Introduction to Network Forensics

ICS/SCADA Environment

Toolset, Document for students

1.1

AUGUST 2019

About ENISA

The European Union Agency for Cybersecurity (ENISA) is a centre of network and information security expertise for the EU, its member states, the private sector and Europe's citizens. ENISA works with these groups to develop advice and recommendations on good practice in information security. It assists EU member states in implementing relevant EU legislation and works to improve the resilience of Europe's critical information infrastructure and networks. ENISA seeks to enhance existing expertise in EU member states by supporting the development of cross-border communities committed to improving network and information security throughout the EU. More information about ENISA and its work can be found at www.enisa.europa.eu.

Contact

For queries in relation to this paper, please use:

csirt-relations@enisa.europa.eu
PGP Key ID: 31E777EC 66B6052A
PGP Key Fingerprint: AAE2 1577 19C4 B3BE EDF7 0669 31E7 77EC 66B6 052A

For media enquiries about this paper, please use:

press@enisa.europa.eu.

Legal notice
Notice must be taken that this publication represents the views and interpretations of ENISA, unless stated otherwise. This publication should not be construed to be a legal action of ENISA or the ENISA bodies unless adopted pursuant to the Regulation (EU) No 526/2013. This publication does not necessarily represent state-of the-art and ENISA may update it from time to time.

Third-party sources are quoted as appropriate. ENISA is not responsible for the content of the external sources including external websites referenced in this publication.

This publication is intended for information purposes only. It must be accessible free of charge. Neither ENISA nor any person acting on its behalf is responsible for the use that might be made of the information contained in this publication.

ISBN: 978-92-9204-288-2, DOI: 10.2824/995110

Table of Contents

PARAMETER	DESCRIPTION	DURATION
Main Objective	In this exercise, the trainees will be taken through an incident response for an attack on an ICS/SCADA environment, starting with the preparation phase, incident analysis and post-incident activity. In the first two tasks, the trainees will have to set up an IDS for the SCADA network using well-established (open source) software solutions. Main goal of this part will be to learn where and how place and configure sensor(s) to gain suitable forensic data given a specific network setup. The latter tasks (3-5) will focus on forensic analysis of three attack stages. For each stage, network traffic captures will be given to the students to analyse with the IDS environment they have set-up in the previous tasks of the scenario.	
Targeted Audience	The exercise is dedicated to (new) CERT staff involved in network forensics. The exercise should be also helpful to (all) CERT staff involved in daily incident response.	
Total Duration	8.0 hours	
Time Schedule	Introduction to the exercise and tools overview	2.0 hours
	Task 1: Setting up the monitoring environment	1.0 hour
	Task 2: Baselining regular traffic	1.0 hour
	Task 3: Initial attack analysis	1.0 hour
	Task 4: Second attack stage analysis	1.0 hour
	Task 5: Analyse the attack on the PLCs	1.0 hour
	Summary of the exercise	1.0 hour
Frequency	It is advised to organise this exercise when new team members join a CERT/CSIRT.	

1. What Will You Learn?

1.1 Summary

This scenario will deal with an attack on an ICS/SCADA environment in the energy sector. The successful completion of this scenario will you how to set up and configure a network security monitoring environment, including the baselining of regular (non-malicious) traffic and finally, the successful analysis of a multi-stage attack on the network. During the exercise, you will have to deal with a previously unseen network architecture and to familiarise with an unknown protocol used to control the industrial environment.

In the first two tasks, you will have to set up an IDS for the SCADA network using well-established (open source) software solutions. Main goal of this part will be to learn where and how place and configure sensor(s) to gain suitable forensic data given a specific network setup.

The latter tasks (3-5) will focus on forensic analysis of three attack stages. For each stage, network traffic captures will be given to you to analyse with the IDS environment you have set-up in the previous tasks of the scenario.

1.2 What is ICS/SCADA?

Industrial plants (power plants, factories, oil refineries, etc.) are large, distributed complexes, where operators must continuously monitor and control many different sections of the plant, to ensure its' proper operation.

Before computers were introduced, industrial plants had to rely on (human) personnel to manually control and monitor equipment and processes through push buttons and dials. As plants grew in size, a solution was needed to control and monitor equipment over long distances. With the introduction of computers, it become possible to remotely control and monitor industrial components and processes through *Industrial Control Systems (ICS)*.

The first ICS were simple point-to-point networks connecting a monitoring panel or command device to a remote sensor or actuator. These have since evolved into complex, large-scale networks interconnecting computers, sensors, actuators, *Remote Terminal Units (RTUs)*, and *Programmable Logic Controllers (PLCs)*.

Supervisory Control and Data Acquisition (SCADA) is a control system architecture that allows high-level management systems to interface with peripheral devices such as PLCs from different vendors to perform a supervisory operation. The general model can be seen below, in Figure 1 where:

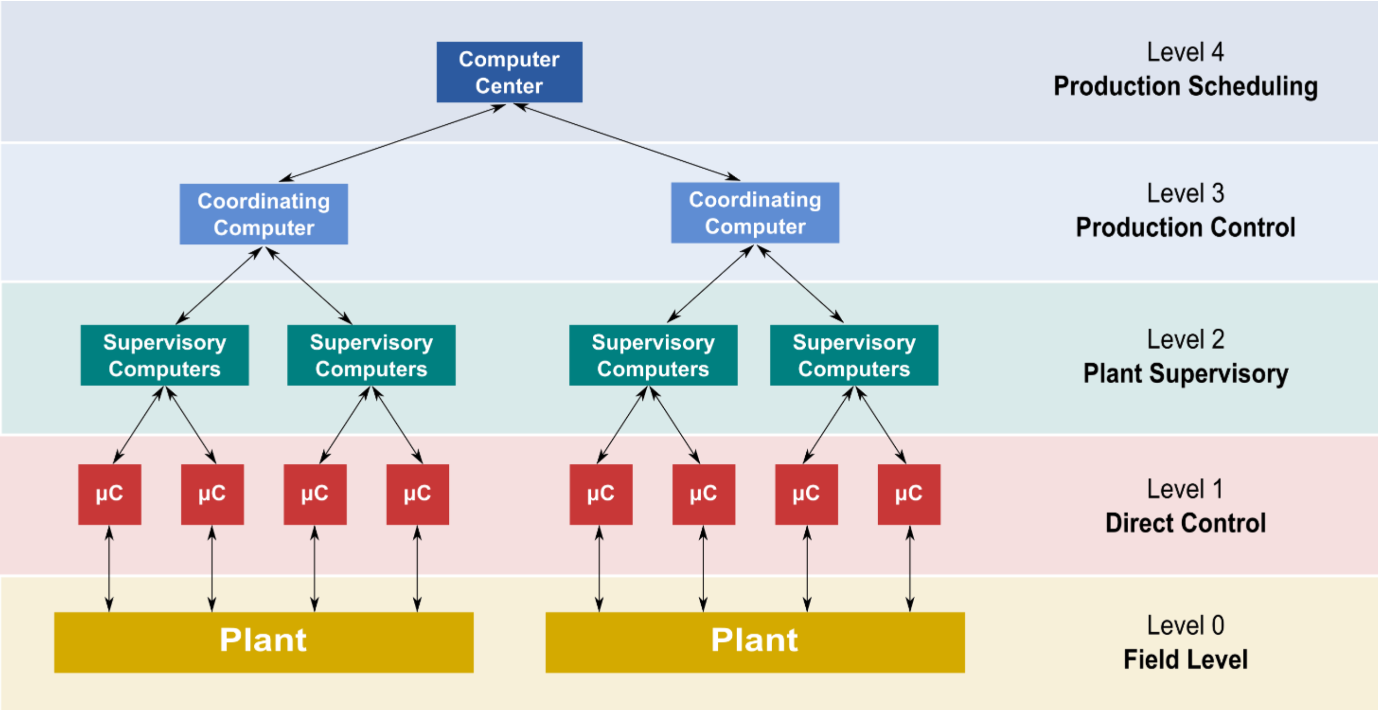


Figure 1. ICS/SCADA control levels

- Level 0
  Contains the field devices such as flow and temperature sensors, and final control elements, such as control valves.
- Level 1
  Contains the industrialised input/output (I/O) modules, and their associated distributed electronic processors.
- Level 2
  Contains the supervisory computers, which collate information from processor nodes on the system, and provide the operator control screens.
- Level 3
  Is the production control level, which does not directly control the process, but is concerned with monitoring production and targets.
- Level 4
  Is the production scheduling level.

1.2.1 What is challenging about SCADA security?

The consequences of intrusions to SCADA systems may be much more severe than in traditional IT-systems. Equipment may be damaged, hazardous (poisonous, radioactive) material released to the environment, or human life may be endangered, even that of people outside the plant. When SCADA systems are attacked that control critical infrastructures, such as transmission of electricity, transportation of gas and oil in pipelines, water distribution, traffic lights, etc., the impacts could range much further than the original compromised systems.

The move from proprietary technologies to more standardized and open solutions together with the increased number of connections between SCADA systems, office networks and the Internet has made them more vulnerable to types of network attacks that are relatively common in computer security. This imposes new challenges to traditional IT-security monitoring, including:

- SCADA environments have a different guiding principle. Foremost importance for SCADA systems is the safety, reliability and availability (SRA) of the (industrial) process, because outages would risk damaging equipment or risking catastrophic failures. For traditional IT-systems, confidentiality, integrity and availability (CIA) of data is the guiding principle.
- SCADA systems and networks were originally not planned with IT-security in mind. Particularly, they lack encryption and authentication.
- Furthermore, with availability being the primary concern, systems may not be updated regularly, thus exposing vulnerabilities for months or longer, as testing on live systems is not possible due to the SRA principle and dedicated test environments are deemed to complex or expensive.
- A multitude of SCADA protocols exist and in general traditional IT-security personnel is unfamiliar with them.

There are many threat vectors to a modern SCADA system. One is the threat of unauthorised access to the control software, whether it is human access or changes induced intentionally or accidentally by virus infections and other software threats residing on the control host machine.

Another is the threat of packet access to the network segments hosting SCADA devices. In many cases, the control protocol lacks any form of cryptographic security, allowing an attacker to control a SCADA device by sending commands over a network. In many cases, attackers were also able to compromise the monitoring systems so that operators were unaware of the ongoing attack (ENISA, 2011).

1.3 The toolset

Most of the actual work will be done in a virtual machine that is supplied to you. The virtual machine image is in the *Open Virtualisation Appliance*[1] *(OVA)* format that has been compressed with the *xz*[2] program. After decompression, the image can be imported to run in any contemporary virtualisation environments that supports OVA images, like VMware, VirtualBox, Hyper-V, Qemu, etc. The image can be downloaded from the following location:

https://www.enisa.europa.eu/ftp/ENISA_INF _5.1.ova

Credentials to the machine:

PARAMETER	VALUE
Username	exercise
Password	enisa

The machine consists of a *Security Onion*[3] Linux distribution with custom installation of *Wireshark*[4] that has the dissectors for the S7comm-plus[5] protocol added to it.

The packet captures mentioned in the following sections are in the folder **`traffic`**.

[1] https./www.dmtf.org/standards/ovf
[2] https://tukaani.org/xz
[3] https://securityonion.net/
[4] https://www.wireshark.org/
[5] https://sourceforge.net/projects/s7commwireshark/

2. Introduction

2.1 Background information

The teacher will give a presentation that covers the topics from chapters 1-4 and will familiarise the students with the basic knowledge needed for the upcoming exercises. It is recommended to do this in a workshop-style approach where students and teacher can discuss ideas, which will make this part less dry and more adaptable to the students' prior level of knowledge. This part could be skipped, if the students already have a high enough knowledge about IDS and network forensics.

3. Exercise Tasks

3.1 Task 1: Setting up the monitoring environment

In accordance to what has been laid out in the previous chapters, the exercise will start with a coverage of the preparatory tasks in network monitoring and forensics, i.e. setting about capturing points, selecting monitoring targets and defining a monitoring policy.

3.1.1 The background

This scenario will take place in a power plant, where you will take the role of network monitoring staff tasked with deploying a Network Intrusion Detection System on a small sub-network. The goal of the NIDS is to detect attacks on the PLCs as well as the workstations in the network. If successful, the NIDS deployed, and the processes developed around it will be used as a pilot to other plant systems.

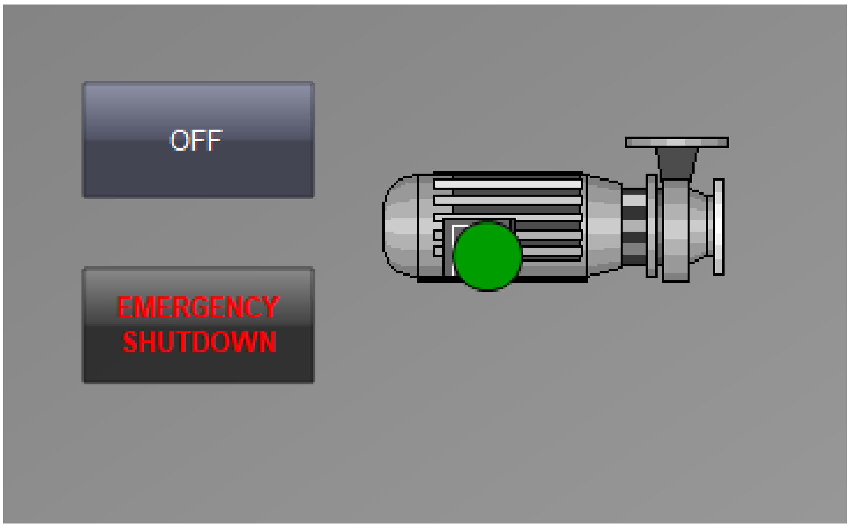


Figure 2. SCADA application

3.1.2 The network

This scenario requires you to analyse the simulated attack on a simulated network in a nuclear power plant, which will include:

- An Engineering workstation for configuring industrial devices, such as programming PLCs
- Two programmable logic controller (PLCs), used to control physical processes, such as opening a valve when a button is pushed on
- A Supervisory control and data acquisition (SCADA) workstation, used to control the industrial process. The application running on the SCADA workstation gives the operator two buttons to control the operation of a pump. One to power the pump off and another button for emergency shutdown if the first button fails to work for some reason. Despite its apparent simplicity, this system is critical to the operation of the plant (see Figure 2)
- The network has no connection to the other networks.

Within this scenario, those systems will be interconnected through a single hardware switch, as shown in Figure .

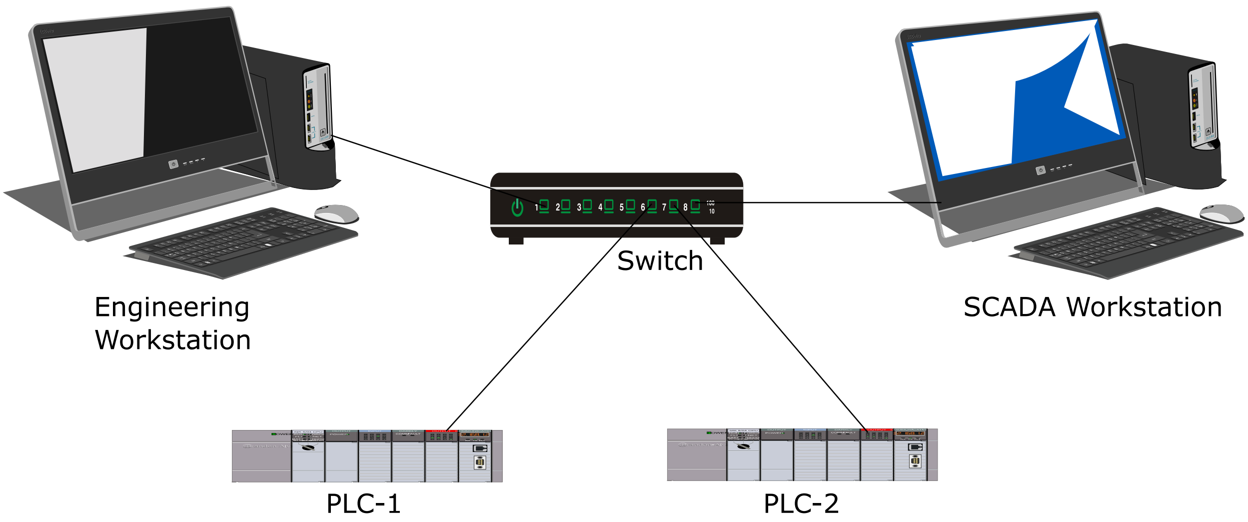


Figure 3. The exercise network

The network traffic data has been generated through the courtesy of National Centre for Nuclear Research (NCBJ, Poland).

