A Fully-Blind False Data Injection on PROFINET I/O Systems

Wael Alsabbagh[1,2] and Peter Langendörfer[1,2]
[1]*IHP – Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany*
[2]*Brandenburg University of Technology Cottbus-Senftenberg, Cottbus, Germany*
E-mail: (Alsabbagh,Langendoerfer)@ihp-microelectronics.com

2021 IEEE 30th International Symposium on Industrial Electronics (ISIE) | 978-1-7281-9023-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ISIE45552.2021.9576496

***Abstract*—This paper presents a fully blind false data injection (FDI) attack against an industrial field-bus i.e. PROFINET that is widely used in Siemens distributed Input/Output (I/O) systems. In contrast to the existing academic efforts in the research community which assume that an attacker is already familiar with the target system, and has a full knowledge of what is being transferred from the sensors or to the actuators in the remote I/O module, our attack overcomes these strong assumptions successfully. For a real scenario, we first sniff and capture real time data packets (PNIO-RT) that are exchanged between the IO-Controller and the IO-Device. Based on the collected data, we create an I/O database that is utilized to replace the correct data with false one automatically and online. Our full attack-chain is implemented on a real industrial setting based on Siemens devices, and tested for two scenarios. In the first one, we manipulate the data that represents the actual sensor readings sent from the IO-Device to the IO-Controller, whereas in the second scenario we aim at manipulating the data that represents the actuator values sent from the IO-Controller to the IO-Device. Our results show that compromising PROFINET I/O systems in the both tested scenarios is feasible, and the physical process to be controlled is affected. Eventually we suggest some possible mitigation solutions to secure our systems from such threats.**

***Index Terms*—Industrial Control Systems, False Data Injection Attack, I/O Database, PROFINET I/O Systems;**

I. Introduction

Industrial Control Systems (ICSs) are used to automate critical control processes such as production lines, electrical power grids, gas plants and others. In the past, security means in ICSs were mostly achieved through isolation based on the control of physical access. But currently, the Ethernet and the IP protocol stack are becoming the main part of any plant and factory network. So as a consequence, thousands of ICS components e.g. PLCs are directly reachable from the internet [1], [2]. Although only one PLC may be reachable from outside, this exposed PLC is likely to be connected to internal networks e.g via PROFINET with many more PLCs. This is what is called the deep industrial network. Therefore attackers can leverage an exposed PLC to extend their access from the internet to the deep industrial network. Another true fact was pointed out in [3]. The report showed that many devices are exposed to the internet without any security in mind, as well as many ICS operators have lack of knowledge of how to secure their devices within their own operations, or organizations may know, but are may not perceive the potential risk costing more than the mitigation. We totally agree with these previous reports that many manufactures are still not willing to secure their industries unless they are specifically told to do so, or they will often perform security upgrades only if it is absolutely necessary i.e. if something severe occurred.

Modern ICS components are increasingly connected over PROFINET I/O communication to exchange data between the controllers and other industrial devices. This concept is based on the Ethernet standard that was given by IEEE 802.3 (Industrial of Electrical and Electronics Engineers), and allows the connected stations to establish and maintain connectivity through three different channels: Real-Time (RT), Non-Real-Time (NRT), and Isochronous Real-Time (IRT). These channels coexist in Application Relation (AR) between nodes, and satisfy all the requirements for industrial automation. Although integrating PROFINET I/O with ICSs provides a better network connectivity and a more streamlined control process, it also comes with its own security challenges. This is due to the fact that PROFINET I/O nodes do not have any endpoint security functionality, which makes them exposed to a variety of attacks such as man-in-the-middle (MITM), denial of service (DoS), replay attack, false data injection (FDI), etc. once a malicious adversary gets access to a target device or its network.

One of the well-known vulnerabilities that might compromise the communication integrity of PROFINET I/O systems are improper/fake sensor readings or actuator values exchanged between the connected stations i.e. between IO-Controllers, IO-Devices, and IO-Supervisor. These threats are known as deception attacks [4] or false data injection (FDI) in the IT world and occur when the physical values of a hardware device are manipulated by sending fake values or signals to the victim device. Such threats are very severe because programmable logic controllers (PLCs) keenly rely on reading accurate sensor measurements to safely control critical processes in real time, and any successful FDI attack might eventually cause a significant damage on the infected system.

In PROFINET I/O systems using real time channels, the nodes normally exchange I/O process data through specific frames namely PROFINET I/O Real Time (PNIO-RT) frames see figure 1. These frames have fixed structure features e.g. packet size, packet type, frame identifier, data size, etc. The

978-1-7281-9023-5/21/$31.00 ©2021 IEEE

RT data field in each PNIO-RT frame contains the bytes that represent either sensor measurements or actuator values depending on the transaction direction between the nodes e.g. from IO-Controllers to IO-Devices for actuator values, and from IO-Devices to IO-Controllers for sensor measurements.

