A Stealth Monitoring Mechanism for Cyber-Physical Systems

Vitor Graveto[a,*], Luís Rosa[a], Tiago Cruz[a], Paulo Simões[a]

DEI-CISUC University of Coimbra Coimbra, Portugal

[a]*Departamento de Eng. Informática*
Polo II da Universidade de Coimbra
3030-290 Coimbra, Portugal

Abstract

Supervisory Control and Data Acquisition (SCADA) systems, which are often used in several types of Essential Systems and Critical Infrastructures, depend on control devices such as Programmable Logic Controllers, Remote Terminal Units and Intelligent Electronic Devices. Such devices, which are deployed at the edge of the SCADA infrastructure, directly interface with the physical processes under control. They are often based on embedded systems with limited capabilities and exposed to significant security and safety-related risks, as demonstrated by past incidents such as Stuxnet. However, despite the recognized relevance of those edge devices, they usually lack monitoring mechanisms able to detect device anomalies and/or cyber-physical threats.

In this paper we propose a novel approach for stealth monitoring of those control devices, for purposes of security and safety management. This approach builds on cost-effective probes, which we designate as Shadow Security Units (SSU), directly attached to the monitored control devices. This privi-

*Corresponding author
Email address: vgraveto@dei.uc.pt (Vitor Graveto)

Preprint submitted to International Journal of Critical Infrastruture Protection July 19, 2018

leged positioning enables the direct and fine-grained observation of both physical inputs/outputs (i.e. the physical processes under control) and network communication flows – allowing the exploitation of various novel monitoring approaches able to address sophisticated security threats not noticeable otherwise. Moreover, the SSU approach is not limited to SCADA scenarios, being also applicable to similar domains such as the Internet of Things (IoT), Avionics and Self-Driving systems.

Keywords: `Safety and Security Monitoring`, SCADA, `Anomaly Detection`, `Industrial Automation and Control Systems`

1. Introduction

In the scope of Industrial and Automation Control Systems (IACS), the need for monitoring the proper operation of physical devices is an old concern, especially for the purpose of fault and anomaly detection. Its main goal is to detect and/or anticipate possible problems, in order to detect malfunctioning devices and to prevent or mitigate their negative impact on the processes they are part of.

That need applies not just to process sensors and actuators (e.g. an engine, a car brake, an electric switch, a temperature sensor), but also to process control devices, such as SCADA's Programmable Logic Controllers (PLC), Remote Terminal Units (RTU) and Intelligent Electronic Devices (IED). These control devices are connected both to the physical systems they are intended to control and to the SCADA process control servers.

Due to their positioning and role, any vulnerability on these devices – due to malfunction or malicious attacks – might compromise the whole pro-

cess under control, leading to considerable economic losses and/or hazardous situations.

The industry has acknowledged this problem long ago, and niche application fields where the implicit safety risks are not acceptable, such as avionics and nuclear plants, regularly adopt fault-tolerance solutions such as redundant (and possibly dissimilar) systems and Byzantine fault-tolerance mechanisms [1]. However, such designs remain too expensive and not practical enough for generalized adoption by mainstream IACS.

Moreover, the recent awareness of risks introduced by cyber-threats further increases the concerns with those control devices. They are exposed not only to the risk of failure, but also to malicious cyberattacks specifically targeting their vulnerabilities and using them to compromise the whole critical infrastructures they control. A compromised RTU or PLC might be undetectable using traditional Intrusion Detection Systems (IDS) but still jeopardize critical assets, by damaging the equipment or process they control and/or by reporting erroneous information back to the control platforms – as demonstrated by incidents such as Stuxnet [2].

This threat is further amplified by the fact that those devices are intrinsically more exposed to direct physical access (as they are deployed in the edge of the field network) and have significant capacity and design constraints that often prevent the use of adequate security mechanisms. This applies even more in the case of legacy equipment, which might be difficult to replace in the short/medium term – due to technical, operational or financial constraints.

In order to mitigate this situation, in this paper we explore the concept of

the Shadow Security Unit (SSU). The SSU is a monitoring probe capable of directly observing all the inputs and outputs of control devices such as RTUs or PLCs, including aspects such as: network communications (e.g. SCADA messages to/from control servers, firmware upgrade operations); analog and digital physical inputs (e.g. temperature, voltage, gas readers, all other types of sensors related with the controlled physical process); and analog and digital outputs (e.g. the actuation of an engine, control of a light, activation of a brake system). This concept, whose early form was first described in a short position paper [3], constituted one of the innovative building blocks of a broader strategy to create an IACS-oriented cyber-security solution, documented in [4].

When compared with classic probes, the SSU stands out due to its capacity of fully observing the interaction of the monitored control device with the outside world. Not just network communications with the SCADA process control servers but also the whole set of physical inputs and outputs of the monitored device.

In order to become successful, the SSU concept needs to address two major challenges. First, its implementation needs to be simple and cost-effective – in order not to fall into the economic and practical constraints that limit the widespread adoption of more complex Byzantine fault tolerance systems. Second, it needs anomaly detection mechanisms able to fully exploit the potential enabled by the different type of monitoring data collected by the SSU.

In this paper we address both challenges. First, we provide an overview of SCADA-based IACS systems (Section 2), followed by the discussion of pro-

cess safety and SCADA security mechanisms for IACS systems (Section 3). Next, we propose the SSU, discussing its reference architecture and implementation aspects (Section 4). Potential applications of the SSU, including possible anomaly detection techniques, are discussed in Section 5. Preliminary validation work is presented in Section 6, and related work is discussed in Section 7. Finally, Section 8 concludes the paper.

2. SCADA-Based IACS Systems

Figure 1 illustrates a generic SCADA-based IACS, such as those used on power plants, power grids, gas and water supply networks, industrial facilities, traffic management systems and water dams. In this scope, the Corporate LAN represents the mainstream Information and Communications Technology (ICT) infrastructure of the organization, interconnected to the IACS infrastructure using a firewall. On the IACS side, SCADA servers supervise the whole industrial process, while Human-Machine Interface (HMI) nodes – based on workstations or local, on-site terminals – interface with human operators. Finally, RTU, PLC and IED devices directly interconnect with process sensors and actuators.

Figure 2 provides a more process-centric look at a SCADA system – based on a simple example of controlling the water level of a tank. This system includes the following components:

- **Master stations**, which are deployed on the process network and supervise processes, controlling and monitoring Slaves and often providing support for HMI consoles. They are also frequently connected to other applications, such as historian databases for logging process data.

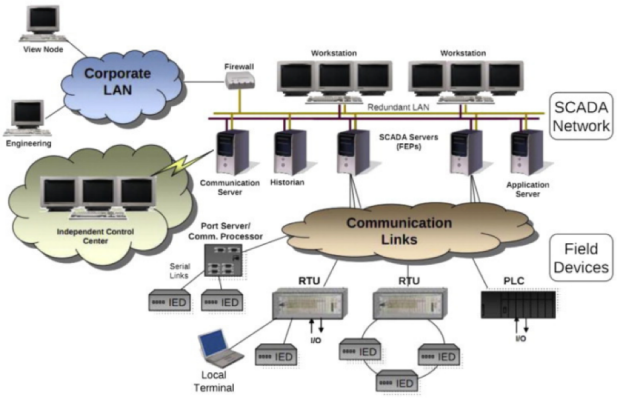

Figure 1: Typical SCADA-based IACS[5]

- **Slave devices** such as RTUs, PLCs and IEDs, deployed on the control network. These devices are typically embedded systems connected to one or more Master Stations, as well as to sensors and actuators. They are responsible for most of the process monitoring and control activities. RTUs have limited control capabilities, connecting to sensors that monitor the controlled process and sending data to master stations, while IEDs and PLCs are more sophisticated, supporting several programming languages and complex control scenarios. Due to their superior capability/cost ratio, IEDs and PLCs frequently replace RTUs in more recent deployments.

- **Field devices**, deployed on the field network and constituting the physical interface with the process under control, providing information about it (sensors) and enabling the execution of actions affecting its behavior (actuators).

As discussed later, devices such as RTUs, PLCs and IEDs are the main target of the stealth monitoring mechanisms proposed in this paper. Those

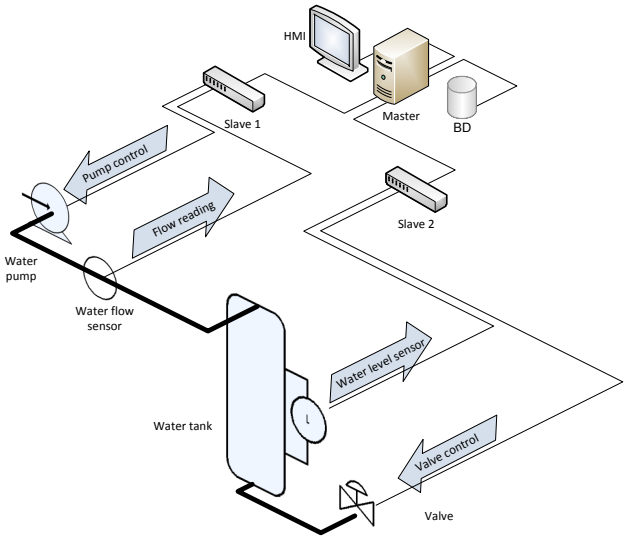

Figure 2: Architecture of a simple SCADA system (adapted from[6])

devices are interconnected using technologies such as RS-485 [7], CAN [8] or forms of Industrial Ethernet technologies such as EtherCAT [9] and Profinet [10], among others, according to the nature of the specific IACS infrastructure they integrate. The control flow between the Master station and the RTU/PLC/IED allows exchanging process-related data or executing actions by modifying process control parameters mapped on device registers. This process involves SCADA specific protocols, such as Modbus (in RTU, ASCII or TCP/IP variants) [11] [12], IEC 60870-5-104 (IEC 104) [13] or DNP3 [14], among others.