3.1.3 Subtask: Decide on monitoring points

In section 1.4 of the Handbook, several different methods of traffic capture have been put forward. It is now on the students to select one for the given network above.

Students: Select one or more capturing points for monitoring the above network. Justify your decision.

Solution: Since traffic to/from all the above systems will need to be monitored, the canonical point for traffic capture is to configure a span-port on the switch where traffic from the four systems (workstations and PLCs) will be mirrored (Figure). This may impose a traffic problem, as the span-port would need 4-8 times the bandwidth of an individual network connection (4 systems times 2 for in- and outgoing traffic). For this exercise, it is assumed that the mirroring port has enough bandwidth.

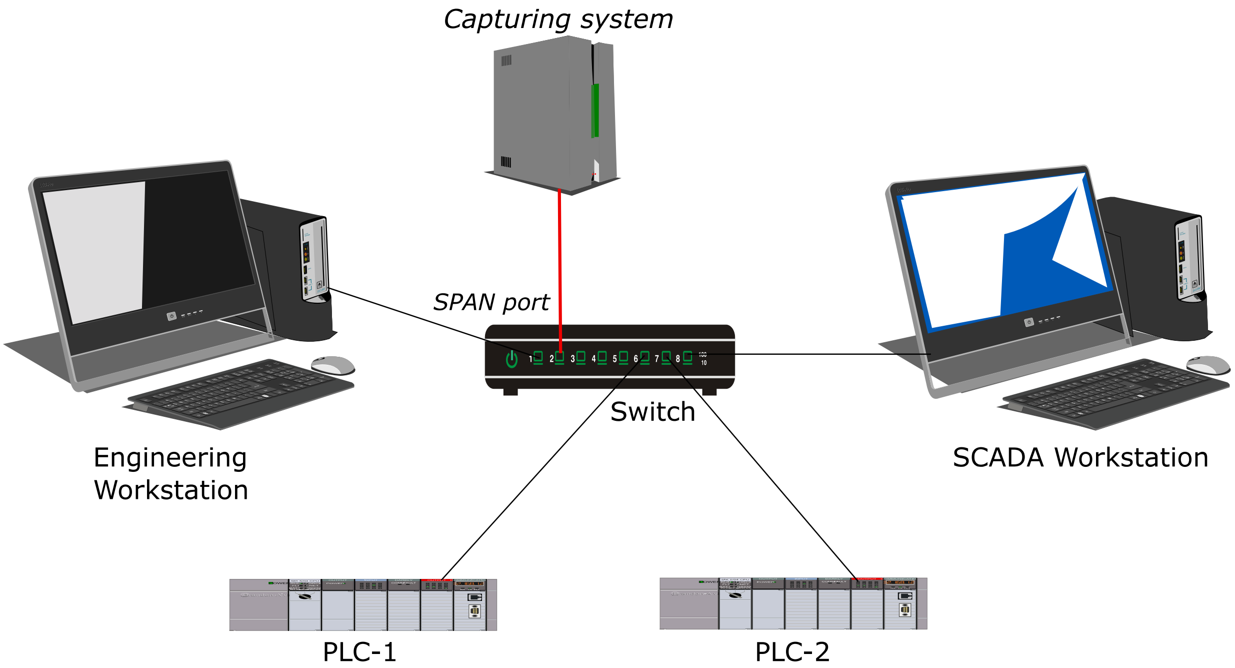


Figure 4. Exercise network with capturing system

Alternative Solution: If the switch does not support port mirroring (or perhaps all ports are already in use), an alternative solution will have to be devised. One could be to use cable taps for both workstations and PLCs as shown in Figure 5. This would require more cables going from the taps to the capturing system: 8 in total, 2 for each system covering in- and outgoing traffic, and correspondingly, 8 network ports on the capturing system, on the other hand, this would avoid bandwidth problems and does not require anything from the switch. The costs for taps, cables and network ports would probably exceed that of a switch with port mirroring support, however.

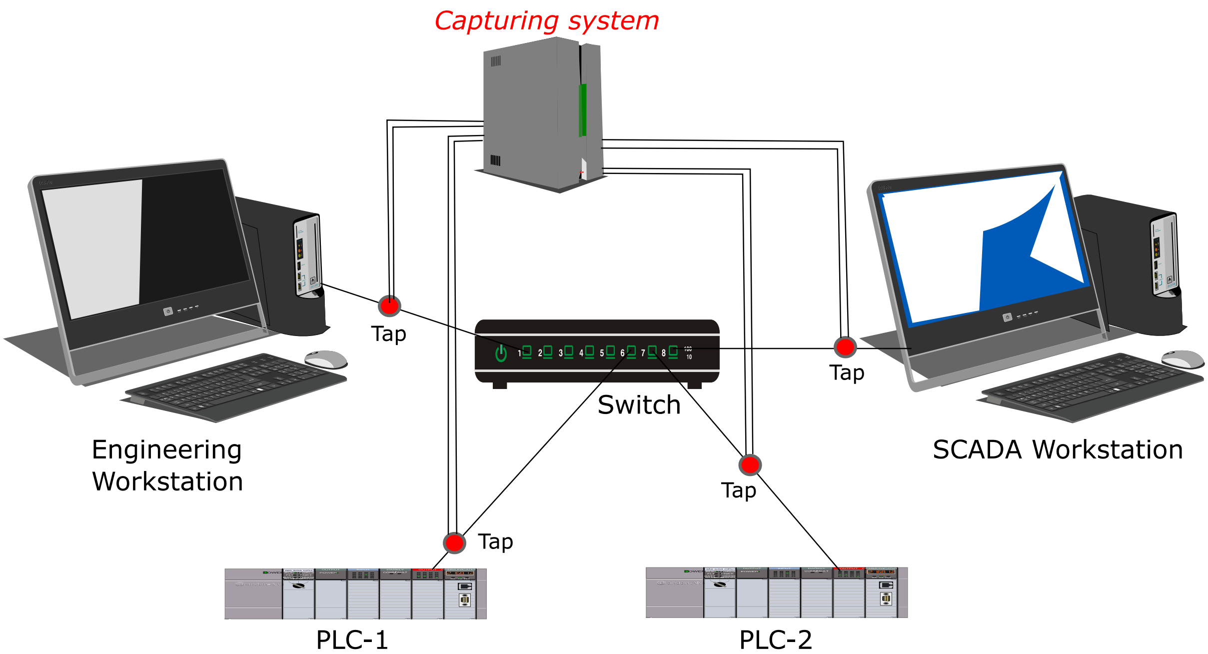


Figure 5. Exercise network with cable taps

3.1.4 Subtask: Develop a monitoring policy

Continuing the path of preparing network monitoring, we now time to decide on how to monitor. In section 2.2, the prerequisites were laid out. This sub-task will let you decide on a monitoring policy as well as targets for the given network.

Students: Select a monitoring policy and target(s) for the above network. Justify your decision.

Solution: Blacklist monitoring is difficult, as there are not enough attack signatures for SCADA networks available, especially for the type of PLCs used in this exercise.

For both anomaly monitoring and policy monitoring, a point can be made.

- The network is small and closed, so it can be expected to have a clear set of traffic patterns that will not change too often. This speaks for anomaly monitoring. In addition, the fact that that the traffic patterns will be known only after baselining (the next sub-task) is another point for anomaly monitoring as one can start right away and refine the policy over time.
  - Details of the policy will have to be postponed until the baselining is done.

- With full control over the systems on the network, a case can also be made for policy monitoring. Only a few key points can already be made:
  - Only the workstations shall communicate with the PLCs
  - Communication shall be limited to port 102/tcp and the S7plus protocols

- The question of with whom (except the PLCs) the workstations should communicate can be left open. If they should communicate, communication should be limited to port 5900/tcp (VNC) and only from the Engineering workstation to the SCDA workstation.
- Both can be combined into a hybrid approach. This should be kept in mind and can be brought back as a point in the summary discussion.

As will be seen later all systems on the network need to be monitored. When talking for individual targets, the following arguments can be made:

- The PLCs should be monitored as they can be attacked from any other system on the network, bypassing any protective measures on the workstations.
- The PLCs should be monitored, as they have no defensive measures on their own. This can be said for the workstations too, but some sort of firewall or IDS/IPS can be retrofitted on them, which is more difficult for the PLCs.
- The SCADA workstation should be monitored, as an attack on this workstation could be used to compromise the SCADA application. It is also the system with the largest attack surface, having two protocols (VNC and S7plus) running.
- In addition, the SCADA workstation could be used to attack the PLCs and as communications between these systems would be considered "normal", the attack would be very hard to detect.
- The engineering workstation will be the one with the largest influence, as it controls the programs that run on the PLCs.

In the end, it depends on how the arguments are weighed.

Since there is no connection to other networks, there is no use of name servers (DNS), NAT or VPN-gateways or automatic address management. Therefore, additional information is not needed or present here. One may argue the lack of NTP, so the investigators should be cautions when comparing timestamps from the different hosts. As this exercise will work only with network packet captures, this will not be a problem.

3.2 Task 2: Baselining of regular traffic

The second part focuses on learning how to get the best out of the IDS system and be able to differentiate between regular traffic patterns and anything malicious/suspicious. One of the main tasks of operating an IDS system is to constantly adjust its' configuration, not only to minimise false alarms, but configuration errors as well. To achieve this goal, you will be presented with a number of prepared network captures they have to analyse and take as input to the IDS configuration.

Students: Assume you had the time to sample some traffic from you network. The file **`normal.pcapng`** will contain traffic without user activity at the SCADA workstation and the file **`button_push.pcapng`** will be that of a button push at the SCADA workstation. Answer the following questions:

1. What systems are on the network? What are the addresses (MAC, IPv4) of the systems? Are there other addresses for these systems?
2. Over what protocols do the systems communicate with each other?

Solution:

A good way to start is to use the endpoint statistic that can be obtained with

```
tshark -q -z endpoints,eth -r normal.pcapng
```

Or from *Wireshark* (Statistics → Endpoints → Ethernet tab). Ignoring the broadcast and multicast addresses a total of seven systems remains. The first column shows the MAC-addresses with the Ethernet vendor part resolved. This can be turned off with the "-n" option for *tshark* or in the Wireshark GUI under View → Name Resolution → Resolve Physical Addresses.

Vendor name resolved	Full MAC address	IP address
Broadcast	ff:ff:ff:ff:ff:ff	255.255.255.255
Dell_9f:7c:74	f4:8e:38:9f:7c:74	10.3.5.3
D-Link_e7:b7:c4	00:26:5a:e7:b7:c4	10.3.5.1
IPv4mcast_7f:ff:64	01:00:5e:7f:ff:64	239.255.255.100
IPv4mcast_7f:ff:fa	01:00:5e:7f:ff:fa	239.255.255.250
IPv4mcast_fc	01:00:5e:00:00:fc	224.0.0.252
LLDP_Multicast	01:80:c2:00:00:0e	
Siemens_ad:91:96	28:63:36:ad:91:96	10.5.3.12
Siemens_ad:91:97	28:63:36:ad:91:97	
Siemens_ae:70:0b	28:63:36:ae:70:0b	
Siemens_f6:8b:bd	00:1b:1b:f6:8b:bd	
Siemens_f7:7c:4f	00:1b:1b:f7:7c:4f	

For completeness, the MAC and IP-addresses for the second PLC and the Engineering workstation are given below. These systems will come up later in the exercise.

Vendor name resolved	Full MAC address	IP-address
Siemens_ae:70:09	28:63:36:ae:70:09	10.3.5.11
Siemens_f7:7c:4f	01:1b:1b:f7:7c:4f	10.3.5.5

The relationship between MAC- and IP-addresses can be obtained from ARP responses exchanged on the network. These responses can be identified by having an opcode of 2. In the *Wireshark GUI*, this can be done by applying a filter for ARP responses, thus **`arp.opcode == 2`**. From the CLI with **`tshark -O arp -Y 'arp.opcode == 2' -n -r normal.pcapng`**. Note that only responses for 10.3.5.3, the SCADA workstation, not 10.3.5.12, the PLC, can be seen.

Seemingly, at the time of the capture the entry for 10.3.5.12 was already in the ARP cache so the system was not asking. However, its IP- and MAC-address can still be seen in the response (see Figure 6):

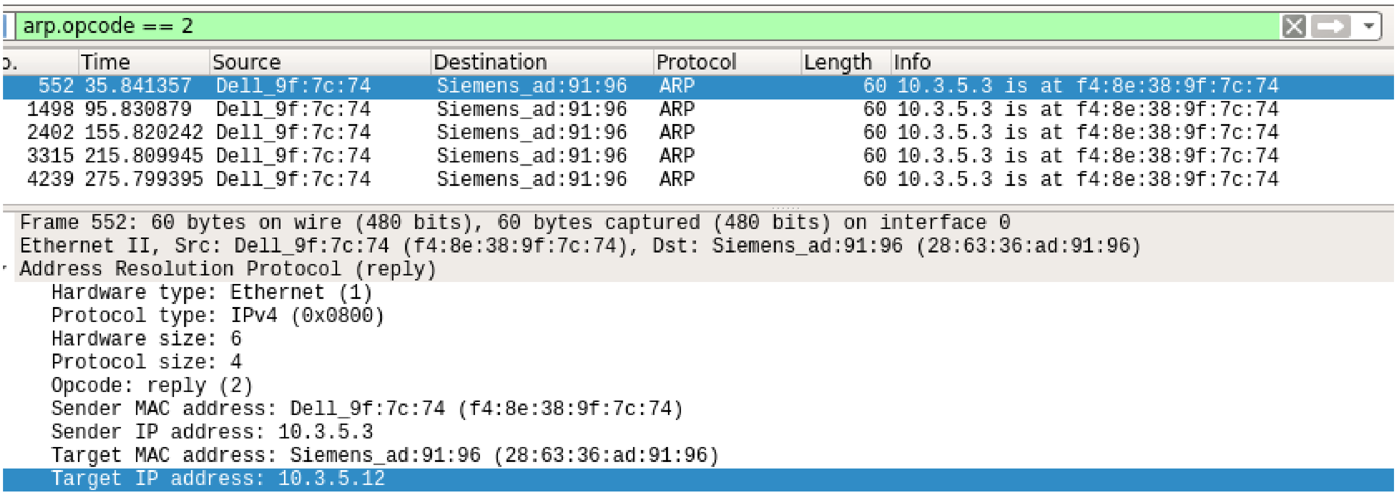


Figure 6. ARP responses in Wireshark

There are no IPv6 or other protocol addresses on the network as can be seen from the empty tab from the endpoints display.

