On the Detection of Replay Attacks in Industrial Automation Networks Operated with Profinet IO

Steffen Pfrang and David Meier
Fraunhofer IOSB, Fraunhoferstr. 1, 76131 Karlsruhe, Germany

Keywords: Industrial Networks, Replay Attacks, Port Stealing, DCP Reconfiguration, Intrusion Detection, Attack Detection Modeling.

Abstract: Modern industrial facilities consist of controllers, actuators and sensors that are connected via traditional IT equipment. The ongoing integration of these systems into the communication network yields to new threats and attack possibilities. In industrial networks, often distinct communication protocols like *Profinet IO (PNIO)* are used. These protocols are often not supported by typical network security tools. In this paper, we present two attack techniques that allow to take over the control of a PNIO device, enabling an attacker to replay formerly recorded traffic. We model attack detection rules and propose an intrusion detection system (IDS) for industrial networks which is capable of detecting those replay attacks by correlating alerts from traditional IT IDS with specific PNIO alarms. Thereafter, we evaluate our IDS in a physical demonstrator and compare it with another IDS dedicated to securing PNIO networks.

1 INTRODUCTION

Automation systems play a key role in modern industrial facilities and are therefore an important part of our economy. New technology tries to maximize the efficiency of such systems by simplifying the installation of automation systems, as well as providing the available data to the business level. In recent years, this was achieved more and more by the introduction of well-known IT technology into areas where in the past dedicated and incompatible technologies were used. While automation and control systems still have certain requirements, new standards aim to provide these requirements using standard IT technology. One such standard is *Profinet IO (PNIO)*, as specified in (IEC 61158-6-10, 2007). This Industrial Ethernet standard has the possibility to connect various automation systems and provide them with real-time communication. PNIO, as most new automation standards, is mostly concerned with providing reliable services and does not provide any distinct security mechanisms.

This situation, combined with the already mentioned process of networks getting connected to more and more services, renders the threat of cyberattacks into a realistic scenario. Even if there are no inherent security controls in PNIO[1], it is crucial to be able to detect attacks targeting PNIO devices. Because such attacks require little to no knowledge about the industrial process being run and are easy to adopt, this makes them a rewarding target for any attacker.

In this work, we will show how the missing security in automation protocols can facilitate the ability to successfully attack control system equipment by using a replay attack method on PNIO. We will then show how it is possible to detect these attacks, taking multiple information sources as an input for an intrusion detection system (IDS). To be able to understand the attack and proposed detection model, this work also includes a brief overview of the relevant details of PNIO.

2 BACKGROUND AND RELATED WORK

In this section, we will give background information on the relevance of PNIO and the protocol specifications which are important in our study. The second

[1]This means that such a PNIO network should be operated in a completely separated environment without any link to the outside.

Pfrang, S. and Meier, D.
On the Detection of Replay Attacks in Industrial Automation Networks Operated with Profinet IO.
DOI: 10.5220/0006288106830693
In *Proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP 2017)*, pages 683-693
ISBN: 978-989-758-209-7

part of this section is dedicated to related work available in the area of control system security and on the security of PNIO.

2.1 Background

According to a recent study (HMS Industrial Networks, 2016b), Ethernet-based fieldbuses have a market share of 38% within industrial automation systems, while classic fieldbuses have 58% and wireless variants 4%. The annual growth of Ethernet-based fieldbuses is estimated with 20%, whereas Siemens is the market leader in Europe and China with its "Profibus family" including PNIO (HMS Industrial Networks, 2016a). In total, 21% of the sold Ethernet-based automation systems are running PNIO.

In the following, we will provide technical details about the PNIO protocol behaviour relevant to the presented scenario: A programmable logic controller (PLC) controlling a PNIO device (a stepping motor). Starting with the discovery and configuration protocol DCP and the context management protocol CM, we describe the startup phase of a PNIO network like it happens when the power returns after an outage. The last subsection covers the alarm management of PNIO devices.

2.1.1 PNIO Discovery and Configuration

DCP (Discovery and Configuration Protocol) is an Ethernet-based protocol within the PNIO protocol suite. Using the *DCP Set Request*, a controller is able to configure a PNIO device. Such a request can contain multiple configuration blocks, for example to set the name and the IP address using only a single packet. The request is then answered by a *DCP Set Response* that signals what configuration options were tried to be set and if the request could executed. It does not contain the actual values from the *DCP Set Request*.

2.1.2 PNIO Context Management

CM (Context Management) is a UDP-based PNIO protocol to setup the application relation (AR) between PNIO controllers and devices. The controller is sending a *CM ConnReq* request, specifying various properties of the communication channel, including its communication relations (CR). These can be data inputs and outputs (IO), as well as properties for alarm handling. The properties of IO CR include values for the *DataHoldFactor*, *SendClockFactor* and *ReductionRatio*. From these values, the *DataHoldTime* of the respective CR is calculated, as shown in equation 1.

$$\begin{aligned} \textbf{DataHoldTime} &= \text{DataHoldFactor} \\ &\times \text{SendClockFactor} \\ &\times \text{ReductionRatio} \\ &\times 31.25\mu s \end{aligned} \tag{1}$$

A watchdog timer of that duration is set every time data is received for the respective CR. When this timer expires, it will trigger an *PN-RT* alarm frame with an appropriate error message.

