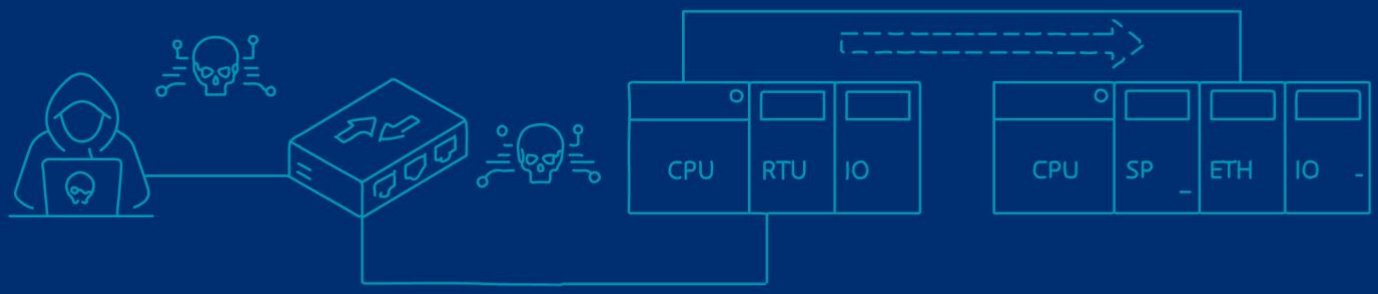
VEDERE LABS



DEEP LATERAL MOVEMENT IN OT NETWORKS

When Is a Perimeter Not a Perimeter?

```
root@jumpbox:/home/administrator# python3 2_object_plc_exploit.py --host 192.168.1.17 --route 10.10.1.6 --hijack --client_
10dbd50a70507708f041ee27a01df8da5f97a969d102e5741e1c00eb3 --reservation_id 96
[CVE-2022-45789, CVE-2022-45788] Schneider Electric Modicon - UMAS Auth bypass & RCE
[*] Connecting ...
[*] Reading PLC identity ...
[i] Model Name: BMX P34 20302
[i] Firmware version: 350
[*] Forging authenticated requests using hijacked nonces ...
[*] Starting CSA session ...
[*] Sending stager payload ...
[*] Injecting code ...
[i] Initiating write-copy sequence
[*] CSA block bytes write operation ...
[*] Executing CSA operational page ...
[*] CSA block bytes copy operation ...
[*] Executing CSA operational page ...
[*] Activating stager payload ...
[*] Injecting code ...
[i] Initiating write-copy sequence
[*] CSA block bytes write operation ...
[*] Executing CSA operational page ...
[*] CSA block bytes copy operation ...
[*] Executing CSA operational page ...
[+] Implant activated
[*] Disconnecting
root@jumpbox:/home/administrator#
```

Jos Wetzels

February 13, 2023

Contents

1. Executive summary

This research report is the first systematic study into how attackers can move laterally between different network segments and types of networks at the controller level – Purdue level 1 (L1) – of OT networks. We show how attackers can cross security perimeters in interfaced Basic Process Control Systems (BPCS) / Safety Instrumented Systems (SIS) architectures or perform detailed manipulation of equipment in fieldbus networks nested behind PLCs to bypass functional and safety constraints that would otherwise prohibit cyber-physical attacks with the most serious consequences.

As part of the proof-of-concept developed for this research, we use two new vulnerabilities that we are publicly disclosing for the first time: CVE-2022-45788 and CVE-2022-45789 allowing for remote code execution and authentication bypass, respectively, on Schneider Electric Modicon PLCs. These issues were found as part of the OT:ICEFALL research in 2022 but were not disclosed at the time at the request of the vendor.

In the past few years, security researchers have demonstrated various approaches to obtaining low-level remote code execution (RCE) on L1 devices from various vendors. Malware such as TRITON and INCONTROLLER have shown that real-world threat actors are both capable of and interested in developing such capabilities as well. When it comes to subsequent post-exploitation of L1 devices however, prior work has primarily focused on stealth, persistence, and demonstrating impact, while lateral movement has received little attention. The focus for lateral movement in the past has been on moving between L1 devices in the same network segment or moving upstream to SCADA systems at level 2 and above but has not considered moving deeper into nested devices networks or across perimeters at level 1.

As a result, asset owners frequently overlook security perimeters at level 1 and consider the kind of granular control required to bypass functional and safety limitations enforced by controllers and field devices as infeasible. This is despite the fact that L1 devices that sit at the intersection of multiple, mixed networks should be treated as security perimeters and ought to have the corresponding hardening and risk profiles that would be accorded to workstations in a similar position.

In this report, we present:

- Two new vulnerabilities affecting Schneider Electric Modicon PLCs and allowing for remote code execution and authentication bypass (Section □).
- An overview of lateral movement on level 1, including different real-world BPCS/SIS architectures and third-party package unit setups, relevant lateral movement options and related attacker use-cases (Sections 3 and 4).
- A realistic attack scenario on critical infrastructure where lateral movement on level 1 allows an attacker to cause physical damage to a movable bridge (Section 5).
- An in-depth discussion and demonstration of an L1 RCE and lateral movement proof-of-concept using previously undisclosed authentication bypass and RCE vulnerabilities against fully patched Schneider Electric M340 & M580 PLCs (Section 6).
- Our conclusions and thoughts on hardening L1 devices and networks against the discussed threats (Section 7).

