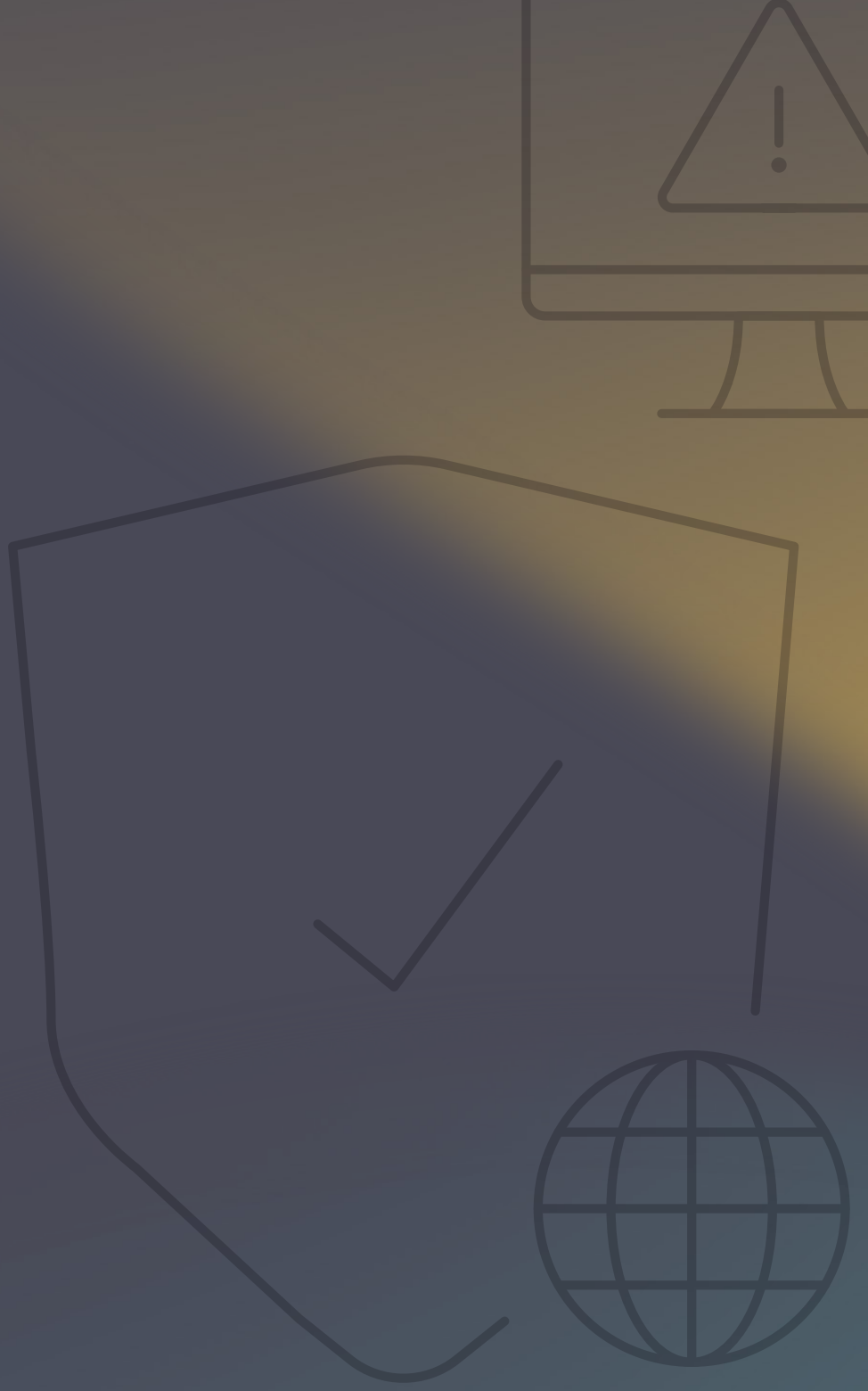
NOZOMI NETWORKS LABS



WHITE PAPER

Systemic Security Issues in SoftPLCs

Table of Contents

Table of Contents

1. Abstract

A Programmable Logic Controller (PLC) is a specialized hardware device widely used in industrial automation to control machinery and industrial processes. In contrast, a Software-based PLC (SoftPLC), which is not a hardware device, implements the same control logic on general-purpose computing platforms, offering increased flexibility, scalability, and integration capabilities. However, this architectural transition from dedicated hardware to software running on conventional operating systems significantly expands the attack surface, introducing threat scenarios, traditionally associated with IT systems.

This white paper presents the results of a focused security assessment of modern SoftPLC platforms. The study draws on extensive testing and analysis of six widely adopted SoftPLC implementations from different vendors, providing a representative picture of the current SoftPLC security landscape across the market.

Through this research, we identify and demonstrate multiple attack scenarios in which vendor security models can be compromised. These scenarios primarily stem from insufficient hardening and isolation at the operating system level, exposing SoftPLC platforms to vulnerabilities that are uncommon in traditional PLC environments. The findings highlight a structural gap in current industrial threat modeling approaches and underline the need for improved security practices and hybrid IT/OT threat models to ensure the safe deployment of SoftPLC technologies.

2. Introduction

2.1 Programmable Logic Controller (PLC)

A **Programmable Logic Controller (PLC)** is a specialized industrial computer designed to automate electromechanical processes, such as control of machinery on factory assembly lines, amusement rides, or lighting fixtures. Unlike general-purpose computers, PLCs are engineered to withstand harsh industrial environments characterized by extreme temperatures, humidity, electrical noise, and vibrations.

Due to their usage in critical environments, a PLC must guarantee the following properties:

- **Reliability:** Capable of operating continuously for years with minimal maintenance.
- **Deterministic Operation:** Ensures predictable response times essential for real-time control tasks.
- **Ruggedness:** Resistant to environmental stressors, including dust, moisture, and electromagnetic interference.
- **Ease of Programming:** Uses specialized programming languages such as Ladder Logic, Structured Text or Function Block Diagram, which are intuitive for engineers familiar with control systems.

Normally, a PLC operates by continuously cycling through three primary stages:

1. **Input Scanning:** The PLC reads signals from connected field devices such as sensors, switches, and encoders.
2. **Program Execution:** The control logic, programmed by engineers, processes the input data and determines appropriate actions.
3. **Output Updating:** The PLC sends commands to actuators, motors, valves, or other output devices to carry out the control tasks.

Figure 1 - A PLC installed inside an industrial site

Additional functionalities include communication with other PLCs or supervisory systems (via industrial protocols such as Modbus, PROFIBUS, or Ethernet/IP), diagnostic capabilities, support for safety and redundancy features in critical environments.

2.2 The SoftPLC model

A Software-based Programmable Logic Controller (SoftPLC) is a control system that replicates the functions of a traditional hardware PLC but operates as software on general-purpose computing hardware, such as industrial PCs, servers, or embedded platforms. While it executes the same industrial control logic as a standard PLC, its software-based architecture enables greater flexibility in deployment, easier updates, and seamless integration with modern IT infrastructure. This makes SoftPLCs particularly attractive for industries undergoing digital transformation and adopting Industrial Internet of Things (IIoT) technologies, where connectivity and scalability are increasingly important.

The main distinction between a SoftPLC and a standard PLC lies in the hardware and operating environment. Traditional PLCs are purpose-built, ruggedized devices designed for deterministic performance and reliability under extreme industrial conditions. They run on proprietary real-time operating systems or optimized firmware, ensuring a predictable and stable behavior. In contrast, SoftPLCs depend on general-purpose operating systems like Windows or Linux, which do not inherently guarantee the same level of real-time responsiveness. To achieve similar performance, SoftPLCs often require additional configurations, real-time extensions, or specialized hardware support.

SoftPLCs offer several benefits, including support for more complex control algorithms, the ability to handle larger data volumes, and easier integration with advanced analytics and cloud-based solutions. However, these advantages come with certain trade-offs. Because they run on non-dedicated platforms, SoftPLCs inherit potential vulnerabilities from the underlying OS and hardware, making them more susceptible to malware, unauthorized access, and performance issues if not properly hardened. They also rely on system resources that may be affected by other processes, increasing the risk of instability in critical operations.

3. The SoftPLC

3.1 Software and Operating Systems

SoftPLC (Software-based Programmable Logic Controller) systems replace traditional, hardware-based PLCs by running control software on general-purpose computing platforms using standard operating systems. This shift unlocks significant benefits in terms of flexibility, scalability, and cost-efficiency, though it does introduce a greater level of complexity in software configuration and system maintenance. Traditional PLCs often depend on real-time operating systems (RTOS) or minimal firmware for deterministic performance, whereas SoftPLCs typically operate on general-purpose operating systems (GPOS) such as Windows, Linux, or BSD. These platforms offer multiple advantages:

- Seamless integration with enterprise IT systems, cloud platforms, and modern software frameworks
- Broad hardware compatibility, reducing vendor lock-in and allowing use of commodity or industrial PCs
- Lower deployment and operational costs, particularly in systems that require virtualization or remote access
- Advanced networking features, including support for IP-based protocols, secure tunnels, firewalls, and software-defined networking (SDN)
- Enhanced extensibility through the installation of third-party applications and modules

Figures 2a and 2b depict a Beckhoff SoftPLC platform based on Windows, delivered as an industrial PC running a full-featured Windows operating system.

