connected with each other and require care to not be destroyed by attacks. On the other hand, scalability requires many of these devices to be installed which leads to an impossible equation for many universities or companies. To tackle this contradiction, we propose an emulation platform called WonderCloud, allowing to execute real devices' firmwares without the need of the physical devices themselves. Benefits are manyfold:

1. *Realism:* Firmwares run are taken from vendors' website, allowing for proper behavior. Hardware is emulated with respect to original architecture and peripheral to match as close as possible with real device.

2. *Scalability:* Many devices can be emulated with limited real hardware (one server can run many devices simultaneously).

3. *Repeatability:* Emulated devices can be damaged by invasive attacks and faithfully recreated without any cost.

4. *Deceptive Security:* Emulated devices can safely be run as honeypots to assess how real attackers would attack them in real life.

We first introduced the WonderCloud framework in [22]. It is based on the emulation framework Firmadyne [11] that we extend to support peripherals and architectures. It allows to faithfully reproduce the software contained in the device, its true behavior and its interfaces. We first recall the inner-workings of WonderCloud, then detail its integration in HIL platforms with other components described above. To our knowledge, very few solutions allow to emulate IoT objects only based on their firmware. One can mention FirmPin [14], a software suite developed by DeepBitsTechnology since 2018 and integrating the TriforceAFL fuzzing tool, the Firmadyne emulation platform and the dynamic code analysis tool DECAF. We can also mention the Avatar tool [49] combining emulation with real hardware by forwarding I/O accesses from the emulator to the hardware. In addition, several frameworks have been developed to automatically analyze the security of consumer or industrial IoT objects. These include Expliot [26] in 2018, Hercules [33], and P-SCAN [28]. These tools allow the identification of vulnerabilities in IoT or industrial IoT objects by supporting many protocols. They allow locating objects on a network, listening to communications, as well as send or replay malicious messages. Similarly to the state of the art, WonderCloud relies on model based on: (i) firmware acquisition, often based on the binwalk tool binwalk tool [39]; (ii) preparation of the firmware for emulation, using an appropriate kernel; and (iii)

[6]`https://gforge.inria.fr/projects/eastman/`

the emulation of the firmware via the multi-architecture emulator qemu [7]. We improve the existing Firmadyne framework with the objective to increase success rate of the emulation by adding support for several interfaces and hardware architectures related to industrial systems, but also to add steps to bypass certain protections.

Firmware Extraction: The first step to emulate a device is the recovery of its embedded software. Two main solutions main solutions to retrieve this data:

- The first solution consists in browsing the web site of vendors in order to find firmwares. Having the firmware associated to the devices available online is a common practice, as it allows users to update their products. Many scrapers allow to automatically browse vendors' website such as FKIE [21].
- The second solution consists in physically extracting the software from a device using the programming interface present on the electronic board. This interface, commonly called JTAG, is generally used to program and debug digital components such as processors, micro-controllers or FPGAs, but can also be used to extract the binary file from the program memory. Open source tools such as *openocd* can be used to connect to the processor/ micro-controller to extract the binary file from its memory.

Emulation: Firmware emulation must allow the firmware code to be executed in its entirety. One of the first steps is the discovery of the characteristics of the hardware on which this firmware must be able to run (architecture of the microprocessor, instruction set, system peripherals, etc). We can either study the hardware if it is available, or the binwalk tool will help us to identify the instruction set of the various binaries. Once the physical environment of the firmware is known, it is necessary to emulate it via a compatible emulator. Qemu [7] offers a number of microprocessor architectures, allowing the execution of code for various common architectures such as ARM, MIPS, PowerPC, Sh4, etc. An alternative project project, named xPack QEMU ARM [48], also allows the simulation of lighter architectures that are frequently found on connected objects. To facilitate the emulation, we opt for the use of a precompiled Linux kernel for the architecture rather than trying to emulate the original kernel. This approach is used in the state of the art and allows a better success rate of the firmware boot on Linux systems. It also allows to introduce new drivers or to rename some of them in order to to allow some user-space applications to run. Thus, we have extended the kernel proposed by Firmadyne in order to support hardware architectures such as PowerPC, but also to be able to add modules or drivers emulating peripherals such as WiFi interfaces (module MAC80211_hwsim module) or video devices (v4l2loopback). In our approach, emulation is considered sufficient when the emulated firmware allows to execute the functionalities for which it is intended. For example, in the case of a video surveillance camera, we expect a video stream to be present in the system and that the human/machine interfaces or APIs to be functional.

Instantiation and Network Composition: The tools of the community mentioned in the state of the art have for finality the research of vulnerabilities within the firmware. In our case, we wish to use emulated systems in order to simulate realistic environments. We thus rely on the OpenStack [1] framework to instantiate a virtual product and manage its network through bridge connections and mounted physical interfaces (e.g., serial TTY). As an example, we emulated an industrial building-automation gateway composed of several specific wired interfaces (Modbus/TCP, KNX, RS485 (Modbus/RTU), RS232) and hosting a web server for the control and supervision. The firmware of this gateway is publicly available on the manufacturer's website. The downloaded image includes the entire file system that could be rebuilt and formatted as a virtual hard disk. Coupled with a generic Linux kernel, the system is run in the xPack QEMU emulator for ARM. The first emulation attempts generated some errors in the target system that were corrected by iteration: adding a network card, mounting required TTY interfaces, changing some *sysctl* parameters, etc. The emulated instance was first compared to a physical instance to validate that the whole system was operational. We were able to verify that the emulated system could connect to a physical MODBUS bus and execute all its services.

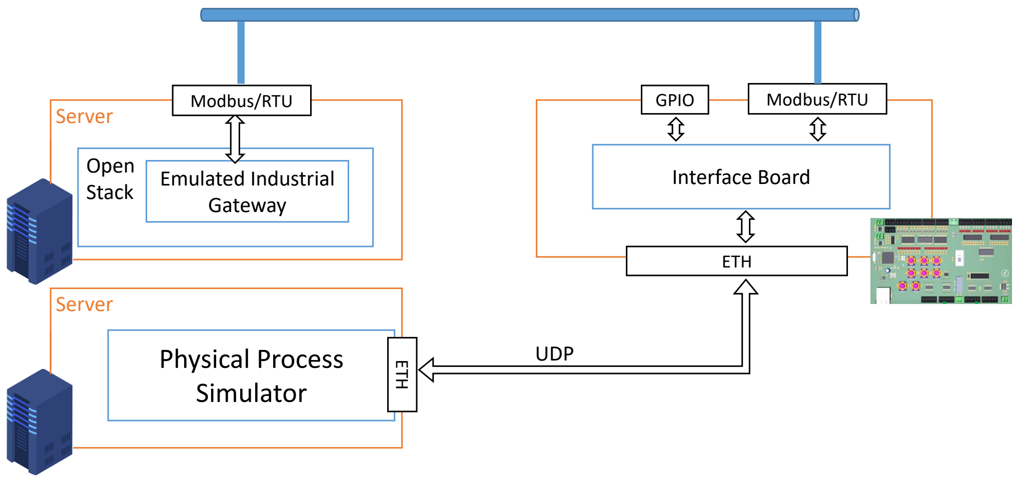


Figure 4: Integration of WonderCloud emulation in HIL platforms

Figure 4 depicts a simple use-case where the emulated industrial gateway is used to control a smart-building temperature and humidity management system. It interacts through a Modbus/RTU over RS232 connection with an interface board presented in Section 2.1 (which interacts with the physical process simulator presented in Section 2.2). The simulator simulates temperature and humidity values changing over time, which are accessed by the interface board. Through its Modbus/RTU connection with the interface board, the emulated industrial gateway can retrieve temperature and humidity values for its monitoring feature. The connection between the industrial gateway with a real PLC over Modbus/RTU, for it to send commands to real or simulated actuators is currently left as a future work.

3 Awareness Training and APT Demonstrators

WonderICS is a hardware-software co-simulation environment mainly dedicated to raise awareness of cybersecurity issues in industrial control systems and to experiment innovative security solutions. This platform integrates the simulator presented in Section 2.2 that emulates physical processes based on three different use cases: hazardous gases management, hydroelectric power plant and the chemical process of Tennessee-Eastman. A complete infrastructure including some virtual machines and a set of dedicated tools have been developed to attack the industrial control system in different ways (phishing mails, corrupted USB key, hardware trojan, etc). This section will describe the infrastructure allowing to attack the industrial system and some attacks developed especially for the different use cases. Figure 5 shows a view from the WonderICS platform with, from left to right, we have a 3D view of the physical process, the SCADA software, off-the-shelf industrial devices, the computer of an operator targeted by attacks, and finally the attacker's computer. It is worth noting that the 3D view of the process represents the real state of the simulated process (and would be replaced by the actual physical process itself in a real factory); while the SCADA only show the local vision of the process obtain by communicating with captors and actuators.

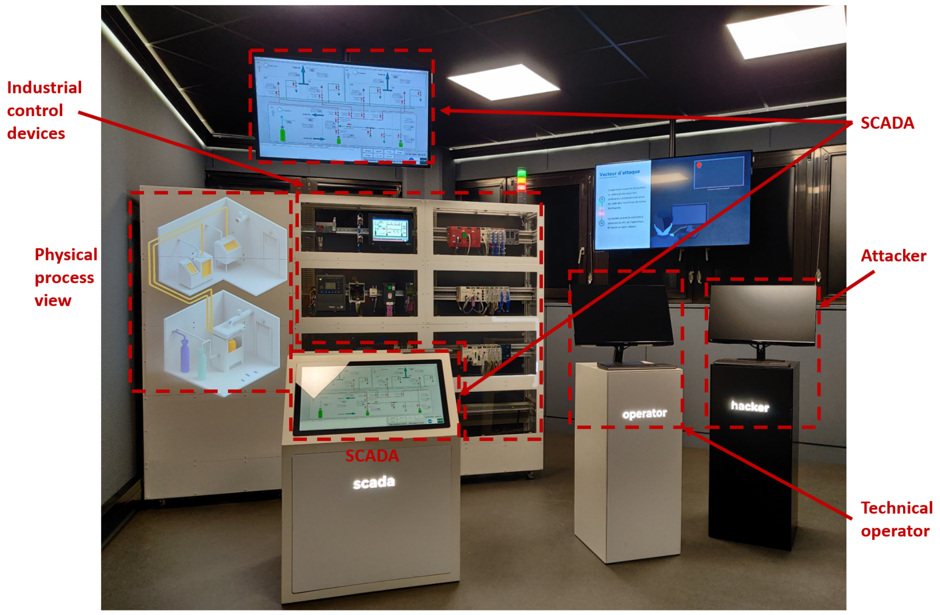


Figure 5: WonderICS platform

3.1 Infrastructure of the WonderICS platform

To create realistic attacks that we can play on demand (and easily recover from), we have developed a complete network of virtual machines. These virtual machines allow us to implement complex attack scenarios with potentially high impact on the system but high isolation from hosts and seamless reset process. Figure 6 describes this virtual infrastructure.