2. Summary of new vulnerabilities

As part of this research, we are publicly disclosing two new vulnerabilities affecting Schneider Electric Modicon *Unity* PLCs.

The Modicon family of PLCs is one of the most popular in the world, used in several critical infrastructure sectors, and traces its roots to 1968, with the Modicon 084 being the first PLC in the world. This product family can be organized into several distinct (informal) groups based on runtime internals and engineering protocols (M171/M172 HVAC PLCs excluded) as shown in Table 1. Broadly speaking, the Machine Expert PLCs are used

for machine automation purposes while the Control Expert PLCs being used for process automation, with the M340 and M580 being the *Unity* line's most prominent offerings.

Table 1 – Schneider Electric Modicon PLCs

Group	PLCs	Primary engineering protocol
Machine Expert Basic	M221	Machine Expert Basic dialect
CODESYS-based (EcoStruxure Machine Expert)	M238, M241, M251, M258, M262	CODESYS
Unity (EcoStruxure Control Expert)	M340 (BMXP34*), M580 (BMEP*, BMEH*), M580 Safety (BMEP58*S, BMEH58*S), MC80 (BMKC80), Momentum Unity M1E (171CBU*), Quantum Unity (140CPU65*), Premium Unity (TSXP57*)	UMAS
Concept (Schneider Concept)	Quantum, Premium	Concept

The uncovered issues, summarized in Table 2, only affect the *Unity* PLC line and were discovered as part of the OT:ICEFALL research in 2022 but were not disclosed at the time, at the request of the vendor. CVE-2022-45788 is an example of RCE via an undocumented memory write operation (described in Section 6.3), while CVE-2022-45788 exemplifies a broken authentication scheme (described in Section 6.2). More details about these issues are available on Schneider Electric's advisories SEVD-2023-010-05 and SEVD-2023-010-06. As we show in Section 6 of this report, as part of the proof-of-concept for L1 lateral movement, when combined these vulnerabilities can lead to remote code execution on Modicon *Unity* PLCs.

It should be noted that while Schneider Electric describes CVE-2022-45788 as relating to downloading malicious project files, this vulnerability actually operates on a completely different – undocumented – set of functionalities that allows for modifying internal PLC memory without affecting the PLC run state or requiring a project download.

Table 2 – New Vulnerabilities

CVE ID	Affected Products & Versions	Description	CVSS	Potential Impact
CVE-2022-45788	**EcoStruxure Control Expert** – *All versions* **EcoStruxure Process Expert** – *Version V2020 and prior* **Modicon *Unity* PLCs** (BMXP34*, BMEP*, BMEH*, BMEP58*S, BMEH58*S, 171CBU*, BMKC80, 140CPU65*, TSXP57*) – *All versions*	A vulnerability exists that could cause arbitrary code execution, denial of service and loss of confidentiality & integrity when undocumented Modbus UMAS CSA commands (service code 0x50) are executed. See **SEVD-2023-010-05**.	7.5	RCE
CVE-2022-45789	**EcoStruxure Control Expert** – *All versions* **EcoStruxure Process Expert** – *Version V2020 and prior* **Modicon *Unity* PLCs**	A vulnerability exists that could cause execution of unauthorized Modbus functions on the controller when hijacking an authenticated Modbus session.	8.1	Auth Bypass

	(BMXP34*, BMEP*, BMEH*, BMEP58*S, BMEH58*S) – *All versions*	See **SEVD-2023-010-06**.		

2.1. Popularity of Modicon PLCs

Modicon PLCs are used in a wide range of industrial processes and critical infrastructure, including in industries such as water and wastewater, mining, manufacturing, and energy.

Estimating the number of affected devices based on public data is difficult since reliable market share data is hard to come by. Regardless, Schneider Electric's Modicon PLCs are known to be highly popular in specific sectors like the ones mentioned above as well as regions like EMEA. Several publicly available surveys consistently indicate Schneider Electric is among the top 4 leaders of the PLC industry.

Another data point comes from internet-connected PLCs. Despite the fact that this practice has been widely discouraged for a long time, a non-negligible number of PLCs continues to be publicly connected to the internet as shown by Shodan. Figure 1 shows that France (33%), Spain (17%), Italy (15%), and the United States (6%) are the countries with the most exposed devices. Querying Shodan for all relevant Modicon *Unity* models yields over 1,000 exposed PLCs, the vast majority of which do not appear to be trivial honeypots.