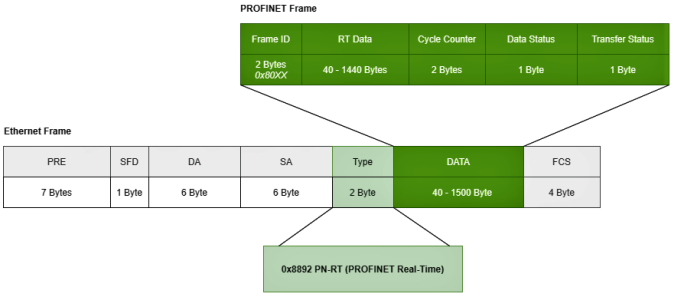

Fig. 1: Ethernet message structure

The approach we take in this paper is to manipulate these bytes by performing an FDI attack scenario without any prior knowledge of the target system, the data exchanged between stations, the physical process, or even the system parameters. For achieving this, we introduce a new attack approach based on integrating an I/O Database during the exploit scenario. This method is based on collecting network captures that contain actual sensor and output values from the target system (prior to launch our FDI attack) which allows the attacker to interrupt, compare, and then replace the correct I/O process data with false one from the I/O database. This technique does not require from the adversary to map the I/O data bytes to its readable version as most of the previous works assumed e.g. [16] which is, in our opinion, not practical as the attacker is not familiar with the system he aims to exploit. However, our new approach is more realistic and easy to implement as we use a simple Python script based on Scapy to filter, extract and store the PNIO-RT frames in the I/O Database. Our full attack-chain consists of two main phases: **Offline**: this includes sniffing and collecting data prior to our attack. **Online**: injecting and forwarding false data to the victims in real time.

This work is summarized as follows: we discover all PROFINET I/O enabled devices in the network using our PN-IO DCP (Discovery and Configuration Protocol) scanner. Then we create an I/O database by collecting multiple network captures from the target system, and processing each network capture by extracting and grouping the PNIO-RT frames in inputs/outputs pair. Afterwards we compromise the trust connection between the PROFINET I/O stations by implementing MITM attack based on port stealing. Finally, we inject false data in the network based on our I/O database see section IV. Please note that in this work, we are only interested in compromising the deep network i.e. having access to the ICS network from outside is out of the scope of this work and can be achieved via typical attack vectors in our IT world such as infected USB, vulnerable web server, etc. Our full attack-chain is implemented on a real industrial setting using an S7-300 PLC as IO-Controller, and an S7 CP 343-1 Lean as IO-Device. The rest of the paper is organized as follows. We compare our work with related ones in section II. Section III gives an overview of our experimental setup, and our attack is illustrated in IV. We discuss the results of our new approach in V, and finally conclude the paper in VI.

II. Related Work

In the recent years, many previous efforts [5]–[15] discussed that sensors on a field bus network send information to the PLC where it is used as an input into the control program, and showed that if the data was modified or inserted, the PLC reads consequently incorrect information and makes wrong decisions that effect on the physical process. In a similar vein, output data could be manipulated or inserted between the PLC and the controlled device causing a similar situation. All the above works were done on field-bus communications based on MODBUS and/or Ethernet protocols assuming that an attacker is familiar beforehand with the target system, and has a prior knowledge of the system parameters. However, this strong assumption makes their attacks hard to be implemented for a practical scenario in the real world. Opposed to their works, we present in this paper a real-time false data injection attack that is implemented on PROFINET I/O field-bus protocol and that does not require from the adversary to be familiar with neither the physical process nor the data packets exchanged between the stations.

The port stealing approach is widely used in compromising PROFINET I/O systems. As an earlier attempt to inject false data into PLCs through port stealing is presented in [17]. The authors showed that it is possible to attack and gain control over PROFINET I/O nodes. However, their approach was not implemented on a real hardware. Another research group managed to exploit the vulnerability of the PROFINET Discovery and Basic Configuration Protocol (DCP) [18]. They performed Denial of Service (DoS) attacks through port stealing against the application relation (AR) between the IO-Controller and the IO-Device. Their attack aimed only at breaking the trust relation between PROFINET stations, and was not designed to inject fake data after stealing the port like ours does. The authors of [19] described also a false data injection attack via the port stealing technique on PROFINET IO-Controllers. They managed to set up an AR with the IO-Controller and send fake input data by crafting DCP identify responses. Their attack was detectable and could be exploited only in the set up process and had no influence on the availability of the automation system during operation. Our work differs in that after implementing the port stealing attack, we replace the correct data with false one based on utilizing an I/O Database that contains already collected data from previous network captures. This makes our method hard to detect as we use the same data that the stations exchanged. In [20] a paper presented a signature-based IDS for industrial control systems. As motivation for this IDS, an attack aiming to disturb a stepping motor with no knowledge about the