1. To get an overview of the protocols used, *Wireshark* offers the protocol hierarchy display, which can be used with Statistics → Protocol Hierarchy or with `tshark -r normal.pcapng -z io,phs` with the GUI giving more detailed information (Figure 77).

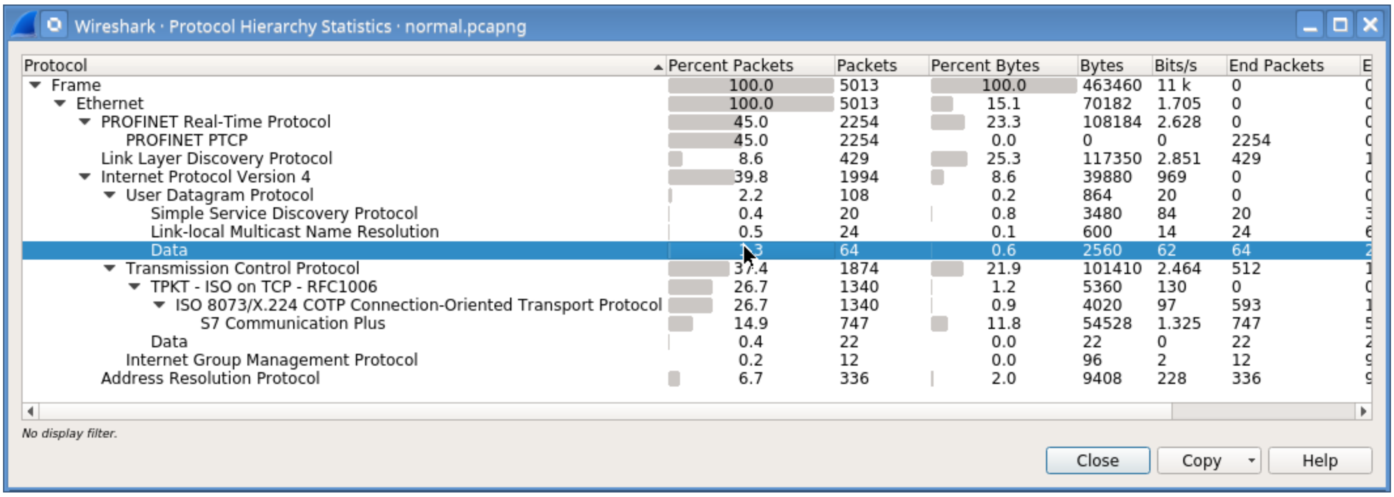


Figure 7. Protocol hierarchy

As the Layer 2 protocols play no larger role in this exercise, the focus will be on IP. There are four different protocols used: Two UDP-based (SSDP and LLMNR), one TCP-based (S7 Communication Plus, shortened to S7plus in this document) and IGMP. SSDP and LLMNR are artefacts from Microsoft Windows, which can be ignored here, as can IGMP.

As can be seen from the hierarchy, S7plus is encapsulated via two more protocols, TPKT and COTP. Being originally from the OSI suite of protocols, S7plus is being transported over TCP through encapsulation of its own transport protocol, COTP (short for Connection Oriented Transport Protocol) which plays the same role as TCP in the OSI world. The encapsulation is done through a small intermediate protocol layer, TPKT[6] (see Figure 8, the third and second rightmost columns[7]).

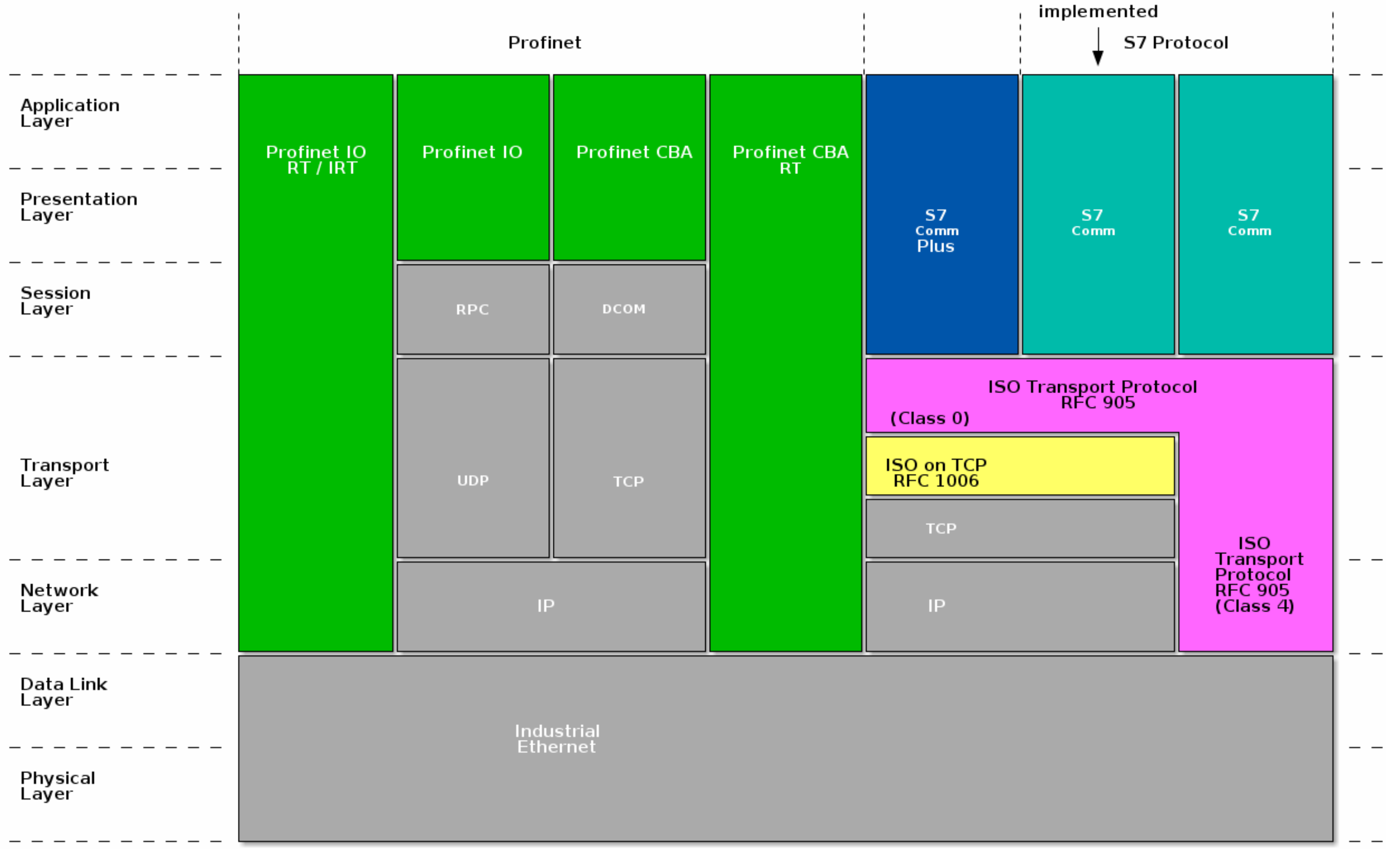


Figure 8. S7 protocol layering on top of TCP/IP

While redundant, it once made porting OSI applications to the TCP/IP world easier. The drawback is that TPKT uses one TCP port (102) for all transported OSI protocols. One cannot see what OSI protocol is transported without looking at the higher protocol layers. The TPKT header is just four bytes long, the first byte being the version (3), one reserved byte (0) and the other two bytes being the length of the encapsulated OSI packet including the TPKT header (see Figure).

[6] TPKT is specified in RFC 1006: https://tools.ietf.org/html/rfc1006
[7] Taken from: https://plc4x.incubator.apache.org/img/protocols-s7-osi.png

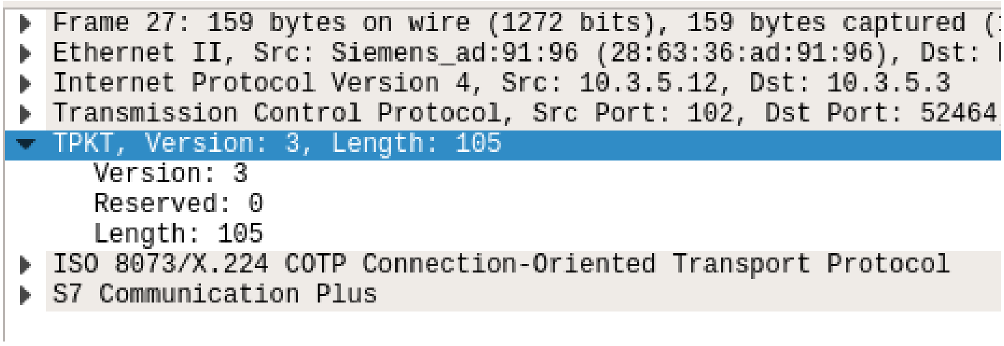


Figure 9. TPKT header in Wireshark

COTP defines five classes of transport protocols. In this exercise, only class 0 is used, which is also referred to as "TP0" (with class 1 being TP1, etc.) and each higher class defining more functions. TP0 has only a minimal set of functions (its use was planned for connection-oriented layer 3 protocols like X.25, where most functions were already supplied by the lower level protocol) and with TP4 being roughly equivalent to TCP[8] in functionality. Since TCP is already used and supplying most of the needed functionality, only TP0 needs to be used. COTP connections are initiated by the initiator sending a TPDU with a type of 0x0e (Connect Request), the other party responding with a Connect Confirm (type 0x0d) packet. Data is exchanged with TPDUs of type 0x0f (Data) and an ordered connection release is done by sending a TPDU of type 0x08 (Disconnect), there is no disconnect response in COTP.

TPDU	Type code
Connection request	0x0e
Connection response	0x0d
Data	0x0f
Disconnect	0x08

The S7comm and S7comm-plus protocols are layered on top of COTP. However, unlike TCP and IP, one cannot see directly from the COTP header what protocol is transported in it. Instead, one has to look at the S7comm or S7comm-plus header, where the first byte tells which type of protocol is used. Figure and Figure show a sample of each protocol version. They will be needed later in the exercise.

S7 protocol	Version code
S7comm	0x32
S7comm-plus	0x72

[8] For a comparison of COTP class functionality, see https://en.wikipedia.org/wiki/OSI_model#Layer_4:_Transport_Layer

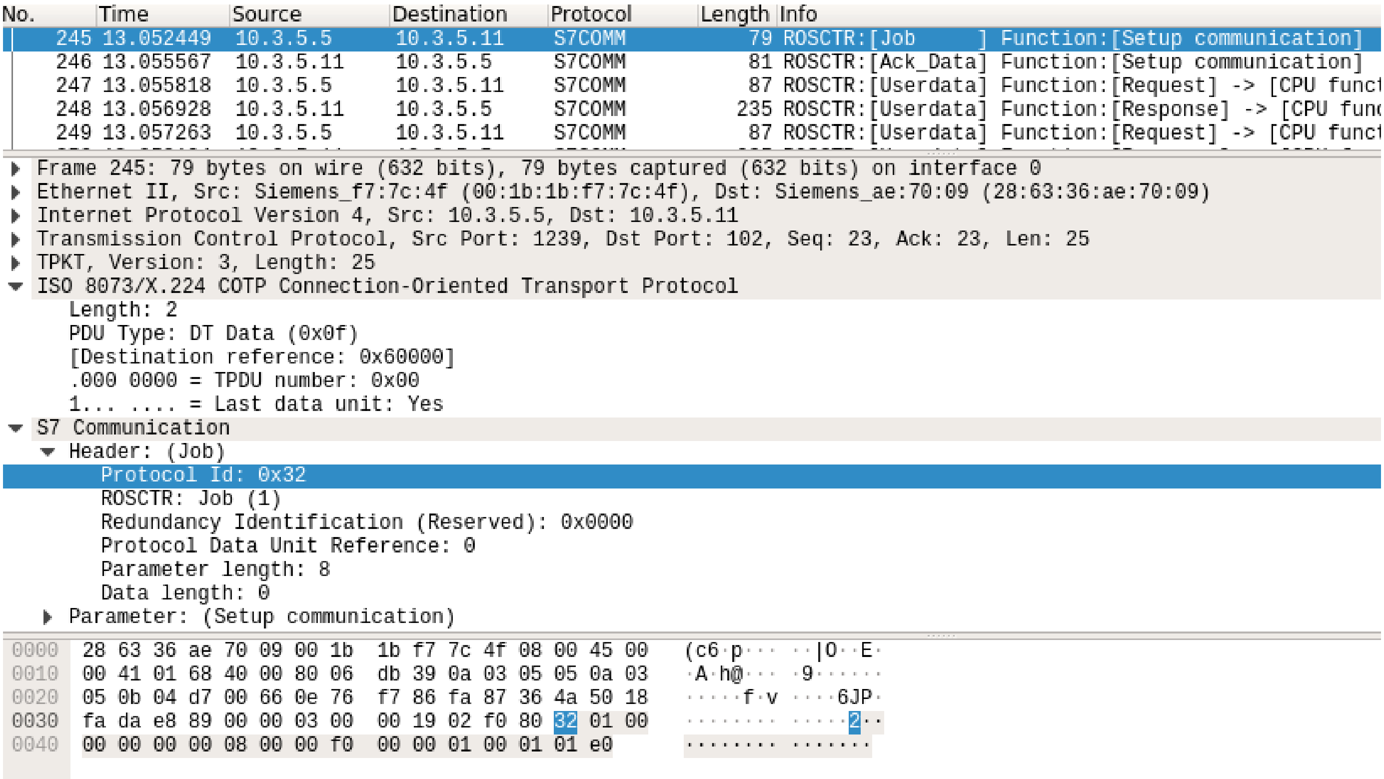


Figure 10. S7comm PDU (type 0x32)

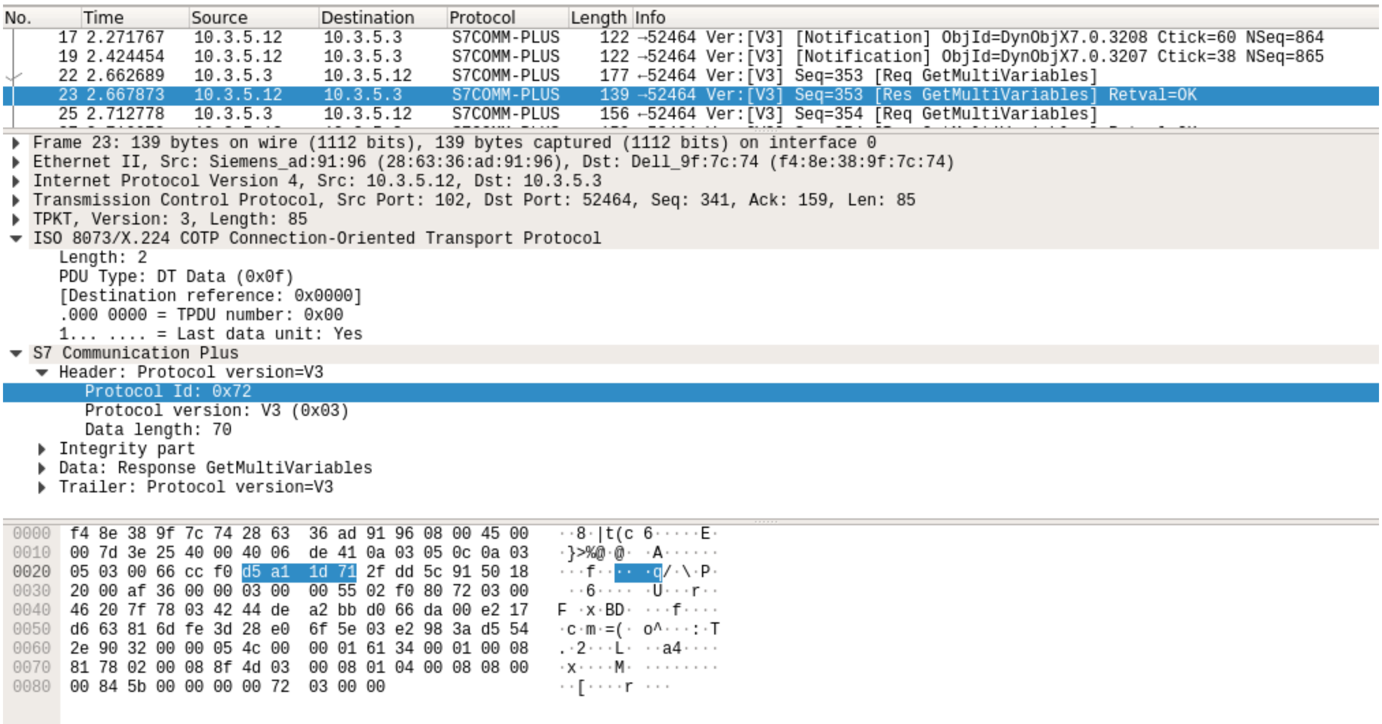


Figure 11. S7comm-plus PDU (type 0x72)