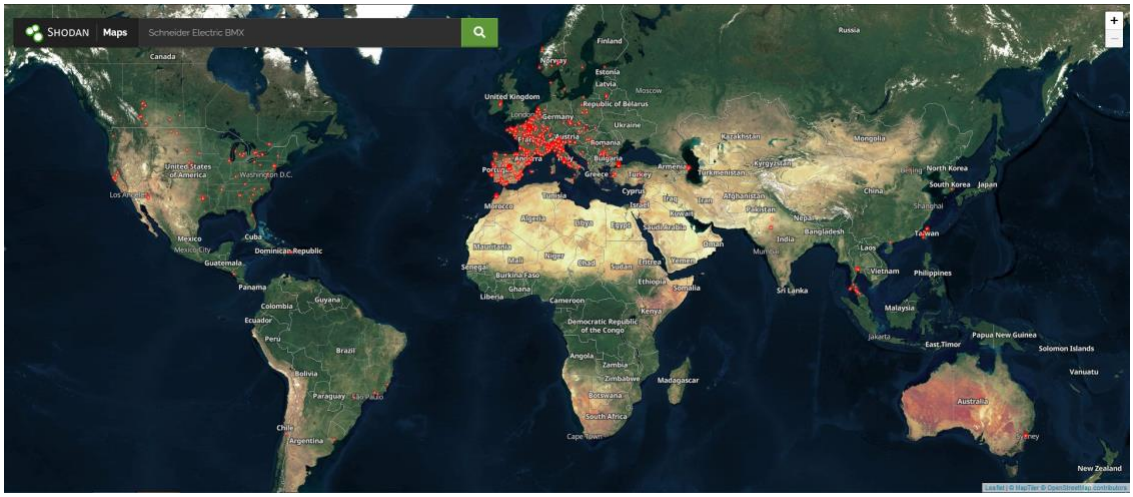


Figure 1 – Modicon PLCs exposed online across the world

The number of devices visible on Shodan is just a small indication of the popularity of these PLCs, but these devices also highlight some of the critical facilities that rely on Modicon PLCs. A quick search for the affected models on Shodan has shown exposed Modicon PLCs in everything from airports, mining, and solar and hydro power generation to chemical manufacturing. For instance, Figure 2 shows one of the exposed M340 PLCs with a project loaded for a "*impianto di frantumazione*", the Italian term for a crushing plant used in mining operations. Smaller power generation facilities (often owned by local communities) seem particularly overrepresented in the set of exposed PLCs.

```
Unit ID: 0
-- Device Identification: Schneider Electric BMX P34 2020 v2.9
-- CPU module: BMX P34 2020
-- Memory card: BMXRMS008MP
-- Project information: [redacted] Impianto di frantumazione [redacted]
-- Project revision: 2.6.225
-- Project last modified: 2022-01-27 16:02:43

Unit ID: 1
-- Device Identification: Schneider Electric BMX P34 2020 v2.9

Unit ID: 255
-- Device Identification: Schneider Electric BMX P34 2020 v2.9
```

Figure 2 – Example of exposed PLC connected to a crushing plant

Another interesting observation was that we found *multiple* instances of **public subnets**, likely used by system integrators or contractors, exposing Modicon PLCs for different power generation projects. In each instance, project naming conventions (sometimes including full filepaths and workstation names) clearly indicated these PLCs were programmed and commissioned by the same party. Such indicators might provide fruitful targeting for opportunists.

We mapped close to 30 of the exposed devices to actual critical infrastructure organizations based on open-source intelligence and got the following distribution per sector.

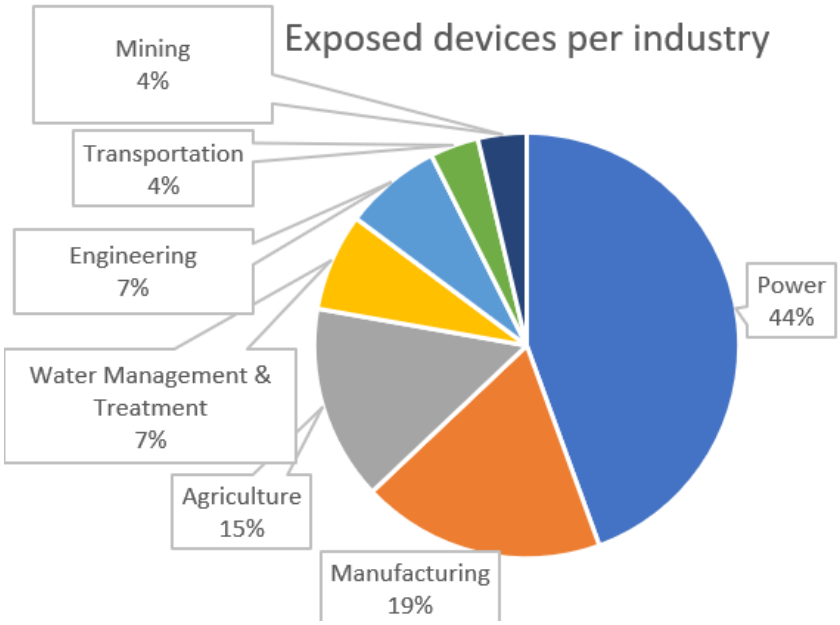


Figure 3 – Exposed Modicon PLCs per industry

Almost half of the power industry findings are related to a set of solar power parks built by a single system integrator in a European country. The others are mostly hydroelectric power plants in the US, again part of a single subnet. The manufacturing organizations are mostly in the chemical sector.

The popularity of Modicon PLCs led them to be the subject of previous security research [1] [2] [3] [4] [5] [6] [7] [8] and also to be targeted by threat actors. As a part of the recent wave of hacktivist attacks targeting OT, the

GhostSec group has targeted an exposed M340 belonging to the Nicaraguan ISP UFINET by writing the value 0 to all its Modbus registers, while the Pro-Ukrainian Team OneFist claims to have caused a fire in a Russian industrial site by targeting M258 PLCs.