The PNIO device replies to the *CM ConnReq* with a *CM ConnResp* response. This response can acknowledge the request or deny it, for example if there are no resources left for a new AR. In case of a successful connection request, the already established CRs will start exchanging data and the controller can proceed establishing further CMs. When all desired CMs are established, the controller will send a *CM Control Req* request signaling the end of parameter transmission, that will again be acknowledged by the PNIO device through an *CM Control Resp*. Afterwards, the device is sending a *CM Control Req* signaling application readiness. After the controller has acknowledged this request, the communication establishment is complete.

2.1.3 PNIO Startup

When an industrial setup operated with PNIO starts up, the sequence of network packets shown in Figure 1 can be discovered. We will focus on the connection establishment between the PLC (as PNIO controller) and the motor (as PNIO device). Additionally, we consider the case that all the components are configured for operation. That means, that the process is already running which is a reasonable assumption when dealing with real production environments.

From the Profinet protocol suite, there are three protocols involved: DCP, CM and RT (Real Time). Additionally, PNIO makes use of ARP (Address Resolution Protocol). CM is an UDP-based protocol, while the other protocols are operating directly on the MAC layer (Ethernet).

The startup process is initiated by the PNIO controller. It asks with a multicast *DCP Ident Request* for a device with the PNIO name that has been configured initially in the PLC program. The PNIO device with that particular name replies to that request with an unicast *DCP Ident OK* that is directed to the PLC. The *Ident OK* message includes, amongst other information, the IP address configuration of that device.

Subsequently, the controller starts an *ARP request* for the IP address given by the PNIO device. If the device is not yet configured with that IP address, the controller sends a *DCP Set Request*, setting the IP address configuration of the PNIO device in order to be

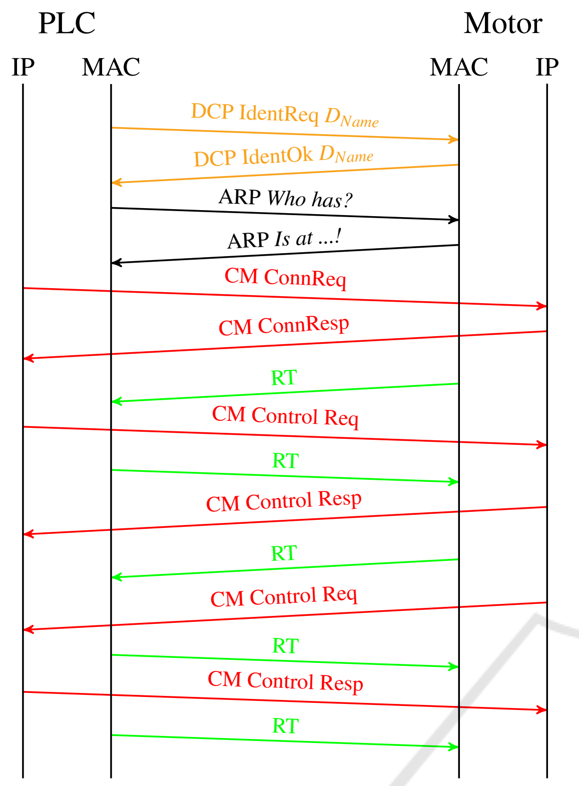

Figure 1: The regular startup process of a PNIO network at the example of a PLC and a motor.

able to communicate with it using UDP for the next step, the PNIO-CM.

2.1.4 PNIO Alarm Management

The signaling of errors and warnings is also an integral part of PNIO. On the network level, this signaling is achieved by the usage of special *alarm frames*. There are multiple alarm types, depending on the origin and cause of the alarm. For example, *process alarms* are originating from the executed process on a field device, while *diagnosis alarms* are originating from the field device itself (Popp, 2014). Only *process alarms* have their priority set to `HIGH`, while all other type have a `LOW` priority. Alarms are queued into a buffer according to their priority and only removed from the buffer once they are acknowledged. Every alarm is therefore assigned a sequence number which also needs to be part of the acknowledgment of the communication partner, i.e. the PNIO controller or device. This means other alarms of the same priority are not reported until all alarms prior to them are removed from the queue buffer. The alarm buffer is cleared when the application relation (AR) between controller and device is terminated, for example by a new connection request.

The PNIO standard holds a predefined list of alarm messages that can be extended by manufacturer specific messages. Alarms are sent within an acyclic *PN-RT* frame, the alarm frame (*PN-AL*), specifying the alarm type, alarm origin and structure of the alarm message.

2.2 Related Work

A broad overview of the state of the art in industrial control system security has already been given (McLaughlin et al., 2016). This work promotes the idea of using industrial control system (ICS) testbeds for helping to identify, exploiting and mitigating cybersecurity vulnerabilities. It follows a security assessment approach to analyze the ICS architecture and identify vulnerabilities. The authors identify the attacks on controllers and sensors as two attack classes in the current ICS threat landscape and the false data injection (FDI) as one of the emerging threats in this domain.

A compact overview of the system architecture of PNIO and its functionality can be found in (Popp, 2014). More details about the alarm management in PNIO networks are being described in (Ferrari et al., 2006). However, this work focuses mainly on the performance aspects of a PNIO system.

Replay attacks in the control system domain have been studied in (Mo and Sinopoli, 2009). Here, the authors are assuming an attacker trying to disrupt the operation of a control system by injecting control inputs without being detected. The main contribution of this work is a new problem formulation providing information on the feasibility of replay attacks on control systems.