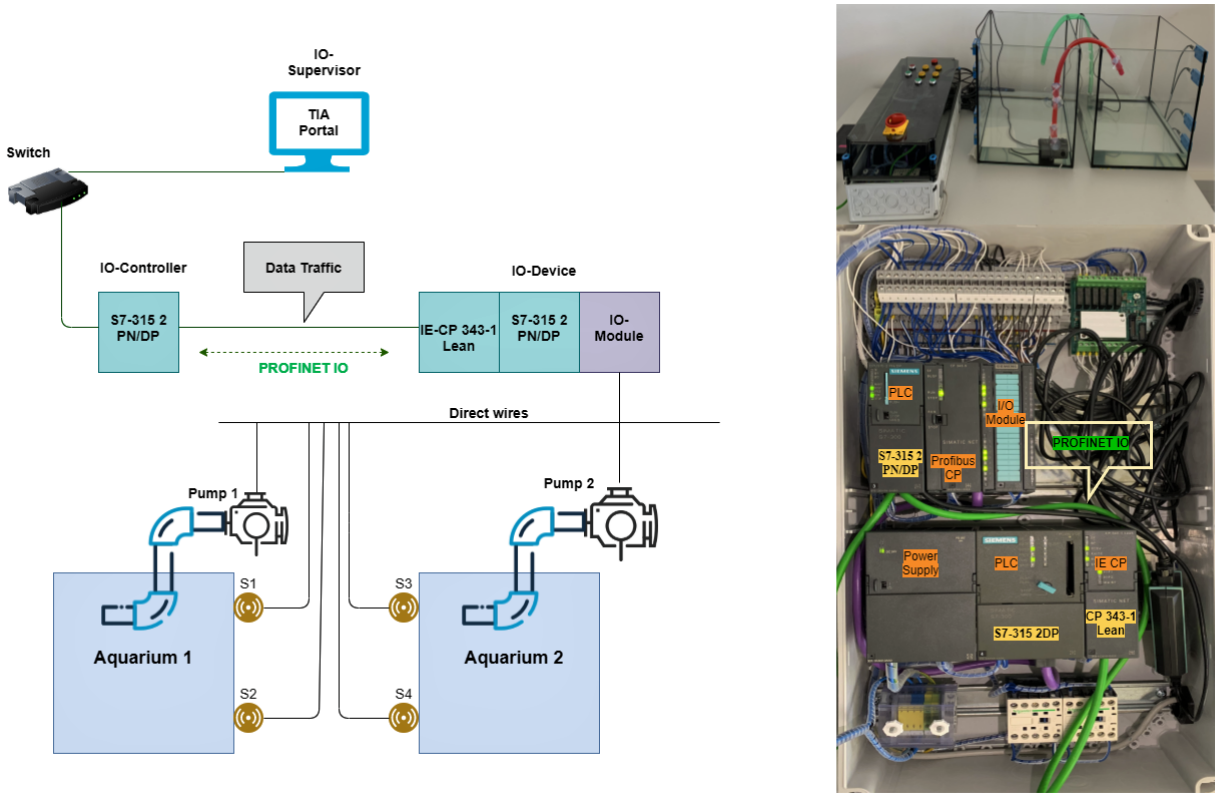

Fig. 2: Experimental Set-up

industrial process was performed. The authors focused on replaying sniffed network packets to achieve a successful attack based on the port stealing technique. We believe that the attack scenario described in their paper is not feasible for a real setup, as well as the authors did not describe clearly how they managed to capture the packets between the motor and the PLC to replay the traffic back. In 2021, a recent published paper [16] implemented a false data injection against a PROFINET I/O system. They showed that an attacker can sniff and manipulate the data exchanged after stealing the port from the IO-Controller. The authors assumed that the attacker has a previous knowledge of the physical process and the control system. Therefore he can map the raw I/O data to the actual sensor readings. We believe that this given assumption makes their work also not practical for a real scenario as the attacker is not supposed to be familiar with the target system in advance.

III. EXPERIMENTAL SET-UP

In this section, we describe our experimental set-up used to test our attack scenario presented in this paper. As shown in figure 2[1] there are two aquariums filled with water that is pumped from one to the other until a certain level is reached and then the pumping direction is inverted. The entire system is controlled by a PLC (S7-315 2 PN/DP) which is connected to a remote I/O module using PROFINET I/O standard and exchanging data cyclically (sensor and actuator values) over the network via industrial Ethernet communication processor IE-CP. The physical process is monitored by the TIA Portal software installed on the engineering station which is here a normal PC. However, in our example application we have three stations. In the first station, an S7 315-2 PN/DP CPU was set as IO-Controller. The second station is the IO-Device. it consists of an S7 315 DP CPU and a communication processor CP 343-1 Lean. this station is connected directly to an external I/O module which is attached to the inputs and outputs hardware i.e. the two pumps and the four digital sensors as shown in figure 2. Both stations are exchanging input and output data cyclically in real time using the PROFINET I/O protocol. The third station is the IO-Supervisor which represents the engineering station in this example, and all the three stations are connected to a 100 Mbit/s industrial switch.

[1]please note that we are using these industrial settings also in experiments run in our earlier publications [23], [24], [25]

IV. ATTACK DESCRIPTION

Figure 3 shows an overview of the attack scenarios we perform to inject false data in the network traffic exchanged between the PROFINET I/O nodes. To achieve a fully blind FDI attack scenario, we need first to discover the network topology of the target PROFINET I/O system, and then collect PNIO-RT packets from the traffic to create our I/O database that contains actual sensor and actuator values. However, these two steps are done prior to our injection attack. After