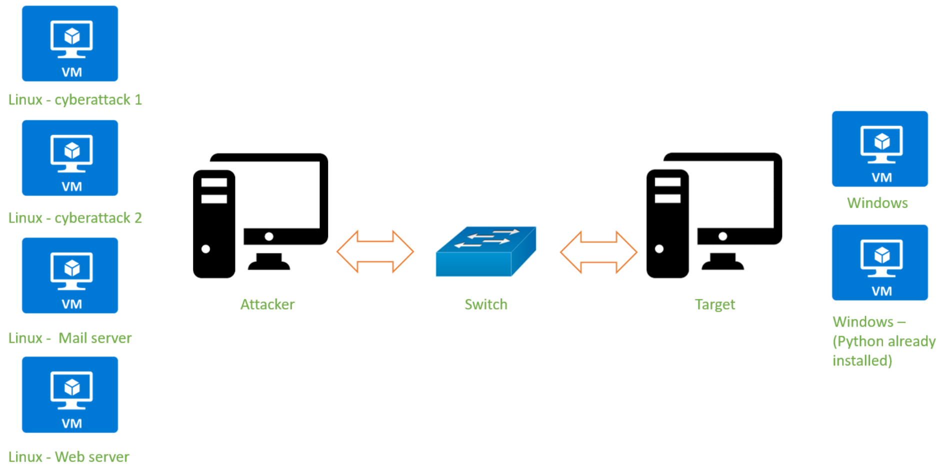


Figure 6: Network of virtual machines

Two physical computers host the different virtual machines. The attacker's computer is both used to launch attacks and to host attacker's downloadable resources (local mail server, website, etc). The following virtual machines are implemented on the attacker's computer:

- Cyberattack 1: This virtual machine contains a Kali Linux operating system. The first cyberattack, explained in Section 3.2, is based on a Rubber Ducky key that emulates a keyboard. The tools needed for the first cyberattack have been added in this virtual machine. For example, the duckencoder program used to encode scripts on the Rubber Ducky key.

- Cyberattack 2: This machine also contains a Kali Linux OS but the APT attack materials are different. The second cyberattack, described in Section 3.2 integrates a RAT (Remote Access Trojan) server and mail client such as Mozilla Thunderbird to send some malicious emails as part of a spear phishing campaign.
- Mail server: This machine represents the local mail server of the industrial facility, allowing the attacker to send emails to the operator. The mail server is composed by a Postfix server for sending and a Dovecot IMAP server for receiving.
- Web server: A Nginx Web server is launched automatically at the start of this VM and allows to retrieve different programs part of attacks (charges, cryptolocker for example).
- Windows 7: This machine allows the use Window's Remote Desktop Control module that can give a complete access to the target's computer.

The majority of attacks target the computer of a technical operator of the industrial system because his computer has a direct access to the industrial network. The operating system used by the target is mainly Windows so we have chosen virtual machines running on Windows 7 operating system (which remains rather common in industrial facilities). These machines integrate Microsoft Office 2010 and Mozilla Thunderbird for spear phishing attacks.

3.2 Presentation of Attacks

As explained in Section 3.1, we implemented two complete cyberattacks reproducing the behavior of Advanced Persistent Threats (APT). Thus, they include several phases going from open-source intelligence to complete exploitation and include persistence. Global attack paths are represented in Figure 7. Both attacks start with a reconnaissance phase (1), then the delivery phase (2) allows the attacker to gain access to the vulnerable machine. Technically, at this step, both attacks allow for a control command server (3. C&C) to perform remote operation in order to gain privilege and/or information gathering. However, as detailed later, nature of Cyberattack 1 requires fast exploitation and would usually skip this phase. Both attacks then perform the actual charge delivery (4), harming the system and potentially the industrial process behind (5). The rest of the section details both cyberattacks (depicted in green and blue on Figure 7), then Section 3.3 will detail how these attacks can impact the industrial process controlled by the system.

Presentation of Cyberattack 1: Cyberattack 1 involves the use of a Rubber Ducky[7]. The attack is based on the lack of precaution of an employee who, having found what looks like a USB key, inserts it into a USB port on his workstation. The Rubber Ducky is then considered as a keyboard, and programmed to perform a large number of actions at high speed. As the USB stick acts as a keyboard, all actions are visible on the operator's screen, leading to an immediate disclosure of the attack and requiring a quick charge delivery and exploitation. As this attack is completely remote and does not let much time for the attacker to perform manual operations, reconnaissance phase will mainly resume to scanning and information gathering to find vulnerabilities to exploit in the OS or applications. It then mainly relies on Python to execute actions and is able to disable other existing keyboards to prevent the operator stopping the attack. It will also be able to fetch a malware from a remote server if needed. As the charge is

[7]`https://shop.hak5.org/products/usb-rubber-ducky-deluxe`

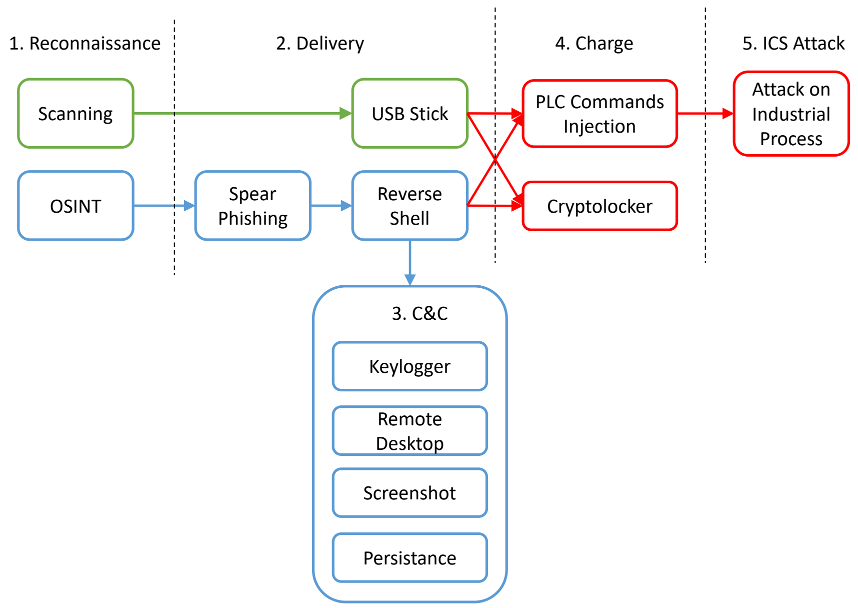


Figure 7: Attacks Steps

launched without any control from the attacker, it is usually more realistic to conclude this attack with automatically launching a cryptolocker. An overview of cyberattack 1 is depicted in Figure 8. In an awareness demonstration, this cyberattack is very visual and impressive for decision-makers as a lot of actions are quickly performed by the USB stick and they loose the control of the target system.

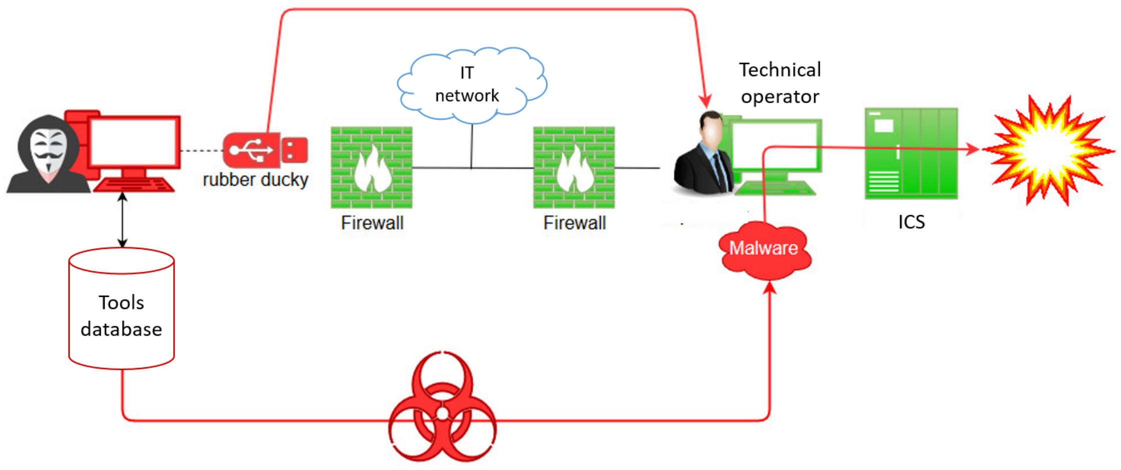


Figure 8: Overview of attack 1

Presentation of Cyberattack 2: Cyberattack 2 involves sending a fake email containing a Word document with a rogue macro that sets up a Remote Access Trojan (RAT) on the Target PC. Once this RAT is set up, the attacker can initiate the download of charges from a controlled web server to the Target machine, and launch their execution. This attack has a lot from Advanced Persistent Threats and starts with Open-Source Intelligence (OSINT) in order to gather information on a vulnerable employee (his work email, his position, his hobbies, etc). It then relies on spear phishing with an email specifically destined to him and designed around his involvement in the associative world to maximize his chances to open the malicious office document attached. This document starts a RAT in the background that will automatically connect the a Control Command server (C&C) setup by the attacker and allow him to use various commands on the target system. These commands include launching a keylogger, enabling Windows remote desktop, taking a screenshot from the screen or making the malware persistent in case of reboot via various vulnerability exploits. Once the attacker estimates that he gathered enough knowledge on the system, he can download and run various charges in order to affect the industrial system behind the target system. These industrial focused charges are described in Section 3.3. An overview of cyberattack 2 is depicted in Figure 9.