The vulnerability of PNIO to Man-in-the-middle (MitM) attacks when using PNIO conformance class A has already been addressed (Baud and Felser, 2006). For this conformance class, classical network infrastructure components, like standard Ethernet switches, are allowed and do not need to be certified for the usage in a PNIO network. In this work, the authors describe the communication setup process and possible errors that can occur during the setup, like the duplicate assignment of station names or IP addresses. By using open-source tools like `ettercap`, the authors tried a MitM approach that was not successful because of the PNIO cycle speeds, as suggested.

The security of PNIO was also explored in (Åkerberg and Björkman, 2009b). Here, the authors followed two attack scenarios. In the first scenario, the systems are using a shared medium for their communication, while in the second one, the systems are communicating via a packet switched network. In the shared media scenario the attack method is to inject PNIO frames so that they will reach the target before the correct frame which is requiring highly accurate timing and sound understanding of the protocol. In the packet switched scenario, a MitM attack is used to intercept and alter PNIO frames. This is a far more simple approach than in the first scenario, as the attacker only has to wait for frames to arrive and forward them after they are altered without the need to know any details about the AR between PNIO controller and device. While being successful in implementing the attack, the author noticed a certain unreliability due to the fact that their PC-based attack system is partially delaying the frame processing.

Two attack types, the Denial-of-service (DoS) and the MitM attack, and corresponding attacks on PNIO have also been studied by (Paul et al., 2013). The attack on the DCP protocol relies on fabricated *DCP Ident Requests* frames preventing the successful assignment of a device name. Similar to that, the IP address assignment can also be prevented by sending malicious DCP or ARP frames. Both of these attacks will result in a DoS. The presented MitM attacks are divided into attacks during the startup process and during the operational stage. The MitM attacks rely on the *port stealing* method which is manipulating the switch to be able to capture the traffic between PNIO controller and device. In this work, the authors also investigate the setup of an intrusion detection system for ICS based on anomaly detection and show that this IDS is able to detect the described DoS and MitM attacks. The proposed anomaly detection is thereby only using the sequence of DCP frames to determine abnormal behavior, without incorporating indicators from the PNIO protocol or using means to detect the port stealing itself. In section 4.1, we will use this work for a comparative evaluation of our proposed solution.

There have been further efforts to study the security of PNIO. The security of *PROFIsafe* (an additional layer on top of PNIO dealing with safety aspects) has been examined in (Åkerberg and Björkman, 2009a). Recommendations on the enhancement of PNIO in terms of security have also already been given by (Åkerberg and Björkman, 2009c), where PNIO is recommended to be extended by the introduction of security modules based on the concept of the safety profile in *PROFIsafe*. This extension should be comprised of an additional layer ensuring authenticity, integrity and confidentiality for PNIO process data. Currently, no such extension was introduced in the PNIO standard.

3 ATTACK CASE STUDY

This section is divided in three main sections: The environment setup, the attack selection and execution, and the attack detection.

3.1 Environment Setup

The scenario for this paper is implemented in an industrial IT security laboratory. This lab consists of both a physical part and a virtualization environment. (Reference removed for anonymity reasons.) Within the physical part, real industrial components like PLCs, sensors and actors, switches etc. are connected to each other using typical industrial protocols like PNIO, Siemens S7 and OPC UA.

In the virtualization environment (in the following, we will call it “cloud”), different networks have been set up using virtual switches as well as virtual firewalls. Virtual machines connected to these networks provide components of automation systems (e.g. PLC programming stations, SCADA servers), management tools as well as different attack and detection tools.

This scenario (see Figure 2) makes use of a Siemens PLC (S7-300), a stepping motor (Schneider Electric mdrive), a human machine interface (Siemens KTP-700) and an industrial switch (Hirschmann RS-20) in the physical part. The switch is connected to the virtualization environment providing data exchange as well as a mirror stream of the network traffic within the physical part to be used by the detection tools.

The physical process implemented in the scenario turns a disk which is mounted on the motor shaft clockwise from one position to the next with a pause of 2 seconds. The real industrial process from which our demonstration setup was derived, consists additionally of a driller which drills a hole in a workpiece if and only if a workpiece was transported by the disk to the drilling position.[2]

3.2 Attack Selection and Execution

In order to steer the stepping motor, we focus on replaying network traffic. This a very general attack which can be performed even without having knowledge about the process that is implemented, only the ability and time for watching and recording the traffic that is flowing through the network is needed. A limitation of that approach is that you can only record

[2]For obvious reasons, attacks causing irreversible damages and possibly injuring people are not allowed to be performed in our lab.