Figure 2a - Beckhoff C6015 TwinCAT-based SoftPLC

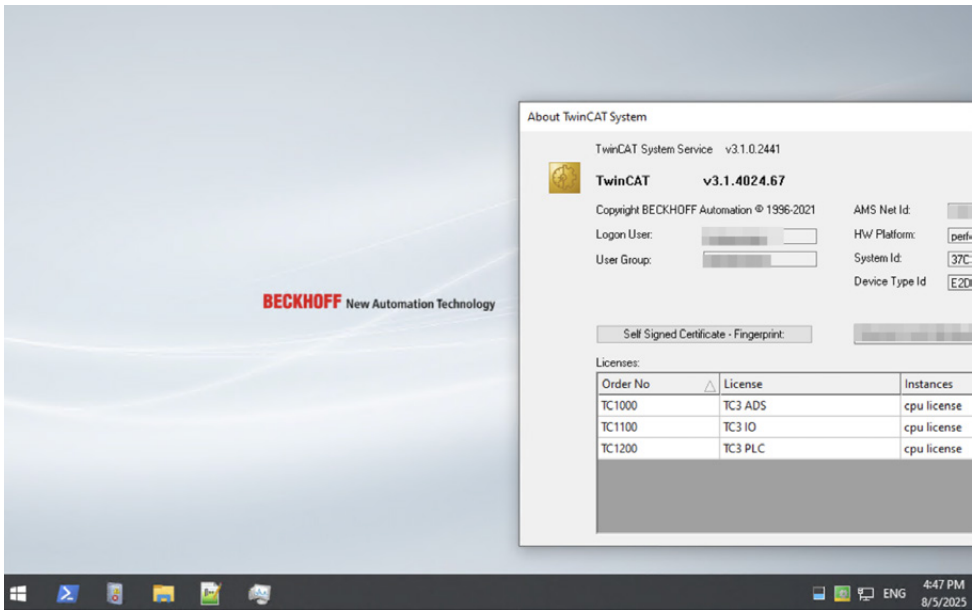


Figure 2b - Beckhoff C6015 Windows Operating System interface

Solutions like Codesys, Beckhoff TwinCAT, and OpenPLC offer extensive development ecosystems, including simulation environments, version control integration, and remote debugging tools. These capabilities are essential for implementing distributed control architectures, edge computing, or cloud-connected automation.

However, as SoftPLCs increasingly connect to enterprise networks and the internet, cybersecurity becomes a critical concern. Unlike isolated hardware PLCs, SoftPLCs running on standard OSs are more exposed to threats and therefore require robust security hardening, including user authentication, patch management, network segmentation, and intrusion detection systems—often provided through integration with industrial cybersecurity platforms like Nozomi Networks Guardian.

3.2 Security Model

SoftPLC systems typically implement a granular and role-based security model that defines what each user can view, modify, or control within the system. The complexity of this model varies: in basic configurations, access may be limited to just two roles, user and administrator (simple ACLs: Access Control List), providing a clear division between general operation and system configuration privileges.

However, many modern SoftPLC platforms support more sophisticated user hierarchies, with roles such as engineer, auditor, and viewer-only, each assigned a tailored set of capabilities (fixed or customizable RBAC: Role Based Access Control).

For example:

- An administrator may have unrestricted access to all configuration settings, including network interfaces, logic deployment, firmware updates, and user management.
- An engineer typically has permissions to edit and upload control logic, adjust parameters, and perform system diagnostics, but may not be allowed to create new users or modify core security settings.
- Auditors are often granted read-only access to system logs, configuration history, and change tracking features for compliance purposes.
- View-only users, such as operators or plant personnel, may only be able to monitor system status and alarms, without the ability to modify any control logic or configurations.

Configuration		
PLC Runtime	**PLC Runtime Configuration**	**user**
Networking		
TCP/IP Configuration	TCP/IP Configuration	user
Ethernet Configuration	Ethernet Configuration	user
Host/Domain Name	Configuration of Host and Domain Name	user
Routing	Routing	user
Clock	**Clock Settings**	**user**
Administration		
Serial Interface	Configuration of Serial Interface RS232/RS48	admin
Service Interface	Configuration of Service Interface	admin
Create Image	Create bootable Image	admin

Figure 3a - Wago PFC200 750-8216 roles (user and admin) define in the user manual

Application or Service	Access Permission for:	User Role														
		Admin	SecurityAdmin	SecurityAuditor	CertificateManager	UserManager	Engineer	Commissioner	Service	DateViewer	DateChanger	Viewer	FileReader	FileWriter	EHmiLevel1 → EHmiLevel10	EHmiViewer	EHmiChanger
SD card, parameterization memory	SFTP access to the file system with an SFTP client note	✓															
Shell	SSH access to the shell note Configuration	✓															
PLCnext Engineer	View values in the cockpit (e.g., utilization) Configuration of Host and Domain Name	✓	✓				✓	✓	✓	✓	✓	✓					

Figure 3b - Phoenix Contact AXC F 3152 PLCNext multiple roles define in the user manual

Additionally, access to the underlying OS-level superuser account, such as root in Unix-like systems or SYSTEM on Windows, is usually prohibited or tightly restricted, even from the highest-level SoftPLC roles. This design minimizes the risk of unauthorized system-level changes or exposure to malware.

Some vendors go further by allowing dynamic role creation, where administrators can define custom user groups and assign granular permissions across specific features, projects, or devices. This flexibility is particularly useful in large-scale industrial environments where different teams, such as maintenance, IT security, or third-party integrators, need tailored access to specific parts of the system without compromising overall security.

3.3 Expanded Attack Surface: PLC vs SoftPLC

Traditional PLCs are intentionally designed with a minimal attack surface. Their firmware-based execution environments typically expose no general-purpose file systems, no dynamic service management, and no support for arbitrary local code execution beyond the control logic runtime.

SoftPLCs fundamentally change this paradigm by introducing a full-fledged operating system under the hood. By doing this, they inherently expand the attack surface by adding several new components, such as:

- General-purpose operating systems, inheriting decades of IT security complexity, introduce a broad and well-documented vulnerability landscape. These platforms include numerous subsystems, libraries, and services that were not originally designed for safety-critical or highly constrained industrial environments. As a result, SoftPLCs inherit classes of vulnerabilities, such as privilege escalation, misconfigurations, and service abuse, that are largely absent from firmware-based PLCs.
- Full file system access enables users and processes to read from and write to directories containing binaries, configuration files, logs, and trusted data. Improper permission models or weak isolation mechanisms can allow attackers to tamper with executables, modify configuration files to alter system behavior, or replace trusted components, facilitating persistence and privilege escalation.
- Inter-process communication (IPC) mechanisms, including sockets, shared memory, pipes, and message queues, enable complex interactions between system components. When these mechanisms are insufficiently protected, attackers can intercept, manipulate, or inject data into trusted communication channels, leading to unauthorized resource access, execution flow manipulation, or elevation of privileges.
- System services and daemons, often running with elevated privileges, expand the attack surface by providing long-lived execution contexts. Misconfigured or vulnerable services can be abused to hijack execution flows, execute attacker-controlled code, or gain persistent access, particularly when services trust user-controllable inputs or files.
- Dynamic modules, plugins, and application frameworks, sometimes distributed via app stores or package managers, allow SoftPLCs to be extended with additional functionality. While this improves flexibility, it also introduces risks related to untrusted code execution, supply chain vulnerabilities, and insufficient validation of third-party components, potentially enabling arbitrary code execution within the system.
- Full-fledged command-line shells, such as Bash or PowerShell, provide powerful interfaces for system interaction and automation. Even when accessed by low-privileged users, these shells enable the execution of complex command sequences, scripting, and environment manipulation, significantly increasing the potential impact of initial access and facilitating subsequent execution and privilege escalation stages.

These components enable attacker behaviors that are largely absent from traditional PLC environments. Specifically, they introduce execution contexts, privilege hierarchies, and process interaction mechanisms that resemble enterprise systems. As a result, SoftPLC security challenges cannot be fully understood using traditional PLC threat assumptions.

4. Research Scope and Methodology

This research is intentionally scoped and does not attempt to provide an exhaustive security assessment of SoftPLC platforms. Rather than surveying all possible vulnerabilities or attack vectors, the analysis focuses on a well-defined subset of attacker behaviors that emerge specifically because of the additional attack surface introduced by the new SoftPLC architecture.

In particular, the study focuses on behaviors that are enabled by the presence of a general-purpose operating system, such as:

- Local execution environments (interactive shells and process creation)
- Privilege hierarchies and OS-level separation
- File system access to configuration, binaries, and trusted data
- Local IPC mechanisms used by privileged services

These elements fundamentally distinguish SoftPLCs from traditional PLCs and introduce attack paths that are uncommon, or entirely absent, in firmware-based controllers. By narrowing the scope in this way, the research aims to provide depth rather than breadth, allowing for a detailed examination of how these OS-level capabilities can be abused following initial access. This targeted approach enables a precise evaluation of whether existing industrial threat models adequately represent the risks introduced by SoftPLC platforms, while avoiding overgeneralization beyond the defined attack surface.

4.1 MITRE ATT&CK for ICS Overview

The MITRE ATT&CK for ICS framework is a structured knowledge base that documents adversary tactics, techniques, and procedures (TTPs) observed in real-world attacks targeting industrial control systems. It is specifically designed to model malicious behaviors affecting industrial assets such as PLCs, Human–Machine Interfaces (HMIs), Distributed Control Systems (DCS), and Supervisory Control and Data Acquisition (SCADA) environments. The framework places strong emphasis on actions that can influence industrial operations, safety mechanisms, and physical processes, reflecting the unique priorities and risks of OT environments.

The framework is widely adopted by asset owners, system integrators, and security vendors for threat modeling, detection engineering, and defensive strategy development in industrial contexts. It provides a common language for describing attacker behavior, supports the development of security use cases and monitoring logic, and facilitates communication between engineering, security, and incident response teams operating in ICS environments.

In this research, the MITRE ATT&CK for ICS framework was used as the primary reference for threat modeling SoftPLC platforms, with the goal of evaluating their security posture under realistic attack assumptions. By grounding the analysis in a framework derived from real-world adversary behavior, this study aims to assess how effectively SoftPLC security mechanisms withstand practical exploitation scenarios, rather than theoretical or purely compliance-driven evaluations. This approach allows the identified weaknesses to be contextualized within established attack patterns and highlights gaps between expected and actual security guarantees in modern SoftPLC environments.

4.2 Research Focus and Scope

SoftPLC platforms challenge several historical assumptions embedded in ICS threat modeling. Traditional controllers often run proprietary firmware or RTOS platforms with tightly constrained execution environments. In such systems, local execution capabilities are limited, operating system services are minimal or inaccessible, and privilege boundaries are often simplified or vendor-specific. As a result, the framework intentionally abstracts or omits many operating system–level behaviors that are common in enterprise environments but historically irrelevant to classical PLC deployments.

In contrast, SoftPLCs introduce a general-purpose operating system and an enterprise-style execution environment into OT systems, significantly increasing internal complexity. This research intentionally focuses on the Execution and Privilege Escalation phases to highlight how this added complexity can undermine the security models defined by SoftPLC vendors. By examining how low-privileged users can interact with operating system components, services, and trusted execution paths, the study aims to demonstrate how internal mechanisms—rather than external attack vectors alone—can be abused to bypass role-based controls and escalate privileges within the device.