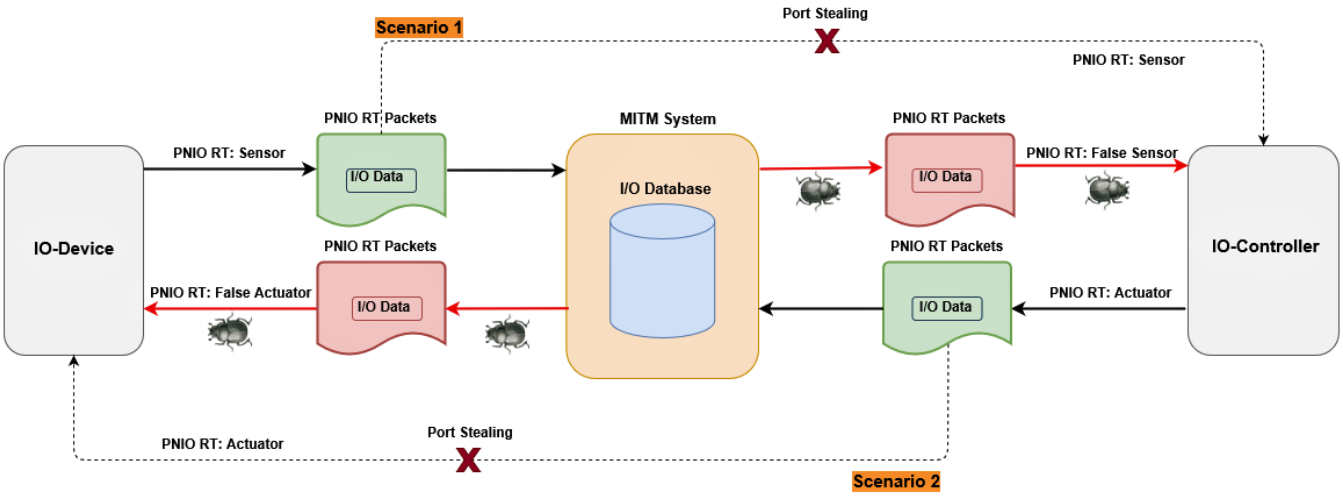

Fig. 3: High-level overview of our false data injection attack: scenario 1 manipulating senor data upper part of the figure; scenario 2 manipulating control commands lower part of the figure

collecting the needed data from the target, we start our main attack by stealing the port from the nodes and send false data packets to the victim devices using our I/O database created earlier. In the following we illustrate the two phases of our full attack-chain in detail.

A. *Pre-Attack Phase (Offline)*

Here, the attacker aims to get an overview of the network and the devices' roles that are connected in the target system as a first step, and then to sniff and to collect data packets that the stations exchange cyclically over the PROFINET I/O frames.

1) Discovering the Network Topology:

For obtaining all the required information about the target system for our attack, we use a PN-IO DCP scanner introduced in our paper [23]. Our scanner is a python script based on Scapy, and sends a DCP identify request via multicast to the network. Each connected device returns its identifying parameters. Table 1 shows information gathered about all PROFINET-enabled devices such as names, MAC addresses, IP addresses, vendors, etc. For our example, our scanner managed to find two devices available. The first node is an IO-Controller located at the IP address *192.168.0.1* using the MAC address *00:1b:1b:23:fb:fe*, whereas the second one is an IO-Device located at the IP address *192.168.0.2* using the MAC address *20:87:56:05:06:15* to connect with the other station.

2) Sniffing and Collecting Data:

After discovering and determining the role of each device in our PROFINET I/O system, the next step is to collect sensor and actuator values. To create an I/O Database, we first sniff and record the entire network stream between the stations using a sniffing network software e.g. Wireshark. The nodes exchange I/O process data through PNIO-RT frames, precisely class 1 (RTC1) frames see figure 4. These frames have the same structure features which make them easy to be recognized and extracted out during an ongoing network traffic.

TABLE I: Output of executing our PN-IO DCP scanner

Parameter	Device 1	Device 2
MAC Address	00:1b:1b:23:fb:fe	20:87:56:05:06:15
Device ID	257	515
Device Role	IO Controller	IO Device
Device vendor	b' S7-300	b' S7-300 CP
IP Address	192.168.0.1	192.168.0.2
Network Mask	255.255.255.0	255.255.255.0
Vendor ID	42	42

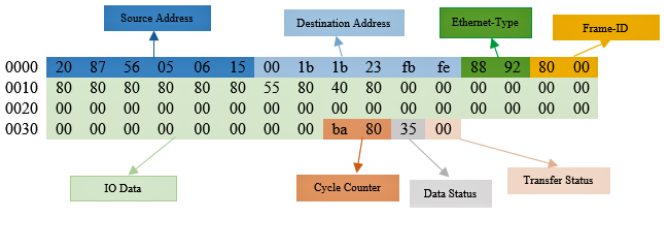

Fig. 4: PROFINET I/O Real Time frame structure

As shown in figure 5, For collecting a sufficient number of sensor measurements and actuator values, the sniffing process should last for a reasonably long period of time e.g. in this work, we sniff the network for approx. 30 minutes. Then the stream captured is filtered to retrieve the PNIO-RT frames only using the unique packet type (0x8892) and frame ID (0x80xx) bytecode. According to our prior knowledge to the devices roles (from the previous step), we can check the source and destination MAC addresses of each PNIO-RT frame and group the captured frames into a pair of Pcap files (sensor data and control command files) accordingly.