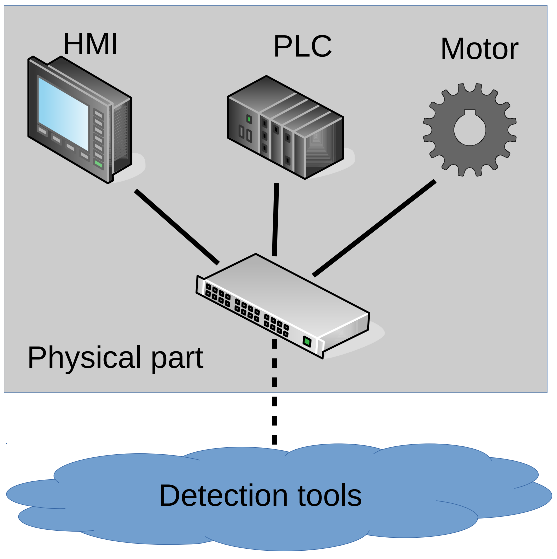

Figure 2: The base scenario is set up using a PLC with a HMI and a motor in a star topology with a switch that connects the local network to a cloud network which hosts the detection tools.

actions that occur while you are watching the traffic. But we don't consider this as a major drawback because most industrial processes rely on a very correct timing[3], so you can harm the system even by triggering valid actions in an invalid time slot. For example, see the drilling example from above with a driller drilling a hole into the disk instead of the workpiece.

3.2.1 Taking Over Control

Replaying traffic in order to attack a specific PNIO device requires the attacker to take over control of at least one connection to it. For that, we selected two types of attacks which turned out to be suitable. The first one is a classical way to attack Ethernet-based protocols: Using *port stealing*.

Usually, a network switch learns which device is connected to which hardware port by maintaining a lookup table. If it receives an Ethernet packet with the source MAC address M_i from its port P_j, it learns that packets with the destination MAC address M_i have to be delivered to port P_j. Former entries with the same MAC address get lost. This standard mechanism is useful because it adapts itself when new devices are added to the switch or when the topology changes.

The port stealing attack exploits this behaviour as follows: The device to be attacked has the MAC address M_{victim} and is attached to port P_{victim}. The attacker is attached to port $P_{attacker}$ and sends any kind of Ethernet-based packets with the source MAC address M_{victim}. The switch now learns that packets for M_{victim} have to be delivered to $P_{attacker}$. If the victim sends itself a network packet, the switch will recover the original relation of M_{victim} on port P_{victim}, but the attacker sends its port stealing packets in such a high frequency that these changes won't last for a longer period of time.

In our work, the PLC acted as the victim for the port stealing attack. Figure 3 illustrates the attack sequence with the corresponding network communication. When the motor sends packets to the PLC, they will not arrive at the PLC anymore but at the switch port of the attacker. Meanwhile, the attacker is able to send packets directly to the motor. The real PLC would be able to send packets to the motor as well, but that does not pose as a problem for the attacker, because the PNIO standard specifies, that if there is no real-time response from the communication partner, an existing connection has to be closed.

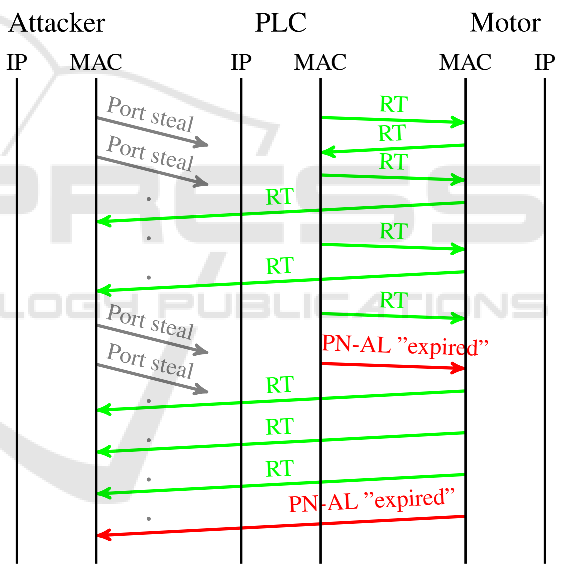

Figure 3: The attacker performs a port stealing attack targeting the PLC via the hardware switch. This results in the attacker taking over the RT connection to the motor from the PLC.

The second attack type is specific for PNIO operated devices: A *reconfiguration attack* using DCP. Caused by the design of this protocol without any authentication measures, each device that is located in the same subnet is able to reconfigure a PNIO device. There are two options available: *Set IP* and *Set Name*. Since the identifier of a PNIO device is its name, we chose that option for the attack.

In Figure 4, the sequence diagram of the DCP reconfiguration attack is shown. There are three devices involved with both their IP and MAC interface: The

[3]PNIO supports up to 5ms cycle time (1ms in the isochronous real-time class).

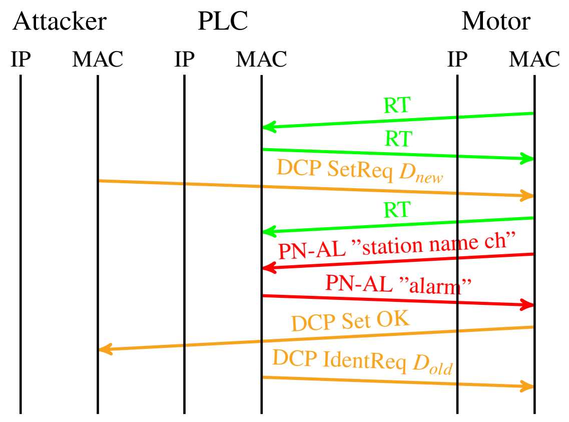

Figure 4: The attacker reconfigures the motor using DCP. In consequence, the real-time communication between the PLC and the motor will be terminated.

PLC, the motor with its original device name D_{old} and the attacker. Before the attack happens, PLC and motor exchange real-time data (RT).