Let us go a little deeper and try to answer the question: “what sort of S7plus packets are being sent here?” The type of S7plus packets can be inferred from the opcode (in *Wireshark* filter terminology: **`s7comm-plus.data.opcode`**). The table below gives an overview of the opcodes used in this exercise:

s7comm-plus.data.opcode	
Hex	Mnemonic
0x31	Request
0x32	Response
0x33	Notification

First, there are notifications; these are going from the PLC (10.3.5.12 to the workstation (10.3.5.12). Notification are used to inform the SCADA application about the value of a set of variables. A sample packet is shown below. These packets form the bulk of the S7plus traffic in the captures. The SCADA application "subscribes" to a set of variables it wants to be notified of and each “subscription” is identified by its "Subscription Object Id" (or **`s7comm-plus.notification.subscrobjectid`**). A sample notification frame is shown below

```
Frame 494: 122 bytes on wire (976 bits), 122 bytes captured (976 bits) on interface 0
Ethernet II, Src: 28:63:36:ad:91:96, Dst: f4:8e:38:9f:7c:74
Internet Protocol Version 4, Src: 10.3.5.12, Dst: 10.3.5.3
Transmission Control Protocol, Src Port: 102, Dst Port: 52464, Seq: 42600, Ack: 21167,
Len: 68
TPKT, Version: 3, Length: 68
ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
S7 Communication Plus
    Header: Protocol version=V3
        Protocol Id: 0x72
        Protocol version: V3 (0x03)
        Data length: 53
    Integrity part
        Digest Length: 32
        Packet Digest: dbf6f38588aded43bfdc37245f084b8561ef494046aa7cb1...
    Data: Notification
        Opcode: Notification (0x33)
        Notification Data Set
            Subscription Object Id: 0x70000c87
            Unknown 2: 0x0400
            Unknown 3: 0x0000
            Unknown 4: 0x0000
           Notification Credit tickcount: 201
            Notification sequence number (VLQ): 1354
            Unknown5: 0x01
           ValueList
                Terminating Item/List
            Unknown additional 3 bytes, because 1st Object ID > 0x70000000: 0x000000
        Data unknown: 00
    Trailer: Protocol version=V3
        Protocol Id: 0x72
        Protocol version: V3 (0x03)
        Data length: 0
```

While one can play around with *Wireshark* display filters like this:

`s7comm-plus.data.opcode == 0x33 and (s7comm-plus.notification.subscrobjectid == 0x70000c87 or s7comm-plus.notification.subscrobjectid == 0x7000c88)` to get an overview of all the values in a capture file, it is easier to use *tshark* and UNIX sorting.

To find all opcode values in a capture file (*uniq -c* output: first column being the number of occurrences, second column being the content of the line):

```
$ tshark -n -r normal.pcapng -Ys7comm-plus -Tfields -e s7comm-plus.data.opcode
| sort -n | uniq -c
    153 0x00000031
     99 0x00000032
    495 0x00000033
```

To find all Subscription Object Ids in a capture file

```
$ tshark -n -r normal.pcapng -Y 's7comm-plus.data.opcode == 0x33' -Tfields -e
s7comm-plus.notification.subscrobjectid | sort -n | uniq -c
    165 0x70000c87
    330 0x70000c88
```

The Subscription Id will change with each S7plus session, so it will not be the same in other captures, although the variables subscribed to may be the same. Unfortunately, it cannot be inferred from the capture, what variables exactly are meant.

Let us move on to the other opcodes; Requests (0x31) and Responses (0x32). "What" exactly is requested is identified by the *function* subfield:

```
$ tshark -n -r normal.pcapng -Y 's7comm-plus.data.opcode == 0x31' -Tfields -e s7comm-
plus.data.function | sort -n | uniq -c
     54 0x000004f2
     99 0x0000054c

$ tshark -n -r button_push.pcapng -Y 's7comm-plus.data.opcode == 0x31' -Tfields -e
s7comm-plus.data.function | sort -n | uniq -c
     51 0x000004f2
      9 0x00000542
     94 0x0000054c
```

Therefore, there are three functions: *SetVariable* (0x04f2) and *GetMultiVariables* (0x054c) that are used in the normal operation, and *SetMultiVariables* (0x0542) that is used when a button is pushed in the SCADA application.

Responses are answers to Requests (obviously) and almost identical in structure, Responses have the same function type as the request they are answering, in the capture files, only two different functions can be seen:

```
$ tshark -n -r normal.pcapng -Y 's7comm-plus.data.opcode == 0x32' -Tfields -e s7comm-
plus.data.function | sort -n | uniq -c
     99 0x0000054c
```

```
$ tshark -n -r button_push.pcapng -Y 's7comm-plus.data.opcode == 0x32' -Tfields -e
s7comm-plus.data.function | sort -n | uniq -c
      9 0x00000542
     94 0x0000054c
```

Note that the number of Responses is equal to that of the corresponding Requests. It seems that "SetVariables" request do not trigger a response. The following table gives an overview about functions used in this exercise:

Hex	Function
0x04bb	Explore
0x04ca	CreateObject
0x04d4	DeleteObject
0x04f2	SetVariable
0x0524	GetLink
0x0542	SetMultiVariables
0x054c	GetMultiVariables
0x0556	BeginSequence
0x0560	EndSequence
0x056b	Invoke

3.3 Task 2: Initial attack detection

During this first stage of the attack, the intruder first gets onto the SCADA network. You will have two tasks: first to analyse the network behaviour during the initial attack stage and then to review and perhaps adapt their monitoring policies, depending on whether you noticed the attack or not.

3.3.1 Initial break-in

An employee opens an office document with embedded macros on an engineering workstation. After the successful infection, the workstation tries to connect to a C&C server via TCP (network activity). Since the network is separated, no connection is established but the malware activates auto exploitation mode

3.3.2 Subtask: Analyse the attack on the engineering workstation

Students: Given the packet capture file **`attack1.pcapng`**, analyse the traffic. Answer the following questions

- Do you see an attack?
- If yes, what do you see?
- What made you suspicious?

Solution:

1. No real attack is in the network capture, only unsuccessful communication attempts that may be noticed:

- The unsuccessful attempts to download pictures from the internet (TCP traffic to 23.95.230.107, port 80, i.e. HTTP).
- The unsuccessful attempts to contact the command and control server over an unknown protocol (UDP traffic to 234.5.6.7 port 8910).

Both communications can be seen by noting the IP-addresses, which are not part of the net 10.3.5.0/24 or the protocols, which are deviating from the traffic patterns in **normal.pcapng** or **button_push.pcapng**. The trick is how to strip away the bulk of the "known good" traffic, i.e. LLDP, PROFINET and S7. With a structured analysis, one would start with an overview of protocols used, like in the previous task. Starting with a simple overview of the communication endpoints:

```
$ tshark -n -r attack1.pcapng -q -z endpoints,ip
================================================================================
IPv4 Endpoints
Filter:<No Filter>
                       |  Packets  | |  Bytes  | | Tx Packets | | Tx Bytes | | Rx Packets | | Rx
Bytes |
10.3.5.3                   268        49559       124           36109         144
13450
10.3.5.12                  240        21263       144           13450          96
7813
234.5.6.7                   21        27722         0               0          21
27722
10.3.5.5                     9          658         9             658           0
0
10.3.5.255                   5          410         0               0           5
410
23.95.230.107                5          330         0               0           5
330
10.3.5.1                     5          300         5             300           0
0
239.255.255.100              5          300         0               0           5
300
255.255.255.255              4          328         0               0           4
328
10.255.255.255               2          164         0               0           2
164
```

Both IP-addresses not from the network 10.3.5.0/24 clearly stand out. But who is talking to them? This can be answered again with the conversations statistic, but this time the output will be limited to the suspicious IP-addresses, which can be done with a filter added to the *conv* selector (the filter for 234.5.6.7 will yield an empty list for TCP)

```
$ tshark -n -r attack1.pcapng -q -z conv,tcp,ip.addr==23.95.230.107
================================================================================
TCP Conversations
Filter:ip.addr==23.95.230.10
                                     |       <-      | |       ->      | |     Total     |Relative
|Duration |
                                     | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |   Start     |
|
10.3.5.5:1232  <-> 23.95.230.107:80    0         0        1        66        1        66   13,031208000
0,0000
10.3.5.5:1233  <-> 23.95.230.107:80    0         0        1        66        1        66   18,337056000
0,0000
10.3.5.5:1234  <-> 23.95.230.107:80    0         0        1        66        1        66   23,656130000
0,0000
10.3.5.5:1235  <-> 23.95.230.107:80    0         0        1        66        1        66   28,975215000
0,0000
10.3.5.5:1236  <-> 23.95.230.107:80    0         0        1        66        1        66   34,294342000
0,0000
```

As can be seen from the port (80), the protocol used is HTTP. Moreover, with just one frame being sent, this must be the initial SYN packet of the TCP connection. As the network has no connection to the outside, no answer will be received (Figure).

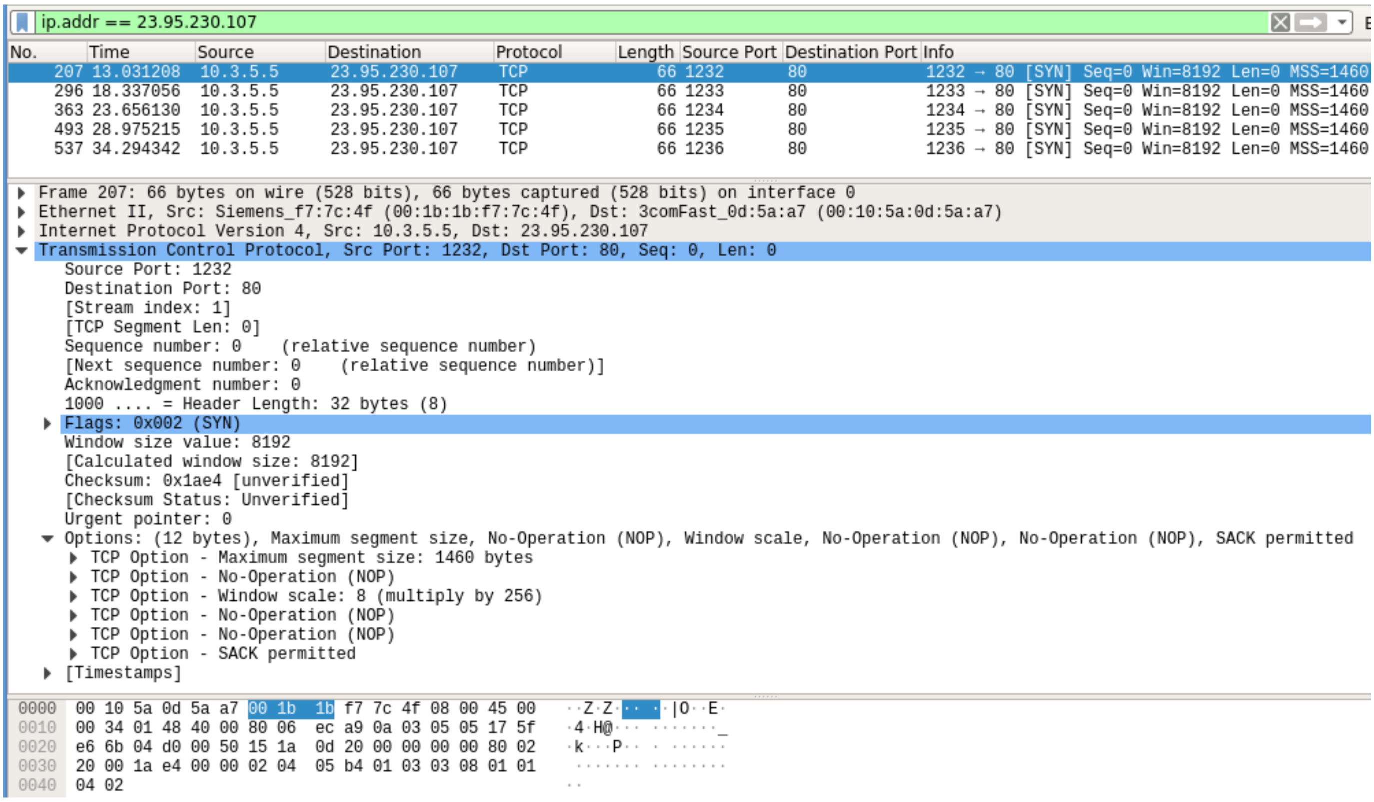


Figure 12. Malware HTTP connection attempts

The same can be done for UDP communications (empty when filtering for 23.95.230.107):

```
$ tshark -n -r attack1.pcapng -q  -z conv,udp,ip.addr==234.5.6.7
================================================================================
UDP Conversations
Filter:ip.addr==234.5.6.7
                                            |       <-        | |       ->        | |     Total      |  Relative     |
Duration |
                                            | Frames  Bytes | | Frames  Bytes | | Frames  Bytes|    Start       |
|
10.3.5.3:60070  <-> 234.5.6.7:8910   0          0        6       6030        6       6030       7,825592000
20,1028
```

What about the S7plus traffic? Let us have a look at the opcodes

```
$ tshark -n -r attack1.pcapng -q  -z conv,tcp,tcp.port==102
================================================================================
TCP Conversations
Filter:tcp.port==102
                                  |       <-        | |       ->        | |     Total       |  Relative    |
Duration   |
                                  | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |     Start     |
|
10.3.5.3:54043 <-> 10.3.5.12:102    140     13210      92      7573     232     20783     0,000000000
41,4049
10.3.5.3:54045 <-> 10.3.5.12:102      2       120       2       120       4       240    18,796773000
0,2014
10.3.5.3:54044 <-> 10.3.5.12:102      2       120       2       120       4       240    24,395835000
0,2003
```

Nothing out of the order so far, looking at the opcodes:

```
$ tshark -n -r attack1.pcapng -Y s7comm-plus -T fields -e s7comm-plus.data.opcode |
sort -n | uniq -c
     19 0x00000031
     12 0x00000032
     62 0x00000033
```

Everything seems to be normal for now.

2. With a monitoring policy that looks for anything that deviates from the laid down rules, the unsuccessful communication attempts are suspicious per definition.
3. At the very least, any communication attempt to IP-addresses other than the workstations and PLCs should raise suspicion, as well as use of any other communication protocol than TCP and port 102.