4.3 Execution and Privilege Escalation in MITRE ATT&CK for ICS

To clearly define the scope of this research, it is necessary to examine how the MITRE ATT&CK for ICS framework models the Execution and Privilege Escalation phases of an attack. These two stages are central to the analysis presented in this white paper, as they represent the primary mechanisms by which attackers can transition from limited access to full system compromise in SoftPLC environments.

The ICS ATT&CK framework was designed around traditional industrial architectures, where controllers typically expose minimal local execution capabilities and limited operating system functionality. As a result, execution and privilege escalation techniques in the ICS framework are intentionally narrow in scope and focus primarily on behaviors that directly affect industrial operations rather than underlying operating system mechanics.

The table below summarizes how the ICS ATT&CK framework currently represents these two tactics and highlights their intended focus.

Tactic	Technique	High-Level Description
Execution (TA0109)	Autorun Image (T1547)	Automatic execution of logic or components during startup or initialization
	Change Operating Mode (T0858)	Switching controller or system modes to enable or alter execution behavior
	Command-Line Interface (T1059.003)	Execution of commands through exposed command-line interfaces
	Execution through API (T1106)	Triggering execution via exposed application or system APIs
	Graphical User Interface (T1061)	Execution of actions through operator or engineering graphical interfaces
	Hooking (T1179)	Intercepting or altering the normal execution flow of functions or tasks
	Modify Controller Tasking (T0839)	Changing task scheduling, priorities, or execution logic on the controller
	Native API (T1106)	Use of native system APIs to execute actions or logic
	Scripting (T1059.005)	Execution of scripts within supported or embedded scripting environments
	User Execution (T1204)	Execution of actions directly initiated by a logged-in user
Privilege Escalation (TA0111)	Exploitation for Privilege Escalation (T1068)	Abuse of vulnerabilities or misconfigurations to gain higher privileges
	Hooking (T1179)	Manipulation of execution flow to bypass or elevate privilege boundaries

4.4 Analysis of ICS Framework Coverage

As shown in Table 1, the ICS ATT&CK framework models execution primarily as interaction with exposed interfaces rather than as arbitrary code execution within a general-purpose operating system. Similarly, privilege escalation is represented as a high-level concept without explicit differentiation between application-layer privilege changes and operating system–level privilege boundaries.

This abstraction is appropriate for traditional PLCs, where:

- Local execution environments are highly constrained
- Privilege separation is minimal or vendor-specific
- Operating system services are either absent or inaccessible

However, SoftPLC platforms diverge significantly from these assumptions. By running on full-featured operating systems, SoftPLCs introduce execution vectors and privilege hierarchies that closely resemble enterprise systems, including process creation, service manipulation, IPC abuse, and file-system–based attacks. These behaviors fall largely outside the intended scope of the ICS ATT&CK execution and privilege escalation tactics.

For this reason, the research presented in this white paper focuses explicitly on execution and privilege escalation paths enabled by the additional attack surface introduced by SoftPLCs, rather than on generic ICS attack techniques.

The analysis assumes initial access and investigates how OS-level capabilities allow attackers to:

- Execute code or commands beyond the intended control logic runtime
- Escalate privileges by abusing operating system mechanisms
- Bypass vendor-defined security models implemented at the application layer

This focused approach enables a precise evaluation of whether existing industrial threat models adequately capture SoftPLC-specific risks and provides the foundation for the framework comparison and hybrid modeling proposal presented in later sections.

5. Findings

5.1 Analyzed Platforms

As part of Nozomi Networks Labs' ongoing security research on SoftPLC technologies, we conducted an in-depth analysis of several widely adopted SoftPLC implementations from leading vendors.

These implementations span both dedicated industrial devices and software platforms that can be deployed on standard PCs or industrial PCs. The objective of this research is to characterize the current security landscape of SoftPLC systems and to derive insights that are broadly applicable across different deployment models. The following section summarizes the platforms evaluated in this study and the attack scenarios identified during the analysis.

5.1.1 Phoenix Contact AXC F 3152 PLCNext

The AXC F 3152 is a high-performance industrial controller from Phoenix Contact's PLCnext Control series, designed for modular automation with Axioline F I/O. It runs a real-time Linux-based OS and supports both IEC 61131-3 programming and high-level languages like C++ and Python. Powered by a dual-core Intel Atom processor, it features multiple Gigabit Ethernet ports, PROFINET capabilities, and robust cybersecurity certifications (IEC 62443).

Figure 4 - Phoenix Contact AXC F 3152 PLCNext

5.1.2 Wago PFC 750-8216

The WAGO PFC200 (model 750-8216) is a second-generation compact industrial controller within WAGO's PFC200 series. It runs a Linux operating system and supports both IEC 61131-3 programming and higher-level IT protocols such as HTTP, FTP, SNMP, Modbus-TCP, MQTT, and Ethernet/IP. With its built-in web server, users can configure and monitor the controller remotely. It supports a wide array of digital, analog, and specialty I/O modules, making it ideal for industrial automation tasks across packaging, processing, manufacturing, and building automation environments.

5.1.3 Beckhoff C6015 with Windows TwinCAT

As a Windows-based SoftPLC platform running TwinCAT, the Beckhoff C6015 transforms into a fully capable real-time controller suitable for modern automation and edge computing environments. Equipped with an embedded Intel Atom CPU and with a full-fledged Windows 10 operating system, the C6015 offers enough performance for machine control, visualization, and communication tasks, all within tight spatial constraints. In essence, using TwinCAT on the C6015 allows engineers to deploy traditional PLC-style automation on a compact, flexible, and scalable PC platform, combining PLC functionality with the openness and extensibility of PC-based control in industrial settings.

Figure 5 - Wago PFC 750-8216

Figure 6 - Beckoff C6015 industrial PC

5.1.4 Beckhoff CX5130 with Unix BSD TwinCAT

The Beckhoff CX5130 BSD-based industrial PC, when combined with TwinCAT software, functions as a compact and reliable SoftPLC for industrial automation tasks. It offers powerful control capabilities in a small, fanless design that's ideal for control cabinets and space-limited environments. TwinCAT enables the CX5130 to handle everything from basic PLC logic to advanced motion control, making it suitable for a wide range of automation projects. With robust construction and versatile connectivity options, it's a practical choice for industries looking to combine traditional PLC functionality with the flexibility of PC-based control.

Figure 7 - Beckoff CX5130 industrial PC

5.1.5 CODESYS on Raspberry Pie

CODESYS is a widely used, hardware-independent automation software platform that enables the development and execution of control programs based on the IEC 61131-3 standard, which defines programming languages for PLCs. It provides a comprehensive development environment for building industrial automation solutions across various hardware platforms—from traditional PLCs to modern embedded systems. As part of our research, we explored CODESYS on a Raspberry Pi, a cost-effective and versatile single-board computer. By installing the CODESYS runtime on the Pi, we effectively transformed it into a fully functional SoftPLC, capable of handling real-time control tasks, managing I/O, and communicating over standard protocols such as Modbus TCP.

5.1.6 Bosch Rexroth ctrlX CORE

The Bosch Rexroth ctrlX CORE is a compact industrial automation controller. Built on a Linux-based architecture, it combines automation, IT, and IoT technologies in one device. Its app-based approach allows users to install, update, and customize functions, offering flexibility compared to traditional fixed-function PLCs. The platform is open, enabling developers to create or add third-party apps using container technology such as Docker. Designed with Industry 4.0 in mind, it supports cloud connectivity and AI applications.

Figure 8 - Bosch Rexroth ctrlX CORE based device

5.2 Finding List

Our investigation into SoftPLC security led to the identification of five distinct categories of issues, covering multiple vulnerabilities and misconfigurations observed across the platforms analyzed in this study. These findings are not isolated defects or incidental weaknesses; rather, they emerge from the expanded attack surface introduced by SoftPLC architectures when compared to traditional PLCs.

This classification forms the structural foundation of the white paper and provides the basis for a detailed analysis of the underlying security flaws. By grouping the findings in this way, we aim to clarify the broader threat landscape associated with SoftPLC deployments and to highlight how architectural design choices, such as the adoption of general-purpose operating systems, introduce classes of vulnerabilities that are uncommon in firmware-based PLC environments. Collectively, these issues underscore the potential risks faced by industrial systems that rely on SoftPLC technologies.

Here is the table with the provided content:

Finding	
1	Improper protection on local IPCs
2	Weak privileges on files
3	Trusted data tampering
4	Symbolic link abuse
5	Untrusted search path

In the following sections of this white paper, we shift from theory to practice, presenting real-world vulnerabilities uncovered during our research.

5.3 Finding Details

5.3.1 Improper protection on local IPCs

This class of vulnerability arises from insufficient access control and isolation applied to local inter-process communication (IPC) mechanisms used by privileged SoftPLC components. During our analysis, this issue was identified on Beckhoff TwinCAT platforms running on both BSD-based and Windows-based operating systems, indicating that the problem is architectural rather than platform-specific.

In the affected implementations, privileged TwinCAT services rely on local IPC channels to receive commands and requests from other system components (such as the web interface).

Figure 9 illustrates the architecture of the Windows-based TwinCAT platform, highlighting the communication flow between the unprivileged web interface and the privileged MDP system service. Through this design, remote users can authenticate to the web interface and, when logged in with sufficient privileges, request administrative operations, such as user management, which are executed by the privileged service.