The extensive reliance on RTUs and IEDs/PLCs, within SCADA systems, means that overall reliability of the process under control is largely dependent on those devices. Depending on specific models and range, RTU, IED and PLCs may incorporate different features, going from watchdogs and basic on-board self-diagnostic and recovery features to more sophisticated techniques such as the inclusion of redundant hot-standby modules

for reinforcing the control loop or doubled execution units (sometimes using independent memory bocks), supporting double code generation and execution. However such mechanisms are only concerned with device-level safety and health of the CPU, RAM, I/O and communication modules, and cannot even be taken for granted in all models, as several features are only available in high-end product lines – ultimately, they do not ensure safety or security at the process or control infrastructure levels.

3. Process Safety and SCADA Security

Originally, the IACS industry mindset traditionally relied on isolation (the "airgap principle") and/or on the "black box" approaches, based on obscurity and use of proprietary and insufficiently documented technologies to ensure security. With convenience and cost pushing for the introduction of commodity technologies from the ICT world in IACS, such as TCP/IP networking, the "airgap" started to close. This process has been matched by a significant effort from equipment manufacturers, operators, standardization organisms and security communities in order to improve the overall security of SCADA systems. As a result, today's best practices already address the most obvious security weaknesses and attack vectors, such as the use of clear-text communications, lack of Authentication, Authorization and Accounting (AAA) capabilities or the absence of integrity checking mechanisms.

However, in parallel with this evolution, IACS became increasingly exposed to a range of more sophisticated threats that were unconceivable before on this ecosystem, such as stealthy Advanced Persistent Threats (APTs) running over a long time frame as part of a cyber-warfare strategy to gather

intelligence and later cripple public and/or military Critical Infrastructures (CIs). The Stuxnet worm [2, 15] is a prominent example that has raised significant awareness to the problem of Critical Infrastructure Protection (CIP). By exploiting a series of zero-day flaws, Stuxnet was able to attack the PLCs for a series of nuclear fuel processing centrifuges, damaging them by increasing their rotational speed, while concealing its actions.

3.1. Process Safety and Security

Incidents such as Stuxnet demonstrated that safety and security are interdependent characteristics, as the first may become compromised due to problems with the latter. Safety is defined as "freedom from unacceptable risk of harm" [16], being concerned with the avoidance of unintended accidents involving human or environmental harm. This is different from process security, which relates to protecting from intentional harm, targeting the specific implementation of the control process, managed by the in-band devices which implement and supervise the control discipline.

Outside the scope of the process loop, extra monitoring may be implemented using sources such as historian databases or OLE for Process Control (OPC) [17] servers, which can provide a detailed log of the system operation, for analysis and knowledge extraction. For instance, a server supporting the OPC-HDA (for Historical Data Access) may provide historian database logs, while real-time information may be acquired via OPC-DA (for Data Access) or OPC-AE (for Alarms and Events), if such profiles are supported – the newer OPC-UA (for Unified Architecture) integrates the DA, AE and HDA functionality within a common updated framework. Yet, this information is not useful without knowledge about the particular characteristics of the

controlled process and its specific details: if it uses a closed or open-loop (therefore requiring feedback), the type of control process (discrete, continuous), I/O sensors, actuators and respective electrical interface characteristics. Also, information about device operation, such as its programming, the PLC scan cycle or the type of I/O modules (discrete, analog) may also be important, for purposes of formal validation.

Moreover, there are shortcomings related to the use of middleware components for safety and security monitoring: the communications chain (Device-Server-Client) means that there is a latency overhead whose impact may vary according to the nature of the process. Furthermore, the OPC or historian servers constitute potential targets that can be potentially compromised, using Man-In-The-Middle attacks (to feed wrong information to the servers) or Denial-of-Service techniques (inducing a loss of visibility regarding controlled process). Without reliable information about the variables of the process, it is impossible to implement some sort of parallel, formal validation process. This latter example also gives a strong hint about the importance of security monitoring in SCADA, discussed next.

3.2. SCADA Security

Having become a cornerstone of many IACS and CIs, SCADA systems constitute a potential target for malicious activity, having shown a poor security record in recent years by succumbing to several successful attacks. This is due to several reasons. As they were originally restricted to isolated environments, SCADA systems were considered relatively safe from external intrusion. However, as architectures evolved, these systems started to assimilate technologies from the ICT world, such as TCP/IP and Ethernet

networking, encouraging the interconnection of the IACS with organizational ICT network infrastructures and even with the exterior (e.g., for remote management). This trend, together with the increasing adoption of open, documented protocols, exposed serious weaknesses in SCADA architectures and brought a new wave of security problems that were not conceivable when such systems were first designed, prompting a significant increase in the number of externally initiated attacks on IACS systems, especially when compared with internal attacks [18].

This meant that several issues that were familiar only to the ICT context suddenly became a source of concern for IACS operators – until them, IACS and ICT infrastructures were clearly differentiated (and even separated), mainly because of the considerable degree of independence among them, but also due to their fundamental governing principles. As noted by ISA-99 [19] (later updated by IEC 62443 [20]), the priorities for IACS and ICT systems are reversed: for IACS, availability comes first, even if at the cost of integrity and confidentiality – just the opposite of the ICT philosophy.

To a certain extent, the focus on availability influenced the IACS mindset to progressively regard technology and platform maturity as an implicit recognition of value and reliability. As an example, communication protocols such as Modbus [11], originally developed by Modicon in 1979, are still very popular in production systems, due to their simplicity and ease of use, despite suffering from security problems such as the lack of encryption or any other protection measures [21]. Eventually, these and other similar vulnerabilities became the root cause of many IACS security issues that have been successfully exploited in recent times, such as those mentioned next.

3.3. Attacks on IACS and SCADA Systems

The growing interest in critical infrastructures, the interconnection between the ICT network and the control network, the growing knowledge about the processes and protocols used, and in particular the amplitude of the possible damage in these infrastructures, have led to an increasing number of attacks. In this subsection, we present an overview of three well-known attacks which directly involved PLCs, RTUs or IEDs and could have been detected earlier with SSU-like devices (Stuxnet [2, 15], Black energy [22] and Maroochy Water Breach [23]), followed by the presentation of a simplified taxonomy of attacks on SCADA/IACS systems.

Stuxnet: At the Natanz Fuel Enrichment Plant in Iran at middle 2010, a number of problems were identified, the causes of which could be attributed to malfunctioning of uranium enrichment centrifuges. However, the analysis of existing forensic data shows that it was an attack, aimed at delaying the Iranian nuclear program, called Stuxnet.

The development and implementation of a cyber physical attack goes far beyond the code and the use of unknown vulnerabilities. It implies the knowledge and understanding of the architecture, the existing devices and the parameters of the SCADA system. This type of attack is based on actions at three distinct levels: the ICT network used to propagate malicious software; the control network used to manipulate the process; and the physical level, which in this case allowed to change the speed of rotation of the centrifuges (Siemens S7-315 controller) and / or their pressures (Siemens S7-417 controller) causing irreparable physical damage.

The malicious code used a Man in The Middle strategy, in the control net-

work, allowing input/output signals to be passed between peripherals, PLCs and SCADA control servers during normal operation. After the implementation of the attack, some previously recorded data (period of 21 seconds) were retransmitted in a cycle, allowing the display of a normal operating scenario in the SCADA monitors in the control room.

The spread of Stuxnet happened through local networks, USB sticks or through the laptops of contractors, with access to these local networks. That is, the global internet network was not used for the proliferation of Stuxnet neither to reach the final target — the attackers assumed the existence of an air gap between the facility and the global networks. The use of Zero Day exploits and valid certificates, illegally obtained, allowed Stuxnet to survive as a non-detectable system driver by antivirus or other security software, even with Windows operating system updates.

All facilities, either industrial or military, rely on their contractors, so at a certain instant a support team will have access to the ground zero, and deploy the worm, with no need to break firewalls, air gaps, data diodes or any other security system, as they have physical access to the control system. When Stuxnet reached the Iran facility it identified the known architecture and *waked up* and started its malicious purpose.

Black Energy: This attack was able to outage the power distribution of 27 substations, affecting at least 225,000 customers. It was named after the Trojan BlackEnergy 3 (Industroyer [24]), used by the attackers on an early stage to get access the ICT systems. The attack was developed in two stages: the intrusion – that started about nine months before – and the IACS attack – that occurred on December 23, 2015.

During the intrusion stage, several steps where taken. First, e-mails were sent to authorised users, as a phishing tool, to deploy the BackEnergy 3 Trojan through Microsoft Office documents. Afterwards, using macros from Office documents (Word, Excel) to deploy a Trojan that enables a foothold into the system. This happened for about 6 months before the outage attack. Next, when inside the system, attackers collected users credentials to access the ICT and control networks (using tools such as keystroke loggers). Afterwards, they established new connections to the control and/or field networks, using VPN connections. Then, they scanned the whole system to identify and recognize the existing control devices (RTUs and PLCs) and how they operated. At this stage, the UPS network of the offices and data-center was also identified and latter used to outage the power supply to those systems, after the attack. Afterwards, attackers uploaded the malicious firmware to the control devices and installed and customized KillDisk, to remove some external access points (used by remote managers of the system), to remove bootloaders from some workstations and servers, and to swipe logs.

The second step corresponds to the IACS attack. Attackers used their way in, previously obtained credentials and the existing HMIs to trigger an outage, in a very synchronized move, through three different electric providers.