The attacker asks the motor to change its name to D_{new} by sending a *DCP Set Request*. The motor emits a PNIO alarm frame "device name changed" to its Profinet communication partner, the PLC. The PLC acknowledges the alarm frame, and the connection between the PLC and the motor will be terminated. As a last step, the motor acknowledges the *DCP Set Request* by sending a *DCP Set Response* to the attacker. On a regular basis, the PLC keeps on asking for a device with the name D_{old} which is still configured in its setup, but will not receive any response, as no device is configured to react on that name now.

3.2.2 Replaying Traffic

Replaying traffic successfully in a PNIO operated system requires, besides a communication connection, the application relation (AR) as mentioned before (section 2.1.2). Once a PNIO device like the stepping motor has established an application relation with its PLC, it refuses each further communication request, responding with an error reply. Hence, the existing communication relation between the PLC and the motor has to be interrupted before continuing the replay.

Fortunately, both types of attacks described in section 3.2.1 ensure that the original communication relation gets interrupted: In the port stealing case, the PLC does not receive real-time data from the motor anymore, because this data is now redirected to the attacker. After the *DataHoldTime*, as described in section 2.1.2, has passed without any message from the motor, it terminates the connection and sends an alarm frame. As a consequence, the motor will also stop receiving real-time data from the PLC and will terminate its communication in the same way. In the case of the DCP reconfiguration attack, the connections gets lost immediately.

Once the motor isn't involved in a communication relation anymore, the attacker is able to setup a new connection with the motor. The attacker now starts with *CM Connect Request* that is answered by the motor with a *CM Connect Response*. After that, the attacker sends a *CM Control Request*, which is answered with a *CM Control Response* by the motor. In the original traffic, the motor proceeds by sending a *CM Control Request*, but the attacker does not have to reply to that request and can directly start replaying the PNIO real-time packets. As a side note, the attacker does not even need to process or react on the replies of the motor.

3.3 Attack Detection

In this section, we will start with a model for detecting the attacks being described in the previous sections. In a second part, we explain how we implemented or plan to implement this using our existing open-source based intrusion detection framework.

3.3.1 Model for Attack Detection

In the case of the port stealing based replay attack, there is the evidence of the packets used for the port stealing. So in general, one could be suspicious if such unusual traffic can be observed in the network. Since the port stealing approach is based on the attacker sending forged packets in a limited period of time[4] between a correct packet from the device under attack and the response to this packet, the attacker has to produce lots of packets in time. A method to detect this huge amount of traffic is by using flow-based intrusion detection systems. Those systems aggregate single packets by considering source and destination as well as some other configurable keys to flows of packets. Another method for detection would be a switch configuration that emits alerts when MAC addresses change from one interface to another. We will call that evidence ⟨ *port-stealing* ⟩ and the tool performing that analysis *Port stealing detector*.

Additionally, there will appear two PNIO alarm frames of the type "*AR consumer DHT/WDT expired (RTA_ERR_ABORT)*" at the network: The first one from the PLC that does not receive cyclic data from the motor anymore (⟨ *al-plc-expired* ⟩), and a second one from the motor complaining about the PLC stopping to send cyclic data (⟨ *al-motor-expired* ⟩).

[4]In tests, we discovered answer times less than 1 ms.

Detecting the DCP reconfiguration attack is very different because there will not be any packets present that are unusual for the specific networks. Nevertheless, industrial networks tend to be very stable and are run for a long time. So reconfiguring devices isn't part of the daily work and for that reason, detecting DCP packets that change the name of a device seems to be reasonable[5]. We will call that packet ⟨ *dcp-setname* ⟩.

In order to refine the detection, a PNIO alarm frame "*station name changed*" will appear on the network if the exchange of name was successfully completed (⟨ *al-station-name-changed* ⟩). This alarm gets acknowledged by the PLC with an alarm (⟨ *al-generic* ⟩). And further on, the PLC will send regularly *DCP Ident Requests* asking for the initial name of the motor (⟨ *dcp-ident-req-oldname* ⟩).

Summarizing, the port stealing attack can be discovered using the following evidence:

⟨ *port-stealing* ⟩ *AND*
⟨ *al-plc-expired* ⟩ *AND*
⟨ *al-motor-expired* ⟩

In case of the DCP reconfiguration attack, we experience the following evidence:

⟨ *dcp-setname* ⟩ *AND*
⟨ *al-station-name-changed* ⟩ *AND*
⟨ *al-generic* ⟩ *AND*
⟨ *dcp-ident-req-oldname* ⟩ (repeated)

It has to be noted that not each evidence of the AND clauses is really needed for assuming that there is an attack ongoing: In the first case, it would be sufficient if you discover ⟨ *port-stealing* ⟩ and one of ⟨ *al-plc-expired* ⟩ or ⟨ *al-motor-expired* ⟩ because you are able to infer that when one partner of the communication quits, the other one will follow soon. In the second case, ⟨ *al-generic* ⟩ and ⟨ *dcp-ident-req-oldname* ⟩ only improve the accuracy of the detection. This is an important notice because in some topological constellations, not every packet is visible for the IDS (see section 4).

3.3.2 Attack Detection Implementation

Figure 5 depicts a schema of the attack detection systems. It has to be noted that parts of the implementation are still work in progress. The mirroring of the network traffic in the physical environment is ensured by the hardware switch which monitors the specific ports of the PLC and the motor. Since the distributed IDS resides in the virtual part of the laboratory, the mirrored traffic has to be forwarded using a generic routing encapsulation (GRE) tunnel into the cloud.