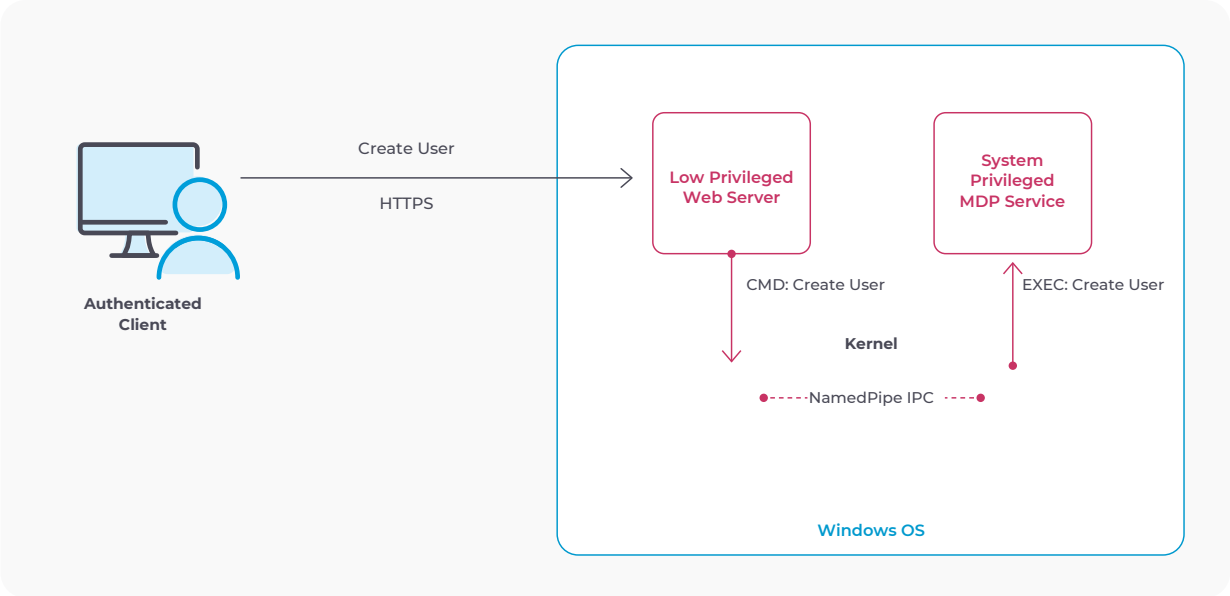


Figure 9 - IPC-based communication on Beckhoff TwinCAT for Windows

However, this IPC mechanism is not adequately protected, allowing low-privileged local users to interact with it (CVE-2025-41727). As a result, users who gain local access, either through a shell interface such as SSH (Windows and BSD) or through a graphical login (Windows only), can send crafted messages or commands directly to privileged processes. As illustrated in Figure 10, analysis of the MDP Windows executable through reverse engineering reveals the implementation details of the IPC mechanism, including how the `Windows Named Pipe` is created and the security descriptor applied to it.

```
CoInitialize(0);
ConvertStringSecurityDescriptorToSecurityDescriptorA(
  "D:(D;OICI;GA;;;BG)(D;OICI;GA;;;AN)(A;OICI;GRGWGX;;;AU)(A;OICI;GA;;;BA)(A;OICI;GA;;;LS)(A;OICI;GA;;;SY)",
  1u,
  &SecurityAttributes.lpSecurityDescriptor,
  0);
v1 = lpThreadParameter;
```

Figure 10 - Weak protection for NamedPipe IPC channel on MDP service

The extracted string security descriptor is

`D:(D;OICI;GA;;;BG)(D;OICI;GA;;;AN)(A;OICI;GRGWGX;;;AU)(A;OICI;GA;;;BA)(A;OICI;GA;;;LS)(A;OICI;GA;;;SY),`

which can be interpreted as follows:

- Full control is denied to Guests
- Full control is denied to Anonymous users
- Read, write, and execute permissions are granted to Authenticated Users
- Full control is granted to Administrators
- Full control is granted to Local Service
- Full control is granted to SYSTEM

Notably, granting read, write, and execute permissions to Authenticated Users allows any authenticated local account, including low-privileged users, to interact with the IPC channel. This configuration enables users with minimal privileges to send data and commands to the privileged process communicating over the IPC mechanism.

Because the receiving privileged MDP process implicitly trust the IPC requests, it may execute operations with elevated privileges on behalf of the attacker, without enforcing proper authentication or authorization checks. This effectively breaks the intended security boundary between low-privileged users and administrative-level operations within the SoftPLC environment.

The impact of this vulnerability is significant. By abusing the improperly protected IPC channels, an attacker can perform authentication bypass and privilege escalation, gaining administrative control over the SoftPLC system. One example of an exploitation scenario observed during the research involves instructing a privileged TwinCAT process to create a new administrator user with arbitrary credentials. Once the account is created, the attacker can simply authenticate using those credentials, achieving full administrative access to the system.

A typical attach chain for this issue is:

1. **Preparation of the malicious executable:** A low-privileged user compiles a script into an executable capable of interacting with the vulnerable IPC channel.
2. **Initial access with limited privileges:** The attacker logs into the target device using a low-privileged account, either through an SSH session or a Windows graphical login.
3. **Deployment from a low-privileged shell:** Using a low-privileged shell environment, such as SSH or PowerShell, the attacker uploads the compiled executable to the system.
4. **Execution and IPC abuse:** The executable is launched with low privileges and abuses the improperly protected IPC mechanism to request privileged operations from a trusted service.
5. **Creation of an administrative account:** As a result of the unauthorized IPC interaction, a new administrative user is created on the system.
6. **Privilege escalation and full access:** The attacker authenticates using the newly created account, achieving full administrative access to the SoftPLC platform.

The BSD-based TwinCAT implementation is affected by a closely related vulnerability, identified as CVE-2024-41173. While the underlying architectural issue is the same, the specific implementation differs due to the use of a different operating system.

5.3.2 Weak privileges on files

Weak privileges on files represent a vulnerability class that arises when sensitive files within a SoftPLC environment, such as configuration data, authentication material, logs, or diagnostic artifacts, are protected by overly permissive file system access controls.

Because SoftPLCs rely on full-featured operating systems and complex file systems, improper permission models can allow low-privileged users to read or manipulate data that should be restricted to administrators or privileged services.

Unlike traditional PLCs, where internal data structures are tightly constrained by firmware, SoftPLCs expose standard file systems that can be accessed through local shells or remote management interfaces, enabling attackers to bypass application-layer security controls by directly interacting with the underlying operating system.

A representative example of this vulnerability class is CVE-2025-41658, identified in CODESYS Control V3, which allows low-privileged users with access to the host system to read private authentication keys belonging to other users due to insufficient file access restrictions.

```
drwxr-xr-x 2 root root 4.0K Jan 24 02:38 .
drwxr-xr-x 5 root root 4.0K Oct  1 13:20 ..
-rw-r--r-- 1 root root 1.2K Jan 24 02:06 024db545a22e29237c1d8855cce0dd7c80ee30a9.key
-rw-r--r-- 1 root root 1.2K Oct  2 00:44 11d1d9c36cfa21a8b3ce182a4723f20c73cd80c7.key
-rw-r--r-- 1 root root 1.2K Jan 22 16:07 207dd73d486ca476a913ecb319da2cd045acd954.key
-rw-r--r-- 1 root root 1.2K Jan 23 01:52 58d22b5e3bba71c65b66199554af1ee004a714fe.key
-rw-r--r-- 1 root root 1.2K Jan 23 18:38 6d4cb6167c4c6858055468f9aa17707a61eac748.key
-rw-r--r-- 1 root root 1.2K Jan 22 07:02 bfb5aa47f8db4b3fbcb8b8c9029c7a19e64eb220.key
-rw-r--r-- 1 root root 1.8K Jan 24 02:38 cd3aff51a1938fad0c622685b348d46f1bdc847d.key
-rw-r--r-- 1 root root 1.2K Jan 22 08:36 d487c5f9533f21629e99fd7bdac1088021a139d1.key
-rw-r--r-- 1 root root 1.2K Jan 23 21:02 fb6d22f7184740a29a399dda457753ce87454a25.key
```

Figure 11 - Weak privileges for user private keys on CODESYS V3 for Raspberry

By obtaining these authentication artifacts, an attacker can impersonate other users, potentially achieving horizontal privilege escalation or, in cases where administrative credentials are exposed, vertical privilege escalation.

A second example is CVE-2024-41970, affecting multiple Wago PFC200 models as confirmed by the vendor, where weak file permissions allow unauthorized access to diagnostic data stored on the device. Depending on system configuration, this data may include plain network traffic captures or backup files, enabling attackers to extract administrative credentials transmitted over the network and further compromise the system.

Figure 12 highlights insufficient file protection applied to a diagnostic backup file, `log _ 2024-04-10 _ 12-36-03.pcapng`, located under a world readable directory. The file permissions `(-rw-r--r--)` allow read access to all users, enabling any authenticated local user to view its contents.

```
root@PFC200V3-6017EB:~ ls -l
-rw-r--r--    1 root     root      14661956 Apr 10 12:41 log_2024-04-10_12-36-
03.pcapng
```

Figure 12 - Weak privileges on backupd data on WAFO PFC200

This leads to a possible attack that allows a low privileged user to gain administrative access.

1. **Initial access with limited privileges:** A low-privileged user authenticates to the system using a restricted SSH shell.
2. **Extraction of diagnostic artifacts:** The attacker retrieves a diagnostic file that is insufficiently protected and accessible to low-privileged users.
3. **Offline analysis of captured data:** The extracted diagnostic dump is analyzed offline to inspect recorded network traffic.

Together, these scenarios demonstrate how weak file privilege models on complex file systems can undermine the intended security model of SoftPLC platforms, particularly when combined with the availability of full-fledged shells such as Bash over SSH. Such shells allow authenticated users, even with low privileges, to freely navigate the file system and execute complex commands, making it easier to discover and exploit sensitive files and ultimately bypass role-based security controls enforced at the application level.

5.3.3 Trusted Data Tampering

Trusted Data Tampering refers to a class of vulnerabilities in which data that is implicitly trusted by the system, such as configuration files, application resources, environment variables, or runtime artifacts, can be modified by an attacker in an unauthorized manner. Because this data is consumed by privileged components or system services without sufficient validation or integrity checks, tampering with it can directly influence privileged execution paths and lead to escalation of privileges or compromise of system integrity.

In SoftPLC environments, this vulnerability class is particularly relevant due to the presence of general-purpose operating systems, complex file systems, and extensible application frameworks. Unlike traditional PLCs, where trusted data is often embedded in firmware or protected by hardware-enforced boundaries, SoftPLCs rely on software-managed trust assumptions that can be broken if file permissions, integrity checks, or isolation mechanisms are insufficient.

In the following sections, we present two representative attacks that illustrate how Trusted Data Tampering can be abused in modern SoftPLC platforms.