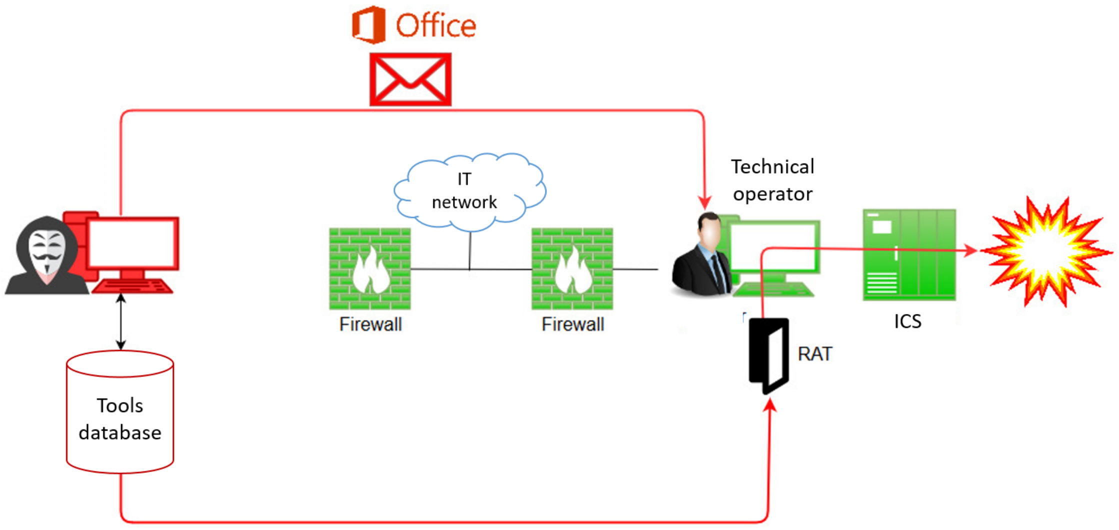


Figure 9: Overview of Attack 2

3.3 Modbus Injection, Man-In-The-Middle Attacks and Impacts on the Physical Process

As soon as the attacker takes control of the operator's computer, he obtains direct access to industrial controllers (PLCs) and the SCADA. In our configuration, the protocol Modbus TCP is used between the SCADA, the targeted computer and the Programmable Logic Controller (PLC). This protocol is legacy but is still used in most of industrial systems. As in the majority of industrial protocols, there are no builtin security and a lot of vulnerabilities can be used to eavesdrop or inject false commands. Within the WonderICS platform, we consider two main types of attacks: (i) injection attacks, and (ii) man-in-the-middle attacks (hardware and software).

3.3.1 Injection attacks

Injection attacks usually consists for an attacker in hijacking an already open and insecure session, and send malicious frames in between legitimate ones. However, when the targeted protocol allows multiple sessions in parallel and does not provide any authentication measures, the attacker is simply free to open a session with the target server using a perfectly legitimate client. This is typically the case of the Modbus protocol family (TCP or RTU), making such attacks really easy to carry out on industrial systems. Yet, depending on the industrial process, they can be easily noticeable. Also, the attacker will potentially have to compete with legitimate clients periodically sending requests, and will require some luck for his message to be fulfilled by the server instead of others.

Depending on the use case chosen in the WonderICS platform (hazardous gas management, hydroelectric plant, chemical process), we thus inject Modbus frames to change the status of a sensor or send a new order to an actuator. Targeted devices of this attack are mainly PLCs but the final target can also be a simulated device (valve, sensor, etc) or a real physical actuator connected to the controllers (protection relay, breaker, motor management, etc). For instance, one of our attack vectors sends a malicious Modbus/TCP frame to write a fake value on a register of the controller. This address encodes the value of the intensity given by an electrical protection relay. By sending a very high value of intensity, the controller detects an overcurrent and opens a real industrial breaker connected to the platform in order to cut the current in the system. Code for this attack is presented in Listing 3. It relies on the *pymodbus* Python package to send Modbus frames.

```python
from pymodbus.client.sync import ModbusTcpClient
import time

PLC_ADDR = "10.1.2.3"
MODBUS_PORT = 502
REG_BREAKER = 4000
REG_PLAYLOAD = 1024

client = ModbusTcpClient(PLC_ADDR, MODBUS_PORT)

while True:
    client.write_register(REG_BREAKER, REG_PAYLOAD)
    time.sleep(0.001)
```

Listing 3: Code for injection attack

3.3.2 Software Man-In-The-Middle Attack

The objective of this attack is to interface between the computer running the supervision software and the PLCs to demonstrate the feasibility of modifying the exchanged traffic in real time. The SCADA visualization software periodically performs read requests on the PLCs' variables in order to retrieve their most updated values. These values are then analyzed by the SCADA software of the WonderICS platform and represented on the synoptic view. Thus, an attacker can modify the commands sent by operators to the PLCs or distort the information feedback to hide a dangerous state. The computer running the supervision software and the PLCs communicate live via an industry proprietary protocol based on Modbus/TCP. This proprietary protocol was studied and analyzed by the company Digital Bond in 2016 [36] and then by the consulting cabinet Wavestone in 2017 [44]. It has no security features in terms of cryptographic authentication and can therefore be the target of Man-In-The-Middle attacks.

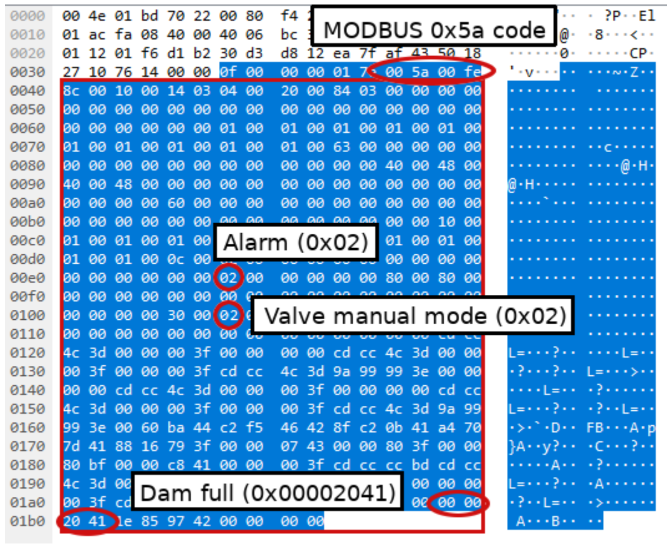
Attack Scenario: We have chosen to perform this attack on an hydroelectric plant scenario. A dam feeds water into a power-plant working with an alternator and the dam level is regulated by a valve which opens automatically when the dam level is too high and closes when the level becomes low enough. This valve can also be switched to "manual" mode to be controlled via the SCADA view. The objective is to overflow the dam by:

1. Switching the dam level control valve to manual mode;
2. Forcing the dam level control valve to a closed state (preventing water from flowing through);

3. Falsifying the feedback to the SCADA with:
    (a) The dam level displaying a normal level,
    (b) The status of the dam level control valve displaying automatic mode,
    (c) By blocking alarm feedback.

In this way, the safety functions of the PLC allowing the regulation of the water level by the valve are deactivated by switching to manual mode. Then, via the Man-In-The-Middle attack, the supervision software wrongly estimates a normal state of the dam while the real state, represented by a 3D view connected directly to the process simulator, will show an overflowing dam.

Man-In-The-Middle attack's unfolding: The SCADA makes requests to read variables (Modbus function code 0x5A followed by proprietary function code 0x0022) on the PLCs. All the variables declared in the PLC programs (.stu files) are thus retrieved at once. We show an example of response to these requests in Figure 10. The function code 0x5a appears at the beginning of the frame, followed by the return code 0x00fe indicating an "OK" return code. The rest of the message consists of the values of the variables of the automaton. The large number of zeros and ones comes from the fact that most of the variables are Boolean. By varying the parameters of the process simulator, we were able to isolate the bytes corresponding to the level of the dam, as well as the presence of an alarm. This capture was done with the dam at its maximum level (corresponding to the value 0x00002041 in the frame) and with an alarm raised on the water level (corresponding to the value 0x02 in the frame).



```
0000  00 4e 01 bd 70 22 00 80  f4                        ·  ?P··El
0010  01 ac fa 08 40 00 40 06  bc                       @·  ·8···<··
0020  01 12 01 f6 d1 b2 30 d3  d8 12 ea 7f af 43 50 18  ······0·  ·····CP·
0030  27 10 76 14 00 00 0f 00  00 00 01 7  00 5a 00 fe  '·v·····  ····~·Z··
0040  8c 00 10 00 14 03 04 00  20 00 84 03 00 00 00 00  ········  ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ········  ········
0060  00 00 00 00 00 00 01 00  01 00 01 00 01 00 01 00  ········  ········
0070  01 00 01 00 01 00 01 00  01 00 63 00 00 00 00 00  ········  ··c·····
0080  00 00 00 00 00 00 00 00  00 00 00 00 40 00 48 00  ········  ····@·H·
0090  40 00 48 00 00 00 00 00  00 00 00 00 00 00 00 00  @·H·····  ········
00a0  00 00 00 00 60 00 00 00  00 00 00 00 00 00 00 00  ····`···  ········
00b0  00 00 00 00 00 00                    00 00 10 00  ········  ········
00c0  01 00 01 00 01 00                    01 00 01 00  ········  ········
00d0  01 00 01 00 0c 00                    00 00 00 00  ········  ········
00e0  00 00 00 00 00 00 02 00  00 00 00 00 80 00 80 00  ········  ········
00f0  00 00 00 00 00 00 00                                        ········
0100  00 00 00 00 30 00 02                                        ········
0110  00 00 00 00 00 00 00                                        ········
0120  4c 3d 00 00 00 3f 00 00  00 00 cd cc 4c 3d 00 00  L=···?··  ····L=··
0130  00 3f 00 00 00 3f cd cc  4c 3d 9a 99 99 3e 00 00  ·?···?··  L=···>··
0140  00 00 cd cc 4c 3d 00 00  00 3f 00 00 00 00 cd cc  ····L=··  ·?······
0150  4c 3d 00 00 00 3f 00 00  00 3f cd cc 4c 3d 9a 99  L=···?··  ·?··L=··
0160  99 3e 00 60 ba 44 c2 f5  46 42 8f c2 0b 41 a4 70  ·>·`·D··  FB···A·p
0170  7d 41 88 16 79 3f 00 00  07 43 00 00 80 3f 00 00  }A··y?··  ·C···?··
0180  80 bf 00 00 c8 41 00 00  00 3f cd cc cc bd cd cc  ·····A··  ·?······
0190  4c 3d 00                             00 00 00  L=···?··  ·A······
01a0  00 3f cd                             00 00 00  ·?··L=··  ·>······
01b0  20 41 1e 85 97 42 00 00  00 00                     A···B··  ··
```