2.2. Disclosure and mitigation

We disclosed CVE-2022-45788 and CVE-2022-45789 to Schneider Electric in April and July, respectively, of 2022. Advisories were released this January. As discussed in Section 7.3, for the authentication bypass we suggested a fix at the protocol level using a protocol such as Secure Remote Password for mutual authentication and then a key derivation function plus freshness to feed into an HMAC based authentication of messages. We believe this could be retrofitted into the existing protocol structure.

Regarding mitigations for these specific vulnerabilities, we refer to Section 7.3.

3. Deep lateral movement on level 1

3.1. Background

Operational technology (OT) is “hardware and software that detects or causes a change, through the direct monitoring and/or control of industrial equipment, assets, processes and events.” This includes industrial control systems (ICS). OT/ICS systems rely on a variety of specialized embedded devices such as programmable logic controllers (PLCs) and remote terminal units (RTUs), which are directly connected to sensors/actuators and implement control loops; as well as more traditional computers with specialized software that act as human machine interfaces (HMI), allowing operators to graphically see and effect changes on the industrial process; data historians, which store time-stamped data and events collected from the process; or engineering workstations, which allow operators to program the field devices.

It is traditional to discuss the equipment described above as part of the Purdue Enterprise Reference Architecture, which is a reference model for the different levels of enterprise integration. Figure 4 depicts the types of systems in each level of the Purdue model: level 0 contains sensors and actuators connected to the physical world; level 1 contains the devices that control the physical process (such as PLCs and RTUs); level 2 contains the SCADA and HMI systems used by humans to monitor and control the process; level 3 contains the historian and other operation management software; level 4 contains business-related devices and software, such as enterprise resource planning (ERP) or financial systems; finally, level 5 contains the remaining enterprise network, with other IT servers. Levels 0-2 are replicated in several cell/area network zones within an organization’s site (such as production lines), level 3 tends to be present across several sites within an organization (such as manufacturing plants), while the upper levels are typically part of an organization’s global network.

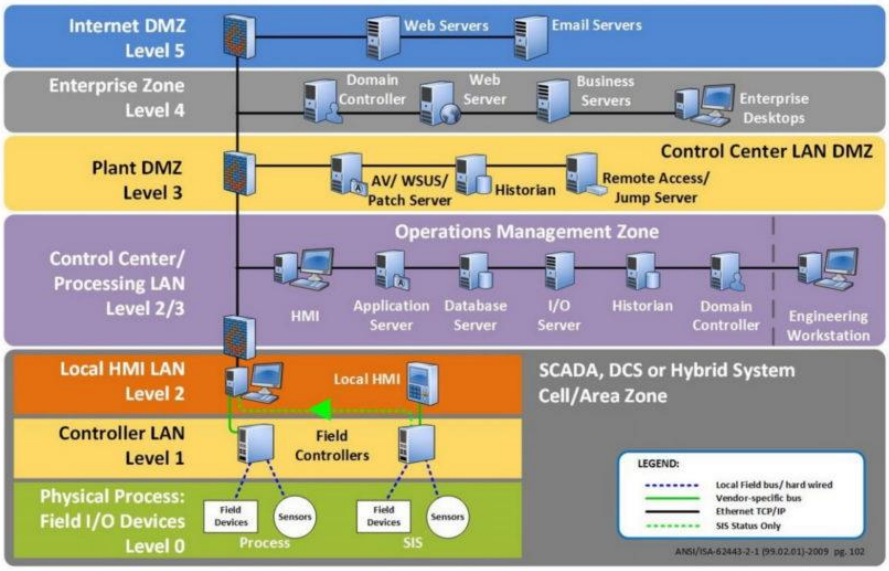


Figure 4 – Purdue Enterprise Reference Architecture – from ANSI/ISA-62443-2-1

It is well known that OT security, especially OT device security, has continued to lag behind IT for years as recently emphasized in Forescout's OT:ICEFALL research. As a result, OT security has traditionally put heavy emphasis on perimeter hardening, both at the outer perimeter (especially the DMZ between IT and OT typically located at Purdue level 3.5) and the different security segment perimeters such as segmentation between the basic process control system (BPCS) and safety instrumented systems (SIS) or east-west segmentation between different production cells or remote sites typically located at Purdue level 2.5.

However, level 1 devices also frequently sit at the intersection of multiple, mixed networks which sometimes have different security profiles. As shown in Figure 5, these links may consist of connections to industrial wireless networks (such as WirelessHART and ISA-100.11a), serial links to WAN radio modems (e.g. LTE, TETRA, DMR, or proprietary 900 MHz FHSS), nested fieldbus networks, and serial or Ethernet point-to-point links to safety systems and third-party Packaged Units (PU).