The first case affects the Phoenix Contact PLCnext device family. Through CVE-2025-41669, we demonstrated that by modifying a third-party application installed on the device, it is possible to escalate an engineering-level user profile to root. This attack highlights the security cost of SoftPLC extensibility: while support for third-party applications increases flexibility and functionality, it also expands the trusted computing base. If third-party components or their associated data are insufficiently protected, they can become a vehicle for privilege escalation and compromise of the underlying operating system.

According to the device documentation, the Engineer role is intended to be a restricted user profile with a limited set of capabilities on the PLC device. However, this role is granted access to the PLCnext Apps section of the web-based management interface. Through this interface, an Engineer user is allowed to manually install applications that are normally distributed via the official PLCnext Store. This capability introduces a critical trust boundary, as it allows users with limited privileges to influence how third-party applications are deployed and executed on the system.

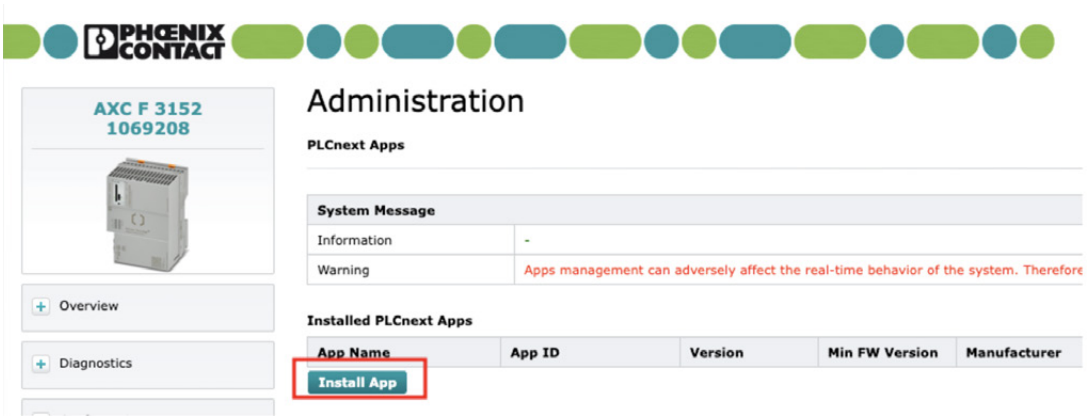


Figure 13 - Phoenix Contact PLCnext application installation feature on the web interface

The following steps describe how this functionality can be abused to achieve privilege escalation.

1. **Analysis of a legitimate third-party application:** An application downloaded from the official store (in this case, the Telegraf application for x64 CPUs) is analyzed, revealing that it is packaged as a filesystem image.

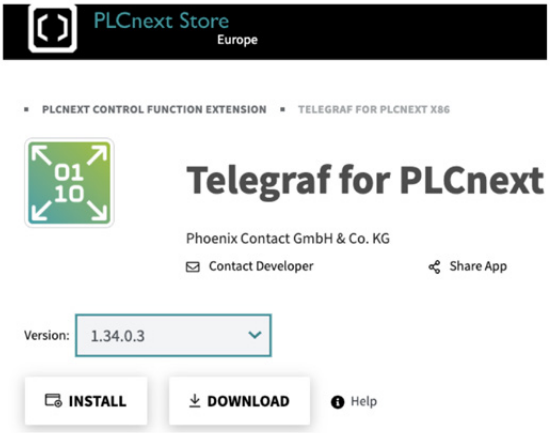


Figure 14 - Telegraf application download from the PLCnext Store

2. **Inspection of application metadata and execution logic:** By extracting the filesystem image, it is possible to examine the application contents, including the application configuration file, which defines how the application is launched on the device and specifies the main executable.

Figure 15 - Telegraf application internals

3. **Replacement of the trusted executable:** To assess the security impact, the original Telegraf executable is replaced with a malicious payload. For demonstration purposes, the payload simply executes the Linux id command to reveal the execution context.
4. **Repackaging of the modified application:** The filesystem image is rebuilt using the modified executable, producing a new application package that appears structurally valid.
5. **Installation of the modified application:** Using an Engineer-level session in the web-

based management interface, the modified application is successfully installed on the device without triggering validation or integrity checks.

6. **Verification of privilege escalation:** Analysis of the output file generated by the payload (e.g., `/tmp/exp`) shows that the malicious code executed with root privileges on the underlying Linux operating system.

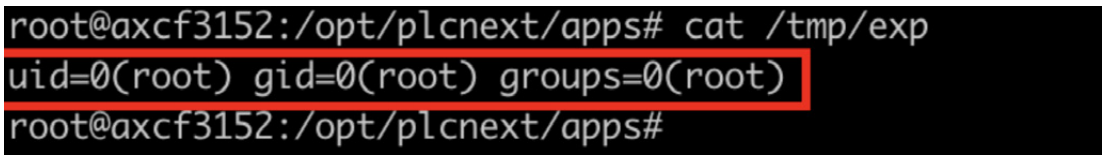


Figure 16 - Code execution with root privileges achieved

This attack demonstrates a privilege escalation from the Engineer role to root, enabled by insufficient protection of trusted application data and a lack of integrity validation during application installation.

The second case involves the Bosch Rexroth ctrlX CORE platform. In CVE-2025-24346, a low-privileged remote user is able to tamper with trusted elements of the Linux operating system environment, corrupting its execution context and ultimately achieving root-level access. This vulnerability demonstrates how the complexity of a full Linux-based SoftPLC environment introduces additional attack surfaces, where manipulation of trusted OS-level data can have system-wide security implications.

This attack is made possible by CVE-2025-24346, which allows a low-privileged user to tamper with the `/etc/environment` file on the target system. On Linux systems, this file defines environment variables that are applied globally to all running services. By manipulating this trusted configuration, an attacker can influence the execution context of privileged processes. In the analyzed Bosch Rexroth ctrlX CORE system, this weakness can be combined with the behavior of a privileged service, `snapd`, responsible for managing installed packages, which invokes the `systemctl` binary when listing applications. By corrupting the execution environment, a low-privileged user can hijack this execution flow and achieve privilege escalation.

The attack proceeds as follows:

1. **Initial access with limited privileges:** A low-privileged user authenticates to the Bosch Rexroth ctrlX CORE web interface using an account with restricted permissions.
2. **Abuse of backup restoration functionality:** The attacker exploits a legitimate feature that allows restoration of backups to upload an arbitrary file to the system. In this case, the uploaded file contains a legitimate backup together with a malicious script designed to execute the Linux id command to reveal the privilege context and redirecting its output to the file `/tmp/exp`. The backup file is named attack.zip.

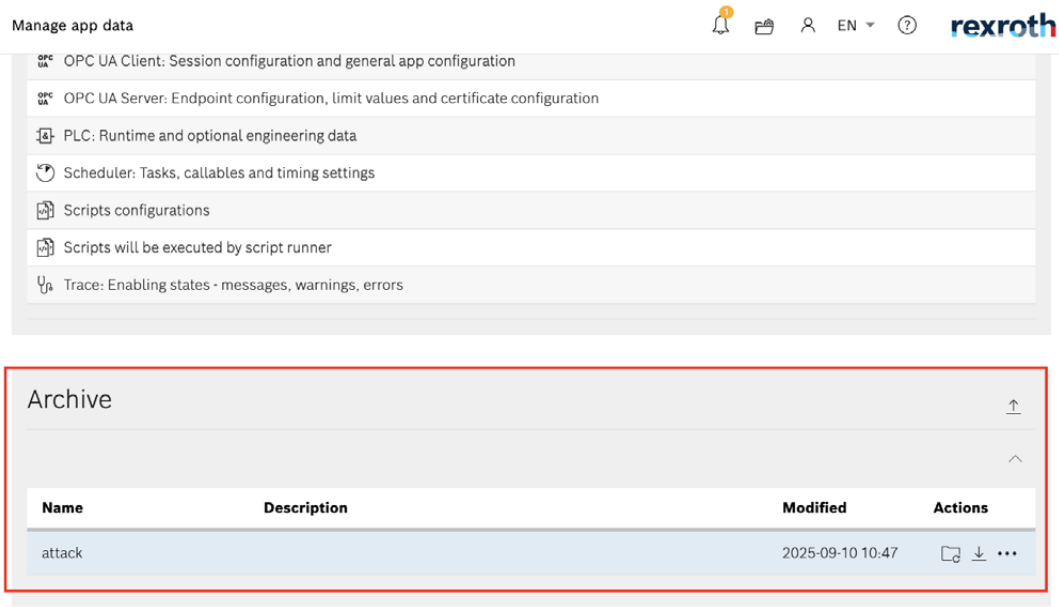


Figure 17 - Backup restore on **Bosch Rexroth ctrlX CORE**

3. **Environment manipulation via /etc/environment:** Leveraging CVE-2025-24346, the attacker modifies the `/etc/environment` file to prepend the controlled directory containing the malicious binary to the system `PATH` environment variable. The exploit operates by sending a crafted HTTP request containing a malicious newline character, which disrupts request parsing and enables the attacker to control the contents written to the target file. Figure 18 shows the resulting modification of the `/etc/environment` file.

```
root@ctrlX-CORE:/# cat /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin"
http_proxy=http://user:p4ssw0rd%21@192.168.1.101:8080/
https_proxy=http://user:p4ssw0rd%21@192.168.1.101
PATH="/writable/system-data/var/snap/rexroth-solutions/common/solutions/DefaultSolution/configurations/appdata:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:
/usr/games:/usr/local/games:/snap/bin"
:8080/root@ctrlX-CORE:/#
root@ctrlX-CORE:/#
```

Figure 18 - File /etc/environment corrupted

4. **System reboot to apply the corrupted environment:** The device is rebooted so that the modified `/etc/environment` file is applied globally to all services at startup.
5. **Execution flow hijacking of a privileged service:** After reboot, the privileged `snapd` process starts with the attacker-controlled `PATH`. When the web interface requests a listing of installed applications, `snapd` attempts to invoke systemctl.
6. **Execution of attacker-controlled code as root:** Due to the manipulated `PATH`, `snapd` resolves `systemctl` from the attacker-controlled directory and executes the malicious payload with root privileges. By examining the contents of the newly generated `/tmp/exp` file, it is possible to confirm that arbitrary code was executed with root privileges, as illustrated in Figure 19.

```
root@ctrlX-CORE:/#
root@ctrlX-CORE:/# ls -l /tmp/exp
-rw-r--r-- 1 root root 663 Sep 10 11:31 /tmp/exp
root@ctrlX-CORE:/#
root@ctrlX-CORE:/#
root@ctrlX-CORE:/# cat /tmp/exp | head -n1
uid=0(root) gid=0(root) groups=0(root)
root@ctrlX-CORE:/#
root@ctrlX-CORE:/#
```

Figure 19 - Root code execution achieved

This attack demonstrates how a low-privileged remote user can corrupt trusted operating system configuration to hijack the execution flow of a root-running service. By abusing both OS-level complexity and trusted data assumptions, the attacker achieves arbitrary code execution as root, resulting in a full privilege escalation on the SoftPLC platform.