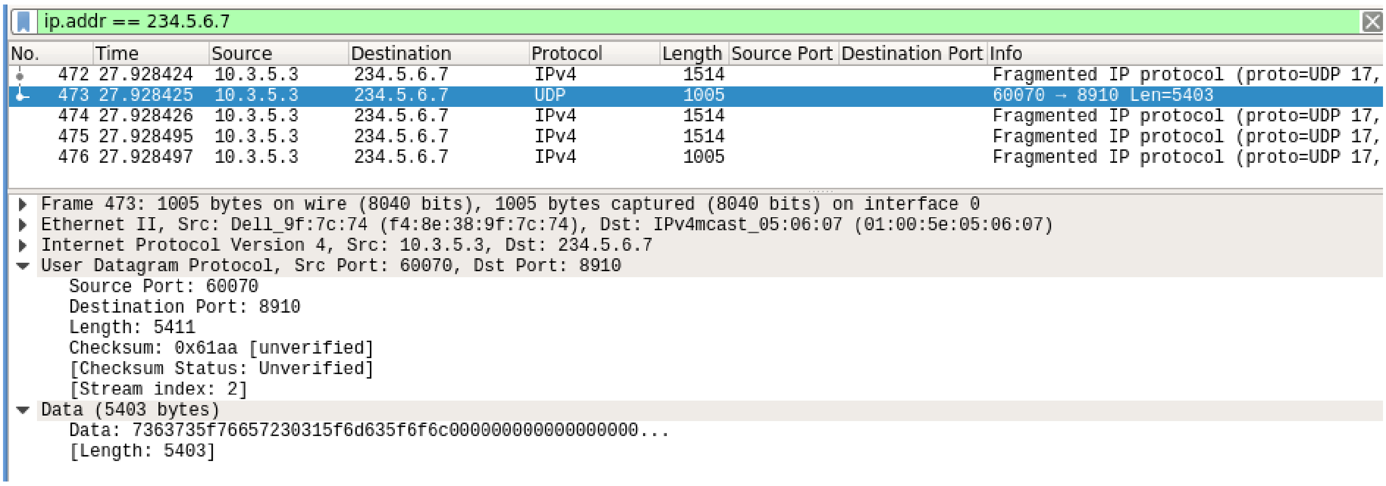


Figure 13. Malware UDP communication

3.3.3 Subtask: Review your monitoring policy

Students: Try to answer the following questions:

- Would your monitoring policy notice the intruder activity?
- All of it? Which one would it miss?

Solution: The answer to this question depends on the policy the students developed in section 3.1.4.

When the sample policies key points are used (repeated below),

- Only the workstations shall communicate with the PLCs
- Communication shall be limited to port 102/tcp and the S7plus protocols
- The question of with whom (except the PLCs) the workstations should communicate can be left open. If they should communicate, communication should be limited to port 5900/tcp (VNC) and only from the Engineering workstation to the SCDA workstation.

The HTTP and UDP connections are clearly detected by destination IP-addresses, which are neither that of the PLCs nor of one of the workstations. They can also easily be detected by protocol, HTTP using port 80/tcp and UDP port 8190, that are not whitelisted in the policy.

For this part, the policy would detect all of the adversaries' activities.

When using the sample policy given in the solutions section of 3.1.4, the port scan and the VNC password brute force would be noticed, because they involve a connection from the engineering to the SCADA workstation (10.3.5.5 to 10.3.5.12, port 5900/tcp) that is not whitelisted in the policy. In addition, the S7scan is discovered as plain S7comm uses a different protocol version (0x32) than S7plus (0x72).

3.4 Task 4: Second attack stage analysis

Typical attacks nowadays do not get direct access to critical systems. Usually, attackers compromise a less secured system and then move on to other systems, exploiting internal trust relationships.

3.4.1 Lateral movement

Since the malware cannot connect to its C&C[9] server, it activates a fall-back mode for offline operation. In this mode, the malware scans the local network and tries to attack whatever targets it finds. The malware discovers a SCADA workstation and two Siemens PLCs in the same subnet as the engineering workstation. As part of the scanning, an open VNC port on the SCADA workstation is discovered.

The VNC username and passwords are brute-forced; the malware successfully logs in to the SCADA workstation (through VNC) (network activity) and stops an industrial process through the SCADA panel (emergency shutdown).

3.4.2 Subtask: Analyse the lateral movement

Students: Given the packet captures **`attack2.pcapng`**, **`attack3.pcapng`**, and **`attack4.pcapng`** analyse the attack(s).

- Describe and classify the activities? Who is doing what to whom?
- Assess the damage done by the end of the attack, i.e. all three packet captures.

Solution: Three activities are to be noticed:

1. In **`attack1.pcapng`**, the engineering workstation is scanning/probing the network. This is typical scan like the one outlined in section 3.2.2.
2. In **`attack2.pcapng`**, the engineering workstation is specifically scanning for S7 enabled systems (i.e. PLCs)

---

9 Command and Control

3. **`attack3.pcapng`** contains the VNC attack on the SCADA workstation which consists of a brute-force attempt on the password.

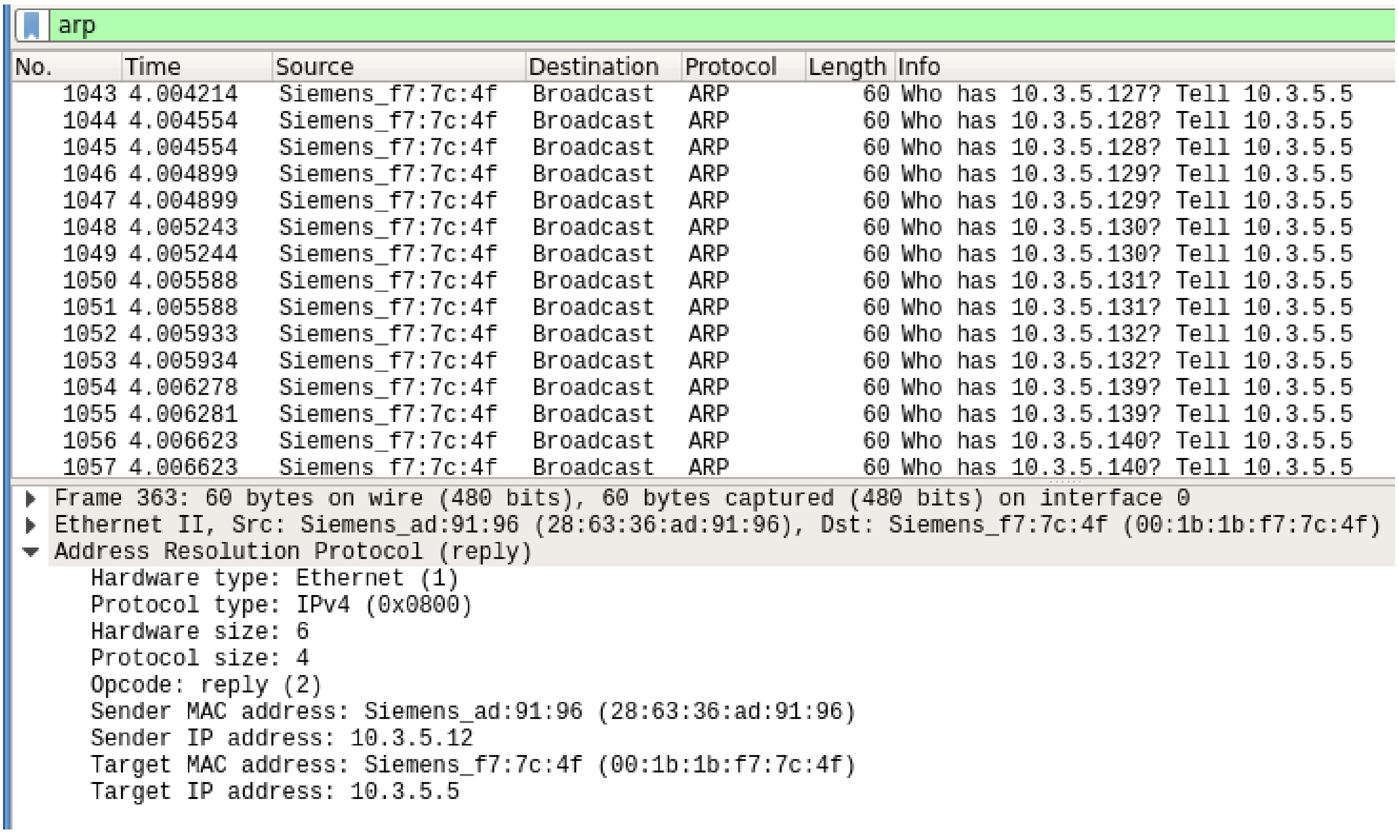
The port scan is easily seen in the conversations overview:

```
$ tshark -n -r attack2.pcapng -q -z conv,ip
================================================================================
IPv4 Conversations
Filter:<No Filter>
                              |       <-      | |       ->      | |     Total     |    Relative    |
Duration    |
                              | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |
|
10.3.5.3 <-> 10.3.5.5           195     11700      15       900     210     12600     4,345429000
10,4537
10.3.5.5 <-> 10.3.5.11          100      6000     100      6000     200     12000     4,345564000
0,1676
10.3.5.5 <-> 10.3.5.12          100      6000     100      6000     200     12000     4,345761000
0,1684
10.3.5.3 <-> 10.3.5.12           79      7416      55      4501     134     11917     0,000000000
21,4088
10.3.5.1 <-> 10.3.5.5             0         0      12       744      12       744     4,697235000
16,0501
10.3.5.1 <-> 239.255.255.100      0         0       5       300       5       300    19,782942000
0,0015
10.3.5.3 <-> 255.255.255.255      0         0       3       246       3       246    20,666588000
0,0000
```

```
$ tshark -n -r attack2.pcapng -q -z conv,tcp
================================================================================
TCP Conversations
Filter:<No Filter>
                                   |       <-      | |       ->      | |     Total     |    Relative |
Duration    |
                                   | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start  |
|
10.3.5.3:54043 <-> 10.3.5.12:102   75      7176      51      4261     126     11437     0,000000000
21,4088
10.3.5.5:59416 <-> 10.3.5.1:80      6       372       0         0       6       372     4,697235000
14,0504
10.3.5.5:59423 <-> 10.3.5.1:80      6       372       0         0       6       372     6,349966000
14,3973
10.3.5.5:59416 <-> 10.3.5.3:135     3       180       1        60       4       240     4,363587000
9,0002
10.3.5.5:59416 <-> 10.3.5.3:3389    3       180       1        60       4       240     4,377779000
8,9980
10.3.5.5:59416 <-> 10.3.5.3:445     3       180       1        60       4       240     5,562540000
8,9987
10.3.5.5:59416 <-> 10.3.5.3:5900    3       180       1        60       4       240     5,564956000
9,0004
10.3.5.5:59416 <-> 10.3.5.3:5800    3       180       1        60       4       240     5,798956000
9,0002
10.3.5.3:54045 <-> 10.3.5.12:102    2       120       2       120       4       240    10,998112000
0,0075
10.3.5.3:54044 <-> 10.3.5.12:102    2       120       2       120       4       240    15,597341000
0,2017
10.3.5.5:59416 <-> 10.3.5.11:443    1        60       1        60       2       120     4,345564000
0,0001
10.3.5.5:59416 <-> 10.3.5.12:443    1        60       1        60       2       120     4,345761000
0,0002
10.3.5.5:59416 <-> 10.3.5.11:8080   1        60       1        60       2       120     4,346828000
0,0001
10.3.5.5:59416 <-> 10.3.5.12:8080   1        60       1        60       2       120     4,347199000
0,0002
10.3.5.5:59416 <-> 10.3.5.11:110    1        60       1        60       2       120     4,348204000
0,0001
10.3.5.5:59416 <-> 10.3.5.11:554    1        60       1        60       2       120     4,361129000
0,0002
```

```
10.3.5.5:59416 <-> 10.3.5.12:110    1       60      1       60      2       120     4,361533000
0,0001
10.3.5.5:59416 <-> 10.3.5.11:8888   1       60      1       60      2       120     4,362535000
0,0001
...
```

Lots of ports that were not in use before. But wait, if this is a port scan, what about other IP-addresses in the network? Why are only 4 IP-addresses in the network? This can be answered by a look at the ARP requests in *wireshark* (see Figure below).



```
arp
No.   Time      Source             Destination  Protocol  Length Info
 1043 4.004214  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.127? Tell 10.3.5.5
 1044 4.004554  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.128? Tell 10.3.5.5
 1045 4.004554  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.128? Tell 10.3.5.5
 1046 4.004899  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.129? Tell 10.3.5.5
 1047 4.004899  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.129? Tell 10.3.5.5
 1048 4.005243  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.130? Tell 10.3.5.5
 1049 4.005244  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.130? Tell 10.3.5.5
 1050 4.005588  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.131? Tell 10.3.5.5
 1051 4.005588  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.131? Tell 10.3.5.5
 1052 4.005933  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.132? Tell 10.3.5.5
 1053 4.005934  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.132? Tell 10.3.5.5
 1054 4.006278  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.139? Tell 10.3.5.5
 1055 4.006281  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.139? Tell 10.3.5.5
 1056 4.006623  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.140? Tell 10.3.5.5
 1057 4.006623  Siemens_f7:7c:4f   Broadcast    ARP          60 Who has 10.3.5.140? Tell 10.3.5.5
Frame 363: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: Siemens_ad:91:96 (28:63:36:ad:91:96), Dst: Siemens_f7:7c:4f (00:1b:1b:f7:7c:4f)
Address Resolution Protocol (reply)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: reply (2)
    Sender MAC address: Siemens_ad:91:96 (28:63:36:ad:91:96)
    Sender IP address: 10.3.5.12
    Target MAC address: Siemens_f7:7c:4f (00:1b:1b:f7:7c:4f)
    Target IP address: 10.3.5.5
```

Figure 14. ARP responses

From the list of unanswered ARP queries, it can be seen that the adversary really tries to probe the whole network.