At this same time, a DoS against the electric provides call centers was also performed – customers were not able to report the outage situation – and other substations where forced to operate in manual (human supported) mode.

Maroochy Water Breach: This 2000 security incident, in Queensland, Australia, was also a SCADA incident. Mr. Vitek Boden, as a revenge

for not being able to maintain his job, used a laptop to control 150 sewage pumping stations. He caused the release of millions of liters of sewage to the final local waterways (rivers, streams or lakes), polluting the existing ecosystem. He was then arrested and judged for these actions. The SCADA system was hijacked in order to bypass the treatment process, using the storm water drains. The attack included: (i) accessing the control SCADA network; (ii) reconfiguration of the pump stations software; (iii) DoS attacks of radio communications; (iv) lockup of computer communications without any logging; and finally, (v) malicious operation of pumps, changing the normal process flows.

From a taxonomy standpoint, attacks on SCADA/IACS systems can be classified as show on Table 1. This table classifies attacks into three main categories: layer 2 and 3 attacks, based on frame or stream manipulation for layer 2 or layer 3 protocol frames; protocol-level attacks, which relate to higher level protocols, both SCADA-related (such as Modbus) and general purpose (such as FTP or SNMP), implemented by abusing specifications or service vulnerabilities; and process-level attacks, which take place at the semantic process level, taking advantage of this knowledge to disrupt specific aspects of its operation. While the first two types of attacks can be detected through network and system-level monitoring mechanisms, the third type constitutes the most demanding kind in terms of detection – moreover if considering that these attacks constitute the ones with the highest potential of negative impact on operational safety and security.

Table 1: Simplified Taxonomy of SCADA/IACS attacks

Level	Class	Impact	Attack examples
Layer 2/3	Scanning/ Scouting	Getting information about network topology and devices	ARP or LLDP queries can be used to track devices; Probe for available services and protocols using a FIN or SYN scan
	Attack on data integrity	Unstable and/or unpredictable behaviour	Corrupt inflight data through packet manipulation
	Denial-of-Service and/or service degradation	Loss of visibility and/or control	Overwhelm or crash device, via SYN or ICMP flooding; Employment of CAM table overflow to disrupt communications
Protocol/ service level	Scanning/ Scouting	Getting information about service and device capabilities	Function Code scan attacks for device profiling; Use of MITM to analyse used services and protocols
	Integrity	Unstable and/or unpredictable behaviour	Abuse of protocol specifications and features, such as the Modbus Invalid Read Payload Size or the Negative Sensor Measurement attacks
	Denial-of-Service and/or service degradation	Loss of visibility and/or control	Exploit vulnerability to crash or disable service or device (such as a FTP buffer overflow); Introduce latency or communications failures through MITM attack; Use of administrative modes to start/stop PLC; Modbus Invalid CRC attack
Process level/ semantic	Scanning/ Scouting	Reveal details about the nature of the process	MITM attack for scouting purposes or preparation of replay attack; Use of of Modbus FC 90 to download ladder logic; Structural analysis of memory map thorugh probing and exception interception
	Direct manipulation	Manipulation of process variables	Manipulation of process variables to alter behaviour, through direct device access
	Interception and fuzzing	Interception and manipulation of process values	Manipulation of process variables to alter behaviour, through command injection or protocol fuzzing, using a MITM (via ARP poisoning or CAM table) attack to intercept communications and conceal the intruder; Process-aware response injection or replay attacks
	Reprogramming	Process behaviour is modified and/or hijacked	Use of of Modbus FC 90 to upload ladder logic rogue code

3.4. The Need for Domain-Specific Monitoring

While both the IACS community and cyber-security experts are actively working towards identifying and solving cyber-security issues, the differences between the ICT and IACS contexts mean that there is no "one size fits all" solution when it comes to choosing and deploying security mechanisms. While importing solutions from the ICT world is often a necessity, it might lead to undesirable side effects. The fundamental premises for ICT security tools and commonplace lifecycle management procedures, such as patching and updating a system, can become troublesome in an IACS when faced with situations such as the impossibility, the high cost of stopping production or even the explicit prohibition by the system's manufacturer, as it may happen with operating system updates or patches not previously certified by the equipment provider. Moreover, mature systems are often kept in operation, sometimes far beyond their projected lifetime, constraining the implementation of some security measures, as existing equipment may lack the necessary requisites [25].

Altogether, this situation requires the development of domain-specific safety and security mechanisms for IACS, such as the Shadow Security Unit for stealth monitoring of RTUs, PLCs, IEDs and similar devices, which is the main subject of this paper and will be presented in the next section.

4. The Shadow Security Unit

As already discussed, many successful attacks on SCADA systems have used intrinsic vulnerabilities of RTUs, PLCs and IEDs which traditional security monitoring frameworks failed to timely detect. This state of affairs is

the main driver for our proposal of a specialized monitoring probe focused on such devices: the Shadow Security Unit (SSU).

4.1. The SSU concept

The role of the SSU on a IACS security management framework is depicted in Figure 3. The SSU is essentially a probe that is attached in parallel to the RTU/PLC/IED device, passively monitoring the network communication flows and the physical process interfaces, in order to detect anomalies with potential impact on system safety and security. The SSU is managed by and reports to the global IACS Security Management Platform – preferably using out-of-band communications for increased stealthiness and resilience. Depending on the adopted strategies, collected monitoring data is processed at local and/or global level.

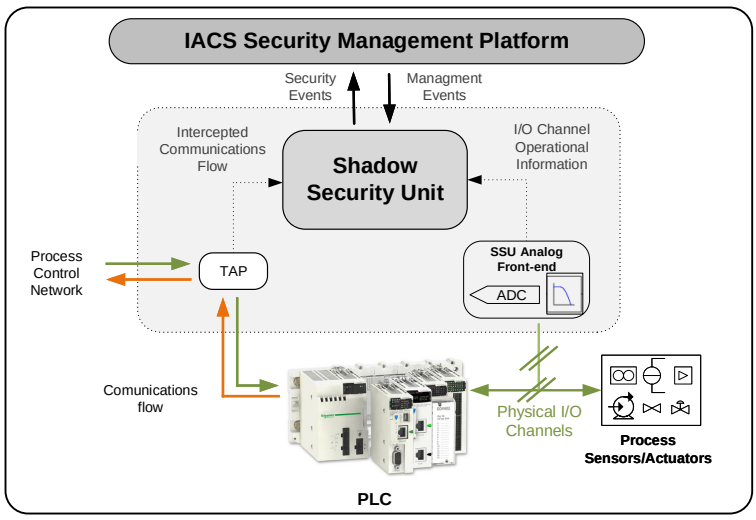

Figure 3: Deployment of the Shadow Security Unit

The SSU concept builds on the following key principles and requirements:

- **Passive Monitoring** – by design, the SSU should not be able to interfere in the normal systems operation, thus avoiding the introduction of additional points of failure from the viewpoints of security or resilience.

- **Stealthiness** – the SSU should not be visible to outside attackers, which in addition to the fore mentioned passive monitoring implicates the possibility of using off-band communication channels with the IACS security management platforms.

- **Cost-effectiveness** – The SSU must be cost-effective when compared with traditional resilience mechanisms of mission-critical systems, such as byzantine voting mechanisms (even if it is not directly competing with those systems).

- **Close coupling** – each SSU focuses on the monitoring of a single device, to which it is physically coupled (especially to simplify the deployment process and the monitoring of physical I/O).

- **Complementarity with existing solutions** – the SSU is intended to complement and integrate with, rather then replace, existing security monitoring platforms.

In addition to these design principles, the SSU is expected to support the following functionalities:

- **Semantic Command Stream Processing** – the SSU should be able to transparently capture and decode the SCADA protocols that provide data and control synchronization between devices and supervisory/master stations, whether polling-based (e.g. Modbus, IEC-104)

or event-based/hybrid (e.g. DNP3). The SSU passively captures data at the lowest level to provide a copy of the communications feed for semantic analysis purposes, but also allowing the detection of issues such as various types of communication errors.

- **Continuous Network Flow Monitoring** – once configured or trained with thresholds for typical bandwidth usage and traffic patterns on certain control channels (such as Ethernet), the SSU should be able to detect when some of these parameters change significantly (e.g. packet rates, inter-message arrival times). Such changes may be related with malware infection, Distributed Denial-of-Service attack (DDoS), flooding, brute-force attacks or equipment failures. Additionally, it is also an useful mechanism for detection of zero-day threats.

- **Message Integrity/Trust Checks** – the SSU must provide the means to detect and report scouting attempts, spoofing attacks or the sending of unauthorized commands (from potential attackers) from a rogue host (a typical attack scenario for many SCADA protocols that accept multi-master operations). Communication flow monitoring provides the means to detect: the presence of rogue hosts not involved in process awareness or control interactions; attacks against protocol integrity; and situations where certain commands not used on a daily basis are issued (e.g. administrative commands). Moreover, with the support of an external service, the SSU should be able to detect inconsistencies in process control flow messages exchanged with other hosts – therefore performing tampering checks by verifying if commands have

not been changed in transit – or creating a closed loop for communication flow checking at the endpoints.

- **I/O sampling and processing** – the SSU should able to acquire information about the status of the I/O device lines attached to the physical process, for integration into its local analysis or simple reporting for the upper security components. The integration of this information is a distinctive feature of the SSU concept, since it enables detecting situations where the monitored device is not operating as expected – due to failure or malicious behavior.