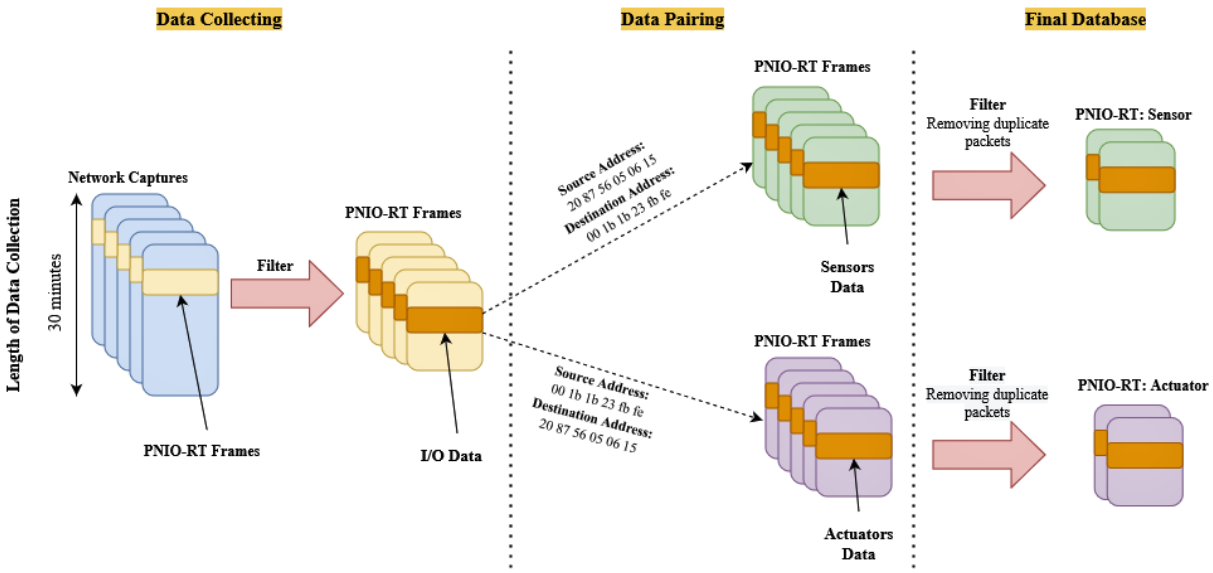

Fig. 5: Scheme of creating I/O Database

To quicken the comparison process during our injection, we need to remove the duplicate packets received during the sniffing period i.e. those containing similar I/O data for each Pcap file, keeping only the packets that differ from each in the I/O process data bytes. This is done by comparing the I/O data bytes of each PNIO-RT frame to the one of the other frames using byte comparison tools (Burp Suite Comparator[2]). Please note that the location of I/O data bytes is always static in PNIO-RT frames of the same size e.g. for our example application, the nodes exchange PNIO-RT frames of a size of 60 bytes, and the I/O data bytes are located between the byte number 17 and 56 as shown in figure 4.

For our example application given in section III, we managed to create an I/O Database containing 7 sensor reading frames as inputs, and 2 actuator value frames as outputs. It's worth mentioning that pairing the captured frames in our I/O database into inputs and outputs Pcap files, helps the attacker to compare and replace the I/O data bytes with false ones online, and to win the strict race condition that PROFINET I/O nodes must meet (as illustrated in the next subsection), before he replies his forged PNIO-RT packets to the network.

B. Attack Phase (Online)

In this phase, false data is injected in the network traffic based on our I/O database approach.

1) Port Stealing Approach:

Before pushing incorrect data in the network, we need first to interrupt the Application Relation (AR) between the IO-Controller and the IO-Device. Technically, a typical industrial Ethernet switch controls and manages the binding of each MAC address to a certain switch port in an Address Resolution Protocol (ARP) mapping table.

[2]https://portswigger.net/burp

Once the MAC address at any port changes because a new device has been added to the network, the switch updates its mapping table and the old entry is removed. Therefore, we just need to flood the switch with forged gratuitous ARP packets registering the attacker's MAC address in place of the victim host to achieve a successful port stealing as shown in figure 6. This technique is widely used in MITM attacks in

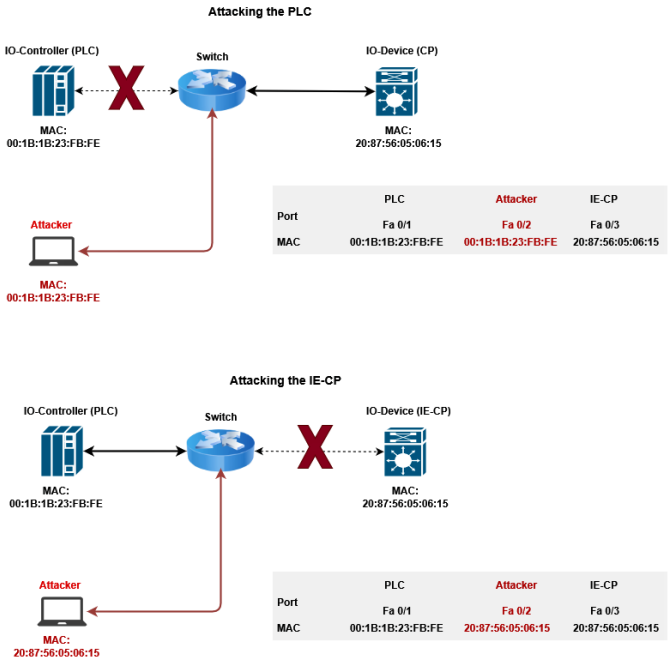

Fig. 6: Data exchange configuration after ARP Poisoning attack: Scenario 1 stealing the port from the IO-Controller; Scenario 2 stealing the port from the IO-Device

traditional IT switched networks, where the switch assumes that the victim device is now using another switch port and forwards the packets to the new port. It is worth mentioning that this attack has a challenge. The frequency of sending ARP packets to the victim must be sufficient. Meaning that if the target device sends ARP packets before the attacker, the switch always updates the binding of the port to the victim MAC address back and forth. To overcome this issue, it is required from the attacker to send ARP packets at a much higher frequency than the victim does. However, in our work a frequency up to 1 ms was sufficient to prevent the switch from updating its mapping table. Now if the packets of two devices are redirected through port stealing to the attacker, he just needs to forward the packets accordingly to achieve a fully MITM attack.

2) Injecting and Forwarding False Data:

The final step is to replace the I/O data exchanged between the stations with false data included in one of the already recorded frames. This is done by using our I/O database approach explained in section IV.A.2. Algorithm 1 gives the main core of our attack script used to inject false data. The algorithm is realized by a python script and the third party library used here is Scapy.