In `attack2.pcapng`, the PLC scan can be seen in when selecting the S7comm protocol (not s7comm-plus)

```
$ tshark -n -r attack3.pcapng -Y 's7comm'
  245  13.052449    10.3.5.5 → 10.3.5.11    S7COMM 79 1239 102 ROSCTR:[Job      ] Function:[Setup
communication]
  246  13.055567    10.3.5.11 → 10.3.5.5    S7COMM 81 102 1239 ROSCTR:[Ack_Data] Function:[Setup
communication]
  247  13.055818    10.3.5.5 → 10.3.5.11    S7COMM 87 1239 102 ROSCTR:[Userdata]
Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0001
  248  13.056928    10.3.5.11 → 10.3.5.5    S7COMM 235 102 1239 ROSCTR:[Userdata]
Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
  249  13.057263    10.3.5.5 → 10.3.5.11    S7COMM 87 1239 102 ROSCTR:[Userdata]
Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0001
  250  13.058194    10.3.5.11 → 10.3.5.5    S7COMM 235 102 1239 ROSCTR:[Userdata]
Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
```

```
  251  13.058573     10.3.5.5 → 10.3.5.11    S7COMM 87 1239 102 ROSCTR:[Userdata]
Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0001
  252  13.059235    10.3.5.11 → 10.3.5.5     S7COMM 435 102 1239 ROSCTR:[Userdata]
Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
  326  19.870650     10.3.5.5 → 10.3.5.12    S7COMM 79 1242 102 ROSCTR:[Job     ] Function:[Setup
communication]
  327  19.874238    10.3.5.12 → 10.3.5.5     S7COMM 81 102 1242 ROSCTR:[Ack_Data] Function:[Setup
communication]
  328  19.874479     10.3.5.5 → 10.3.5.12    S7COMM 87 1242 102 ROSCTR:[Userdata]
Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0001
  329  19.875630    10.3.5.12 → 10.3.5.5     S7COMM 235 102 1242 ROSCTR:[Userdata]
Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
  330  19.876035     10.3.5.5 → 10.3.5.12    S7COMM 87 1242 102 ROSCTR:[Userdata]
Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0001
  331  19.877079    10.3.5.12 → 10.3.5.5     S7COMM 235 102 1242 ROSCTR:[Userdata]
Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x0011 Index=0x0000
  332  19.877488     10.3.5.5 → 10.3.5.12    S7COMM 87 1242 102 ROSCTR:[Userdata]
Function:[Request] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0001
  333  19.878402    10.3.5.12 → 10.3.5.5     S7COMM 435 102 1242 ROSCTR:[Userdata]
Function:[Response] -> [CPU functions] -> [Read SZL] ID=0x001c Index=0x0000
```

The scan conveys information similar to that what can be obtained with the *nmap* "s7-info.nse" script shown below:

```
nmap -Pn -sT -p102 --script s7-info.nse 10.3.5.12
Starting Nmap 7.70 ( https://nmap.org ) at 2018-06-08 13:53 Europa Zachodnia (cz
as letni)
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled.
 Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 10.3.5.12
Host is up (0.00s latency).

PORT    STATE SERVICE
102/tcp open  iso-tsap
| s7-info:
|   Module: 6ES7 511-1AK01-0AB0
|   Basic Hardware: 6ES7 511-1AK01-0AB0
|   Version: 2.0.1
|   System Name: S71500/ET200MP station_1
|   Module Type: PLC_1
|   Serial Number: S C-HDN715522016
|   Plant Identification:
|_  Copyright: Original Siemens Equipment
Service Info: Device: specialized
Nmap done: 1 IP address (1 host up) scanned in 1.11 seconds
```

Here is a breakdown of one of the answer packets (Figure):

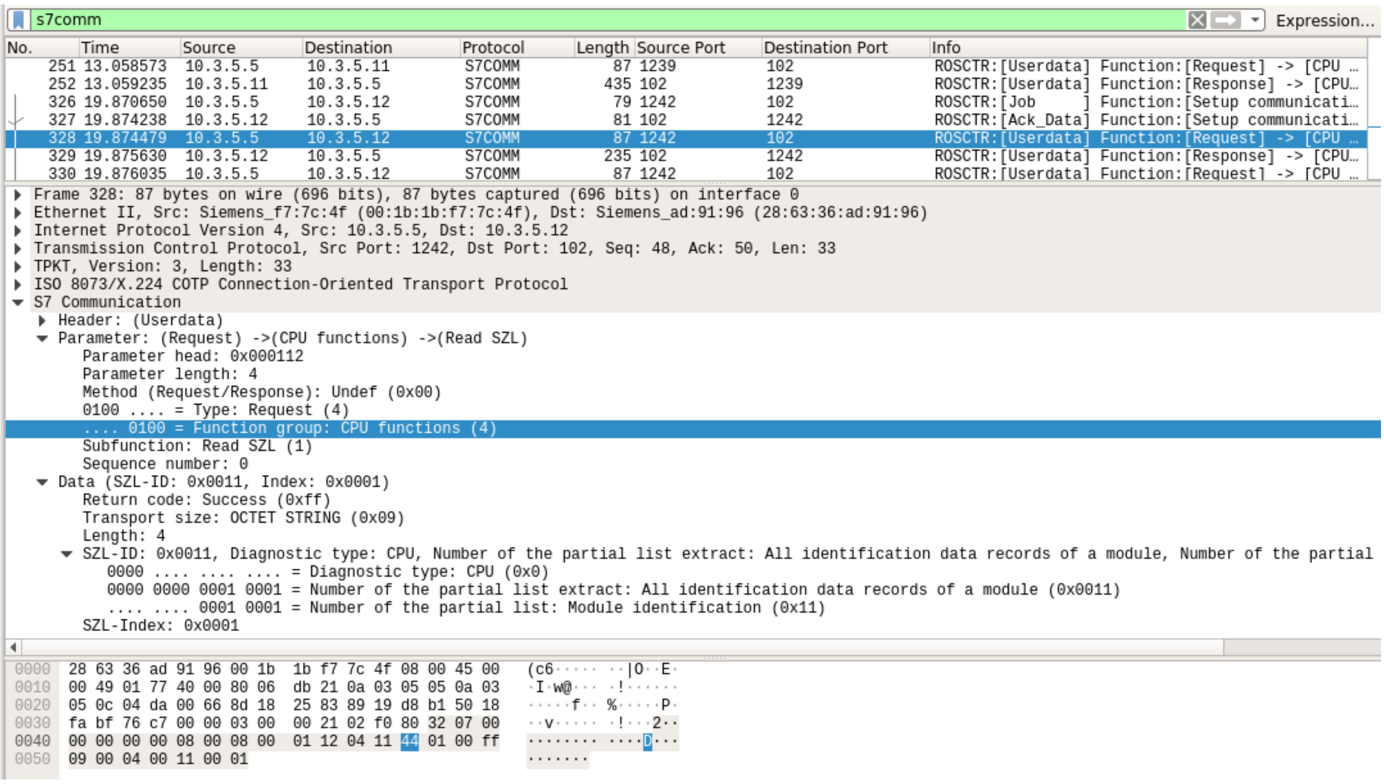


Figure 15. S7comm packets

Last, **attack4.pcapng** is the VNC brute-force attempt on the passwords.

Six sessions can be seen in the packet capture, each starting from successively increasing source ports: 1396, 1397, 1398, ...

VNC authentication (of the type used here, VNC) is a challenge response process[10]

- The server sends an authentication challenge, a random 16-byte string.
- The client sends an authentication response, containing also a 16-byte string, consisting of the DES encrypted challenge with the password being the encryption key.
- The server responds with an authentication result packet. The first four bytes encode an integer, a value of 1 means that the authentication was unsuccessful; a value of 0 meaning the authentication was successful.
- In case of an unsuccessful authentication, the server will append a string describing the reason for the failure, and then close the connection.

However, as can be seen from the packet capture, in the first five sessions, the server responds with three authentication result packets, the first two of them containing a code of 1 (failure) which is what one would expect. The third however, has a code of 0, but also the string "Authentication failed" attached.

[10] https://tools.ietf.org/html/rfc6143

The sixth session is different, there is only one authentication result packet and this time, it has a value of 0 and no additional string attached (see Figure). Also, the client is closing the connection, which can be seen from the TCP packet coming next.

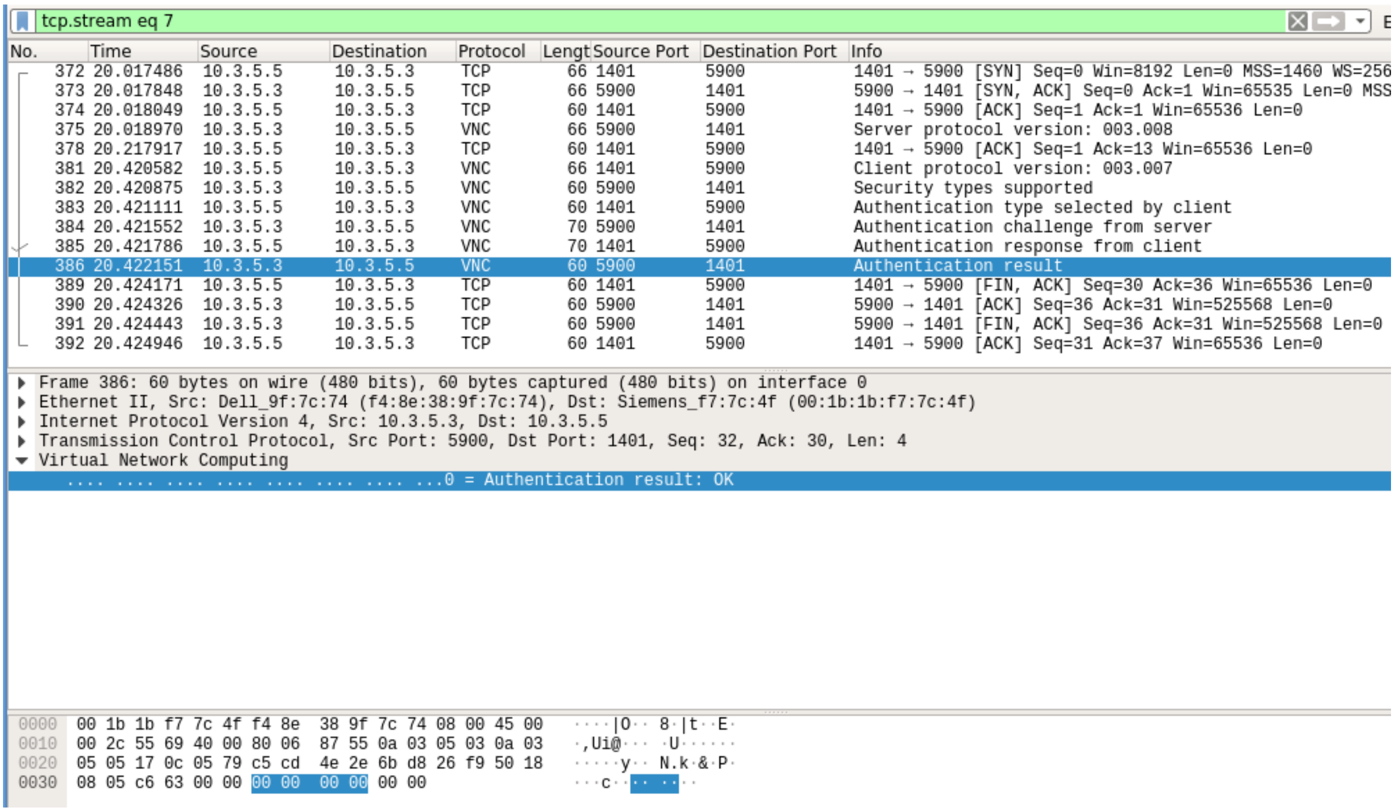


Figure 16. Successful VNC authentication

It can be safely assumed that the adversary did successfully guess the password in the last session.

The port scan and the PLC scan did no damage, except that the adversary gained information about the network and the systems on it. The VNC brute-force attack did give the adversary a login to the SCADA workstation. However, as no other activity can be seen in the capture, it is unclear whether this is already used to compromise or misuse the SCADA workstation.

3.4.3 Subtask: Review and revise the policy

Students:

- Review your policy, would it catch the adversaries' activity?
- Revise your policy to catch the adversaries' activity.

Solution:

Once again, a recap of the initial policy:

- Only the workstations shall communicate with the PLCs
- Communication shall be limited to port 102/tcp and the S7plus protocols
- The question of with whom (except the PLCs) the workstations should communicate can be left open. If they should communicate, communication should be limited to port 5900/tcp (VNC) and only from the Engineering workstation to the SCDA workstation.

When using just the first two points, the port scan and the VNC password brute force would be noticed, because they involve a connection from the engineering to the SCADA workstation (10.3.5.5 to 10.3.5.12, port 5900/tcp) that is not whitelisted in the policy. Also, the S7 scan is discovered as plain S7comm uses a different protocol version (0x32) than S7plus (0x72).

However, the VNC brute-force attack would not be noticed, if this protocol is included in the whitelist.

One more fine point: The policy specifies S7plus protocols (note the plural). If this is taken as both S7comm and S7comm-plus, the S7 scan would not be noticed. Taken more narrowly as S7comm-plus (i.e. only protocol type 0x72) than the scan will be noticed.

For the revision, the VNC brute-force will have to be discovered and the S7 scan. As has been said, the latter is easily recognised by the protocol type, so the policy should clearly state that only type 0x72 (i.e. s7comm-plus) connections are meant.

The brute-force attempts will need a closer look into the protocol. When looking closer into the packet capture, the following *wireshark* filter rules can be worked out with respect to login packets (the length field in the *wireshark* windows is that of the frame, the IP packet length can be seen when looking into the IP header):

a) A vnc.auth_result code of 0 and a packet length of 44 (i.e. without the trailing Authentication failure) denotes a successful login: `vnc.auth_result == 0 and ip.len == 44`
b) A vnc.auth_result code of 1 or vnc_auth_result of 0 and packet length > 44 denotes a login failure: `vnc.auth_result == 1 or (vnc.auth_result == 0 and ip.len > 44)`

The revised policy would look something like this:

- Only the workstations shall communicate with the PLCs
- Communication shall be limited to port 102/tcp and the S7plus (type 0x72) protocol.
- Only the engineering workstation shall communicate with the SCADA workstation over VNC (port 5900).
- Multiple (more than 3 in one minute) login failures will be monitored and investigated.

3.5 Task 5: Analysing the attack on the PLCs

The attack has reached its final goal, harming/disabling the industrial process. Since availability is of foremost importance in SCADA systems, avoiding the attack by shutting down the affected systems is not an option. This puts new challenges to network administrators and investigators.

Later, the pump is again disabled, but this time it could not be changed back to the original by an operator (using the SCADA workstation).

3.5.1 The pump disabling attack

The first thing the plant operators notice is that the pump is being disabled. Fortunately, it was possible to re-enable it. Since the operator convincingly states that it wasn't his action, the investigators now have to find out how this happened.

3.5.2 Subtask: Analyse the attack

Students: Given the packet captures `attack5.pcapng`, analyse the pump disabling attack. Try to answer the following questions

- How was the attack carried out?

- How could the attack have been spotted?

Solution: The overview of the conversations shows four TCP connections, one VNC and three S7plus.

```
$ tshark -q -n -r attack5.pcapng -z conv,tcp
================================================================================
TCP Conversations
Filter:<No Filter>
                                         |       <-        | |       ->        | |     Total       |    Relative    |
Duration |
                                         | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |
|
10.3.5.3:5900  <-> 10.3.5.5:1404    430     25950     630     249236     1060     275186     6,622980000
20,6494
10.3.5.3:54238 <-> 10.3.5.12:102    113     10481      70       5726      183      16207     0,000000000
30,9946
10.3.5.3:54239 <-> 10.3.5.12:102      4       358       4        452        8        810    15,458449000
6,2378
10.3.5.3:54240 <-> 10.3.5.12:102      2       120       2        120        4        240    19,796503000
0,0149
```

The VNC session behaves differently like the one from the last task. More authentication result packets are seen in the session (see below), all with code 0 (success) and a little larger (60 bytes).