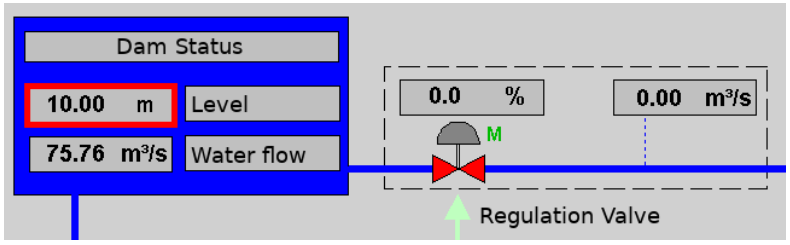
Figure 10: Protocol frame before Man-In-The-Middle attack

From the attacker VM presented in Section 3.1, we carry out an ARP poisoning attack, targeting the computer running the supervision software on the one hand and the PLCs on the other hand in order to modify the correspondence table between the IP and MAC addresses stored on each of them. Thus, the MAC addresses used in the Ethernet layer of each of the packets sent are replaced by that of the attacker. This attack is performed using the Ettercap software, present in the Kali-Linux distribution. It requires knowledge of the IP addresses of the targets and a physical connection to the network (Ethernet cable or authenticated WiFi). All network traffic between the supervision software and the PLCs now passes through the attacker. It is then possible to specify to Ettercap filtering rules describing the behavior of the attacker according to the type of packet received. We have created the Ettercap filter shown in Listing 4 in order to modify on the fly the bytes of the responses of the PLC corresponding to the value of the level of the barrage and the presence of an alarm. It is then possible to overwrite the value of each byte, by specifying its index in the frame. Therefore, we force the bytes 0x00 to mask the presence of an alarm and show automatic mode for the valve (instead of 0x02 when an alarm is present) and 0xd44dd640 to indicate a normal level (instead of 0x002140 for the full barrier).

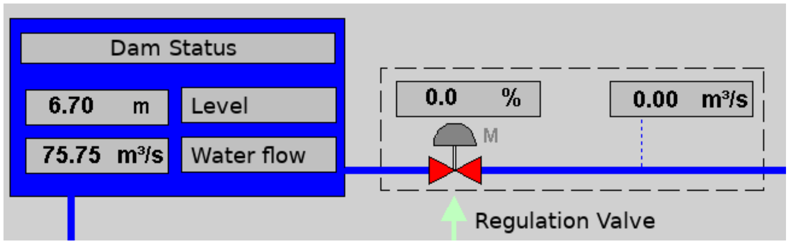
```
if (ip.src == '10.1.2.3') { // Detects responses from PLC
    DATA.data + 0x00a0 = 0x00; // No alarm
    DATA.data + 0x00c0 = 0x00; // Automatic mode
    DATA.data + 0x0178 = 0xd4; // Normal level 1/4
    DATA.data + 0x0179 = 0x4d; // Normal level 2/4
    DATA.data + 0x017a = 0xd6; // Normal level 3/4
    DATA.data + 0x017b = 0x40; // Normal level 4/4
}
```

Listing 4: Ettercap filter

It is then possible to force the manual of mode the valve and its closed state by a Modbus frame injection attack as shown in Section 3.3.1. The effect of the Man-In-The-Middle attack can be visualized on the SCADA. After forcing the state of the valve, but before modifying the variables' values in the Modbus frames, the level is maxed at 10m, there is an alert raised on the level (red frame around) and the valve appears closed in manual mode (green "M" next to it), as shown in Figure 11a. Obviously, in a real attack, injection of malicious commands and Man-In-The-Middle attack would be carried simultaneously to not disclose the attack. We only run them one by one here to showcase the process.



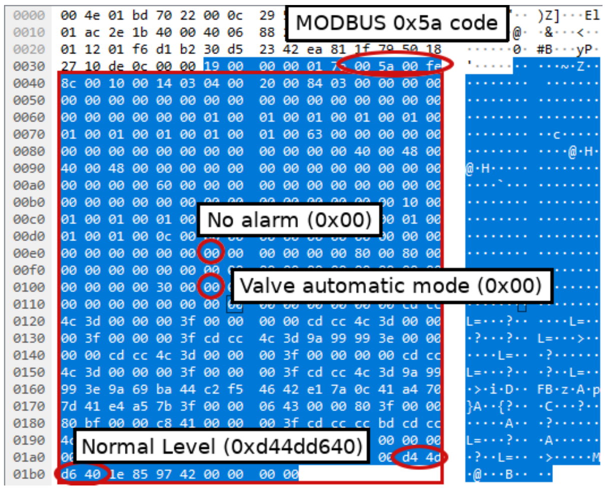
(a) Dam before Man-In-The-Middle attack



(b) Dam after Man-In-The-Middle attack

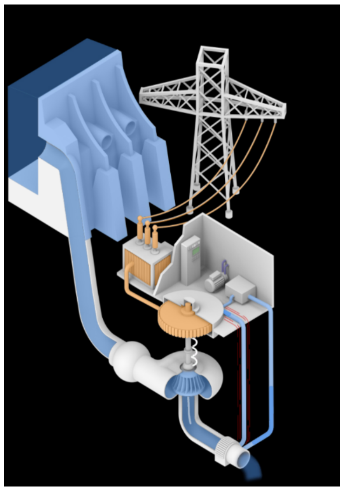
Figure 11: SCADA before (a) and after (b) Man-In-The-Middle attack

After the Man-In-The-Middle attack, the dam level only appears at 6.70m (corresponding to the value 0xd44dd640 forced by the Ettercap filter) in Figure 12a. No alert is sent to the supervision anymore and the valve now appears in automatic mode ("M" symbol is grayed), as pictured in Figure 11b. However, we can see that the attack is effective by looking at the real view of the process, connected directly to the process simulator. Indeed, Figure 12b clearly shows the dam overflowing.



```
0000  00 4e 01 bd 70 22 00 0c  29 [illegible]            )Z]···El
0010  01 ac 2e 1b 40 00 40 06  88 [illegible]          @·  ·&···<··
0020  01 12 01 f6 d1 b2 30 d5  23 42 ea 81 1f 79 50 18  ······0· #B···yP·
0030  27 10 de 0c 00 00 19 00  00 00 01 7? 00 5a 00 fe  '······· ···~·Z··
0040  8c 00 10 00 14 03 04 00  20 00 84 03 00 00 00 00  ········ ········
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ········ ········
0060  00 00 00 00 00 00 01 00  01 00 01 00 01 00 01 00  ········ ········
0070  01 00 01 00 01 00 01 00  01 00 63 00 00 00 00 00  ········ ··c·····
0080  00 00 00 00 00 00 00 00  00 00 00 00 40 00 48 00  ········ ····@·H·
0090  40 00 48 00 00 00 00 00  00 00 00 00 00 00 00 00  @·H····· ········
00a0  00 00 00 00 60 00 00 00  00 00 00 00 00 00 00 00  ····`··· ········
00b0  00 00 00 00 00 00 [illegible]    00 10 00  ········ ········
00c0  01 00 01 00 01 00 [illegible]    00 01 00  ········ ········
00d0  01 00 01 00 0c 00 [illegible]    00 00 00  ········ ········
00e0  00 00 00 00 00 00 00 00  00 00 00 00 80 00 80 00  ········ ········
00f0  00 00 00 00 00 00 00 [illegible]               ········ ········
0100  00 00 00 00 30 00 00 [illegible]               ········ ········
0110  00 00 00 00 00 00 00 00  [illegible]               ········ ········
0120  4c 3d 00 00 00 3f 00 00  00 00 cd cc 4c 3d 00 00  L=···?·· ····L=··
0130  00 3f 00 00 00 3f cd cc  4c 3d 9a 99 99 3e 00 00  ·?···?·· L=···>··
0140  00 00 cd cc 4c 3d 00 00  00 3f 00 00 00 00 cd cc  ····L=·· ·?······
0150  4c 3d 00 00 00 3f 00 00  00 3f cd cc 4c 3d 9a 99  L=···?·· ·?··L=··
0160  99 3e 9a 69 ba 44 c2 f5  46 42 e1 7a 0c 41 a4 70  ·>·i·D·· FB·z·A·p
0170  7d 41 e4 a5 7b 3f 00 00  06 43 00 00 80 3f 00 00  }A··{?·· ·C···?··
0180  80 bf 00 00 c8 41 00 00  00 3f cd cc cc bd cd cc  ·····A·· ·?······
0190  4[illegible]                      00 00 00  L=···?·· ·A······
01a0  0[illegible]                   00 d4 4d  ·?··L=·· ·>·····M
01b0  d6 40 1e 85 97 42 00 00  00 00                    ·@···B·· ··
```

(a) Protocol frame after Man-In-The-Middle attack



(b) Real state of the dam

Figure 12: Effects of the Man-In-The-Middle attack

This attack shows us that it is possible to distort the synoptic view of SCADA software by launching a Man-In-The-Middle attack on the proprietary Modbus/TCP based protocol. This is not surprising since this protocol was not designed to include security features allowing authentication [16] (yet similar results have been shown on secure protocols such as OPC-UA [37, 17]). However, although potentially devastating, this attack is very complex to implement because it requires an advanced knowledge of the industrial process in order to force the right variables via Modbus write requests, as well as the right values to modify in the Ettercap script.