As can be seen from algorithm 1, after interrupting the AR between the stations in the previous step, the attacker listens and receives a PNIO-RT frame in the very next PROFINET update cycle. The I/O Data field of the frame received is then compared to the data fields of the already recorded frames existing in our I/O database, taking into account only frames recorded for the same communication direction. This comparison process aims at finding an appropriate frame having different I/O data bytes to the one captured and is repeated starting from the first frame in the database. Once a frame is found, its I/O data field will be used as the new I/O data field in our forged PNIO-RT packet, and the port stealing attack is then stopped. Finally the malicious packet that contains the false I/O data is forwarded to its final destination.

Please note that forwarding the crafted packet back to the network has two challenges. First, the malicious packet cannot be directly forwarded due to the fact that each PNIO-RT frame has a cycle counter see figure 4. The 2 bytes value that the cycle counter has is always read, and the number of missing packets between the consecutive cycles is set inside the TIA Portal software. To overcome this security challenge, the forged packet should always have the cycle counter value of the next expected PNIO-RT packet to be received at the final destination. However, this is easy to be solved as the cycle counter values differ in a constant number always e.g. in our system the counter cycle number always increases by 256 per cycle. The second challenge is that after stopping the port stealing, the attacker should always win the race condition by sending the malicious PNIO-RT frame to the victim before the correct data is sent from the original source. In PROFINET I/O systems, the transmission interval (PROFINET update time cycle) divided into four phases named Send Clock as shown in figure 7. This parameter represents the frequency of exchanging data between the IO-Device and IO-Controller. In fact the PROFINET update time cycle results from the Send Clock × Reduction Ratio. Therefore, a Send Clock of 1ms and a Reduction Ratio of 4 means that I/O data is sent every 4 ms. However, the Send Clock is normally set from 2 to 512 milliseconds and differs from one system to another based on their requirements. For the most industrial PROFINET I/O systems, the Send Clock is set at 128 milliseconds to avoid extreme network traffic overloads. Meaning that each PROFINET node gets updated every 0.5 second. By assuming that, the Send Clock in our example application is set to 128 milliseconds. Meaning that, the attacker needs to send his false data in less than 0.5 second to avoid updating the target PROFINET node with correct I/O data.

Algorithm 1 FDI Attack based on I/O Database using Scapy

```
Function Inject (iface=eno1, SrcPort)
packet = sniff (iface = eno1, timeout = cfg_sniff_time)
save_Pcap (packet, filter = '0x8892, Frame_id = '0x8000', sniff.Pcap)
for pkt in rdpcap (sniff.Pcap) do
    src_mac = pkt [1:6], dest_mac = pkt [7:12], data = pkt [17:56], coun_cyc = pkt [57:58], data_status = pkt [59]
    if (src_mac ≠ plc_src_mac) then
        for p in rdPcap (inputs_Pcapfile) do
            if (data ≠ load_packet (p[17:56])) then
                fgd_data = load_packet (p[17:56]) break
            end if
            P = P+1
        end for
    else
        for p in rdPcap (outputs_Pcapfile) do
            if (data ≠ load_packet (p[17:56])) then
                fgd_data = load_packet (p[17:56]) break
            end if
            P = P+1
        end for
    end if
    if data ≠ fgd_data then
        break
    end if
    pkt = pkt + 1
end for
fgd_pkt = padd_pkt (raw (PNIORealTime (CycleCounter= 'coun_cyc + 256', fgd_data, data_status, len = '60')
stop_port_stealing ()
while time_slot() do
    sendp(fgd_pkt, iface=eno1)
end while
END Function
```

V. Results and Discussion

In this work, we test our false injection attack approach in the following two scenarios:

- False Sensor Data: figure 8 describes this scenario. First the port is stolen from the PLC and as a consequence the PLC stops receiving any real-time data from the IE-CP and the data is redirected to the attacker. The packet received on the attacker machine is then compared and the I/O data bytes are replaced with false ones based on our I/O database. The

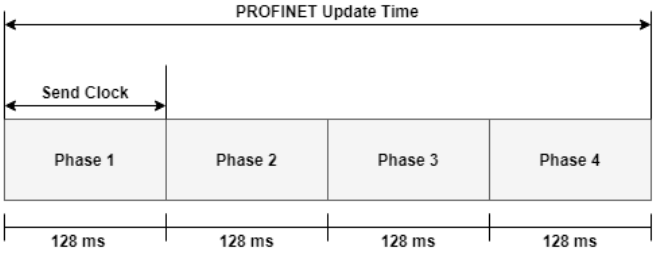

Fig. 7: PROFINET update time cycle

cycle counter value is read and increased by 256 to match the expected counter value of the next PNIO-RT frame, before the port stealing attack is stopped. Finally, our forged packet is sent at the next PROFINET update cycle taking into account the race condition i.e. in less than 0.5 second.