Packaged Units (PU): A PU is a black-box control system with a specific function (such as HVAC, chemical injection, water treatment, or gas turbine) that can range from a specific subsystem to an entire plant 'plugged' into an asset owner's wider control system. PUs typically expose only a limited control- and/or monitoring interface to the asset owner's Process Control Network (PCN) or SCADA via protocols like Modbus and OPC. This interface consists of a few process variables and setpoints that allow the PU to be integrated into the wider control system at a high level, but do not allow direct or detailed control over the PU's internals. In some cases, additional restrictions on PLCs and network interfaces inside the PU are in place for reasons of Intellectual Property (IP) protection.

Commissioning, integration, and maintenance of PUs is often done by a third party who may have remote access to the PU's internals through a cellular modem, indirectly exposing the asset owner's control networks to an external perimeter through the PU interface.

When a level 1 device is directly connected to networks with different security profiles, it is effectively a *perimeter device* and ought to be accorded the corresponding protection profile that would be accorded to a multi-homed workstation or server in its position. After all, an attacker compromising the device through any of its links could use this foothold to move laterally into the other networks. This is especially a concern when one of the connected networks can be accessed without crossing the traditional perimeters at levels 2.5 and 3.5 where most OT security hardening is currently focused. Examples of such networks are externally managed PUs, wireless WAN networks, or geographically remote unmanned locations offering easy physical access allowing upstream infiltration opportunities. The rise of the Industrial Internet of Things (IIoT) and corresponding "*Purdue model compression*" where level 1 devices are connected to cloud platforms has also presented new opportunities for a kind of downstream infiltration where level 1 device hardening becomes crucial.

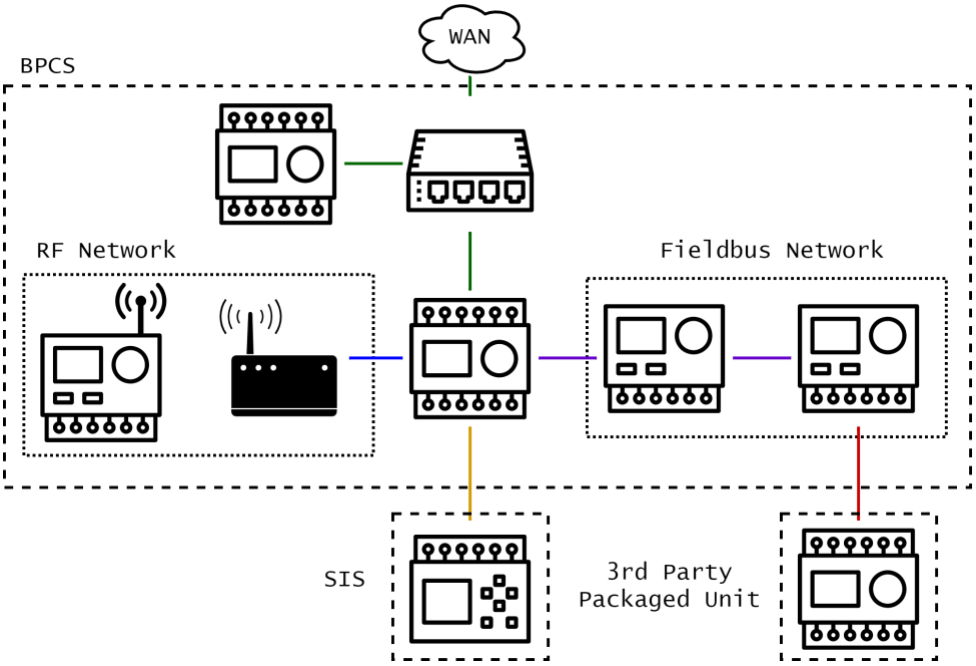


Figure 5 – Level 1 device at the intersection of multiple networks

In the past few years, remote code execution (RCE) on L1 devices has been demonstrated against many vendors using techniques such as insecure engineering interfaces, malicious logic or firmware downloads, and memory corruption vulnerabilities (such as the ones in Project Memoria). Many post-exploitation tactics, techniques and procedures (TTPs) have also been demonstrated for L1 devices, such as: TA0110 – Persistence [9] [10] [11] [12] [13], TA0111 – Privilege escalation [10], TA0103 – Evasion [10] [14], TA0101 – Command and Control [15], TA0010 - Exfiltration [16] [17], and OT payloads (TA0106 – Impair Process Control + TA0107 – Inhibit Response Function) [9] [11] [18] [19] [20].

Notably, there is little prior work on lateral movement for L1 devices. The existing prior work has mostly focused on moving from L1 to identical devices on the same segment (in the form of worms) [11] [21] or upstream hacking to L2 and above (i.e. from RTU/PLC/etc to HMI/EWS/etc) [14] [22] [23] [24] [25]. By contrast, downstream or arbitrary east/west movement from and *through* L1 devices to reach the kind of links discussed above, something we will refer to as ***deep lateral movement*** in this research, has received almost no public attention. The limited related work has focused on conventional protocol routing and proxy forwarding in order to reach nested networks. The reported ability of the INCONTROLLER malware to route Modbus and EtherCAT traffic makes this subject even more interesting.