- **Continuous Behavior Analysis** – a Man-in-the-Middle attack or a direct attack to an HMI or Master Station may provide the means to inject malicious code into the RTU/PLC [26] or to send commands to disturb operation. To address this, the SSU should support correlation of communication stream analysis and anomaly detection on physical I/O interface channels (that interface with sensors and actuators on the field), in order to perform continuous behavior audits of the monitored device. This information can also be used to assess the operational and health status of the device, from a safety point of view.

When integrated into larger IACS security management frameworks, the SSU (i) enlarges the traditional monitoring perimeter – by focusing not only on ICT security but also on process monitoring – and (ii) adds some redundancy to traditional Network IDS. While a conventional signature-based Network IDS may share some of the SSU capabilities (particularly those related to Layer2/3 analysis), there are limitations: when deployed in-line,

a NIDS constitutes an undesirable point-of-failure that might also degrade latency (a sensitive issue in real-time control scenarios); when deployed in passive mode, its effectiveness may be hampered by contention, especially in large scale scenarios – when the monitoring interface capacity of a switch is insufficient to handle the aggregated traffic from the source ports, overflow packets are dropped. Eventually, using a NIDS might not be feasible at all if the RTU/PLC/IED is deployed on a remote location served by a low bandwidth connection (such as VHF links used for telemetry). Moreover, a traditional NIDS lacks the semantic processing and physical interface analysis capabilities that are provided by the SSU, together with its behavior analysis mechanisms.

Once the information is collected (and optionally locally processed), the SSU may feed a Security Information and Event Management (SIEM) platform and/or the Security or SCADA Control Rooms.

4.2. Architecture

The SSU architecture is presented in Figure 4. This architecture is a neutral concept that supports the majority of SCADA protocols (e.g. Modbus, IEC-104, Profinet/(I)RT, DNP3), communications technologies (e.g. Ethernet, RS-485, Profinet, Ethernet/IP) and deployment scenarios. Nevertheless, for sake of readability, some of the technical details discussed in the next sections directly relate with a proof-of-concept built for a SCADA system using Modbus/TCP and IEC-104 – since certain features, such as the communication interception methods, depend on the nature of the physical medium and communications technology being used (for instance, Ethernet-based, star and industrial ring topologies require different approaches than serial buses).

Next, we discuss the main building blocks of the SSU architecture: communication stream analysis, physical I/O probing, automated learning, eventing and reporting, management, and watchdog.

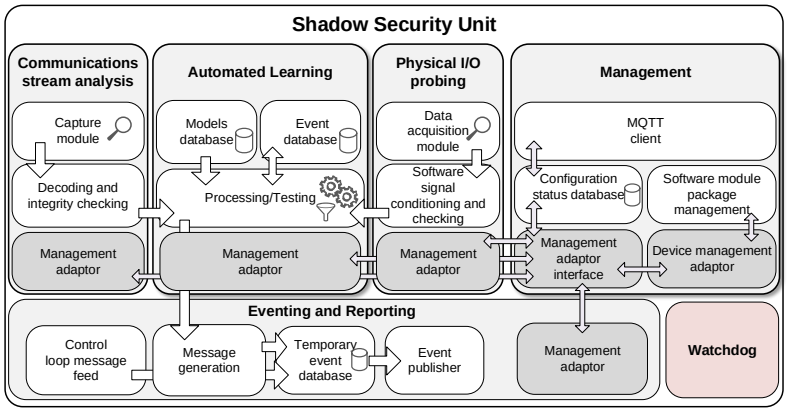

Figure 4: Shadow Security Unit architecture

4.2.1. Communications Stream Analysis

This module is responsible for passively capturing and processing the command flow between the master station and the RTU/PLC/IED. For Ethernet-based systems, for instance, a low-cost TAP module may be deployed between the monitored device and the upstream link of the communications infrastructure, providing a copy of all network traffic to the SSU, while requiring little change to the existing setup.

The SSU network interface connected to the TAP is devoid of IP configuration and does not reply to ARP requests (a requisite for transparency), working in promiscuous mode to capture all the network traffic, redirecting it to an internal software module. Next, a rule-based mechanism filters network events of interest, so that they can be stored and processed afterwards.

Once captured, the protocol stream will be analyzed, looking for inconsistencies in frames and protocol data units – for Modbus/TCP, a protocol decoder allows the SSU to extract and decode the semantics of each Protocol Data Unit (PDU), also enabling tracking the messages exchanged with the monitored device. Moreover, the analysis process calculates bandwidth, packet rate/size and inter-arrival latency metrics for further processing by the *Automated Learning* module.

4.2.2. Physical I/O Probing

Physical I/O probing is the capability of assessing in soft-realtime the state of the physical (discrete and/or continuous) inputs and outputs of the RTU/PLC/IED. This allows the SSU to sense the state of those channels, in order to directly get information about the process. For such purpose, the SSU bundles data acquisition capabilities – for instance a signal preconditioning stage coupled to a precision differential voltage probe using operational amplifier technology feeding an 8 channel, 10-bit, successive approximation (SAR) ADC (Analog to Digital Converter) capable of an overall conversion rate of 200K samples per second (with an effective sampling rate of 6KSps per channel on the prototype described later in this paper, corresponding to a 0.1ms latency – compatible with the scan cycle latency of most RTU/PLC/IED devices), adequate for discrete and most continuous/analog control I/O modules. For more precise acquisition, the increased sensitivity may require a 12 or 16-bit ADC.

Afterwards, a software module uses routines that are executed within the scope of a real-time OS process context to capture the physical I/O data stream at a fixed sampling rate, using time-stamping to label the cap-

tured data for later correlation with the information from the *communications stream analysis* module. In order to properly interpret the captured data, the software requires information about the characteristics of each physical interface such as the direction (input/output), type (discrete/analog) and voltage ranges, which are configured using the SSU management interface. A software signal conditioning and checking module applies a set of processing techniques for anomaly detection on the captured signal feeds, whose output is fed to the *Automated Learning* module.

Due to its nature, the *physical I/O probing* module also provides the means to remotely diagnose the health of the components of the physical process interface. For instance, a floating input might be a symptom of sensor failure.

4.2.3. Automated Learning

The information from the communications and I/O probing modules is fed to a local processing module, performed by an embedded correlator component. This correlator assesses the status of the monitored device, checking if commands arrive from a legitimate source, if they are coherent with the expected control interface flow, and if I/O information is in-line with expected values. For discrete control scenarios, the correlator also incorporates a series of logical state maps (extracted from the ladder logic programming diagrams of the RTU/PLC/IED or manually provided by the operator) that are used to check the behavior of the monitored unit. It also provides preprocessing capabilities, for event reduction and aggregation within predefined time windows. Correlation rules can be customized and managed using the management interface.

Thanks to these capabilities, combined with the communications analysis and physical I/O preprocessing stages, the SSU incorporates behavior-based and behavior-specification based monitoring features, which are often judged to be the most effective in terms of zero-day and rogue threat detection. For discrete control scenarios, the correlator can also relate command interactions with physical I/O behavior, as the protocol decoder module updates a memory map replica of the RTU/PLC/IED device that is kept updated accordingly with the intercepted communication interactions – it must be clarified, however, that this does not guarantee that such a memory map will be completely synchronized with the one on the monitored device. This replica is used to provide a way of storing semantic information about interactions that can be encoded into a correlator rule, using a register reference, therefore making it possible to avoid generating false positives in situations where a legitimate command or process state update was issued, affecting the process behavior.

In the cases where the SSU is implemented using a low-cost Single Board Computer (SBC) system, there is a limitation in terms of processing capabilities that restrict the ability to perform deeper analysis of the information that is captured. For this reason, the SSU is able to operate in a flexible way, either in standalone or as part of a security assessment loop, integrated within a distributed architecture, fully leveraging its capabilities.

4.2.4. Eventing and Reporting

The SSU processing module generates events that are sent to Security Management Platforms such as a SIEM. Generated events are encoded using Apache Avro [27] with a proper data model – its format is flexible enough to

be customized for sending safety-related information or even raw data, when needed. Event transmission uses a publish-subscribe messaging pattern, in which the SSU publishes events into a Message Queue (MQ) system that feeds one or more consumers (subscribers), such as a SIEM. The use of a MQ system provides scalability and reliability for the message stream, with message ordering guarantees.

Moreover, as the SSU can be used to create command flow control loops (using an optional external component), the *event and reporting* module is also able to replicate copies of communication control flows and physical I/O data to a separate queue, for consumption by an external message checking system or a high-level analysis mechanism (such as an anomaly detection component).

4.2.5. Management

The SSU should be seen as one of the components of a wider security management platform – therefore requiring appropriate management capabilities for the purposes of deployment, monitoring and configuration management. The SSU management module exposes the configurations properties and the device status to the integrated security management platform (preferably using out-of-band communications, for the already aforementioned reasons). Some of the functionalities provided by this module include: remotely starting/stopping the component; checking the status of component modules; and read/changing configuration parameters. The specific details of this management module – which uses MQTT [28] – fall outside the scope of this paper.

4.2.6. Watchdog

A watchdog module provides in-device monitoring of component and service operation. This component works at two levels. First, it implements a series of software routines that periodically check component operation and attempt recovery or restart in case of stalled operation. Second, it provides system-level checks through a kernel module working together with a watchdog service that provides a regular feedback to the hardware watchdog timer of the SBC. Using the hardware watchdog allows the possibility of rebooting the entire SSU platform in case of a critical failure, after a predefined number of missed timer events.

4.3. Proof-of-Concept Prototype

In this section we overview a proof-of-concept implementation of the SSU, which has been built for demonstration and testing purposes.