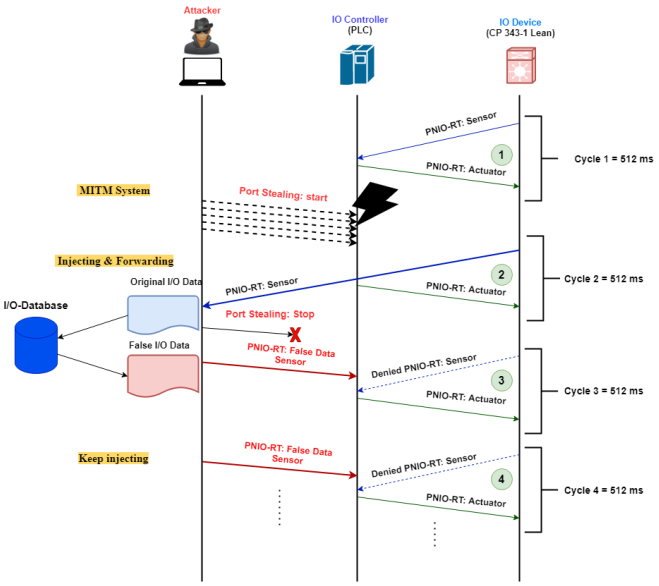

Fig. 8: False Data Injection against IO-Controller

- False Actuator Value: figure 9 shows this scenario. We aim at manipulating the control command sent from the PLC to the IE-CP. Also here first the port is stolen from the IO-Device, and as a result the real-time data is redirected to the attacker machine. The data bytes of the PNIO-RT packet received from the PLC are then compared to the ones in our I/O database, and are replaced with false data bytes in the forged packet. The cycle counter value is increased by 256, and the malicious packet is then replied back to the IO-Device after the port stealing is removed.

As a consequence of executing our injection attack-chain against the example application given in section III, we managed successfully to trick the PLC by reading false sensor measurements (in the first scenario) and the IE-CP by receiving false actuator values (in the second scenario). Our attack approach leads the tested PROFINET I/O system in both

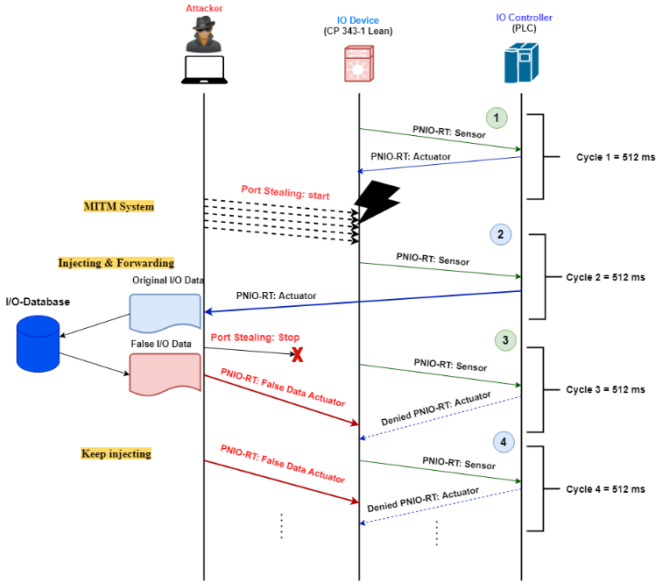

Fig. 9: False Data Injection against IO-Device

scenarios to operate the physical process incorrectly depending on the false data frames chosen from our I/O database, keeping the infected system run at a certain operational state as long as the injection attack lasts e.g. the water exceeds the limits causing water overflow due to reading or acting false data values. However, In order to increase the success probability of such an attack, the PLC/CP should continually receive our crafted data rather than the original one, therefore the attacker needs to send each false data for more than one update cycle as seen in figure 8, 9. Furthermore, if the attacker keeps the port stealing attack run for a long duration, this will disturb the AR communication between the PLC and the CP, and our attack becomes a Denial-of-Service (DoS) not a false data injection (FDI).

VI. Conclusion and Future Work

In this paper, we presented a fully-blind false data injection attack against a PROFINET I/O system based on our new I/O database approach. for a practical implementation we performed our full attack-chain in two scenarios on a real hardware used in industrial settings. We found that the both target PROFINET I/O nodes in our testbed were tricked i.e. the IO-Controller by reading false sensor readings in the first scenario, and the IO-Device by executing false control commands in the second scenario. As a result, the physical process controlled by the infected devices is run incorrectly, and the system remained operating at a certain state as long as the attacker keeps sending false data, depending on the data packet chosen from our I/O database.

To mitigate the effect of such attacks, we highly recommend to improve the isolation from other networks [21], combined with the standard security practices [22]. Furthermore, the detection mechanism introduced in [20] might also be used

to prevent our attack scenarios. Their open-source intrusion detection framework is designed based on port stealing and PN-IO detectors. In the meantime the authors developed a so-called processor for Snort to react on PN-IO Ethernet frames which reveals any forged frame sent from an unauthorized user. However, in our opinion the best solution to make the industrial network more resistant to FDI attacks is when different prevention mechanisms are in place e.g. a demilitarized zone (DMZ) and network segmentation to improve the attack prevention, and layered defense in depth strategy to further improve the detection of successful malicious injections.

The exploit in this paper is efficient but for our future work, we will investigate the ability to perform false data injection attack against modern S7 CPUs such as S7-1200 and S7-1500 PLCs. We are aware to the fact that attacking the PROFINET Systems based on modern PLCs is more challenging as Siemens claims that it provides its new devices with improved security means.

REFERENCES

[1] J.-O. Malchow and J. Klick, Sicherheit in vernetzten Systemen: 21. DFN-Workshop. Paulsen, C., 2014, ch. Erreichbarkeit von digitalen Steuergeräten - ein Lagebild, pp. C2–C19.

[2] B. Radvanovsky, "Project shine: 1,000,000 internetconnected scada and ics systems and counting," Tofino Security, 2013.