In this research, we have focused on two main reasons for *deep lateral movement* at level 1: *perimeter crossing* and *granular control*. In Sections 3.2 and 3.3, we explore each of these reasons in detail. The main reason for including this kind of lateral movement in your attacker calculus is to reevaluate how one looks at perimeters. Firstly, we hope to draw attention to the common fallacy of *1st order connectivity analysis*, where risk assessments only take impact on directly reachable systems or components into consideration. Secondly, we hope to similarly draw attention to the fact that many OT system architects and integrators, as well as some vendor and regulatory guidance, continue to evaluate link security in terms of native routability and explicitly present capabilities and thus consider certain links (serial, point-to-point, non-routable) more robust than they are.

3.2. Perimeter crossing

As discussed above, attackers may need to move around hardened or across unacknowledged perimeters at level 1 to cross into different network segments. One example of zone-crossing at level 1 is the interaction between the BPCS and the SIS. Another, commonly underestimated, level 1 perimeter in OT are connections to third-party control systems (such as PUs or inter-utility interfacing) regulated by fieldbus couplers. Below, we explore these two perimeters in detail.

3.2.1. BPCS/SIS architectures

Based on standards and industry expert discussion [26] [27] [28] [29] [30] [31] we can broadly delineate several types of BPCS/SIS architectures as shown in Figure 6. While historically fully separate (air-gapped) architectures dominated, these have become increasingly rare making way primarily for the *integrated* and *interfaced* architectures.

In the ***integrated*** architecture, communications between the BPCS and SIS go through firewall-based segmentation, typically at level 2, and safety systems have dedicated engineering workstations and HMIs. In the ***interfaced*** architecture, communications between the BPCS and SIS happen at level 1 and go through a restricted link between a BPCS controller and a SIS controller. This restricted link is used to exchange limited SIS status and process values, trip events, (pre-)alarm triggers, interlock resets, and bypasses. At a technical level, this restricted link is typically a point-to-point serial or Ethernet connection between a single BPCS and SIS controller or occasionally a multi-drop serial or nested Ethernet subnet connection between a single BPCS controller and multiple SIS controllers. Since this restricted link constitutes a BPCS/SIS perimeter it is typically set up with restricted capabilities and isolated from the wider control network, for instance a non-routable Modbus interface with limited pre-defined variables.

Less widespread approaches include the ***common*** and ***shared*** architectures. In the ***common*** architecture, which is supported by an increasing number of vendors, BPCS and SIS controllers share a single backplane effectively rendering the (typically insecure-by-design) backplane bus as a security perimeter. In the ***shared*** architecture, which is discouraged to avoid common failure modes, BPCS and SIS controllers share access to some of the same field instruments. If such field instruments are ‘smart’ and hooked up to fieldbuses on both ends they effectively constitute a potentially overlooked conduit between the BPCS and SIS.

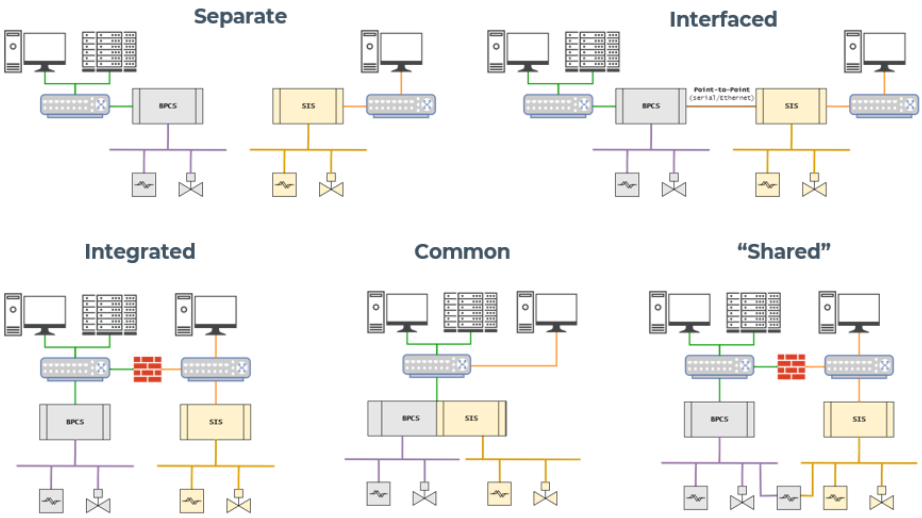


Figure 6 – BPCS/SIS Architectures

SIS Bypasses: An SIS bypass allows engineers to bypass part of the SIS (such as a single sensor or actuator or an entire Safety Instrumented Function (SIF)) as part of process startup, state transition, or maintenance operations to avoid spurious trips. SIS bypasses are typically implemented as a function block within an SIF running as part of the safety task in the SIS logic solver. As long as a bypass is active, the corresponding SIS component will not contribute to triggering its associated safety trip(s).

As shown in Figure 7, SIS bypasses can sometimes be activated from the BPCS, with the activation signal sent over the BPCS-SIS connection link. However, before activation a bypass needs to be enabled first. Historically, activation and enabling were done through hardwired physical switches, but increasingly this is done through software signals. In some cases, a hybrid approach is taken were bypass enabling is done through a hardware switch connected to an SIS remote IO while activation is done through a software signal sent through BPCS faceplate. It is considered good practice to have activation timeouts as well as limits on the number of simultaneously active bypasses.