Together, these vulnerabilities underscore why Trusted Data Tampering is a critical risk in SoftPLC platforms. On one hand, they illustrate the security challenges introduced by extensibility and third-party application support; on the other, they show how the inherent complexity of general-purpose operating systems can be abused to break fundamental trust assumptions. Both cases reinforce the need for stronger integrity controls, strict privilege separation, and careful validation of trusted data in software-defined industrial control systems.

5.3.4 Symbolic Link Abuse

Abuse of symbolic links is a class of operating system–level attacks in which an attacker leverages symbolic links to redirect file operations performed by a privileged process toward unintended targets. When a privileged application does not correctly validate file paths or protect against symbolic link resolution, an attacker can trick it into reading from or writing to sensitive files elsewhere in the file system. This attack technique is particularly effective on Unix-like operating systems, where symbolic links are a native filesystem feature and where privileged services often operate on paths supplied by less trusted components.

To demonstrate how symbolic link abuse becomes a practical and necessary attack primitive in a SoftPLC environment, we analyzed a chain of two vulnerabilities affecting the Wago PFC200 device, which is based on a Linux operating system and uses the CODESYS stack to implement SoftPLC functionality. These vulnerabilities allow a limited-privileged CODESYS user, such as one assigned an Engineer role, to exploit path traversal conditions inside the CODESYS command linte interface (Figure 20) in order to ultimately achieve arbitrary file write to `/etc/shadow`, resulting in the creation of a valid `root` account on the device.

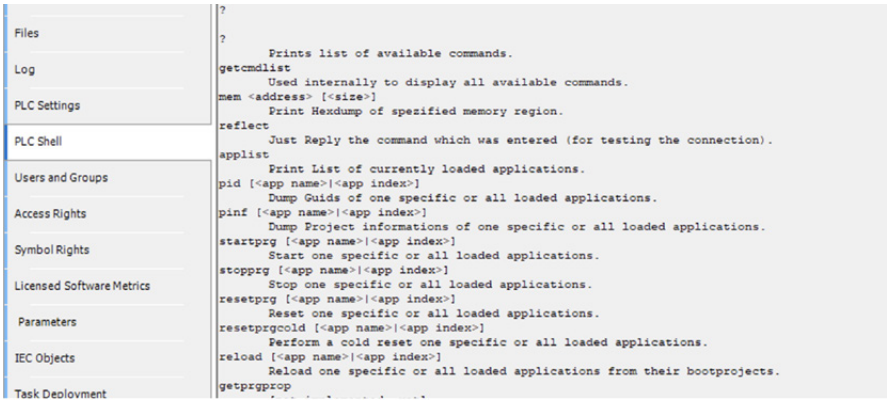


Figure 20 - CODESYS command line interface

It is important to note that the path traversal vulnerabilities alone are not sufficient to directly access sensitive system files. In both cases, the affected CODESYS commands automatically append a fixed file extension to the user-supplied path, preventing direct specification of critical files such as `/etc/passwd` or `/etc/shadow`. The abuse of symbolic links is therefore a mandatory step to bypass this constraint and redirect file operations to arbitrary targets.

While remote exploitation of path traversal issues in the CODESYS application stack is not, by itself, particularly impactful from a SoftPLC security perspective, the attack becomes feasible because the Wago PFC200 provides SSH access to low-privileged users. Although restricted in scope, this shell is fully functional and allows interaction with the Linux filesystem, including the creation of symbolic links. This combination of a full-fledged shell and insufficient path validation transforms otherwise constrained vulnerabilities into a real and complete attack chain.

CVE-2024-41972 affects the CODESYS command interface command `restoresram`, which allows a user to specify a backup file path in order to restore a memory section of the device.

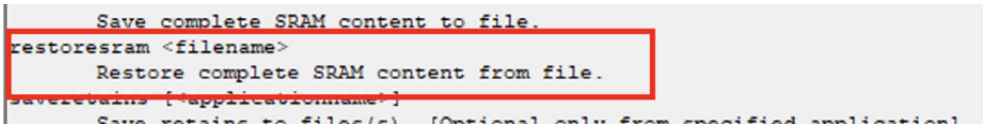


Figure 21 - CODESYS `restoresram` command

Internally, the CODESYS backend service, running with `root` privileges, search for backup files in a dedicated protected folder and automatically appends a `.ret` extension to the specified filename then reading the corresponding file.

A path traversal vulnerability allows an attacker to escape the intended backup protected directory and reference files in other locations. By combining this with symbolic link abuse, the attacker can bypass the enforced file extension. Specifically, a low-privileged user can create a symbolic link in a writable directory that points to `/etc/shadow`, using a filename that ends with `.ret`.

When the `restoresram` command is invoked with a path traversal payload referencing the malicious created file, the backend service follows the symbolic

link and reads the contents of `/etc/shadow`, loading them into memory and displaying them through the PLC command-line interface.

This allows the attacker to retrieve user accounts and password hashes (Figure 22), representing a critical information disclosure.

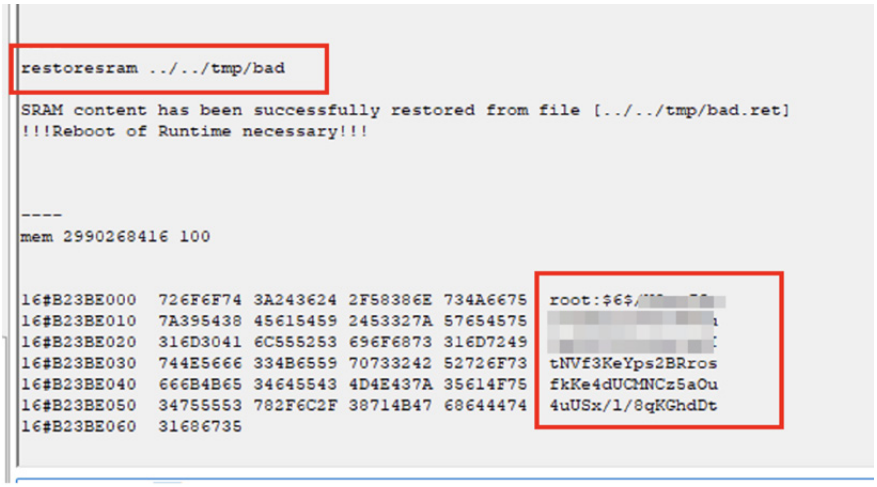


Figure 22 - Retrieving `/etc/shadow` content through CVE-2024-41972 exploitation

CVE-2024-41973 affects the `savesram` command (Figure 23), which allows saving a memory section into a backup file. As with `restoresram`, the backend service appends a fixed extension to the user-supplied filename and writes the resulting file to a protected folder into the disk, executing this operation with `root` privileges. This design prevents direct overwriting of sensitive system files.

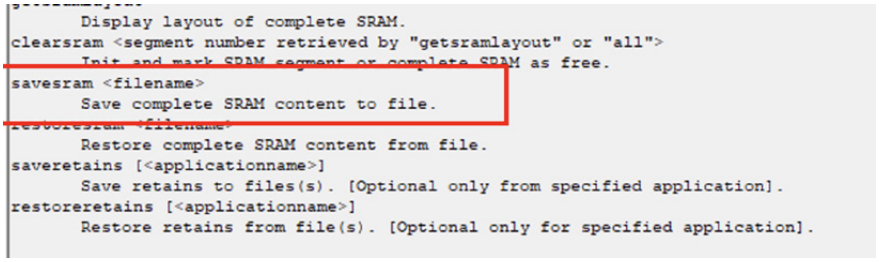

Display layout of complete SRAM.
clearsram <segment number retrieved by "getsramlayout" or "all">
savesram <filename>
Save complete SRAM content to file.
Restore complete SRAM content from file.
saveretains [<applicationname>]
Save retains to files(s). [Optional only from specified application].
restoreretains [<applicationname>]
Restore retains from file(s). [Optional only for specified application].

Figure 23 - CODESYS `savesram` command

Once again, a path traversal vulnerability enables the attacker to specify an arbitrary directory, but symbolic link abuse is required to bypass the enforced filename extension. By creating a symbolic link in a writable directory that points to `/etc/shadow` and has a name compatible with the expected extension, the attacker can induce the privileged backend service to overwrite `/etc/shadow` when the `savesram` command is executed.

By chaining these two vulnerabilities and leveraging symbolic links, an attacker can achieve a full privilege escalation:

1. Prepare a crafted shadow file in memory containing attacker-controlled credentials.
2. Exploit CVE-2024-41972 to load attacker-controlled data into memory by abusing path traversal and symbolic links to read arbitrary files.

```
restoresram ../../tmp/bad

SRAM content has been successfully restored from file [../../tmp/bad.ret]
!!!Reboot of Runtime necessary!!!


----
mem 2990268416 100


16#B23BE000  726F6F74 3A243624 61626331 3233245A  root:$6$abc123$Z
16#B23BE010  7036474A 74314B51 364A6D5A 3279306C  p6GJt1KQ6JmZ2y0l
16#B23BE020  51713958 762E2E2E 3A313935 30303A30  Qq9Xv...:19500:0
16#B23BE030  3A393939 39393A37 3A3A3A0A 373A3A3A  :99999:7:::.7:::
16#B23BE040  0A6B4B65 34645543 4D4E437A 35614F75  .kKe4dUCMNCz5aOu
16#B23BE050  34755553 782F6C2F 38714B47 68644474  4uUSx/l/8qKGhdDt
16#B23BE060  31686735
```

Figure 24 - Attacker controlled data loaded in memory

3. Exploit CVE-2024-41973 by abusing path traversal and symbolic links to overwrite `/etc/shadow` with the crafted content.

```
----
savesram ../../tmp/bad

SRAM content has been successfully saved to file [../../tmp/bad.ret]
```

Figure 25 - CVE-2024-41973: Path traversal exploitation

4. Authenticates to the system using the newly created root credentials.

```
~ sshpass -p 'root' ssh root@192.168.

WAGO Linux Terminal on PFC200V3-6017EB.

root@PFC200V3-6017EB:~
root@PFC200V3-6017EB:~
```

Figure 26 - Attacker authentication

This attack chain illustrates how symbolic link abuse is a critical enabler that converts limited application-layer vulnerabilities into a complete system compromise. It also highlights how SoftPLC platforms inherit powerful filesystem semantics from general-purpose operating systems, where subtle interactions between path handling, file extensions, and symbolic links can undermine the intended security model.