3.3.3 Hardware Man-In-The-Middle Attack

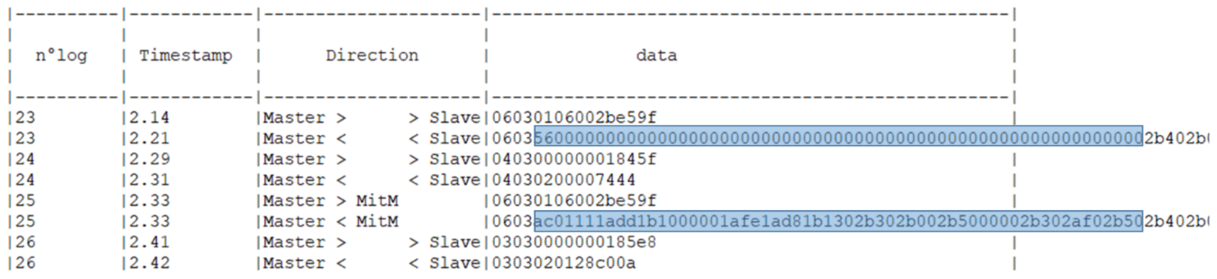
In this specific scenario, an attacker bribes an employee of the target company who has physical access to the industrial system to install a hardware attack device on the industrial network. This device can remotely spy on the industrial network and inject malicious frames to compromise the system. We developed a hardware device to conduct such Man-in-the-Middle (MitM) attacks in Modbus/RTU networks. To facilitate the deployment of this attack, we used a Raspberry Pi electronic board positioned between the master PLC and its slaves. The WonderICS platform has two Modbus/RTU networks. The first one allows exchanging data between the PLCs while the second one interfaces with an industrial circuit breaker, a protection relay and a wireless control unit. For the MitM attack, the choice was made on the second network in order to be as close as possible to the physical process and to attack the trio: protection relay - PLC- circuit breaker. In the hydro-electrical power-plant scenario, the protection relay can detect an electrical anomaly on the three-phase current and alert the PLC, which can then trip the circuit breaker. The developments of this malicious device makes it possible to affect the operation of these devices by corrupting the frames exchanged between the PLC and the protection relay to make it appear that an overcurrent has been detected on the electrical network. A Python script deployed on the Raspberry Pi allows to configure and launch the whole attack but also to spy on the frames exchanged between the PLC and its slaves. Several attack modes have been developed:

- DoS: all the master's requests are blocked so that it can no longer communicate with the slaves.
- ACK : the MitM device answers to the master instead of all the slaves. In the case of a READ request, the content of the frame received by the master will contain only zeros. During a write

request (WRITE), the frame received will indicate that the writing has taken place.

- MitM : this mode allows to target specific frames and to corrupt the data sent back by the slaves. When the master sends a frame whose slave ID and accessed memory address match the configuration, the MITM device does not transmit the request and injects its own data instead of the slave.

All of these attacks are configurable in order to select the function codes and slave IDs that will be targeted by the attack. The Raspberry Pi board has been integrated into the WonderICS platform's Modbus/RTU network in order to demonstrate real attacks. The attacker currently connects to the Raspberry Pi to launch the attack from the attacker's computer through an SSH session, but more realistic alternatives could be considered (BLE, GSM, etc).



```
|----------|------------|---------------------|-----------------------------------------------|
|          |            |                     |                                               |
|  n°log   | Timestamp  |      Direction      |                     data                      |
|          |            |                     |                                               |
|----------|------------|---------------------|-----------------------------------------------|
|23        |2.14        |Master >      > Slave|06030106002be59f                               |
|23        |2.21        |Master <      < Slave|0603560000000000000000000000000000000000000000000000000000000002b402b
|24        |2.29        |Master >      > Slave|040300000001845f                               |
|24        |2.31        |Master <      < Slave|04030200007444                                 |
|25        |2.33        |Master > MitM        |06030106002be59f                               |
|25        |2.33        |Master < MitM        |0603ac01111add1b1000001afe1ad81b1302b302b002b5000002b302af02b502b402b
|26        |2.41        |Master >      > Slave|030300000000185e8                              |
|26        |2.42        |Master <      < Slave|0303020128c00a                                 |
```

Figure 13: Protocol frame before and after Man-In-The-Middle attack

The sample logs presented in Figure 13 present an example of a MitM attack performed. While the logs of transaction 23 display a standard exchange between the PLC and the protection relay, the logs of transaction 25 show that the MitM device sends back a corrupted frame that will trigger the circuit breaker (frames contents are different on the figure).

3.4 An example of awareness training in WonderICS

As soon as they enter in the WonderICS platform, visitors are immediately immersed in the context of industrial cybersecurity due to the presence of real off-the-shelf devices and real visualization softwares. The demonstration is always calibrated to the visitor's knowledge and skills in cybersecurity. Thus, before explaining a cyberattack, the presenter explains them the architecture of the whole platform. A tablet used by the presenter allows him to project pictures on the electrical cabinet to present the different industrial networks or physical devices. Then, the presenter describes the use case; for example the management of an hydro power plant. Using the SCADA software and the projection of a graphical view of the physical process, he explains the different components of the physical process (dam, valves, turbo-alternator, protection relay, etc). Then, at the start-up of the scenario, the visitor sees on the graphical view the given facility running smoothly (e.g., the flow of water in the penstock or the generation of electricity in the generator).

After this, the presenter introduces the context of the attack using some slides depending on the use case and the presented attack (e.g., the company that manages the hydro plant targeted by the attacker, the context of the attack and the profile of the attacker). The first step of attack demonstration shows simple ways for an attacker to perform OSINT and gather information about the targeted company and its employees. His objective is to find an employee whose position allows him to access the industrial system and whom he can corrupt or use his weaknesses to carry out his attack. In our usage scenario, the attacker mainly uses social networks to obtain personal information about the technical operator of the industrial system. Once the attacker obtained enough information on the target, he can forge an email or position a corrupted key on a place to set up the attacks presented in Section 3.2.

The objective of this kind of training is to understand the vulnerabilities used by an attacker to develop his attacks. Even if the attacks just last few minutes during the demonstration, it is important to explain to visitors that this kind of attack can require few months or years to be developed. Moreover, an important knowledge of the industrial system and physical process is needed, it may also be needed to corrupt the company's staff to obtain some secret information. Next, the attack is carried away as described in Section 3.2. At the end of this demonstration, some security guidelines based on the rules of the French ANSSI (*Agence National de la Sécurité des Systèmes d'Information*), the french organization for computer security [4] [5] and the NIST (National Institute of Standards and Technology) [45] in relation with the aforementioned attacks are discussed. These technical and organizational principles are basic rules to improve the security of industrial systems.

4 Pentesting and Reverse Engineering Flexible Training

In this section we present a larger scale implementation of the Hardware-in-the-Loop (HIL) system in a university lab. We also briefly describe the teaching programs developed using the lab material. G-ICS lab (GreEn-ER[8] Industrial Control systems Sandbox) is an industrial control systems research and teaching lab. As well as the WonderICS platform presented in Section 3, G-ICS is built upon the HIL architecture presented in Section 2. More than 100 industrial devices (controllers, protection relays, remote terminal units and industrial HMI's) from several vendors are available together with several supervisory control servers and a few security devices (Stormshield firewalls and Cisco CyberVision IDS). A partial view or the lab is presented in Figure 14.

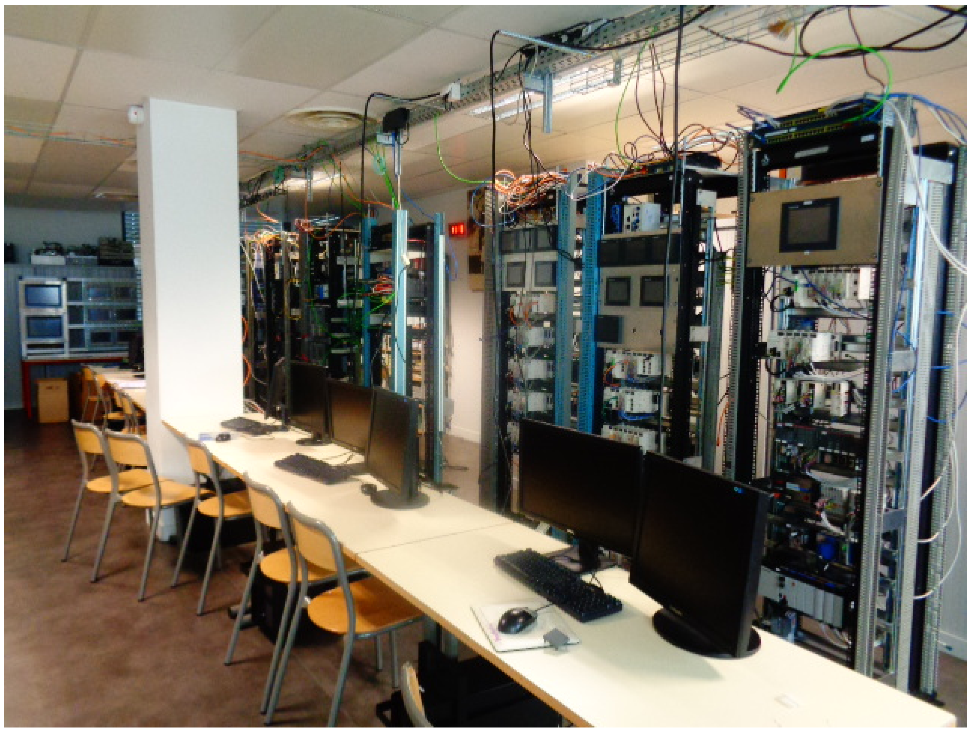


Figure 14: G-ICS lab

[8]Grenoble énergie - enseignement et recherche - an energy research and training center in Grenoble

The lab is used both for research and training. On the research side, the lab was used for the experimental part of two Ph.D. theses on intrusion detection and as a demonstrator for the Grenoble Cybersecurity Institute[9] project. It has also provided a demonstration stand at International Cybersecurity Forum (FIC) at Lille since 2017. On the training side, the lab is used for electrical engineers training in industrial automation and supervisory control, industrial real-time communications and cybersecurity awareness of electrical engineering students but also for the computer science students in Cybersecurity Master major. In the following, we will shortly present the two education programs, as a demonstration on how our HIL platform can be used for student training.

4.1 Electrical Engineering Training Program

For electrical engineering students, the main objective is to acquire PLC programming and supervisory control skills. However, modern control engineering relies heavily on communication networks, thus a basic knowledge of industrial systems cybersecurity is mandatory. Our electrical engineering students are enrolled for an awareness training in cybersecurity compliant with the French CyberEDU[10] educational label. Therefore, their curricula is a mix of SCADA and cybersecurity training (Table 2).

Topic	Type	Duration
Computer networks	Master class	1 day
Industrial protocols	Master Class	1 day
Cybersecurity Primer	Master Class	1 day
SCADA	Lab	2 days
Industrial protocols	Lab	3 days
Basic security controls	Lab	2 days

Table 2: Electrical engineering track.