An attacker seeking to achieve cyber-physical impacts beyond process disruption will likely need to cross from the BPCS into the SIS in order to inhibit response functions, as demonstrated by real-world incidents such as the TRITON attack. One often overlooked vector for response function inhibition that can sometimes be triggered from the BPCS itself is abusing so-called SIS bypasses. However, in order to enable these and circumvent timeout and quantity limits, an attacker will likely need to move deeper into the SIS first in order to spoof enabling signals or manipulate safety logic.

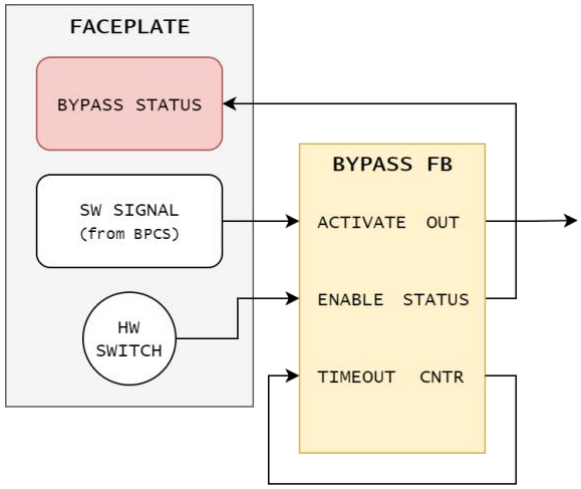


Figure 7 – SIS bypasses

3.2.2. Third-party fieldbus coupler interfaces

Fieldbus couplers are essentially very limited gateways which move specific sets of IO values between two different fieldbuses as shown in Figure 8 and Figure 9. Couplers can be used to connect different networks to each other, such as PROFIBUS DP to PROFINET, but are also used to offer a limited interface to a third-party system like a PU or when two different municipal infrastructure utility systems must talk to each other. The fieldbus coupler acts as a slave device and is configured to exchange a limited number of pre-defined IO values with a master device on one interface. These are then internally mapped to a different set of pre-defined IO values on the other interface which are exchanged with the master on the other side. This effectively allows the two masters to interact through a limited, pre-defined interface. For example, as shown in Figure 9, a master PLC in the BPCS could talk to a (remotely maintained) third-party PU through a coupler. The master PLC can read status values or cause the PU to start or stop, but cannot control the PLCs, valves, or motors inside the PU. Similarly, the vendor only has remote access to the PU's internal network but can't directly interact with the asset owner's BPCS.

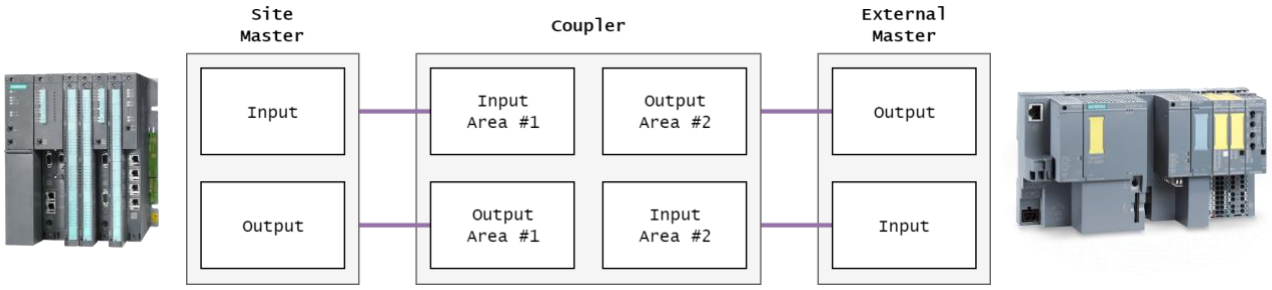


Figure 8 – Example of a fieldbus coupler

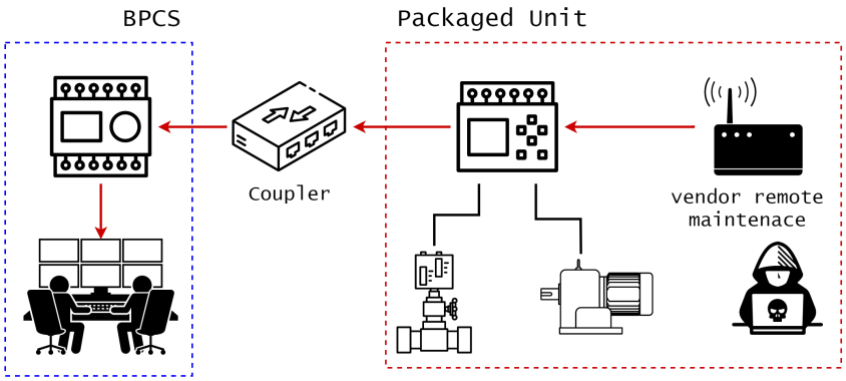


Figure 9 – Fieldbus coupler connecting a BPCS and a packaged unit

Couplers used to be “dumb” devices with static MCU or ASIC logic handling the IO mapping and thus had very limited attack surface. As such, they have been effectively used as perimeter devices with no additional security controls since they only expose a few variables with limited impact deeply in the OT system.