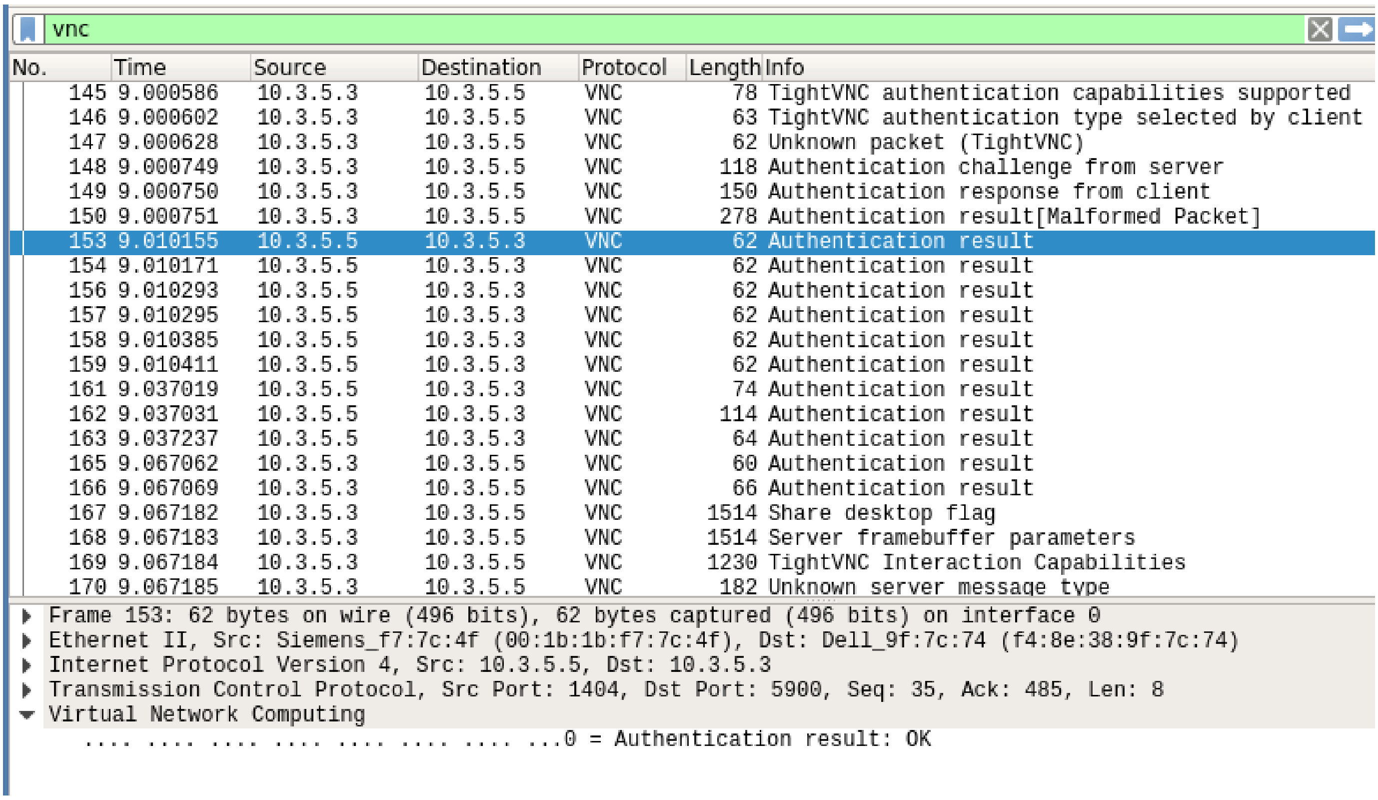


Figure 17. TightVNC authentication result responses

The authentication type selected can explain the difference. In the previous task, the authentication type was 2, for VNC, now the authentication type is 16, for TightVNC (see Figure 17 above).

```
$ tshark -n -r attack5.pcapng -Y 'frame.number in {115 116}' -Ovnc
Frame 115: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: f4:8e:38:9f:7c:74, Dst: 00:1b:1b:f7:7c:4f
Internet Protocol Version 4, Src: 10.3.5.3, Dst: 10.3.5.5
Transmission Control Protocol, Src Port: 5900, Dst Port: 1404, Seq: 13, Ack: 13, Len: 3
Virtual Network Computing
    Number of security types: 2
    Security type: VNC (2)
    Security type: Tight (16)

Frame 116: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
Ethernet II, Src: 00:1b:1b:f7:7c:4f, Dst: f4:8e:38:9f:7c:74
Internet Protocol Version 4, Src: 10.3.5.5, Dst: 10.3.5.3
Transmission Control Protocol, Src Port: 1404, Dst Port: 5900, Seq: 13, Ack: 16, Len: 1
Virtual Network Computing
    Security type selected: Tight (16)
```

When looking further into the VNC connection, the movement and button presses of the mouse can be seen as "client pointer event" packets. At two points, mouse button 1 is pressed:

- In Frame 609 - 630 (while the mouse moves from x=965/y=125 to x=985/y=99). The frames are transmitted within half a second, speculating, this maybe some drag operation.
- Button 1 is pressed again in Frame 1126 at position x=518/y=261, as can be seen below (Figure 18)

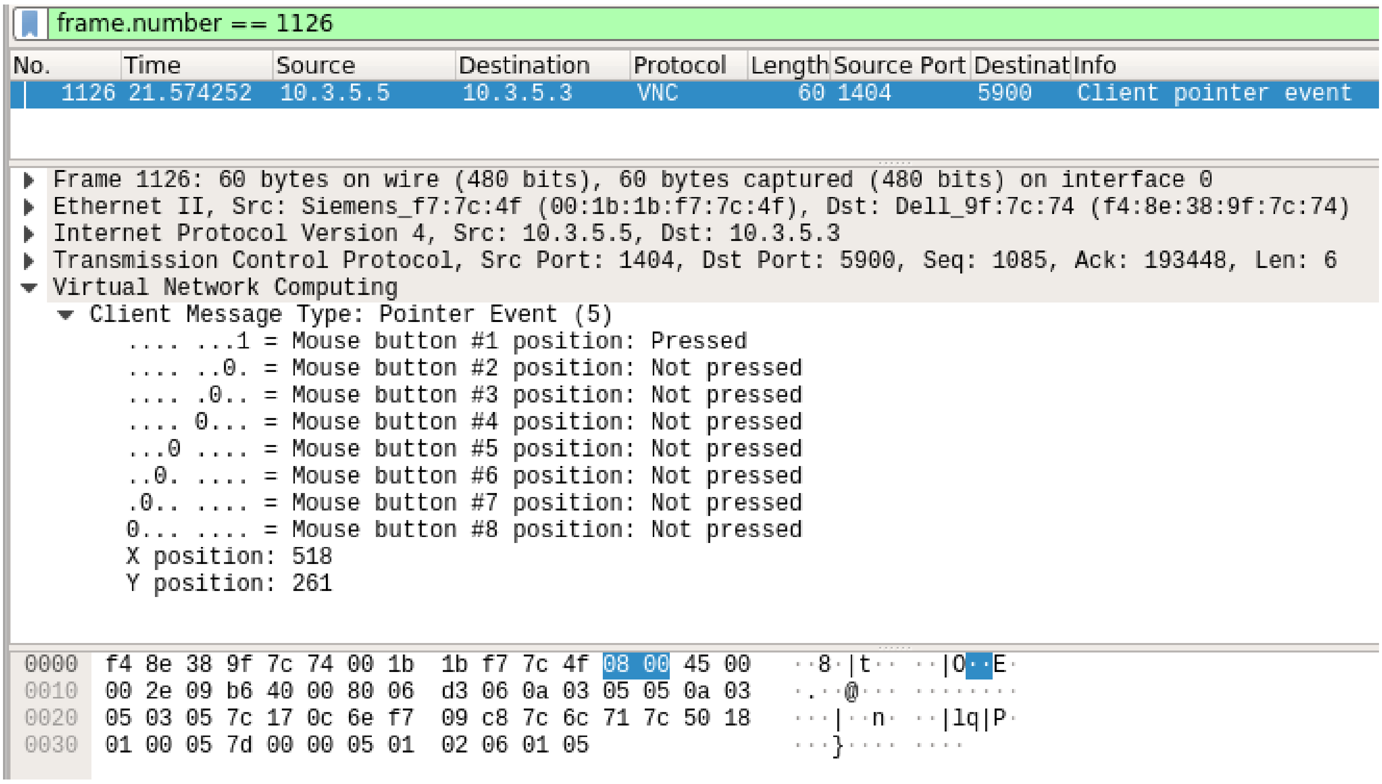


Figure 18. Mouse button press in VNC

Now for the s7plus connections.

Looking at the opcodes, nothing unusual seems to happen:

```
$ tshark -n -r attack5.pcapng -Y 's7comm-plus' -Tfields -e s7comm-
plus.data.opcode | sort -n | uniq -c
     16 0x00000031
     11 0x00000032
     47 0x00000033
```

Looking at the functions used, a similarity to the button_push capture file can be seen:

```
$ tshark -n -r attack5.pcapng -Y 's7comm-plus' -Tfields -e s7comm-
plus.data.function| sort -n | uniq -c
     47
      5 0x000004f2
      4 0x00000542
     18 0x0000054c

$ tshark -n -r button_push.pcapng -Y 's7comm-plus' -Tfields -e s7comm-
plus.data.function| sort -n | uniq -c
    467
     51 0x000004f2
     18 0x00000542
    188 0x0000054c
```

Let’s see if we can break this down by TCP session (TCP stream in *wireshark* terminology), the streams are numbered starting with 0.

Stream 0 seems to be the normal s7plus operation, the "background" so to say.

```
$ tshark -n -r attack5.pcapng -Y 'tcp.stream == 0 and s7comm-plus' -Tfields -e
s7comm-plus.data.opcode| sort -n | uniq -c
     14 0x00000031
      9 0x00000032
     47 0x00000033
$ tshark -n -r attack5.pcapng -Y 'tcp.stream == 0 and s7comm-plus' -Tfields -e
s7comm-plus.data.function| sort -n | uniq -c
     47
      5 0x000004f2
     18 0x0000054c
```

Stream 1 is the VNC connection and stream is again S7plus, but empty with regards to operations.

```
$ tshark -n -r attack5.pcapng -Y 'tcp.stream == 3 and s7comm-plus' -Tfields -e
s7comm-plus.data.function| sort -n | uniq -c
$ tshark -n -r attack5.pcapng -Y 'tcp.stream == 3 and s7comm-plus' -Tfields -e
s7comm-plus.data.opcode| sort -n | uniq -c
```

So, stream 2 is the interesting one, since only this one contains the SetMultiVariables (0x0542) operation.

```
$ tshark -n -r attack5.pcapng -Y 'tcp.stream == 2 and s7comm-plus' -Tfields -e
s7comm-plus.data.function| sort -n | uniq -c
      4 0x00000542
$ tshark -n -r attack5.pcapng -Y 'tcp.stream == 2 and s7comm-plus' -Tfields -e
s7comm-plus.data.opcode| sort -n | uniq -c
      2 0x00000031
      2 0x00000032
```

Two request -- response pairs can be seen in the capture file (we look at the request side only here).

The first seems to set a variable to "false",

```
$ tshark -n -r attack5.pcapng -Y 's7comm-plus.data.opcode == 0x31 and s7comm-
plus.data.function == 0x0542' -Os7comm-plus

Frame 952: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits) on interface 0
Ethernet II, Src: f4:8e:38:9f:7c:74, Dst: 28:63:36:ad:91:96
Internet Protocol Version 4, Src: 10.3.5.3, Dst: 10.3.5.12
Transmission Control Protocol, Src Port: 54239, Dst Port: 102, Seq: 1, Ack: 1, Len: 111
TPKT, Version: 3, Length: 111
ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
S7 Communication Plus
    Header: Protocol version=V3
        Protocol Id: 0x72
        Protocol version: V3 (0x03)
        Data length: 96
    Integrity part
        Digest Length: 32
        Packet Digest: c35ed4e0619e3c1de9ec6694d0f27cd1451dd9c45f7070d8...
    Data: Request SetMultiVariables
        Opcode: Request (0x31)
        Reserved: 0x0000
        Function: SetMultiVariables (0x0542)
        Reserved: 0x0000
        Sequence number: 7
        Session Id: 0x000003ba
        Transport flags: 0x34, Bit2-AlwaysSet?, Bit4-AlwaysSet?, Bit5-AlwaysSet?
            .... ...0 = Bit0: False
            .... ..0. = Bit1-SometimesSet?: False
            .... .1.. = Bit2-AlwaysSet?: True
            .... 0... = Bit3: False
            ...1 .... = Bit4-AlwaysSet?: True
            ..1. .... = Bit5-AlwaysSet?: True
            .0.. .... = Bit6-NoResponseExpected?: False
            0... .... = Bit7: False
        Request Set
            Unknown: 0x00000000
            Item Count: 1
            Number of fields in complete Item-Dataset: 5
            AddressList
                Item Address [1]: (82), SYM-CRC=df6ac14c, (3736), LID=9
                    Symbol CRC: 0xdf6ac14c
                    Access base-area: Unknown (82)
                    Number of following IDs: 2
                    Access sub-area: Unknown (3736)
                    LID Value: 9
            ValueList
                Item Value [1]: (Bool) = False
                    Item Number: 1
                    Datatype flags: 0x00
                        ...0 .... = Array: False
                        ..0. .... = Addressarray: False
                        .0.. .... = Sparsearray: False
                        0... .... = Unknown-Flag1: False
                    Datatype: Bool (0x01)
                    Value: False
        Data unknown: 000000
…
```

The second sets a variable to "true":

```
Frame 1129: 165 bytes on wire (1320 bits), 165 bytes captured (1320 bits) on interface 0
Ethernet II, Src: f4:8e:38:9f:7c:74, Dst: 28:63:36:ad:91:96
Internet Protocol Version 4, Src: 10.3.5.3, Dst: 10.3.5.12
Transmission Control Protocol, Src Port: 54239, Dst Port: 102, Seq: 119, Ack: 66, Len:
111
TPKT, Version: 3, Length: 111
ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
[2 COTP Segments (104 bytes): #954(0), #1129(104)]
S7 Communication Plus
    Header: Protocol version=V3
        Protocol Id: 0x72
        Protocol version: V3 (0x03)
        Data length: 96
    Integrity part
        Digest Length: 32
        Packet Digest: ced5b77ab7ea0919e7c4a5094206bf0f3547e088f06c674f...
    Data: Request SetMultiVariables
        Opcode: Request (0x31)
        Reserved: 0x0000
        Function: SetMultiVariables (0x0542)
        Reserved: 0x0000
        Sequence number: 8
        Session Id: 0x000003ba
        Transport flags: 0x34, Bit2-AlwaysSet?, Bit4-AlwaysSet?, Bit5-AlwaysSet?
            .... ...0 = Bit0: False
            .... ..0. = Bit1-SometimesSet?: False
            .... .1.. = Bit2-AlwaysSet?: True
            .... 0... = Bit3: False
            ...1 .... = Bit4-AlwaysSet?: True
            ..1. .... = Bit5-AlwaysSet?: True
            .0.. .... = Bit6-NoResponseExpected?: False
            0... .... = Bit7: False
        Request Set
            Unknown: 0x00000000
            Item Count: 1
            Number of fields in complete Item-Dataset: 5
            AddressList
                Item Address [1]: (82), SYM-CRC=fc4ae127, (3736), LID=10
                    Symbol CRC: 0xfc4ae127
                    Access base-area: Unknown (82)
                    Number of following IDs: 2
                    Access sub-area: Unknown (3736)
                    LID Value: 10
            ValueList
                Item Value [1]: (Bool) = True
                    Item Number: 1
                    Datatype flags: 0x00
                        ...0 .... = Array: False
                        ..0. .... = Addressarray: False
                        .0.. .... = Sparsearray: False
                        0... .... = Unknown-Flag1: False
                    Datatype: Bool (0x01)
                    Value: True
        Data unknown: 000000
...
```

Now it's time to combine both parts, we can see that the second SetMultiVariables request comes immediately after the mouse button press event in the VNC session (Figure 19).