The master classes address the basics of TCP/IP protocols, fieldbus and industrial real-time protocols and an introduction to cybersecurity with an emphasis on industrial control systems: review of classical attacks and incidents concerning SCADA systems, main differences between IT and OT, common vulnerabilities of industrial devices and basic security controls. SCADA and Industrial protocols labs focuses not only on controller programming and SCADA design but also on traffic observation, network calculations and optimization. For a typical lab, we use a “serious game” plant simulation like Home I/O or Factory I/O from Real Games[11] as they are designed for education [41]. We interfaced the simulators with our HIL system and real PLCs using the Real Games API and the programs are available on-line[12]. In Figure 15, a screen capture shows the final set-up of a student lab: the simulated physical process (Home I/O screen lower right), control program running o the real controller (Schneider Control Expert screen upper left corner), SCADA screen connected with the physical controller (PCVue synoptic lower left corner) and the WireShark network monitoring of the Network optimization lab is based on the analysis of the number of protocol requests versus the number and type of process variables read and write by the SCADA. It implies an organization of the PLC internal variables in contiguous blocks grouped by type (read-only or read-write) and polling period but also the programming of the SCADA client such that the number of requests is minimized. Finally, the cybersecurity lab implies the deployment of the

[9] `https://cybersecurity.univ-grenoble-alpes.fr/`
[10] `https://www.cyberedu.fr/`
[11] `https://realgames.co/`
[12] `http://lig-g-ics.imag.fr`

available security controls. The considered attack scenarios are: Modbus/TCP exchanges (upper right corner).

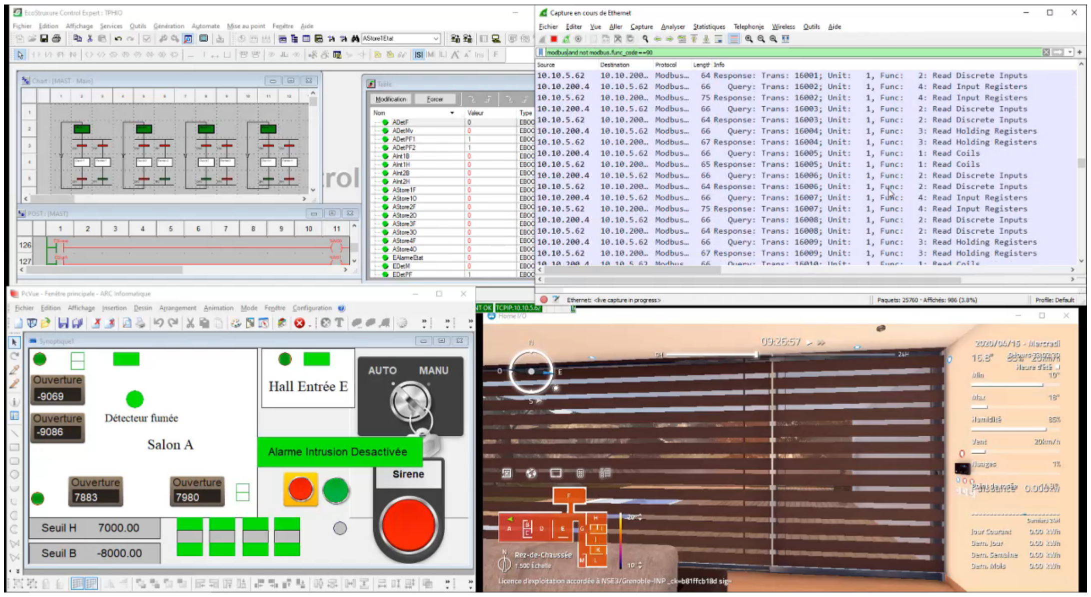

```
modbus and not modbus.func_code==90
Source        Destination   Protocol  Length Info
10.10.5.62    10.10.200…    Modbus…   64 Response: Trans: 16001; Unit:   1, Func:   2: Read Discrete Inputs
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16002; Unit:   1, Func:   4: Read Input Registers
10.10.5.62    10.10.200…    Modbus…   75 Response: Trans: 16002; Unit:   1, Func:   4: Read Input Registers
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16003; Unit:   1, Func:   2: Read Discrete Inputs
10.10.5.62    10.10.200…    Modbus…   64 Response: Trans: 16003; Unit:   1, Func:   2: Read Discrete Inputs
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16004; Unit:   1, Func:   3: Read Holding Registers
10.10.5.62    10.10.200…    Modbus…   67 Response: Trans: 16004; Unit:   1, Func:   3: Read Holding Registers
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16005; Unit:   1, Func:   1: Read Coils
10.10.5.62    10.10.200…    Modbus…   65 Response: Trans: 16005; Unit:   1, Func:   1: Read Coils
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16006; Unit:   1, Func:   2: Read Discrete Inputs
10.10.5.62    10.10.200…    Modbus…   64 Response: Trans: 16006; Unit:   1, Func:   2: Read Discrete Inputs
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16007; Unit:   1, Func:   4: Read Input Registers
10.10.5.62    10.10.200…    Modbus…   75 Response: Trans: 16007; Unit:   1, Func:   4: Read Input Registers
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16008; Unit:   1, Func:   2: Read Discrete Inputs
10.10.5.62    10.10.200…    Modbus…   64 Response: Trans: 16008; Unit:   1, Func:   2: Read Discrete Inputs
10.10.200.4   10.10.5.62    Modbus…   66    Query: Trans: 16009; Unit:   1, Func:   3: Read Holding Registers
10.10.5.62    10.10.200…    Modbus…   67 Response: Trans: 16009; Unit:   1, Func:   3: Read Holding Registers
Ethernet: <live capture in progress>   Paquets: 25760 · Affichés: 986 (3.8%)   Profile: Default
```

Figure 15: Plant/Controller/SCADA/Network traffic view

- False data injection into the PLC by a threat present on the SCADA network;
- PLC program modification via an unauthorized developer access;
- PLC internal data modification using an unsecured service (FTP, or embedded web site with default credentials).

PLC security controls: Students will consult the manufacturer specifications and establish the list of available controls then justify for each control the necessity and deployment level. The most common available controls are :

- Whitelist: a list of allowed IP addresses to connect and exchange data with a controller can be established. As only the IP address is verified this security control can be easily avoided and the students are aware to never rely on whitelist only;
- Protected memory variables: exchanges between SCADA and PLC are using directly the PLC memory variables. On some PLC models a variable protection can be applied. Students have to list the used PLC memory addresses in the SCADA communication, check that they are organized by category in the PLC memory (read and read-write distinct blocks) then set-up the adequate access rights in the PLC configuration;
- Password protection of the PLC control program: By default programming access to the PLC is not password protected. Practically all recent programming environments allow password protection of the control program and raise a warning when password is absent.
- Shutting down unsecured services and securing the needed services: traditionally, control devices are using unsecured services like FTP, SNMP, SMTP and HTTP for non-real time communications

like firmware upgrade, diagnostics and network and device monitoring. They are still available even on the last generation models but, hopefully, no more activated by default or, at least, a warning is raised by the development environment if an unsecured service is activated. Students have to list the available services, check the utility of each service for their application, set-up a secured version where possible (use of HTTPS instead of HTTP and SNMPv3 instead of SNMPv2, for instance) check that default login/passwords are not used, shut down unnecessary services (usually FTP) and list the remaining unsecured services (typically NTP and SMTP clients). A network protection mechanism will be required for the unsecured remaining services.

- Syslog setup: Some recent PLC models implement a syslog service and are even able to log messages on a remote syslog server. Students have to check the presence of the syslog service and set-up the client.

Network security controls: As the training addresses electrical engineering students, we do not ask them to do advanced tasks like setting up secure gateways and VPN channels. The labs are targeting the network monitoring and basic understanding of firewall rules. We are using CISCO Cisco Cyber Vision[13] for industrial traffic monitoring and alert raising. Using a cartography of normal traffic students shall propose firewall rules to block potentially abnormal requests. Using deep packet inspection capability of industrial firewalls (we use the Stormshield SNi40[14] in particular), one can detail the analysis down to Modbus/TCP function identifiers and memory addresses accessed. Most off the shelf SCADA software will not provide the full lists of requests and protocols used in the implementation. Usually, one has to observe the traffic in order to decide which subset of requests of a given protocol are used in a given application. A critical analysis is used to define the limits of the monitoring based flow cartography while a firewall ruling, allowing only observed request may exclude and, therefore, it may block ”rare” but legitimate requests not observed in the regular traffic (like alerts, for instance or device configuration requests).

4.2 Cybersecurity Master Training Program

A second track was set-up for cybersecurity students enrolled to a specialized Master program. While their future jobs will be related either to information security systems deployment and management or to cybersecurity controls development and device testing, the track is a mix between normative document study, risk analysis and pentesting.

Topic	Type	Duration
Cybersecurity of Industrial Systems	Master Class	0.5 day
Hands-on SCADA	Lab Tutorial	2 days
ISO/IEC 27000	Master class	1 day
EBIOS for Industrial systems	Master Class	1 day
STRIDE		0.5 day
IEC 62443	Master Class	1 day
Pentesting SCADA	Lab	1 days
Reverse engineering	Lab	1 days

Table 3: Cybersecurity Master track.

[13]https://www.cisco.com/c/en/us/products/security/cyber-vision/index.html
[14]https://www.stormshield.com/products/sni40/

The first master class introduces the main concepts of SCADA systems and their cybersecurity. We present the basic notions of control systems, PLC and supervisory control and we focus on the industrial real-time aspects and their consequences to the device and communication design. Cybersecurity is introduced via classical examples of cyber-incidents and attacks (2006 U.S. blackout started by an accidental false-data injection, Stuxnet and Black Energy) we present the main differences between IT and OT systems from the point of view of cybersecurity and a we briefly introduce the standards that apply. The ”hands-on SCADA tutorial” is a fast initiation to PLC programming and supervisory control for non-control-system students. The trainees have to follow an example and set-up a basic SCADA system (one PLC and one supervision screen). The secondary objective is to let students understand the close link between the communication protocols and the control of the system. This point will be exploited in the pentesting labs and the studied use-case (Figure 16) will be used for cybersecurity master class applications. The physical process is a mixer of two products. The PLC program has to control the filling of the tank with the two products, control the mixer motor and the evacuation of final product. Only three sensors (level sensors E, P1 and P2) and four actuators (filling valves VP1 and VP2, the motor M and the flush valve VE) have to be controlled. The SCADA screen will visualize the states of the sensors and actuators and allow manual operations.