5.3.5 Untrusted Search Path

An Untrusted Search Path attack occurs when a privileged process relies on external files, paths, or execution context elements that can be influenced by a less privileged user. If a privileged service loads configuration files, binaries, or runtime parameters from locations that are writable or partially controllable by non-administrative users, an attacker can manipulate those inputs to alter the behavior of the privileged process. This class of attack does not require memory corruption or code injection; instead, it exploits implicit trust in execution paths and runtime dependencies, often leading to privilege abuse, unauthorized actions, or system instability.

This attack pattern is particularly relevant in SoftPLC environments, where general-purpose operating systems introduce complex service orchestration, configuration files, and watchdog mechanisms that are uncommon in traditional PLC architectures. When access controls on these components are insufficient, low-privileged users may indirectly influence privileged execution flows, undermining vendor-defined security models.

CVE-2025-41667 affecting the Phoenix Contact PLCnext AXC F 3152, it is presented to illustrate this attack class in a SoftPLC context. According to the documented security model of the PLCnext platform, non-administrative users are explicitly not allowed to reboot the device (Figure 27). However, this vulnerability allows a low-privileged user to violate this restriction by modifying the behavior of a privileged system component through local access.

Rights	plcnext group	admin
Setting and inspecting IP settings (including `ifconfig`, `ping`, `netstat`, etc.)	√	√
Configuring the firewall	√	√
Starting/stopping the firewall (`init` script)	√	√
Inspecting the firewall with `nft`	√	√
Configuring VPN (IPsec and OpenVPN™)	–	√
Starting/stopping VPN services (IPsec and OpenVPN™)	–	√
Editing the *Default* PLCnext folder for individual ACF, ESM, GDS configurations, and **.so*	√	√
Starting/stopping the PLCnext Technology firmware processes with `sudo /etc/init.d/plcnext start|stop|restart`	–	√
Reading PLCnext log files	√	√
Calling and configuring TOP/HTOP	√	√
Firmware update via update script with `sudo update-plcnext` –	–	√
Configuring the NTP server	√	√
Setting the root password with `passwd`	–	√
Requesting the system time with `date`	√	√
Setting the system time with `sudo date -s`	√	√
Restarting/shutting down the controller with `reboot` or `shutdown`	–	√

Figure 27 - Phoenix Contact PLCnext AXC F 3152 user manual

By analyzing the privileged daemons running on the underlying Linux operating system, it is possible to observe that a watchdog process is executed automatically at boot time. This process relies on a configuration file located at `/opt/plcnext/data/System/Watchdog/WatchdogDaemon.config`, as shown in the referenced Figure 28.

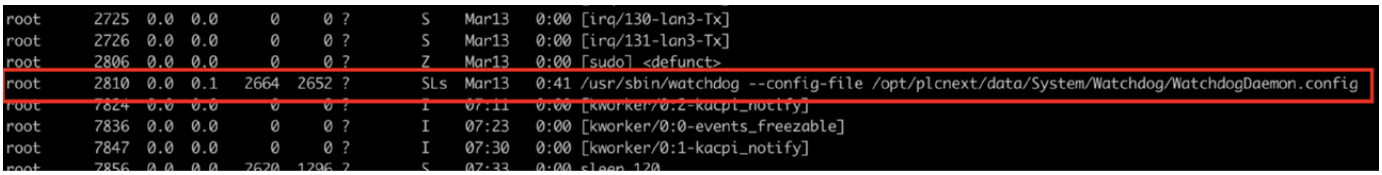


Figure 28 - Privileged `watchdog` process

Inspection of this configuration file reveals that the watchdog daemon monitors the health of three processes, each identified by a process ID (PID) stored in a dedicated file (Figure 29).

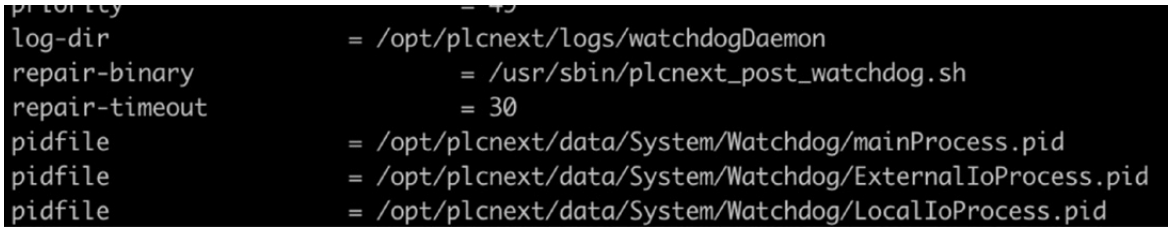


Figure 29 - Privileged `watchdog` configuration file

When any of the monitored processes terminates unexpectedly, the watchdog daemon invokes the recovery script `plcnext_post_watchdog.sh`, which generates log entries and initiates a system reboot as part of the recovery procedure. Notably, the files storing the PIDs of the monitored processes are located under the `/opt/plcnext` directory, which is writable by members of the Linux group `plcnext`. This directory contains tools intended for non-administrative users. As a consequence, a low-privileged user who is a member of this group can modify the PID files (Figure 30) and influence which processes are monitored by the watchdog daemon.

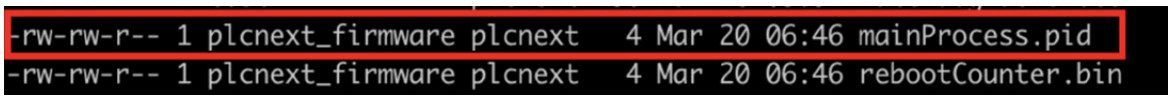


Figure 30 - `mainProcess.pid` file under the control of `plcnext` group

A user belonging to the plcnext group, with SSH access to the system, can abuse this condition by redirecting the watchdog mechanism to monitor a process under their control. By terminating this process, the attacker can trigger an unintended system reboot. In this way, a low-privileged user can indirectly cause a reboot of the PLC, bypassing the explicit security restriction enforced by the platform's role-based access model.

It is worth noting that this attack can be repeatedly triggered, causing continuous reboots of the device. In such a scenario, the vulnerability can be escalated from a policy bypass to a denial-of-service condition, effectively preventing normal operation of the SoftPLC.

5.4 Findings Evaluation

The analysis conducted across the selected SoftPLC platforms revealed a consistent and noteworthy outcome: every platform exhibited at least one weakness capable of undermining the security model proposed by the vendor. While the specific vulnerabilities and exploitation techniques varied from platform to platform, no analyzed system demonstrated a security posture that could be considered fully robust against the attack scenarios explored in this research.

It is important to emphasize that this result does not indicate the presence of a single vulnerability or flaw common to all devices. Instead, the findings highlight a systemic challenge inherent in securing SoftPLC platforms. The vulnerabilities identified arise from different components and mechanisms—such as inter-process communication, file system permissions, trusted data handling, and execution flow control—but collectively they demonstrate the difficulty of effectively hardening a complex and heterogeneous attack surface. This complexity is a direct consequence of adopting general-purpose operating systems, extensible software architectures, and rich management interfaces within industrial control environments.

A second key observation emerged during the evaluation of the findings against threat modeling frameworks. While the research initially focused on Execution and Privilege Escalation as defined in the MITRE ATT&CK for ICS framework, this mapping proved to be insufficient to fully describe the observed attack behaviors. Many of the exploitation techniques leveraged in this research involved abusing local IPC channels, advanced file system features (such as weak permissions and symbolic links), and hijacking of privileged processes, behaviors that extend beyond the abstraction level typically captured by the ICS framework.

These techniques align more naturally with the MITRE ATT&CK for Enterprise framework, which was designed to model attacker behavior in systems built on general-purpose operating systems. This overlap reflects the inherited IT characteristics of SoftPLC platforms and reinforces the notion that SoftPLCs occupy a hybrid space between traditional ICS devices and enterprise systems. As a result, modeling their security risks exclusively through an ICS-specific lens can leave critical attacker behaviors insufficiently represented.

This observation motivates the framework comparison and hybrid modeling discussion presented in the next section, where we analyze in greater detail how SoftPLC attack scenarios bridge the gap between ICS and enterprise threat models.

6. Research Scope and Methodology

The mapping of observed SoftPLC attack scenarios against the MITRE ATT&CK for ICS framework revealed a consistent and structural gap in coverage. While the ICS framework accurately models the final stages of industrial attacks, such as manipulation of control logic, inhibit response functions, and operational impact it provides limited representation of the early and intermediate stages that are critical in SoftPLC compromise.

In contrast, many of the attack behaviors observed during this research align closely with techniques defined in the MITRE ATT&CK for Enterprise framework. These behaviors occur before any direct interaction with industrial processes, yet they are essential enablers for subsequent ICS-specific impact.

6.1 Observed Coverage Gap

Several attack scenarios identified during the analysis could not be fully or accurately mapped to ICS ATT&CK techniques, despite mapping cleanly to Enterprise ATT&CK. This gap is particularly evident in the Execution and Privilege Escalation tactics, which are central to SoftPLC compromise.