Several SBC alternatives were evaluated, in order to strike a balance between cost, size and capabilities. Eventually, the Raspberry Pi (RPi) [29] was selected due to its wide availability, ease of development, expandability and price/performance ratio. The RPi provides reasonable processing capabilities and can accommodate a second network interface (for the out-of-band connectivity), using an USB-Ethernet adapter. Moreover, a hardware watchdog is already available, thanks to the native CPU watchdog driver for the RPi [30].

The SSU Operating System (OS) is based on a customized Linux distribution, using the Xenomai [31] extensions for providing soft real-time scheduling capabilities. An MCP3008 [32] 10-bit ADC coupled to a set of differential voltage probes provides physical data acquisition, feeding the soft real-time

data processing routines implemented using the Xenomai API. The embedded correlator engine used for local correlation rules is based on the Simple Event Correlator [33]. Packet capture is implemented using libpcap [34], together with pcap-filter for packet filtering. Protocol stream decoding was implemented using the pymodbus [35] library.

Figure 5 shows the external look of the proof-of-concept. An IP65 enclosure guarantees protection against dust. External connections use XLR type connectors as specified by IEC 61076-2-203 [36], providing easy assemblage, simple use and reliable and robust connections. A two-line dot matrix display reports the device operational status for on-site interventions. The whole prototype was built using commercial of-the-shelf hardware components and with no special care for packaging or space optimization, and therefore a mass-produced SSU only requires a fraction of the footprint of this prototype.

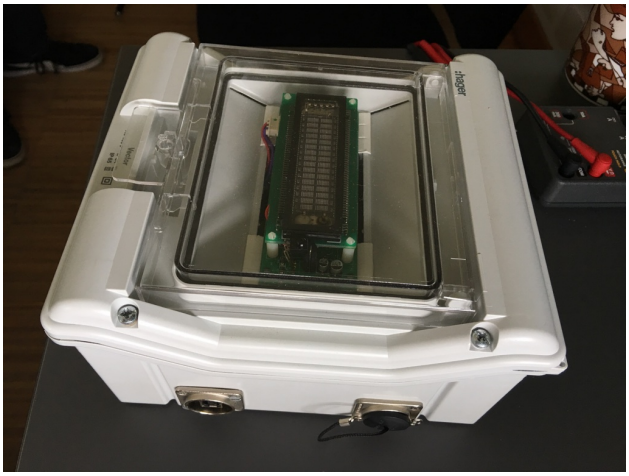

Figure 5: SSU Hardware Prototype

Overall, this proof-of-concept shows it is possible to achieve the main

requirements identified for the SSU:

- The device does not interfere with the normal control process, resorting to fully passive acquisition of data communications and physical I/O.
- The device has enough computing power for meeting the processing requirements of most usage scenarios, despite using a widely available low-cost SBC.
- The overall bill-of-materials is within reasonable boundaries. The proof-of-concept has cost less than 120 EUR, using commercial of-the-shelf components, and a mass produced system could easily cost a fraction of this value (less than 50 EUR, for the most common types of physical I/O tapping scenarios).

5. Leveraging the SSU as an Intelligent Edge Probe

The SSU is a generic device that can assume multiple roles within the scope of an Intrusion and Anomaly Detection Systems (IADS), providing the means to extend its reach towards the edge of the IACS infrastructure. Considering its unique location and capability for combined monitoring of network communications and the physical process, the SSU can add new capabilities to traditional IADS systems, including:

- Fine grained analysis of local process/device, with fast response times.
- Detection of on-site intrusion by means of physical tampering.
- Detection of compromised firmware at the RTU/PLC/IED.

- Enrichment of the knowledge of the detection system, with the monitoring of process I/O.
- Avoidance of propagation of local anomalies.
- Feeding of global/centralized level IADS with enriched local level knowledge.

Next, we identify some of those possible roles, as an overview of the SSU potential capabilities. It should be noted, also, that the criteria for choosing which application profiles may be suitable for SSU deployment is also affected by the amount of prior knowledge regarding the specific characteristics of the process under control. In some cases, a *black-box* approach may be adequate, using a two-stage deployment strategy encompassing a preliminary analysis process, for collection and characterization of the nominal physical and ICT inputs/outputs status, followed by a detection stage. In other situations, pursuing a *white-box* approach may be preferable, using partial knowledge about the controlled process to predetermine the expected behavior of the system.

5.1. Shadow replicator

The simplest way of using the SSU is to feed the information collected by the SSU (control network and/or physical I/O) to an upper processing unit. This approach allows a reliable feed of the global level IADS for mapping, correlating or performing any other processing analysis incorporating local level knowledge. Man-in-the-middle attacks, for instance, are immediately detected by comparing commands issued by SCADA servers and corresponding responses from the PLC at both ends of the communication.

5.2. Protocol inspector

The SSU can host an on-line anomaly detection system that relies on time-series analysis of residual prediction errors to provide semi-supervised anomaly detection. Due to the nature of the SCADA process and field network environments, there is a considerable degree of stability and homogeneity (especially in comparison with ICT networks), regarding the mix of involved protocols, hosts, network traffic flows and physical I/O — making this approach particularly appropriate, not only for cyber-security monitoring but also for safety monitoring and diagnostics, since some detected patterns (such as an unusual number of TCP RST messages, for instance) may be symptomatic of communications issues or equipment malfunction.

Moreover, this technique may be complemented with an authorized IP/MAC address monitor (using a MAC/IP address list for authorized hosts and individual associations). Scanning patterns may be detected with the help of a Portscan Attack Detector suite (e.g. the psad module [37]), fed by packet information — generated for instance by a set of Netfilter iptables rules [38] that receive the feed via an internal bridge, created using ebtables [39]. The iptables rules log communications transactions directly to a file on a small ramdisk (for performance reasons this file may be trimmed at regular intervals, in order to implement a moving window for analysis). Therefore, the SSU is able to detect TCP SYN, NULL and XMAS scans, UDP scans and OS fingerprinting attempts.

5.3. Automated Learning Module

The proximity to data sources, which translates to the richness of these data, guarantees the implementation of classification algorithms to detect

anomalies in the process that can later be considered malfunctions of the equipment or attacks with malicious intentions.

The range of anomaly detection techniques that can be applied in this scenarios is not much different from the techniques used in classic set-ups — the main difference being the already mentioned richness of available data. For instance, the following types of anomaly detection approaches (as summarized by [40]) can be applied: classification based approaches(including neural networks, Bayesian networks, support vector machines and rule-based techniques), nearest neighbour-based techniques, clustering approaches, statistical analysis and spectral analysis.

Hardware limitations, however, may impose specific adjustments to the adopted learning techniques (e.g., models may need to be trained outside the SSU, with the SSU device hosting only the classification phase).

5.4. Logging System

The close proximity of the SSU regarding the physical process provides the means to improve the existing security and/or monitoring systems with enhanced and fine-grained logging capabilities, which can be context sensitive and adjusted to the operator needs. This could be especially useful for forensics and security compliance analysis, due to the potential usefulness of the SSU as an evidence collection mechanism deployed at the edge of the IACS, next to the cyber-physical domain perimeter.

5.5. Data/feature Extraction for Machine Learning Mechanisms

The SSU can also be used for feature extraction to feed external processing units. The proximity to the network and physical IO data is a good reason

to deploy at the edge this feature extraction mechanism providing a reacher feature set to the processing units.

6. Validation

In this section we discuss the validation of the SSU concept. First, we introduce some of the most typical attacks and discuss how the SSU can help detecting them. Second, we present the results of an experimental evaluation work based on the aforementioned proof-of-concept implementation and addressing the SSU functionality, performance and reliability.

6.1. Functional Evaluation

This subsection will discuss the functional evaluation of the SSU, considering a set of attack use cases representative of the categories previously introduced in Table 1. In order to ease the description of the SSU features that provide the detection capabilities for each use case, the corresponding attacks will be also described in detail.

6.1.1. Layer2/3 Device Probing Attack: Network FIN/SYN Scan

A typical first step for an attacker is to perform a scouting survey of the infrastructure. In this perspective, network scan procedures are a valuable tool to get an overview of the network elements and its topology. There are several techniques to perform this on a TCP network, such as SYN or FIN scans [41] — the latter being also known as *stealth scans*, as some firewalls may log SYN (half-open) attempts to restricted ports. Moreover, a FIN packet sent to a closed port on certain hosts (mostly UNIX-based platforms

and most RTU/PLC/IED devices) will pass undetected, generating a RST response or being ignored for an open port.

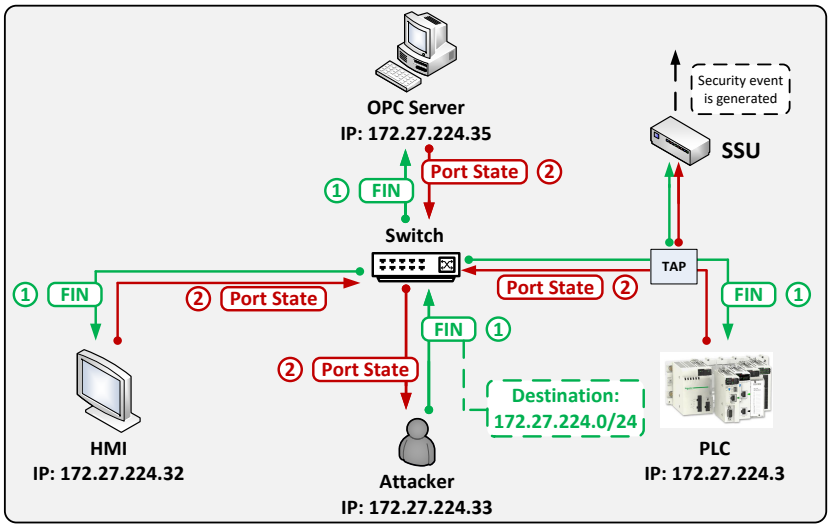

Figure 6: FIN Scan on a network address range

Figure 6 illustrates such a scenario where an attacker continuously sends FIN probes to devices on a network scope and waits for the responses. While a Network Intrusion Detection System (NIDS) may detect such an attack, it must be configured in monitoring mode, receiving the traffic for the relevant hosts — eventually, this might not be feasible if the RTU/PLC/IED is deployed on a remote location, served by a low bandwidth connection (such as VHF links used for telemetry), whereas the SSU will be able to operate.