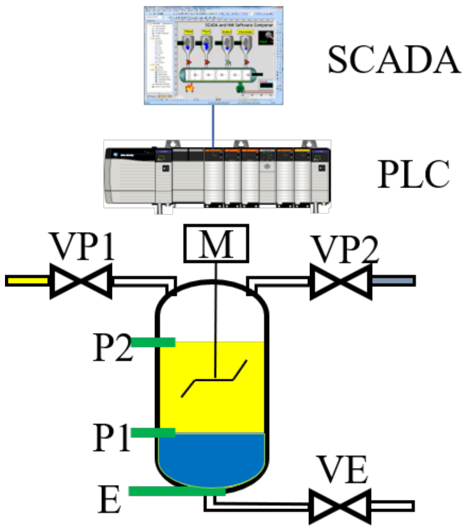


Figure 16: Simple use-case for SCADA tutorial and pentesting

The next three master classes are intended to provide a comprehensive description of the SCADA cybersecurity and the state of the standardization. The IEC 27000 [20] class covers the essentials of the standard (management system, risk management, security controls) and also sector specific information, in particular for energy (ISO 27019). A simple risk-analysis and security controls deployment exercise is conducted on the use-case from the SCADA tutorial. Two risk analysis methods are presented. The EBIOS[15] for industrial systems master class is a one day tutorial presenting the French risk management system applied to industrial systems case. We conduct an interactive EBIOS exercise on the studied use-case. STRIDE is a system threat modeling technique based on the analysis of an information flow diagram. There are six threat categories considered and the STRIDE acronym is a mnemonic of the threat names: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service and Elevation of privilege. The modeling technique uses a graphical representation of the systems based on Data Flow

[15]Expression des Besoins et Identification des Objectifs de Sécurité : Expression of need and security requirements identification in French

Diagrams (DFD) enriched with confidence boundaries which are sets of entities (processes, flows and data storage of same privilege level). The method was initiated by Microsoft and spread especially in the industrial world. We use the Microsoft Threat Modeling Tool[16] to model the communication of simple use-case as an application. One of the automatically generated threat model will be implemented in the lab as a real attack.

IEC 62443 [12] is the newest cybersecurity standard for industrial systems. It was originally initiated by International Society of Automation and some parts are still under development. Several certifications schemes are already defined and all major device manufacturers are certifying their equipments. The 62443 approach redefines the security objectives in a more "control system" manner and proposes a partitioning of the systems in security zones. The idea is that, in a real control system, the security needs and the capability of the devices are not the same at different levels. For instance, it will be very difficult to encrypt communication between sensor and actuators at field level while keeping the real-time performance. Therefore, the 62443 aims to group devices with the same requirements and capabilities into zones and to control the communication between zones. Thus, the security approach is mostly related to network segmentation. We conduct a network segmentation exercise on firstly on the simple use-case then on a complex. The trainees are required to identify zones and flows between zones then write the fire-wall rules for the corresponding network segmentation.

The pentesting SCADA lab contains two parts: the first one consists in a false data injection attack implementation and test on real devices. The students are asked to study the traffic between SCADA and PLC in simple use-cases and identify the network frames used to remotely control the actuators then write a program that will inject malicious controls into the systems and compromise the process (for example open the flush while the motor is still running or continue to fill the tank after the nominal level is attained). The effects of the attack are visible on the simulated physical process. The second part concerns the test of some known exploits. For instance we test the CVE-2013-2763[17], as the exploit can be found on internet. After testing and visualizing the exploit the students are required to write a firewall rule for blocking the known exploit.

The protocol reverse engineering lab continues the pentesting lab with simple illustration of techniques employed to find vulnerabilities and exploits. We study industry proprietary protocols[18] whose specifications are not public. For the reverse engineering part, we use a traffic capture between an HMI and a PLC corresponding to a known operation (for instance forcing an output on a PLC) students are required to find the protocols field used to specify the address of the output then use this partial information to build an attack and finally implement and test the attack. Protocol fuzzing is used to detects some legal requests and even to found the exploit associated to CVE-2013-2763.

5 Related Work and Conclusion

Researches on industrial testbeds have been a hot topic lately. A shown in a survey conducted by Holm et al. [24], there are three kinds of testbeds depending on the virtualization degree:

- *Simulation:* Those approaches will simulate both the behavior of the physical process and the control equipment. These approaches differ from ours since our goal is to rely on real hardware equipments (which are yet to be realistically reproduced by software simulation).

[16]`https://www.microsoft.com/en-us/securityengineering/sdl/threatmodeling`

[17]`https://nvd.nist.gov/vuln/detail/CVE-2013-2763`

[18]`https://documentation.stormshield.eu/SNS/v4/en/Content/User_Configuration_Manual_SNS_v4/Protocols/SCADA-UMAS.htm`

- *Physical process virtualization*: Our testbed fits in this category. Most of existing works rely on commercial simulation software such as Matlab/Simulink, OPAL-RT, LabVIEW, Power-World and most of them only consider a specific kind of physical process (e.g., chemical industry) and are mainly used for productivity forecasting [23, 31, 32, 46, 34, 9]. They are not designed to communicate with other components such as PLCs and thus do not have any network interface such as Modbus. Other are hardly configurable and are not designed to model industrial process [13, 3, 15, 35]. Rather, they are meant to stress test SCADA systems. Close-source alternatives exist but are costly to use for research and/or student projects [40, 8]. Finally, some open-source research exist but seem either abandoned or outdated [42, 2]. Regarding our interface boards specifically, the closest works are Factory I/O Advantech USB 4750 sold by RealGames14 and the one sold by Schneider15. Both solutions are are USB connected so they will not scale and they provide digital I/O only. In general, existing interface boards (mainly Arduino shields, Raspberry Pi modules or specialized data acquisition modules) are limited to digital I/O and, sometime analog input. Analog outputs are rare, and limited one or two. Our card, able to provide eight analog outputs has the most interesting benefit/cost ratio.

- *No virtualization:* These are fully hardware testbeds with real-size processes, built especially for cybersecurity studies. Due to the prohibitive cost and maintenance level, these are nation wide projects. The most known is the Idaho National Laboratory [25] which is a full industrial system. Yet, changing the physical process in these solutions is indeed costly.

In this paper, we presented two twin demonstrators with different purpose: (i) an Advanced Persistent Threat (APT) demonstrator used for awareness training (WonderICS) and (ii) a flexible lab used for students training and pentesting (G-ICS). Both are based on a common Hardware-In-the-Loop technology that includes both industrial process simulation, real firmwares emulation alongside with real industrial control devices. Two teaching courses based on the G-ICS lab started in 2016 (for electrical engineering students), and in 2018 for cybersecurity Master students. Let alone the students enthusiasm for exploit and hacking, a detailed interview showed that, for electrical engineers the training demystified the cybersecurity and even lead some of them to switch their carrier ($\sim 2\%$). Cybersecurity master students, on the other hand, declared that the training was useful for them distinguish from other candidates as skills in industrial control systems cybersecurity is a growing demand. The WonderICS platform was used in multiple demonstrations for industrial vendors and stakeholders ($\sim$ 30 demos per year). Attendees expressed a lot of interest for the underlying problem of the cybersecurity of industrial systems and plebiscited the realism of the demos. Most visitors were also quite shocked by the simplicity and the lack of requirements to launch a cyberattack on an ICS. They were also keen on seeing how an APT is construct and how it can present being detected.

As future works, we aim to continue developing our HIL technology. Many directions are worth studying, such as WonderCloud deeper integration in the existing scenarios (with support for hardware architectures found in industry such as VxWorks), support for of both platforms either by integrating IIoT devices communicating over wireless media (e.g., ZigBee, Bluetooth), or real-life industrial malware based demos.

References

[1] OpenStack. `https://www.openstack.org/`. Accessed: 2021-12-10.

[2] Modrssim: Modbus PLC simulator. `http://www.plcsimulator.org/`, 2010. Accessed: 2021-12-10.

[3] Modbuspal - a Java Modbus simulator. `https://sourceforge.net/p/modbuspal/`, 2013. Accessed: 2021-12-10.

[4] Agence Nationale de la Sécurité des Systèmes d'Information. *La cybersécurité des systèmes industriels : Maîtriser la SSI pour les systèmes industriels*. ANSSI, June 2012.

[5] Agence Nationale de la Sécurité des Systèmes d'Information. *La cybersécurité des systèmes industriels : Mesures détaillées*. ANSSI, Jan. 2014.

[6] G. N. Angafor, I. Yevseyeva, and Y. He. Bridging the cyber security skills gap: Using tabletop exercises to solve the cssg crisis. In M. Ma, B. Fletcher, S. Göbel, J. Baalsrud Hauge, and T. Marsh, editors, *Proc. of the 15th Joint International Conference on Serious Games, Staffordshire University, Stoke-on-Trent, UK, LNCS*, volume 12434 of *Lecture Notes in Computer Science*, pages 117–131. Springer-Verlag, Nov. 2020.

[7] F. Bellard. QEMU, the FAST! processor emulator. `https://qemu.org/`, 2011. Accessed: 2021-12-10.

[8] M. Berutti. Virtual plant using mimic for Schneider electric control systems, 2019. Accessed: 2021-12-10.

[9] A. Borshchev. *The big book of simulation modeling: multimethod modeling with AnyLogic 6*. AnyLogic North America Chicago, 2013.

[10] BSI. Top 10 threats and countermeasures. Technical report, Federal Office for Information Security, 2019.

[11] D. D. Chen, M. Woo, D. Brumley, and M. Egele. Towards automated dynamic analysis for linux-based embedded firmware. In *Proc. of the 23rd Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA*. The Internet Society, Feb. 2016.

[12] I. E. Commission. *IEC/TS 62443-1-1:2009 Industrial communication networks – Network and system security – Part 1-1: Terminology, concepts and models*, volume 6 of *62443*. Afnor, Jan. 2009.

[13] Control Toolbox. Modbus TCP plant simulator. `http://controltoolbox.com/modbus_tcp_plant_simulator`, 2009. Accessed: 2021-12-10.

[14] DeepBitsTechnology. Firmpin. `https://github.com/DeepBitsTechnology/FirmPin`, 2018. Accessed: 2021-12-10.