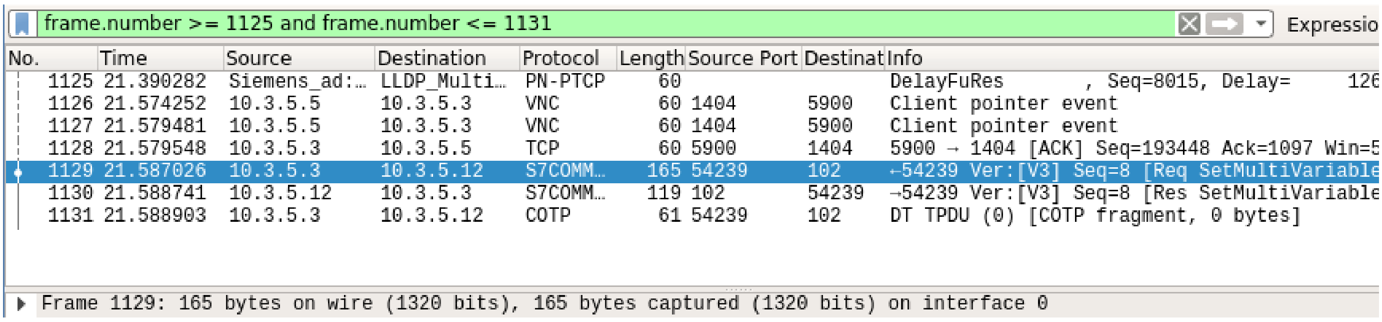


Figure 19. Mouse button press and SetMultiVariable request

So, the answer is:

- The attack was two-staged, with
  - The first stage was a VNC connection from the engineering workstation to the SCADA workstation (probably using the guessed password)
  - The second stage was by S7plus, setting a variable (likely to control the pump)

Although it is not in the packet capture, one can infer that the adversary used the SCADA application to disable the pump through the GUI.

About the spotting of the attack:

- The unusual authentication type (TightVNC) gives away the initial VNC connection that started the attack. Adversaries could improve by using the same authentication type as regular connection, so it would no longer look suspicious.
- The button push, as coming from the SCADA application itself, would not be noticed as there is nothing that differentiates it from normal traffic.
- The combination of a VNC connection and an unusual event (like pump shutdown) would likely raise suspicion, as the SCADA operator would normally not use a remote connection but sit in front of the workstation. Also, this would only be known after the attack had already taken place and a forensic investigation would begin.

3.5.3 The PLC reprogramming attack

The infected engineering workstation is used to reprogram one of PLCs (by downloading the running program from the PLC, modifying it, and re-uploading the changed program to the PLC). The new program changes the industrial process and makes it impossible to be changed back to the original by an operator (using the SCADA workstation).

3.5.4 Subtask: Analyse the last attack stage

Students: Given the packet capture `attack6.pcapng`, analyse the last attack stage. Try to answer the following questions:

- Where did the attack originate?
- Try to correlate the network activity with what is known about the attack (see 3.5.3 above).
- Where are the problems with regards to the correlation?

Solution: This time, there is direct involvement of the compromised engineering workstation, several connections to port 102 on one of the PLCs (10.3.5.3.12) can be seen:

```
$ tshark -q -n -r attack6.pcapng -z conv,tcp
================================================================================
TCP Conversations
Filter:<No Filter>
                                  |       <-      | |       ->      | |     Total     |    Relative    |
Duration |
                                  | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |      Start     |
|
10.3.5.3:54238 <-> 10.3.5.12:102   977     91009     623     51839    1600    142848     0,478472000
279,2012
10.3.5.5:1414  <-> 10.3.5.12:102   423     45991     297     25311     720     71302   113,727670000
111,8493
10.3.5.5:1416  <-> 10.3.5.12:102   150    113624     132      9816     282    123440   126,533451000
11,1994
10.3.5.5:1415  <-> 10.3.5.12:102    82      6733      66      6561     148     13294   115,693498000
109,7862
10.3.5.5:1417  <-> 10.3.5.12:102    52      8500      59     12330     111     20830   212,373235000
6,6389
10.3.5.3:54239 <-> 10.3.5.12:102    53      4244      53      5072     106      9316     5,104313000
274,5798
10.3.5.3:54240 <-> 10.3.5.12:102    21      1321      21      1350      42      2671    22,401286000
257,2871
```

When looking at the number of frames and the number of times an IP-address shows up in a TCP stream, we can collate stream numbers in Wireshark to conversations.

```
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 0' -Tfields -e ip.addr | sort -n | uniq -c
    623 10.3.5.3,10.3.5.12
    977 10.3.5.12,10.3.5.3
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 1' -Tfields -e ip.addr | sort -n | uniq -c
     53 10.3.5.3,10.3.5.12
     53 10.3.5.12,10.3.5.3
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 2' -Tfields -e ip.addr | sort -n | uniq -c
     21 10.3.5.3,10.3.5.12
     21 10.3.5.12,10.3.5.3
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 3' -Tfields -e ip.addr | sort -n | uniq -c
    297 10.3.5.5,10.3.5.12
    423 10.3.5.12,10.3.5.5
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 4' -Tfields -e ip.addr | sort -n | uniq -c
     66 10.3.5.5,10.3.5.12
     82 10.3.5.12,10.3.5.5
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 5' -Tfields -e ip.addr | sort -n | uniq -c
    132 10.3.5.5,10.3.5.12
    150 10.3.5.12,10.3.5.5
$tshark -n -r attack6.pcapng -Y 'tcp.stream == 6' -Tfields -e ip.addr | sort -n | uniq -c
     59 10.3.5.5,10.3.5.12
     52 10.3.5.12,10.3.5.5
```

And it seems like the malware is trying to contact its C&C server (note the communication to port 8910). As it comes late in the packet capture, it looks like it is trying to report its success, but this is just a guess.

```
$ tshark -q -n -r attack6.pcapng -z conv,udp
================================================================================
UDP Conversations
Filter:<No Filter>
                                         |       <-      | |       ->      | |     Total     |   Relative   |
Duration|
                                         | Frames  Bytes | | Frames  Bytes | | Frames  Bytes |   Start      |
|
10.3.5.3:51487 <-> 255.255.255.255:1947   0       0         16      1312      16      1312      11,556976
269,0279
10.3.5.3:51487 <-> 10.3.5.255:1947        0       0         16      1312      16      1312      15,562995
269,0664
10.3.5.5:49152 <-> 255.255.255.255:1947   0       0         14      1148      14      1148      25,246807
232,5779
10.3.5.5:49152 <-> 10.255.255.255:1947    0       0         14      1148      14      1148      29,253409
232,5809
10.3.5.3:60070 <-> 234.5.6.7:8910         0       0         11      5576      11      5576     279,618206
1,4743
...
```

Going back to the TCP streams, going by the number of frames, we now examine the streams from the SCADA workstation (10.3.5.3 to 10.3.5.12)

```
$ tshark -n -r attack6.pcapng -Y 'tcp.stream == 0 and s7comm-plus' -Tfields -e
s7comm-plus.data.opcode | sort -n | uniq -c
    133 0x00000031
     89 0x00000032
    400 0x00000033

tshark -n -r attack6.pcapng -Y 'tcp.stream == 0 and s7comm-plus' -Tfields -e s7comm-
plus.data.function | sort -n | uniq -c
    400
      6 0x000004d4
     48 0x000004f2
    168 0x0000054c
```

So, judging by the IP-addresses, opcodes and functions, this seems to be the normal background S7plus activity, except for the DeleteObject (0x04d4) operations. They seem to happen towards the end of that stream (at frame 5680) at frame 5524, 5534, and 5674 (see Figure 20 below).

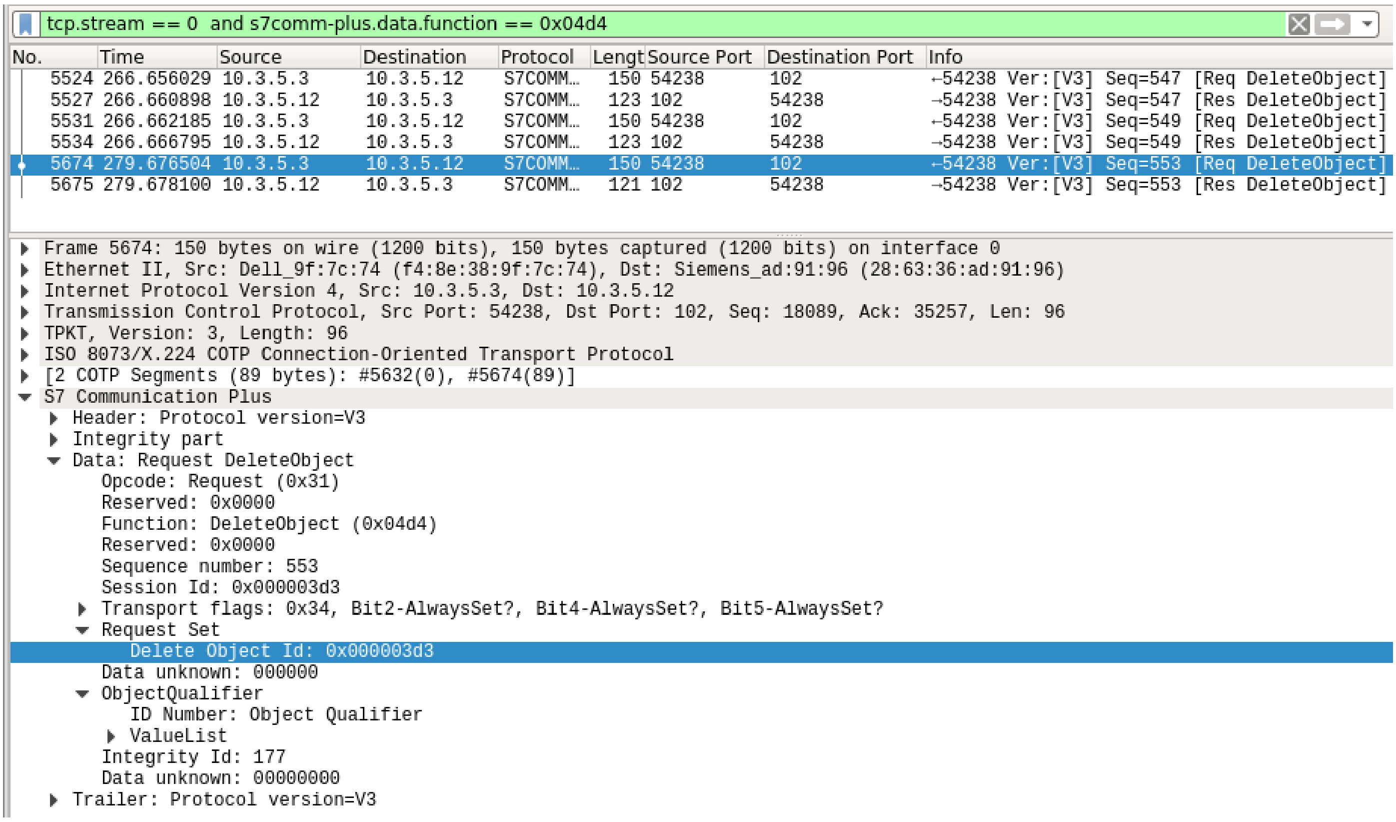


Figure 20. S7comm-plus DeleteObject requests and responses

The other two streams seem to try to delete objects too (inferring from the function codes).

```
Stream 1 opcodes:
     18 0x00000031
     18 0x00000032
Stream 1 functions
      2 0x000004d4
     34 0x00000542

Stream 2 opcodes:
      1 0x00000031
      1 0x00000032
Stream 1 functions:
      2 0x000004d4
```

There are four TCP connections from the engineering workstation to the PLCs, with a breakdown of its opcodes and functions used:

```
Stream 3 opcodes:
     69 0x00000031
     69 0x00000032
    125 0x00000033
functions
     18 0x000004bb
     10 0x000004ca
     10 0x000004d4
      8 0x000004f2
      4 0x00000524
      6 0x00000542
     24 0x0000054c
```

```
      2 0x0000056b
     56 0x00000586

Stream 4 opcodes
     19 0x00000031
     19 0x00000032
functions
      8 0x000004ca
      8 0x000004d4
      8 0x000004f2
      4 0x00000524
      2 0x00000542
      8 0x00000586

Stream 5 opcodes
     13 0x00000031
     13 0x00000032
functions
     12 0x000004bb
      2 0x000004ca
      2 0x000004d4
      2 0x000004f2
      2 0x00000542
      6 0x00000586

Stream 6
opcodes
     24 0x00000031
     24 0x00000032

functions
     12 0x000004bb
     14 0x000004ca
      2 0x000004d4
      4 0x000004f2
      2 0x00000542
      2 0x00000556
      2 0x00000560
     10 0x00000586
```

As can be seen, there are previously unseen functions the composition is also unseen before. While functions like SetVariable or DeleteObject are more or less self-explanatory, functions like Invoke (0x056b) or GetVarSubStreamed (0x0586) are not. Without in-depth knowledge of the PLC operation and its internal memory layout one cannot hope to make any sense out of it.

Even when looking into the request packets, no more information will be gained that will help in resolving the incident. There is still the IP-address and the unusual functions used in S7plus connections which is enough to flag this as suspicious activity, however without prior knowledge that something malicious had happened it would be impossible to infer what has happened (malicious or not) from the packet content.

3.6 Tools used in this use-case

Tool	Homepage
tshark	https://www.wireshark.org/
wireshark	https://www.wireshark.org/

3.7 Further reading

- ENISA Report: *Protecting Industrial Control Systems. Recommendations for Europe and Member States*, https://www.enisa.europa.eu/publications/protecting-industrial-control-systems.-recommendations-for-europe-and-member-states
- ENISA Report: *Analysis of ICS-SCADA Cyber Security Maturity Levels in Critical Sectors*, https://www.enisa.europa.eu/publications/maturity-levels
- ENISA Report: *Certification of Cyber Security skills of ICS/SCADA professionals*, https://www.enisa.europa.eu/publications/certification-of-cyber-security-skills-of-ics-scada-professionals
- ENISA Report*: Good Practices for an EU ICS Testing Coordination Capability*, https://www.enisa.europa.eu/publications/good-practices-for-an-eu-ics-testing-coordination-capability
- ENISA Report: *Window of exposure… a real problem for SCADA systems?*, https://www.enisa.europa.eu/publications/window-of-exposure-a-real-problem-for-scada-systems
- ENISA Report: *Can we learn from SCADA security incidents?*, https://www.enisa.europa.eu/publications/can-we-learn-from-scada-security-incidents

4. Glossary and References

4.1 Glossary

ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
C&C	Command and Control (Server)
CLI	Command Line Interfaces
COTP	Connection Oriented Transport Protocol
GUI	Graphical User Interface
ICS	Industrial Control Systems
IGMP	Internet Group Management Protocol
ISO 27001	International Organization for Standardization
LLDP	Link Local Discovery Protocol
LLMNR	Link Local Multicast Name Resolution
PCAP	Packet CAPture
PLC	Programmable Logic Controller
SCADA	Supervisory Control and Data Acquisition
SMB	Server Message Block
SSDP	Simple Service Discovery Protocol
TCP	Transmission Control Protocol
TPKT	Packet format used to transport OSI TPDUs over TCP
TPDU	(OSI) Transport Protocol Data Uni
UDP	User Datagram Protocol
VNC	Virtual Network Computing

4.2 References

Bejtlich, R. (2013), *The Practice of Network Security Monitoring – Understanding Incident Detection and Response*, No Starch Press, 2013, ISBN-13:1-59327-509-9

ENISA (2011), *Protecting Industrial Control Systems Recommendations for Europe and Member States*, https://www.enisa.europa.eu/topics/critical-information-infrastructures-and-services/scada (last accessed on October 7th, 2018)

ENISA
European Union Agency for Cybersecurity
Science and Technology Park of Crete (ITE)
Vassilika Vouton, 700 13, Heraklion, Greece

Athens Office
1 Vass. Sofias & Meg. Alexandrou

Marousi 151 24, Athens, Greece

PO Box 1309, 710 01 Heraklion, Greece
Tel: +30 28 14 40 9710
info@enisa.europa.eu
www.enisa.europa.eu

ISBN: 978-92-9204-288-2
DOI: 10.2824/995110