[3] R. Radvanosky, & J. Brodsky, " Project Shine (SHodan INtelligence Extraction) Findings Report," USA. 10th SANS ICS Security Summit, Orlando, FL, February 23- 24, 2015.

[4] S. M. Dibaji, M. Pirani, D. Flamholz, A. M. Annaswamy, K. H. Johansson, A. Chakrabortty, (2019), " A Systems and Control Perspective of CPS Security," 10.1016/j.arcontrol.2019.04.011.

[5] D. Urbina, J. Giraldo, N.O. Tippenhauer, and A. Cardenas, "Attacking Fieldbus Communications in ICS: Applications to the SWaT Testbed," 2016, doi:10.3233/978-1-61499-617-0-75.

[6] S. McLaughlin, and S. Zonouz, "Controller-aware false data injection against programmable logic controllers," in 2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)., pages 848-853. dio: 10.1109/SmartGridComm.2014.7007754.

[7] S. Sridhar, and G. Manimaran, "Data integrity attacks and their impacts on SCADA control system", in IEEE PES General Meeting, 2010, pages 1-6. doi: 10.1109/PES.2010.5590115.

[8] G. Bernieri, E. Etchevés Miciolino, F. Pascucci, and R. Setola, "Monitoring system reaction in cyber-physical testbed under cyber-attacks," Computers & Electrical Engineering, vol. 59, pp. 86 – 98, 2017.

[9] A. P. Mathur and N. O. Tippenhauer, "SWAT: a water treatment testbed for research and training on ICS security," in 2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater), pp. 31– 36, 2016.

[10] S. Adepu and A. Mathur, "Distributed attack detection in a water treatment plant: Method and case study," IEEE Transactions on Dependable and Secure Computing, pp. 1–1, 2018. 14 V.

[11] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A dataset to support research in the design of secure water treatment systems," in Critical Information Infrastructures Security (G. Havarneanu, R. Setola, H. Nassopoulos, and S. Wolthusen, eds.), (Cham), pp. 88–99, Springer International Publishing, 2017.

[12] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, "Wadi: A water distribution testbed for research in the design of secure cyber physical systems," p. 25–28, Association for Computing Machinery, 2017.

[13] V. K. Mishra, V. R. Palleti, and A. Mathur, "A modeling framework for critical infrastructure and its application in detecting cyber-attacks on a water distribution system," International Journal of Critical Infrastructure Protection, vol. 26, p. 100298, 2019.

[14] F. Zhang, H. A. D. E. Kodituwakku, J. W. Hines, and J. Coble, "Multilayer data-driven cyber-attack detection system for industrial control systems based on network, system, and process data," IEEE Transactions on Industrial Informatics, vol. 15, no. 7, pp. 4362–4369, 2019.

[15] X. Li, C. Zhou, Y.-C. Tian, N. Xiong, and Y. Qin, "Asset-based dynamic impact assessment of cyberattacks for risk analysis in industrial control systems," IEEE Transactions on Industrial Informatics, vol. 14, pp. 608–618, 2018.

[16] M. Noorizadeh, M. Shakerpour, N. Meskin, D. Ünal, and K. Khorasani, "A Cyber-Security Methodology for a Cyber-Physical Industrial Control System Testbed," IEEE Access. PP. 1-1. 10.1109/ACCESS.2021.3053135, 2021.

[17] J. Akerberg and M. Bjorkman, "Exploring Security in PROFINET IO," 2009 33rd Annual IEEE International Computer Software and Applications Conference, Seattle, WA, 2009, pp. 406-412, doi: 10.1109/COMPSAC.2009.61.

[18] S. Mehner, H. König, "No Need to Marry to Change Your Name! Attacking Profinet IO Automation Networks Using DCP," 2019, doi:10.1007/978-3-030-22038-9_19.

[19] A. Paul, F. Schuster, and H. König, "Towards the protection of industrial control systems – conclusions of a vulnerability analysis of profinet io," In: Rieck, K., Stewin, P., Seifert, J.P. (eds.) Detection of Intrusions and Malware, and Vulnerability Assessment. pp. 160–176. Springer Berlin Heidelberg, Berlin, Heidelberg (2013).

[20] S. Pfrang, and D. Meier, "Detecting and preventing replay attacks in industrial automation networks operated with profinet io," Journal of Computer Virology and Hacking Techniques 14(4), 253–268, Nov. 2018, doi:10.1007/s11416-018-0315-0.

[21] M. Stouffer, V. Pillitteri, "Guide to industrial control systems (ics) security," NIST special publication, 2015.

[22] "Framework for improving critical infrastructure cybersecurity version 1.1,"National Institute of Standards and Technology, Tech. Rep," 2018, Available at: https://doi.org/10.6028/NIST.CSWP.04162018.

[23] W. Alsabbagh and P. Langendörfer, "A Remote Attack Tool against Siemens S7-300 Controllers: A Practical Report," presented at 11. Jahreskolloquium Kommunikation in der Automation (KommA 2020), Lemgo, Germany, Oct. 29, 2020.

[24] W. Alsabbagh and P. Langendörfer, "A Stealth Program Injection Attack against S7-300 PLCs," Proc. 22nd International Conference on Industrial Technology (ICIT 2021), 986 (2021).

[25] W. Alsabbagh and P. Langendörfer, "Patch Now and Attack Later - Exploiting S7 PLCs by Time-Of-Day Block," to be presented at the 4th IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2021), Victoria, Canada, May 10-12, 2021.