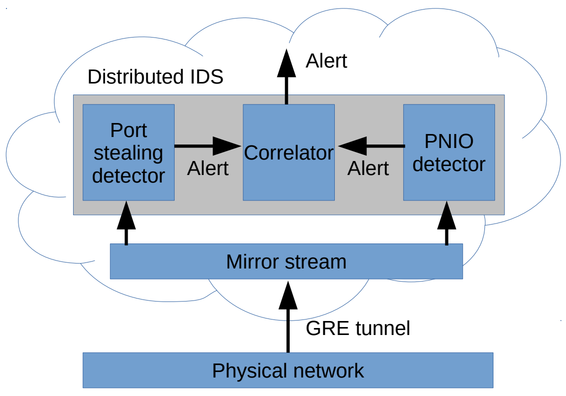

Figure 5: Schema of the implementation comprising attack detection tools in the cloud being fed by a mirror stream from the physical part of the network where the attacks take place.

On the one hand, the *PNIO detector* receives the network packets from the mirror stream. In our case, it consists of a Python script, running in background and employing the "Network packet and pcap file crafting/sniffing/manipulation/visualization security tool" Scapy (Biondi, 2010). It filters the Ethertype *0x8892* of PNIO and generates an alert when it receives either a specific PNIO alarm frame (see section 2.1.4) or a DCP packet (see section 2.1.1). The reason for not using a standard network-based IDS like Snort (Roesch et al., 1999) is that those tools are based on IP traffic, while PNIO is based on Ethernet.

On the other hand, the *Port stealing detector* sniffs the packets from the mirror stream. The open IPFIX standard (Internet Protocol Flow Information Export, (Claise, 2015)) specifies flow keys for layer 2 communication like MAC source and destination addresses. Unfortunately, the current implementations of the common open-source flow-based IDS Flowtools (Fullmer and Romig, 2000), Silk (McHugh, 2004) and nfdump (Haag, 2005) do not support those keys, as they are limited to layer 3 communication. For that reason, we are currently not able to use a flow-based IDS as a port stealing detector.

As an alternative solution for implementing the port stealing detector, we have set up the switch connecting the physical components to send alerts to the correlator when MAC addresses change from one interface to another. The great advantage of this solution is that it is available. The drawback is the fact that it needs reconfiguration of the switches which is not feasible in every production environment.

The *Correlator* receives alerts from both the PNIO detector and the port stealing detector. It is able to

[5] In order to eliminate false positives, we expect the operator receiving alerts from the IDS will be aware of any maintenance work.

aggregate the alerts and generate new alerts which can be sent to the operator.

4 EVALUATION

In this section, we will evaluate our approach to detect the two replay attacks dedicated to PNIO networks described in section 3.2. For that reason, we performed and recorded several tests in different setups. We will conclude this section with a comparative evaluation of our solution.

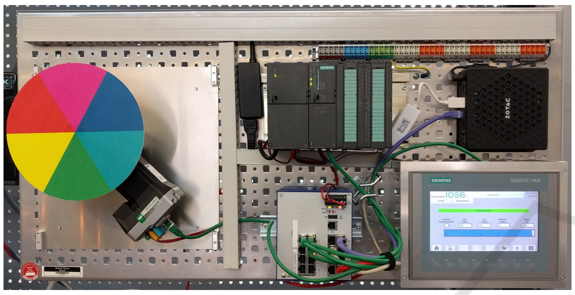
Figure 6: The demonstrator with a stepping motor turning a disk used for performing the replay attacks.

Figure 6 depicts the base setup with the PLC (upper middle) and the motor (on the left) in a star topology connected by a switch (lower middle). In the upper right resides an embedded PC which is connected both to the mirror port of the switch and a cloud uplink. An HMI, reflecting the status of the system, is placed in the lower right.

Table 1 enumerates the scenarios considered in our evaluation. The first column represents the identificator of the scenario. The second column describes the topology of the demonstrator: In case of the "star" topology, both PLC and motor are connected to the switch. In the "line" case, the motor is attached directly to the second Ethernet port of the PLC.

The attacks can be performed either by using a laptop that gets connected to the switch on a port that is configured with the same VLAN as the PLC and the motor, or from the cloud using the same VLAN as well. This distinction is related to the effort an attacker has to make: In the local case, an attacker has to gain physical access to the switch, whereas in the cloud case, an attack even from remote may succeed. This is being represented by the column "Origin".

The column "Action" tells which type of action has been performed in the specific scenario: Mainly the two types of attacks, but additionally, two cases have been considered in which we expected PNIO alarm frames similar to the ones being observed in the attack case. On the one hand, this is the removal of either the plug of the PLC or the plug of the motor from the switch. On the other hand, this is the substitution of the PNIO name by the respective programming station: In case of the PLC the *Siemens TIA portal*, in case of the motor, the *Lexium Software Suite*.

In the column "Evidence", we have listed abbreviations of the evidence being described in section 3.3.1: *PS* stands for $\langle$ *port-stealing* $\rangle$. The different PNIO alarm frames are prefixed with their origin: *P* stands for an alarm that has been issued by the PLC, *M* for one that has been sent by the motor. The index denotes the type of alarm: *ex* for $\langle\, \mathit{al}\text{-}\frac{\mathit{plc}}{\mathit{motor}}\text{-}\mathit{expired} \,\rangle$, *snc* for $\langle$ *al-station-name-changed* $\rangle$ and *gen* for $\langle$ *al-generic* $\rangle$. There are also abbreviations for several DCP packets: *SET* stands for $\langle$ *dcp-setname* $\rangle$, while *ID* stands for $\langle$ *dcp-ident-req-oldname* $\rangle$.