Affected Product	Exploited CVE	Execution	Privilege Escalation
Wago CC100, Edge Controller, PFC100, PFC200, TP600	CVE-2024-4197	Command and Scripting Interpreter	Valid Accounts
	CVE-2024-41971 CVE-2024-41972 CVE-2024-41973	Command and Scripting Interpreter	Hijack Execution Flow
Beckhoff TwinCAT and TwinCAT/BSD	CVE-2024-41173 CVE-2025-41727	Inter-Process Communication	Abuse Elevation Control Mechanism
Bosch Rexroth ctrlX Core OS	CVE-2025-24346	Execution through API	Hijack Execution Flow
Phoenix Contact PLCnext AXC F 1152, AXC F 2152, AXC F 3152, RFC 4072S, BPC 9102S	CVE-2025-41669	Software Deployment Tools	Create or Modify System Process
	CVE-2025-41667 CVE-2025-41665	System Services	Valid Accounts
CODESYS Control V3	CVE-2025-41658 CVE-2025-41660	Command and Scripting Interpreter	Valid Accounts

These techniques are well defined and extensively documented in the Enterprise ATT&CK framework, as they represent common OS-level attacker behaviors in IT environments. However, they are either absent or only abstractly represented in the ICS framework, which does not explicitly model operating system internals or local execution chains.

6.2 Root Cause of the Gap

This discrepancy is not the result of incomplete framework design, but rather of historical architectural assumptions. The ICS ATT&CK framework was developed with traditional PLCs and ICS devices in mind, systems that typically:

- Lack general-purpose operating systems
- Do not expose local execution environments
- Have limited or no OS-level privilege separation

SoftPLC platforms violate these assumptions by design. By embedding full operating systems into industrial controllers, SoftPLCs inherit execution models, privilege hierarchies, and system services that are indistinguishable from those found in enterprise environments.

As a result, SoftPLC attack chains often resemble enterprise compromise paths occurring within OT assets, with OS-level execution and privilege escalation acting as prerequisites for industrial impact.

6.3 Proposed Hybrid ATT&CK Approach

To accurately model these attack chains, this research proposes a hybrid threat modeling approach that extends the ICS ATT&CK framework with selected Enterprise ATT&CK techniques where SoftPLC architectures expose equivalent capabilities.

Under this approach:

- Enterprise ATT&CK techniques are used to describe:
  - OS-level execution
  - Privilege escalation
  - Process and service manipulation on SoftPLC platforms
- ICS ATT&CK techniques remain authoritative for:
  - Process manipulation
  - Safety impact
  - Inhibit response functions
  - Physical and operational consequences

The hybrid model explicitly represents the transition point between IT-style compromise and OT-specific impact, rather than treating it as an implicit assumption. This enables more accurate threat modeling, detection engineering, and defensive planning for modern SoftPLC-based environments.

The following table illustrates this hybrid approach by extending the MITRE ATT&CK for ICS framework with selected techniques from the Enterprise ATT&CK framework that were identified during this research.

Execution	Privilege Escalation
Autorun Image	Exploitation for Privilege Escalation
Change Operating Mode	Hooking
Command-Line Interface	Abuse Elevation Control Mechanism
Execution through API	Create or Modify System Process
Graphical User Interface	Hijack Execution Flow
Hooking	Valid Account
Modify Controller Tasking	
Native API Scripting	
User Execution	
Command and Scripting Interpreter	
Inter-Process Communication	
Software Deployment Tools	
System Services	

Importantly, this proposal does not advocate for a full merger of the two frameworks. Instead, it promotes contextual integration, preserving the strengths of both models while addressing the architectural realities introduced by SoftPLCs.

6.4 Implications for Industrial Security Frameworks

While the MITRE ATT&CK for ICS framework effectively models industrial impact, safety-relevant actions, and process manipulation, it provides limited representation of execution and privilege escalation behaviors that serve as critical enablers in SoftPLC attack chains. These stages often occur before any observable impact on industrial processes and are essential for attackers to transition from initial access to full system compromise.

This gap has several practical implications for organizations relying exclusively on ICS-focused threat models:

- Threat modeling activities may overlook early-stage attacker behavior, particularly OS-level execution and privilege escalation paths that occur prior to manipulation of control logic or physical processes. As a result, important phases of the attack lifecycle may remain unmodeled or underestimated.
- Detection strategies may focus too narrowly on process impact and operational anomalies, missing indicators of compromise that manifest at the operating system or application level. This can delay detection until the attacker has already achieved significant control over the system.
- Defensive controls may fail to address OS-level compromise paths, leaving SoftPLC platforms vulnerable to attacks that bypass application-layer protections and role-based security models. In such cases, security measures designed solely around traditional PLC assumptions may provide a false sense of protection.

Addressing these challenges requires acknowledging that modern ICS asset, particularly SoftPLC platform, are increasingly operating as hybrid IT/OT systems. Effective security strategies and threat models must therefore account for both industrial process impact and enterprise-style attacker behaviors to accurately reflect the realities of contemporary industrial environments.

6.5 Future Research Directions

While this study focused specifically on Execution and Privilege Escalation, the findings suggest that similar modeling gaps may exist across other phases of the ATT&CK lifecycle when applied to SoftPLC platforms. The architectural characteristics that enable OS-level execution and privilege escalation, such as general-purpose operating systems, dynamic services, and extensible application frameworks, are also likely to affect additional stages of attacker behavior.

In particular, future research may extend this work to include:

- Persistence techniques, such as service installation, scheduled task creation, or modification of startup mechanisms, which could allow attackers to maintain long-term access to SoftPLC systems even after reboots or software updates.
- Defense evasion behaviors, including log tampering, manipulation or disabling of security controls, and abuse of trusted system components to evade detection. These behaviors are especially relevant in environments where monitoring and forensic visibility at the OS level is limited or inconsistent.

- Malware deployment and lifecycle analysis within SoftPLC environments, examining how malicious code could be introduced, executed, updated, and removed on software-defined controllers. This includes analyzing how malware interacts with control logic, system services, and industrial processes over time.

These research directions are particularly relevant as SoftPLC platforms increasingly support third-party applications, containerized workloads, and remote update mechanisms. As industrial controllers continue to adopt software deployment models similar to enterprise systems, understanding how persistence, evasion, and malware behaviors manifest in SoftPLC environments will be critical to evolving both defensive strategies and industrial threat modeling frameworks.

7. Key Takeaways

This research demonstrates that SoftPLC platforms fundamentally alter the security landscape of industrial control systems. By introducing general-purpose operating systems into PLC environments, SoftPLCs significantly expand the attack surface and enable attacker behaviors that were historically uncommon, or entirely absent, in traditional PLC deployments.

The findings show that low-privileged access, when combined with operating system–level weaknesses, is sufficient to bypass vendor security models and achieve full system compromise across multiple SoftPLC platforms. These results highlight a systemic issue rooted in architectural design rather than isolated implementation flaws.

7.1 Key Takeaways for End Users

SoftPLC platforms introduce new operational and security considerations compared to traditional PLCs due to their reliance on general-purpose operating systems and expanded connectivity. The following recommendations summarize practical actions that end users can take to reinforce the security posture of SoftPLC deployments and mitigate risks highlighted by this research:

- **Restrict physical and logical access to SoftPLC platforms:** Enforce strong authentication, role-based access control, and strict governance of remote and local access paths.
- **Minimize exposed services and interfaces:** Disable non-essential services, limit management interfaces to trusted networks, and regularly review system configurations.
- **Treat SoftPLCs as hybrid IT/OT assets:** Acknowledge that SoftPLCs inherit OS-level risks and require security controls beyond those traditionally applied to PLCs.
- **Adopt selective IT security controls where feasible: C**onsider host-based IDS, antivirus, integrity monitoring, or application whitelisting, ensuring compatibility with real-time and operational constraints.
- **Improve visibility into OS-level activity:** Monitor execution, privilege escalation, and system changes to detect threats before they impact industrial processes.

7.2 Key Takeaways for Vendors

SoftPLC vendors should recognize that existing industrial threat modeling approaches, including the MITRE ATT&CK for ICS framework, were originally designed around traditional PLC architectures and may not fully capture the security implications introduced by software-defined control platforms. As SoftPLCs increasingly incorporate general-purpose operating systems and IT-like execution environments, relying exclusively on ICS-specific threat models can lead to blind spots in security design and risk assessment.

Vendors are encouraged to adopt a hybrid threat modeling approach that combines ICS-specific techniques with relevant enterprise-grade execution and privilege escalation behaviors. Integrating concepts from enterprise threat models during product design and security validation can help

identify OS-level attack paths that bypass application-layer controls and undermine role-based security models.

By proactively embracing this hybrid perspective, vendors can:

- Design more resilient security architectures aligned with real-world attacker behavior
- Improve internal security testing and validation processes
- Anticipate emerging threats before they are observed in the wild
- Contribute more effectively to the evolution of industrial security frameworks

Recognizing and addressing these gaps early is a critical step toward building secure, future-proof SoftPLC platforms that reflect the realities of modern industrial environments.

8. Conclusion

As industrial automation continues to evolve toward software-defined and IT-integrated architectures, the assumptions that have historically shaped industrial security models are increasingly challenged. SoftPLC platforms exemplify this shift by embedding general-purpose operating systems and IT-like execution environments directly into industrial control assets. As a result, treating SoftPLCs as traditional PLCs, both from a defensive and a threat-modeling perspective, is no longer sufficient to address the risks introduced by these modern architectures.

This research advocates for a hybrid IT/OT security approach that acknowledges the dual nature of SoftPLC systems. Such an approach combines enterprise-grade operating system security practices—such as OS hardening, privilege separation, and host-level monitoring—with industrial-specific threat modeling that accounts for process impact, safety requirements, and operational constraints. By bridging these domains, defenders can more accurately model attacker behavior, detect early-stage compromise, and prevent escalation paths that lead to industrial impact.

Adopting this hybrid perspective provides a realistic and effective foundation for securing modern ICS environments and offers a practical path forward for the evolution of industrial security frameworks. As SoftPLC adoption continues to grow, aligning security models with the architectural realities of these systems will be essential to maintaining the safety, reliability, and resilience of future industrial operations.

NOZOMI NETWORKS

Cybersecurity for OT, IoT and Critical Infrastructure

Nozomi Networks protects the world's critical infrastructure from cyber threats. Our platform uniquely combines network and endpoint visibility, threat detection, and AI-powered analysis for faster, more effective incident response. Customers rely on us to minimize risk and complexity while maximizing operational resilience.

nozominetworks.com

NN-WP-S4-SOFTPLC-A4-001