In these scenarios, two mechanisms allow the SSU to detect suspicious activity. First, the internal authorized device IP/MAC list provides the correlator with information about the devices that are supposed to interact with the monitored device. Second, the psad services of the communications stream analysis module are able to detect inconsistencies such as SYN, FIN,

or even fragmentation scans.

6.1.2. Layer2/3 DoS Flooding Attack: SYN flood

Denial-of-Service attacks can be implemented using flooding techniques such as Smurf or SYN flooding [42]. SYN flood attacks, illustrated in Figure 7, explore the nature of TCP connections by sending a series of SYN packets (signaling the start of a TCP connection) to the RTU/PLC/IED, which responds with a SYN-ACK, never acknowledged (ACK) by the sender.

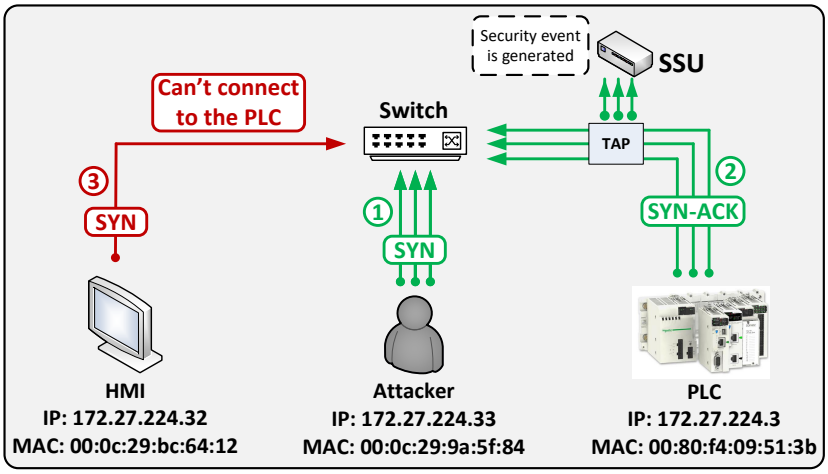

Figure 7: SYN flood attack

A SYN flood creates large numbers of half-open connections on the attacked device (waiting for an ACK or a timeout) that might eventually lead to resource exhaustion, with the device ceasing to answer new requests. A variation of this attack forges raw packets with a false sender, so that the SYN-ACK replies also flood a target, via backscatter (a Smurf attack uses ICMP packets for the same purpose).

In such flooding scenarios, the SSU is able to detect suspicious activity thanks to the internal authorized device IP/MAC list and the communications stream analysis module, which is able to detect the SYN flood pattern over a time window. In the event that the SYN packets are forged with a device with a legitimate sender IP address (to create a backscatter situation), the SSU is able to detect the attack because of the flow pattern analysis features (using behaviour signatures and packet rate analysis).

While a conventional NIDS may deal with some of these threats, there are two limitations that make them unsuitable for large-scale scenarios. First, when deployed in inline mode, it constitutes an undesirable point-of-failure that might also degrade latency (something considered critical, especially in systems dealing with strict, real-time, control). Second, when deployed in passive mode, its effectiveness may be hampered by contention - when the monitoring interface capacity is insufficient to handle the aggregated traffic from the source ports, overflow packets are dropped.

6.1.3. Man-in-the-Middle with ARP Poisoning

A Man-in-the-Middle (MITM) attack corresponds to a situation where a third-party becomes involved in the middle of the communication stream while remaining unnoticed. For instance, an attacker may fool an HMI by directly interacting with it and providing apparently normal process data (e.g. obtained from a previous survey of device interactions and later replayed) while attacking the PLC in the background. MITM attacks can be implemented using several techniques, ranging from ARP poisoning [43] to routing redirection [44]. Figure 8 illustrates the first scenario.

In the first stage of the ARP poisoning MITM, the attacker generates a

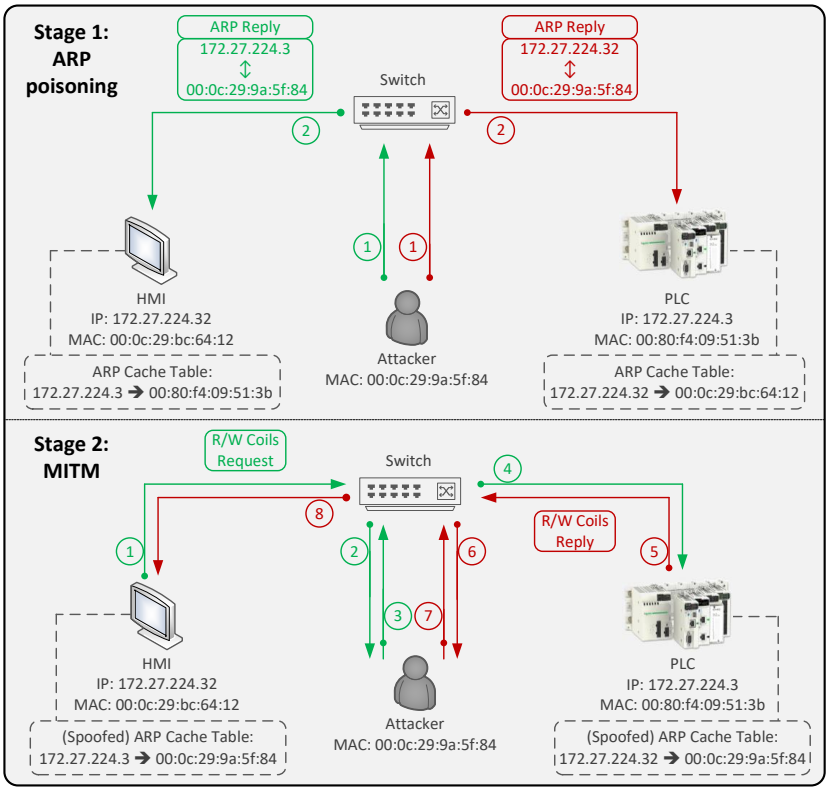

Figure 8: MITM attack using ARP poisoning (adapted from [45])

series of unrequested ARP replies for both the HMI and the PLC (Figure 8, upper half), poisoning the local ARP caches in such a way that the MAC address of the attacker system becomes associated with the IP of the HMI for the PLC and the IP of the PLC for the HMI, respectively. Further interaction attempts from the HMI to the PLC will be redirected to the attacker system, and vice-versa.

In the second stage, the attacker intercepts connections in real-time using a packet manipulation tool (such as SCAPY [46]) to perform session hijacking on the TCP connection (Figure 8, lower half). In alternative, the attacker might provide a fake device for the HMI to interact with, using a Mod-

bus/TCP simulator programmed with information obtained from a previous survey or a script that replies with protocol interactions previously recorded, corresponding to an apparently normal operation scenario – Figure 9 depicts such an attack, including the SSU MITM detection capabilities.

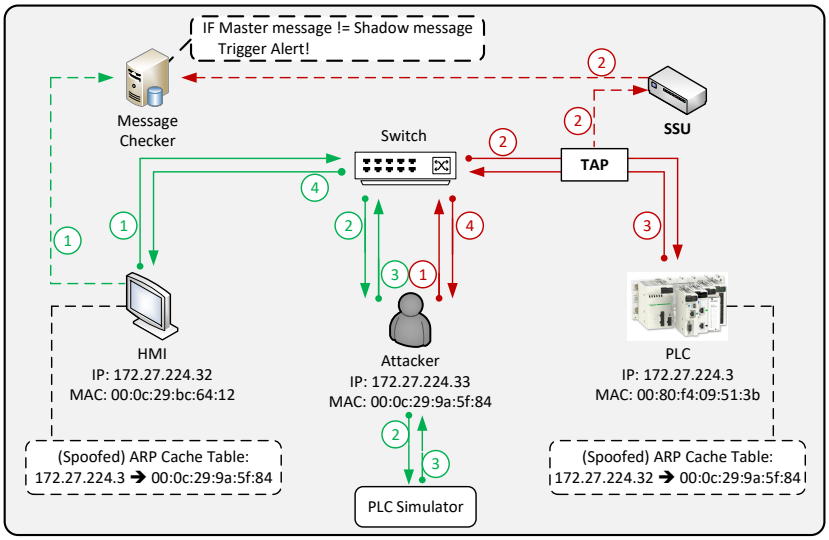

Figure 9: SMU/Message Checker operation against MITM attack (adapted from [4])

After the ARP poisoning is successfully implemented, the attacker system runs a PLC simulator to keep the HMI unaware of the real PLC status. The SSU can be effective against this attack as it is configured with the IP/MAC addresses of the systems that are authorized to interact with the monitored device — this same strategy is equally effective for the real-time protocol manipulation scenario.

However, ARP poisoning attacks can be avoided by using static ARP lists or managed switches featuring port security or dynamic ARP inspection mechanisms (which a large number of SCADA operators do not use, whatsoever). However, despite such security mechanisms, the SSU remains an

effective proposition as it is capable of detecting other kinds of MITM attack techniques. For instance, Spanning Tree Protocol [47] or routing redirection attacks, when used together with MAC address spoofing, may overcome the detection capabilities provided by MAC/IP lists. To deal with these situations, the already mentioned shadow replicator mechanism enabled by the SSU (cf. Section 5.1) can report issued and received commands to a message checker system (cf. Figure 9), therefore providing a closed loop for verification of the integrity of the control flow in an end-to-end basis, overcoming the most sophisticated MITM scenarios.