The next column "Detection" bares witness of the fact whether our proposed IDS detected an attack or not.

In our evaluation, we considered 14 different scenarios. The first eight cases arise from the topology permutation of the demonstrator, the origin and the type of the attack. Case #9 up to #12 reflect the permutation of the location where the cable has been unplugged and the topology, while the last two cases describe the interaction of both the PLC and the motor with their repective management stations located only in the cloud.

First of all, it has to be noted that both types of replay attacks succeed in the case of a star topology. However, if the physical components are connected in a line topology, replaying does not succeed using the port stealing attack, but it still does when employing the DCP reconfiguration attack.

The proposed detection works in case of an attacker that sends the replay packets from the cloud. In case of a line topology, the port stealing attack fails (case #7), but the DCP reconfiguration attack succeeds (case #5) with a *false negative*. The reason for not detecting the attack in that case is that PNIO alarms do not protrude the connection between the PLC and the motor. To overcome this issue, it is necessary to place a network tap in between both physical components and forward these packets also to the mirroring stream. The drawback of this solution is that it does not scale with larger installations. Nevertheless, we consider attacks from the cloud more probable than those where the attacker has to gain physical access to the switch.

In the case of the DCP reconfiguration attacks in the line topology (cases #6 and #8), only the DCP packets can be discovered. Nevertheless, it is not impossible to derive from the observed *DCP Set Request*

Table 1: The different setups examined in the evaluation of the Replay attacks.

#	Topology	Origin	Action	Evidence	Detection	Remarks
1	star	local	port stealing	PS, P_{ex}, M_{ex}	yes	
2	star	local	DCP reconfig	SET, M_{snc}, P_{gen}, ID	yes	
3	star	cloud	port stealing	PS, P_{ex}, M_{ex}	yes	
4	star	cloud	DCP reconfig	SET, M_{snc}, P_{gen}, ID	yes	
5	line	local	port stealing	PS	no	false negative
6	line	local	DCP reconfig	SET, ID	yes	
7	line	cloud	port stealing	PS	no	unsuccessful attack
8	line	cloud	DCP reconfig	SET, ID	yes	
9	star	–	PLC plug removed	M_{ex}	no	
10	star	–	motor plug removed	P_{ex}	no	
11	line	–	PLC plug removed	–	no	motor keeps spinning
12	line	–	motor plug removed	–	no	motor stops
13	star	cloud	PLC change name	–	no	Siemens S7 traffic
14	star	cloud	motor change name	–	no	proprietary protocol traffic

and the subsequent repeated *DCP Ident Requests* that an attack has taken place.

In order to validate the proposed IDS, we undertook tests which might result in similar alarms: Removing a cable from the switch in the star topology case results in either an "expired" alarm from the motor or the PLC. Since there is neither a port stealing attack active nor any DCP packets can be discovered, the IDS does not generate an alert. That happens as well for the management station changing the names of the devices: Neither the *Siemens TIA portal* nor the *Lexium Software Suite* make use of DCP. Both use their own protocols for that purpose: Either Siemens-S7 or a proprietary protocol. Summarizing, in none of the tested cases appeared a *false positive*.

4.1 Comparative Evaluation

The IDS proposed by (Paul et al., 2013) (in the following referred to as *PAUL*) also focuses on intrusion detection for PNIO networks. For that reason, we will try to perform a comparative evaluation with our approach. However, since there is not any implementation of PAUL available, we will have to restrict the comparison to the conceptual level.

PAUL derives n-grams of DCP packets from the mirrored network stream. Each n-gram that appears in a learning phase is considered a good one. Any other n-gram that can be observed in the regular operation phase is considered a bad one, depicting an attack.

In contrast to PAUL, we propose an IDS which does not need any learning phase but works out of the box once it has been configured for detecting security breaches in PNIO networks. Learning phases in industrial automation systems can be leveraged only in very rare cases since availability is the most important parameter. For example, in order to detect DCP packets which rename a PNIO device, you would have to interrupt existing communication relations and probably stop the industrial process.

As a second difference, our solution follows a dual approach combining both intrusion evidence from the well-known IT world with anomaly evidence from automation protocols. Whereas PAUL is limited to PNIO DCP, our IDS takes into account additional protocols of the PNIO protocol suite like CM and alarm frames. That renders our IDS to be more comprehensive than PAUL.

5 CONCLUSION AND FUTURE WORK

In this paper, we took a look at the security of the industrial Ethernet protocol *Profinet IO* which is one of the most common automation protocols being installed in Europe. We explained relevant functions and mechanisms during the startup phase and while running. Afterwards, we introduced the scenario in which we performed our research. In an attack case study, we motivated why we choose replay attacks as the means of attack, how we performed the attacks and conclude with a model for an IDS being able to detect the attacks.

We then evaluated our detection approach which takes into account both alarms from the well-known IT world and different protocols of the Industrial Ethernet world. In 14 different test cases, we ended up with no false positive and one false negative in a special constellation without any proper mirroring in place. Additionally, we compared our approach with another one and were able to show the advantages of the proposed solution.