[15] T. Dogan. MSS Modbus-TCP server simulator 1.0. `http://tdogan.net/mss.html`, 2019. Accessed: 2021-12-10.

[16] J. Dreier, M. Puys, M.-L. Potet, P. Lafourcade, and J.-L. Roch. Formally Verifying Flow Properties in Industrial Systems. In *Proc. of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain*, pages 55–66, July 2017.

[17] J. Dreier, M. Puys, M.-L. Potet, P. Lafourcade, and J.-L. Roch. Formally and practically verifying flow properties in industrial systems. *Computers & Security*, 86:453–470, 2019.

[18] European Parliament and Council of European Union. Directive (EU) no 2016/1148 concerning measures for a high common level of security of network and information systems across the union. `https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016L1148&from=EN`, July 2016. Accessed: 2021-12-10.

[19] FOIA response documents. Aurora Project. United States Department of Homeland Security, July 2014. `http://s3.documentcloud.org/documents/1212530/14f00304-documents.pdf`, Accessed: 2021-12-10.

[20] I. O. for Standardization. *ISO/IEC 27000:2018: Information technology — Security techniques — Information security management systems — Overview and vocabulary*. ISO/IEC, Geneva, 2018.

[21] Fraunhofer. FKIE. `https://github.com/fkie-cad/FirmwareScraper`. Accessed: 2021-12-10.

[22] M. Gallissot, M. Puys, and P.-H. Thevenon. WonderCloud, une plateforme pour l'analyse et l'émulation de micrologiciels ainsi que la composition de pots de miels. In *Proc. of the 27th C&esar conference on Deceptive Security (C&esar 2020)*, Rennes, France, Nov. 2020.

[23] J. E. Hammann and N. A. Markovitch. Introduction to arena [simulation software]. In *Proc. of the 7th IEEE Winter Simulation Conference, Arlington, VA, USA*, pages 519–523. IEEE, 1995.

[24] H. Holm, M. Karresand, A. Vidström, and E. Westring. A survey of industrial control system testbeds. In *Proc. of the 20th Nordic Conference on Secure IT Systems (NordSec), Stockholm, Sweden, LNCS*, volume 9417 of *Lecture Notes in Computer Science*, pages 11–26. Springer-Verlag, Oct. 2015.

[25] INL. Common Cyber Security Vulnerabilities Observed in Control Systems Assessments by INL NSTB Program. . Report, Idaho national Laboratory, Nov 2008.

[26] A. Jakhar. Expliot. `https://expliot.readthedocs.io/en/latest/index.html`, 2018. Accessed:

2021-12-10.

[27] R. Langner. Stuxnet: Dissecting a cyberwarfare weapon. *Proc. of the 32nd IEEE Symposium on Security & Privacy (S&P), Oakland, California, USA*, 9(3):49–51, May 2011.

[28] T. Maurin, L.-F. Ducreux, G. Caraiman, and P. Sissoko. IoT security assessment through the interfaces P-SCAN test bench platform. In *Proc. of the 25th IEEE Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany*, pages 1007–1008. IEEE, Mar. 2018.

[29] T. McAvoy and N. Ye. Base control for the Tennessee Eastman problem. *Computers & Chemical Engineering*, 18(5):383–413, Apr. 1994.

[30] S. Mocanu, M. Puys, and P.-H. Thevenon. An Open-Source Hardware-In-The-Loop Virtualization System for Cybersecurity Studies of SCADA Systems. In *Proc. of the 26th C&esar conference on Virtualization and Cybersecurity (C&esar 2019)*, pages 1–16, Rennes, France, Nov. 2019.

[31] R. Morales-Rodríguez, R. Gani, S. Déchelotte, A. Vacher, and O. Baudouin. Use of CAPE-OPEN standards in the interoperability between modelling tools (MoT) and process simulators (Simulis® Thermodynamics and ProSimPlus). *Chemical Engineering Research and Design*, 86(7):823–833, July 2008.

[32] W. B. Nordgren. Flexible simulation (Flexsim) software: Flexsim simulation environment. In *Proc. of the 35th ACM Conference on Winter simulation: driving innovation, New Orleans, Louisiana, USA*, pages 197–200. ACM, Dec. 2003.

[33] OnwardSecurity. Automated Vulnerability Assessment Tool HERCULES SecDevice. `https://www.onwardsecurity.com/product_cate/item/29`. Accessed: 2021-12-10.

[34] A. Pakonen, T. Mätäsniemi, J. Lahtinen, and T. Karhela. A toolset for model checking of PLC software. In *Proc. of the 18th IEEE Conference on Emerging Technologies & Factory Automation (ETFA), Cagliari, Italy*, pages 1–6. IEEE, 2013.

[35] W. Path. Modmultisim, a programmable Modbus slave simulator. `http://wingpath.co.uk/modbus/modmultisim.php`. Accessed: 2021-12-10.

[36] D. Peterson. Basecamp project. `https://youtu.be/BKJje3Ram2I`, `https://github.com/digitalbond/Basecamp`, 2016. Accessed: 2021-12-10.

[37] M. Puys, M.-L. Potet, and P. Lafourcade. Formal analysis of security properties on the opc-ua scada protocol. In A. Skavhaug, J. Guiochet, and F. Bitsch, editors, *Proc. of the 35th International Conference on Computer Safety, Reliability, and Security (SAFECOMP), Trondheim, Norway, LNCS*, volume 9922 of *Lecture Notes in Computer Science*, pages 67–75. Springer-Verlag, 2016.

[38] M. Puys, P.-H. Thevenon, and S. Mocanu. Hardware-In-The-Loop Labs for SCADA Cybersecurity Awareness and Training. In *Proc. of the 16th ACM International Conference on Availability, Reliability and Security (ARES), Virtual Conference*, pages 1–10. ACM, Aug. 2021.

[39] ReFirmLabs. Binwalk. `https://github.com/ReFirmLabs/Binwalk`, 2013. Accessed: 2021-12-10.

[40] B. Riera, R. Pichard, A. Philippot, R. Saddem, F. Gellot, D. Annebicque, and F. Emprin. HOME I/O et FACTORY I/O: 2 logiciels innovants de simulation de PO pour la formation à l'automatique. In *Colloque consacré à l'Enseignement des Technologies et des Sciences de l'Information et des Systèmes (CETSIS), Le Mans, France*, May 2017.

[41] B. Riera and B. Vigário. HOME I/O and FACTORY I/O: a virtual house and a virtual plant for control education. *Proc. of the 20th IFAC World Congress, Toulouse, France*, 50(1):9144–9149, July 2017.

[42] J. Seidl. Virtuaplant. `https://wroot.org/projects/virtuaplant/`, 2015. Accessed: 2021-12-10.

[43] E. Sitnikova, E. Foo, and R. B. Vaughn. The power of hands-on exercises in SCADA cyber security education. In R. C. Dodge and L. Futcher, editors, *Proc. of the 8th IFIP WG 11.8 World Conference on Information Security Education (WISE 8), Auckland, New Zealand*, volume 406 of *IFIP Advances in Information and Communication Technology*, pages 83–94. Springer-Verlag, July 2013.

[44] A. Soullié. Fun with Modbus 0x5a. `https://youtu.be/A_B69Rifu1g`, `https://github.com/wavestone-cdt/fun-with-modbus-0x5a`, 2017. Accessed: 2021-12-10.

[45] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn. *NIST Special Publication 800-82 - Guide to Industrial Control Systems (ICS) Security*. NIST, May 2015.

[46] D. Wagner. Dwsim - open source process simulator. http://dwsim.inforside.com.br, 2013. Accessed: 2021-

12-10.

[47] T. J. Williams. A reference model for computer integrated manufacturing (CIM). *Proc. of the 11th IFAC World Congress on Automatic Control, Tallinn, Finland*, 23:281–291, Aug. 1990.

[48] xPack$. xPack QEMU Arm. `https://xpack.github.io/qemu-arm/`. Accessed: 2021-12-10.

[49] J. Zaddach, L. Bruno, A. Francillon, D. Balzarotti, et al. Avatar: A framework to support dynamic security analysis of embedded systems' firmwares. In *Proc. of the 21st Annual Network and Distributed System Security Symposium (NDSS), San Diego, California, USA*. The Internet Society, Feb. 2014.

Author Biography

Maxime Puys is a research engineer at the LETI Institute, CEA (Commissariat à l'énergie atomique et aux énergies alternatives). He received his Ph.D in 2018 from University of Grenoble Alpes. His thesis focused on the cybersecurity of industrial control systems against network attacks and risk analysis combining safety and security. He now designs security components and contributes to the development of new tools to analyze the security of Industrial IoT.

Pierre-Henri Thevenon is a research engineer at the LETI institute, CEA (Commissariat à l'énergie atomique et aux énergies alternatives), specialized in the security of IoT and I-IoT. After receiving his Ph.D in 2011 from University of Grenoble Alpes on the security of contactless systems, he worked on the development and security of embedded systems. For the last 4 years, he has been leading multiple Industrial IoT Security projects, within IRT Nanoelec.

Stéphane Mocanu obtained a Ph.D in Control Systems in 1999 from Grenoble-INP. He is assistant professor in Grenoble-INP and in Laboratoire d'Informatique de Grenoble (LIG, UNR 5217 CNRS/G-INP/UGA) in the joint Inria CTRL-A team. He started working on industrial control systems cybersecurity in 2012 and he's running a large size experimental lab for industrial systems cybersecurity pentesting and vulnerability research.

Mathieu Gallissot is a cybersecurity architect at the LETI institute, CEA (Commissariat à l'énergie atomique et aux énergies alternatives), specialized in smart environments (homes, buildings and cities). He received his Ph.D in computer science from University of Grenoble Alpes in 2012 and since have 15 years of experience as an ICT expert in the areas of energy, comfort and well-being management. He joined CEA in 2012 in order to set-up experimental platforms related to emerging technologies and snow animates cybersecurity research activities in various IoT emerging domains.

Camille Sivelle is a research engineer at the LETI Institute, CEA (Commissariat à l'Énergie Atomique et aux Énergies Alternatives), specialized in IoT cybersecurity. She received her MSc in computer science from Lorraine University in 2020 where she was dealing with formal analysis of cryptographic protocols. She joined CEA in 2020 where she now carries out research on firmware emulation, malware detection and cryptographic protocols code generation.