6.1.4. IED/PLC Reprogramming

The SSU can be effective against a IED/PLC reprogramming attack (or even a successful direct attack against the IED/PLC that is not otherwise detected) due to three features: authorized IP/MAC system lists (already discussed, being used to detect unauthorized device access), communications stream analysis and behaviour checking.

The communications stream analysis module can provide information for the correlator about the semantics of a specific operation. Particularly, as code maintenance on a IED/PLC is not a frequent procedure, it makes sense to track such operations at the protocol level – as an example, the Schneider Modicon PLC series use the Modbus function code 90 (0x5a) [48] to exchange ladder logic using unauthenticated transfers, which can be used by an attacker to retrieve or upload the PLC logic programming. The SSU is able to detect such operations and generate security events accordingly.

Behaviour checking includes several techniques, such as those already discussed in Section 5. These behaviour analysis capabilities are effective

Stuxnet-like scenarios, providing the means to detect and report the abnormal behaviour of a reprogrammed PLC – even if the reprogramming is provided by a legitimate, yet compromised, Master Station or HMI.

6.2. Experimental Measurements

For the purpose of testing the SSU, a small scale 100 Mb/s Ethernet LAN testbed was created, reproducing the minimum components of a SCADA system (see Figure 10). Experiments focused on three aspects: functionality, reliability and performance.

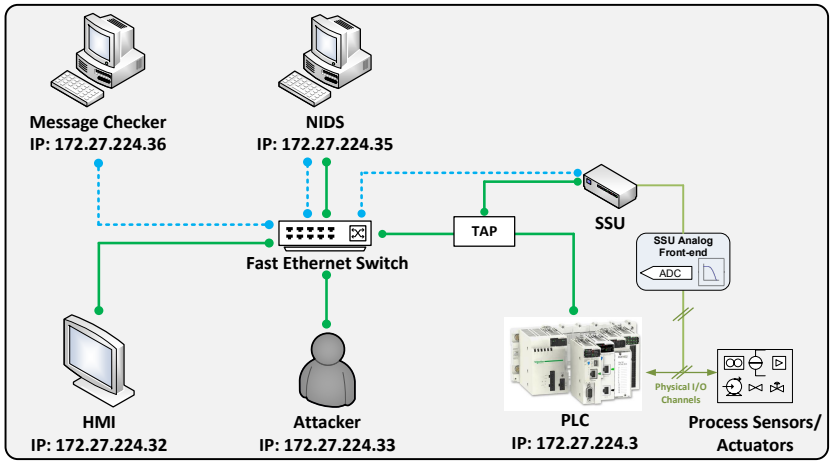

Figure 10: SCADA testbed for SSU evaluation

Functional tests validated the SSU capabilities according to the use cases from Section 6.1, with the SSU behaving as expected.

Reliability tests were conducted using both CPU and storage (SD card) stress tests. These consisted, for the CPU, in using the Livermore Loops Stability Test [49, 50] with a default clock speed of 700MHz, continuously running over a 4-day period in a room with an ambient temperature between

23°C and 18°C, reaching a maximum temperature of 58°C – despite this, a passive heatsink was added to reinforce the thermal operation margin.

Storage stress tests (including continuous write loops and unattended power cycling) revealed a considerable probability of flash memory wearout or filesystem corruption to occur within a short time interval (from days to weeks). To avoid such issues, the SSU firmware image operates from a filesystem in read-only mode, avoiding write cycle wearout, using a USB flash drive for scratch, database and log storage.

Performance validation was focused on analysing the capabilities of the RPi as a packet capture device, using Iperf [51] to execute 15 rounds (120s each) of unidirectional UDP stream-tests between two computers in the testbed, using a 500-byte packet size (a Modbus message has a maximum size of 300 bytes [52]), with different bandwidths (see Table 2).

Table 2: Average results for network overhead tests (STDEV in parenthesis)

Test rate (Mb/s)	10	20	30	40	50	60	70	80	90
Jitter (ms)	0.17 (0.02)	0.27 (0.01)	0.35 (0.02)	0.48 (0.01)	0.53 (0.03)	0.67 (0.01)	0.80 (0.01)	0.91 (0.01)	1.16 (0.13)
Captured packets (%)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	98.1
System load(%)	8.1 (0.4)	20.8 (0.5)	34.5 (2.4)	46.4 (2.1)	57.9 (4.3)	70.0 (1.9)	83.6 (1.7)	91 (3.4)	99.8 (2.1)
Capture overhead (%)	71.1 (0.8)	98.5 (3.2)	98.3 (3.3)	99.1 (3.6)	99.4 (2.1)	98.7 (3.69)	96.8 (4.2)	99.2 (5.2)	99.4 (4.4)

The SSU was deployed to monitor one of the testbed PCs (instead of a PLC/RTU, for test purposes), using tcpdump [34] for packet capture and

mpstat and pidstat [53] to collect system statistics. The SSU was configured with a minimum service set, using a ramdrive to store test data – data collection overhead was also accounted for, constituting 0.6% to 1% of the total load, on average.

Tests demonstrated the SSU to be capable of capturing network streams with no packet loss, in most cases. However, above a 60 Mb/s threshold, the packet capture overwhelms the CPU, leaving little room for other tasks. In extreme cases, (90 Mb/s) CPU starvation causes packet drops. Further analysis (see Figure 11) revealed a considerable overhead for soft IRQ processing, consuming over 50% of the available CPU capacity, for high UDP rates. This was traced back to the Ethernet interface implementation of the RPi that uses a USB device (instead of a dedicated or PCI bus), which is prone to performance and overhead issues.

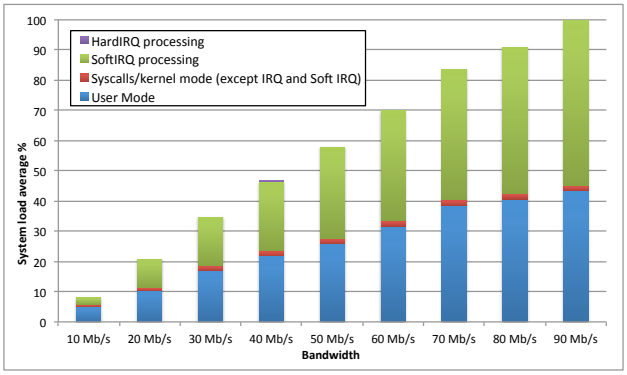

Figure 11: Average CPU overhead for network tests

As typical Modbus/TCP traffic patterns do not exhaust the capacity of a Fast Ethernet link, the problem of network processing overhead does not affect the SSU in typical deployment scenarios, as it only becomes critical for higher packet rates – nevertheless, the next interactions of the prototype

will address this issue.

7. Related Work

To the best of our knowledge, the SSU is the first proposal for adding an external watchdog mechanism to mainstream PLC/RTU/IED devices, capable of observing both network communications and the interactions with the physical process under control (physical I/O tapping). Nevertheless, a few other works have focused on somehow integrating direct and indirect physical process monitoring into IDS frameworks.

Hadziosmanovi et al. [54] presented a detector that continuously monitors the variables of a process and then creates specific prediction models for these variables, allowing to estimate the future activity. The traffic of the control network of two water treatment plants, which serve a total of about one million consumers, was used as reference scenario. The use of a network tap allowed a passive redirection of the PLCs traffic to an implemented IDS. This IDS implements a time series approach for each variable (extracted from the Modbus/TCP messages), to infer its expected activity and to detect attacks. Modeling and detection used auto regression and control limits, as the main goals were techniques that are relevant to the context and do not pose strong assumptions on the monitored time series.

A work addressing semantic attacks that change the way PLCs operate is presented in [55]. These attacks may compromise the firmware of target PLCs, and manifest themselves by changing the normal behaviour of the physical process under control. The authors used an Arduino (embedded micro-controller) to create a non-intrusive device for runtime monitoring

using interval temporal logic, a formal language, for safety and security monitoring.

Hidden Markov Models (HMM) are used by [56] to infer the network protocols from the existing traffic in that same network. This approach uses ϵ-machine reconstruction to infer the minimal deterministic HMM of a protocol. The parameters are obtained directly from the data overcoming the problems of feature selection.

Sequence-based event attacks are covered by [57]. This approach shows that a sequence of events (individually considered normal) goes unnoticed by traditional intrusion detection systems, yet represents a threat to the system. It proposes a Sequence-aware Intrusion Detection System (S-IDS) and a possible architecture to overcome this issue.

8. Conclusion

In this paper we proposed the SSU as a cost-effective solution for monitoring process control devices such as PLCs, RTUs and IEDs. The SSU represents an attractive addition to current mainstream monitoring approaches (which lack the capability to detect sophisticated attacks such as Stuxnet), when compared with the costly and complex redundancy solutions adopted in avionics and similar mission-critical systems.

Overall, the SSU represents an important contribution to enlarge the scope of cyber-detection capabilities to the physical processes under control – helping to bridge the current gap between ICT security and critical infrastructure protection as whole.

In this paper we introduced the SSU concept, together with a reference

architecture and a proof-of-concept implementation. We also discussed some of the possible approaches for exploiting the cyber-detection capabilities enabled by the SSU device.

Acknowledgement

This work was partially funded by the ATENA H2020 Project (H2020-DS-2015-1 Project 700581).