Taking care of replay attacks is very crucial since PNIO devices are a worthwhile target. This is the case because an attacker does not need any special skills or knowledge of the process being controlled, making these attacks easy to perform.

In order to enable data exchange with other systems in the use case of the so-called "Industry 4.0" or "Internet of Things", many companies connect their industrial networks with their office networks and the Internet. Improper and unsecure configurations of these connections are raising risk and danger. And the danger is proverbial: Industrial processes control real devices. If e.g. motors, valves, robots do not behave as expected, serious physical damage to property and lives may happen.

This strengthens the need for intrusion detection systems adapted for the industrial environment. Otherwise, if anything gets damaged by a cyber attack, like the replay attacks presented in this paper, it will not be possible to identify the reason of the attack. This is the case because of the PNIO inherent features which restore the initial connection and do not persist any evidence in the network.

5.1 Future Work

Performing replay attacks in automation systems is only one type of possible means to disturb or destroy an industrial processes, yet a simple and easy to perform one. There are many more attack vectors, like the tampering of data that is presented in visualizations, worth being considered and assessed. Evaluation criteria include the amount of knowledge about the industrial process an attacker needs, how easy the attacks are to employ and the extend of possible damage. This work will be extended by an overall attack case study of the used demonstrator including the PLC, the motor, the local HMI as well as an HMI that accesses data from the PLC and presents it on a web interface in the cloud.

Many automation systems in Europe are operated with PNIO. Nevertheless, there are many other automation protocols which are widespread. We will setup demonstrators with protocols besides PNIO (for example with EtherCAT) and will try to transfer our attack and detection techniques.

An ongoing task is the implementation work in our distributed open-source IDS. This task deals with the main problem that common IDS start with the IP layer. They do not support Ethernet-based protocols as they are usually used in industrial automation networks and are not able to pay attention to the real-time requirements of such networks. An effort to close this gap is the development of a preprocessor for the popular IDS Snort which enables it to deal with PNIO traffic. Another effort is the work on different correlation techniques that have to be employed in a central correlation engine that controls the possible flood of different alerts from both the IT and the automation process side.

REFERENCES

Åkerberg, J. and Björkman, M. (2009a). Exploring network security in profisafe. In *International Conference on Computer Safety, Reliability, and Security*, pages 67–80. Springer.

Åkerberg, J. and Björkman, M. (2009b). Exploring security in profinet io. In *Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference - Volume 01*, COMPSAC '09, pages 406–412, Washington, DC, USA. IEEE Computer Society.

Åkerberg, J. and Björkman, M. (2009c). Introducing security modules in profinet io. In *2009 IEEE Conference on Emerging Technologies & Factory Automation*, pages 1–8. IEEE.

Baud, M. and Felser, M. (2006). Profinet io-device emulator based on the man-in-the-middle attack. In *ETFA*, pages 437–440.

Biondi, P. (2010). Scapy documentation. http://www.secdev.org/projects/scapy/doc/. [Online; accessed 08-December-2016].

Claise, B. (2015). Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of IP Traffic Flow Information. RFC 5101.

Ferrari, P., Flammini, A., and Vitturi, S. (2006). Performance analysis of profinet networks. *Computer standards & interfaces*, 28(4):369–385.

Fullmer, M. and Romig, S. (2000). The osu flowtools package and cisco netflow logs. In *Proceedings of the 2000 USENIX LISA Conference*.

Haag, P. (2005). Watch your flows with nfsen and nfdump. In *50th RIPE Meeting*.

HMS Industrial Networks (2016a). Feldbusse heute. http://www.feldbusse.de/trends/status-feldbusse.shtml. [Online; accessed 08-December-2016].

HMS Industrial Networks (2016b). Variantenvielfalt bei Kommunikationssystemen. http://www.feldbusse.de/Trends/trends.shtml. [Online; accessed 08-December-2016].

IEC 61158-6-10 (2007). Industrial communication networks - Fieldbus specifications - Part 6-10: Application layer protocol specification - Type 10 elements. Standard, International Electrotechnical Commission, Geneva, CH.

McHugh, J. (2004). Sets, bags, and rock and roll. In *European Symposium on Research in Computer Security*, pages 407–422. Springer.

McLaughlin, S., Konstantinou, C., Wang, X., Davi, L., Sadeghi, A.-R., Maniatakos, M., and Karri, R. (2016). The Cybersecurity Landscape in Industrial Control Systems. *Proceedings of the IEEE*, 104(5):1039–1057.

Mo, Y. and Sinopoli, B. (2009). Secure control against replay attacks. In *Communication, Control, and Computing, 2009. Allerton 2009. 47th Annual Allerton Conference on*, pages 911–918. IEEE.

Paul, A., Schuster, F., and Knig, H. (2013). Towards the Protection of Industrial Control Systems: Conclusions of a Vulnerability Analysis of Profinet IO. In *Proceedings of the 10th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, DIMVA'13, pages 160–176, Berlin, Heidelberg. Springer-Verlag.

Popp, M. (2014). *Industrial Communication with PROFINET*. PROFIBUS Nutzerorganisation e.V., Karlsruhe.

Roesch, M. et al. (1999). Snort: Lightweight intrusion detection for networks. In *LISA*, volume 99, pages 229–238.