References

[1] M. Castro, B. Liskov, Practical byzantine fault tolerance and proactive recovery, ACM Transactions on Computer Systems 20 (4) (2002) 398–461. arXiv:arXiv:1203.6049v1, doi:10.1145/571637.571640.
URL http://portal.acm.org/citation.cfm?doid=571637.571640

[2] R. Langner, To kill a centrifuge: A technical analysis of what stuxnet's creators tried to achieve.
URL https://www.langner.com/wp-content/uploads/2017/03/to-kill-a-centrifuge.pdf

[3] T. Cruz, J. Barrigas, J. Proenca, A. Graziano, S. Panzieri, L. Lev, P. Simoes, Improving network security monitoring for industrial control systems, 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM) (2015) 878–881doi:10.1109/INM.2015.7140399.
URL http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7140399

[4] T. Cruz, L. Rosa, J. Proença, L. Maglaras, M. Aubigny, L. Lev, J. Jiang, P. Simões, A Cybersecurity Detection Framework for Supervisory Control and Data Acquisition Systems, IEEE Transactions on Industrial Informatics 12 (6) (2016) 2236–2246. doi:10.1109/TII.2016.2599841.

[5] P. Laboratory, Northwest National, The Role of Authenticated Communications for Electric Power Distribution, National Workshop Beyond SCADA: Networked Embedded Control for Cyber Physical Systems (HCSS-NEC4CPS).

[6] D. Baily, E. W. Nipenz, Practical SCADA for Industry (IDC Technology)(Paperback), Newnes Publishers, Oxford UK. ISBN 7506 (2003) 58053.

[7] Electronic Industries Association, Standard for Electrical Characteristics of Generators and Receivers For Use In Balanced Multipoint Systems (EIA-485) (1983).

[8] ISO, 11898-1–Road vehicles–Controller area network (CAN)–Part 1: Data link layer and physical signalling, International Organization for Standardization.

[9] IEC, PAS 62407: Real-time ethernet control automation technology (EtherCAT), FR USA: EtherCAT Technology Group.

[10] J. Feld, PROFINET-scalable factory communication for all applications, in: Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on, IEEE, 2004, pp. 33–38.

[11] Modicon Inc., Modicon Modbus Protocol Reference Guide-PI-MBUS-300, Rev. J.

[12] Modbus-IDA, Modbus Application Protocol V1.1b., December 2006.

[13] IEC, Equipment, Telecontrol Systems—Part 5-104: Transmission Protocols—Network Access for IEC 60870-5-101 Using Standard Transport Profiles, IEC Standard 60870.

[14] IEEE Standard for Electric Power Systems Communications-Distributed Network Protocol (DNP3), IEEE Std 1815-2012 (Revision of IEEE Std 1815-2010) (2012) 1–821doi:10.1109/IEEESTD.2012.6327578.

[15] N. Falliere, L. O. Murchu, E. Chien, W32. stuxnet dossier, White paper, Symantec Corp., Security Response 5.

[16] IEC, ISO/IEC 61508-1:2010 Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 1: General requirements, IEC Standard.

[17] OPC Foundation, OPC Classic Specifications.
URL https://opcfoundation.org/about/opc-technologies/opc-classic/

[18] D. J. Kang, J. J. Lee, B. H. Kim, D. Hur, Proposal strategies of key management for data encryption in {SCADA} network of electric power systems, International Journal of Electrical Power & Energy Systems 33 (9) (2011) 1521–1526. doi:http://dx.doi.org/10.1016/j.ijepes.2009.03.004.

[19] ISA, ANSI, ISA–99.00. 01–2007 Security for Industrial Automation and Control Systems Part 1: Terminology, Concepts, and Models, International Society for Automation.

[20] IEC, ISO/IEC 62443-3-1:2009 Industrial communication networks - Network and system security - Part 3-1: Security technologies for industrial automation and control systems, IEC Standard.

[21] Triangle MicroWorks, Inc., DNP3 overview, Raleigh, North Carolina (www. trianglemicroworks. com/documents/DNP3 Overview. pdf).

[22] R. M. Lee, M. J. Assante, T. Conway, Analysis of the Cyber Attack on the Ukrainian Power Grid, Sans (2016) 23.
URL https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf

[23] J. Slay, M. Miller, Lessons learned from the maroochy water breach, IFIP International Federation for Information Processing 253 (2007) 73–82. doi:10.1007/978-0-387-75462-8_6.

[24] D. Cohen-Sason, Industroyer / CrashOverride IT to OT Malware That Changes Industrial Security Paradigms (2017).

[25] V. M. Igure, S. A. Laughter, R. D. Williams, Security issues in {SCADA} networks, Computers & Security 25 (7) (2006) 498–506. doi:http://dx.doi.org/10.1016/j.cose.2006.03.001.

[26] Y. Yang, K. McLaughlin, T. Littler, S. Sezer, E. G. Im, Z. Q. Yao, B. Pranggono, H. F. Wang, Man-in-the-middle attack test-bed investigating cyber-security vulnerabilities in Smart Grid SCADA systems, in:

Sustainable Power Generation and Supply (SUPERGEN 2012), International Conference on, 2012, pp. 1–8. doi:10.1049/cp.2012.1831.

[27] The Apache Software Foundation, Apche Avro.
URL https://avro.apache.org/

[28] N. A. Dr Stanford-Clark Andy, MQTT (1999).
URL http://mqtt.org/

[29] E. Upton, G. Halfacree, Meet the Raspberry Pi, John Wiley & Sons, 2012.

[30] H. Philip, Who Watches the Watcher? (aug 2012).
URL http://pi.gadgetoid.com/article/who-watches-the-watcher

[31] J. H. Brown, B. Martin, How fast is fast enough? Choosing between Xenomai and Linux for real-time applications, in: proc. of the 12th Real-Time Linux Workshop (RTLWS'12), 2010, pp. 1–17.

[32] Microchip, Inc., MCP3004/3008: 2.7V 4-Channel/8-Channel 10-Bit A/D Converters with SPI Serial Interface (2008).
URL http://ww1.microchip.com/downloads/en/DeviceDoc/21295d.pdf

[33] R. Vaarandi, SEC, Simple Event Correlator (2006).

[34] M. G. Luis, TCPDUMP/LIBPCAP public repository, Online document.

[35] Pymodbus Project, pymodbus - A full modbus protocol written in python.
URL https://github.com/bashwork/pymodbus

[36] C. Prix, P. Code, INTERNATIONALE INTERNATIONAL STANDARD.

[37] R. Michael, Portscan Attack Detector Suite.
URL http://www.cipherdyne.org/psad/

[38] H. Harnisch, Netfilter / IP-Tables (2007).
URL https://www.netfilter.org/

[39] ebtables - linux ethernet bridge firewalling.
URL http://ebtables.netfilter.org/

[40] V. Chandola, A. Banerjee, V. Kumar, Anomaly detection: A survey, ACM computing surveys (CSUR) 41 (3) (2009) 15.

[41] R. Christopher, Port Scanning Techniques and the Defense Against Them, SANS Institute.

[42] R. K. C. Chang, Defending against flooding-based distributed denial-of-service attacks: a tutorial, Communications Magazine, IEEE 40 (10) (2002) 42–51. doi:10.1109/MCOM.2002.1039856.

[43] R. Siles, Real world ARP spoofing, GIAC Certified Incident Handler (GCIH) Practical, Version 2.

[44] A. Ornaghi, M. Valleri, Man in the middle attacks, in: Blackhat Conference Europe, 2003.

[45] C. Foglietta, D. Masucci, C. Palazzo, R. Santini, S. Panzieri, L. Rosa, T. Cruz, L. Lev, From Detecting Cyber Attacks to Mitigating Risk Within a Hybrid Environment (2018) 1–12.doi:10.1109/JSYST.2018.2824252.

[46] P. Biondi, Scapy, a powerful interactive packet manipulation program (2010).

[47] E. Vyncke, C. Paggen, LAN switch security: What hackers know about your switches, Cisco Press, 2007.

[48] Digital Bond, Inc., Metasploit Modules.
URL http://www.digitalbond.com/tools/basecamp/metasploit-modules

[49] F. H. McMahon, Tech. rep., Lawrence Livermore National Lab., CA, (USA), month = dec, number = UCRL-53745, title = The Livermore FORTRAN Kernels: A computer test of the numerical performance range, year = 1986.

[50] R. Longbottom, Raspberry Pi Benchmarks.
URL http://www.roylongbottom.org.uk/RaspberryPiBenchmarks.htm

[51] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, K. Gibbs, Iperf: The TCP/UDP bandwidth measurement tool, htt p://dast. nlanr. net/Projects.

[52] Modbus, Modbus messaging on TCP/IP implementation guide v1. 0b, North Grafton, Massachusetts.

[53] S. Godard, SYSSTAT utilities home page (2010).
URL http://sebastien.godard.pagesperso-orange.fr

[54] D. Hadžiosmanovi, R. Sommer, P. H. Hartel, Through the Eye of the PLC : Semantic Security Monitoring for Industrial Processes.

[55] H. Janicke, A. Nicholson, S. Webber, A. Cau, Runtime-Monitoring for Industrial Control Systems, Electronics 4 (4) (2015) 995–1017. doi:10.3390/electronics4040995.

[56] S. Whalen, M. Bishop, J. P. Crutchfield, Hidden Markov Models for Automated Protocol Inference, Computer 1–8.

[57] M. Caselli, E. Zambon, F. Kargl, Sequence-aware intrusion detection in industrial control systems, Proceedings of the 1st ACM Workshop on Cyber-Physical System Security (2015) 13–24.doi:10.1145/2732198.2732200.