But increasingly these devices have turned smart, have complicated protocol conversion capabilities and in-band firmware updates. We have encountered several real-world cases of asset owners who had designed security architectures based on assumptions about the “dumb” nature of fieldbus couplers which turned out to actually be “smart” device with a far larger attack surface and potential for lateral movement than expected.

3.3. Granular control

The second reason why an attacker might want to move deeply into level 1 systems is because they need a very *granular kind of control over nested devices* or *bypass functional and safety constraints*.

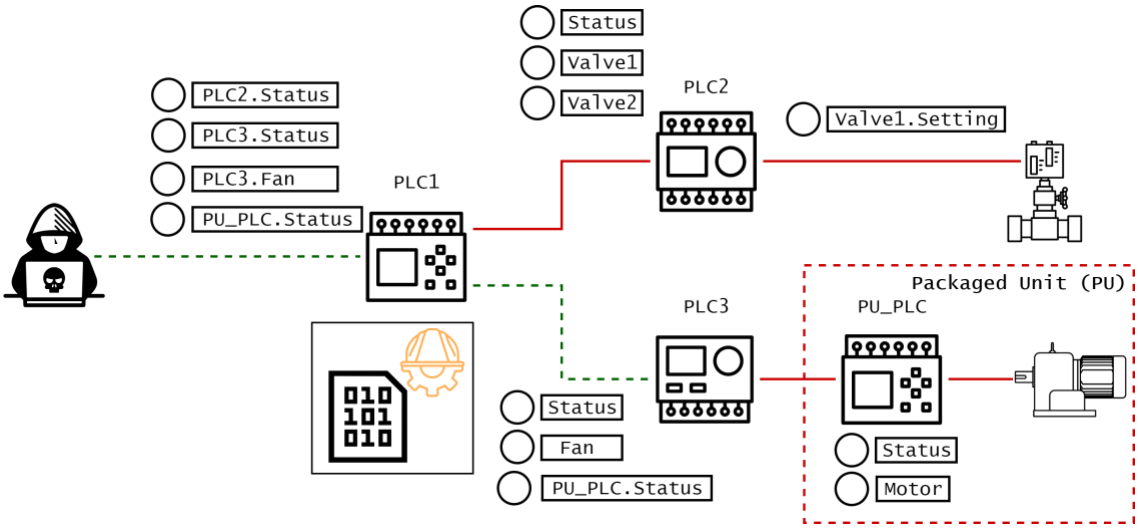


Figure 10 – Example architecture with nested systems

3.3.1. Granular access to nested devices

In many of the more complex industrial architectures, it is common for the PLCs in level 1 *control networks* which communicate with level 2 systems to be master PLCs connected to multiple nested Ethernet or fieldbus *device networks* consisting of slave PLCs and remote IOs, possibly with multiple levels of such nesting. As illustrated in Figure 10, nested devices like *PLC2* and *PLC3* expose their parameters, including a limited subset of their field devices’ parameters, to the master *PLC1*. This master PLC in turn exposes only a subset of those parameters to the level 2 network, giving an attacker interacting with it limited abilities to manipulate the control system which might be insufficient for achieving their desired cyber-physical impact.

For example, if the attacker in Figure 10 wished to change settings for valve1 or manipulate the motor of the PU, they could not achieve this from their current position in the network. Instead, they would need to have direct access to the associated interfaces on nested devices *PLC2* and *PLC3*, likely requiring code execution on *PLC1* first.

While nested devices can often be reached through routable protocols such as CIP or S7comm (as discussed in Section 4.1), nested networks sometimes are not fully routable due to employing non-routable or restricted links at some point as shown in Figure 10 where only the green links are routable. In addition, even though a nested device might be reachable through a routable protocol, this might still mean maintenance interfaces over other protocols are unreachable. Finally, attackers might not be able to get arbitrary or malformed packets required for certain attacks routed for proper delivery.

3.3.2. Bypass functional and safety constraints

Simply because an attacker can communicate with a PLC or field device, this does not mean the device will immediately do everything the attacker instructs them to. An attacker might want to manipulate variable-frequency drive (VFD) skip frequency settings, sensor calibration and signal reporting, or valve closing speed in a manner impossible through regular functionality.

For example, there might be internal safety limits on setpoints and configuration parameters. These sanity checks could be enforced at PLC logic level, as suggested in the PLC Secure Coding Practices Top 20, but can also be enforced in device firmware in a manner only configurable through local serial ports or physical display and control interfaces. To override such checks, an attacker will need code execution on the device itself.

Alternatively, consider the kind of communication modules frequently used on smart solar inverters and smart UPS devices. Typically, these communication modules allow for status monitoring and maybe scheduling disconnects or altering triggering/alert policies. An attacker seeking to achieve a more damaging scenario (such as output voltage and frequency manipulation, grid desynchronization, or bypassing current spike edge case protection) will need to obtain code execution on the responsible digital signals processor (DSP) of the solar inverter or UPS main module. In order to obtain this access, the attacker will first need to move laterally through the communication module and possibly through the main module's Application Processor (AP).

This kind of *deep lateral movement* has implications for evaluating the potential impact of individual vulnerabilities. Consider an RCE vulnerability on a communications module. Evaluated in isolation one might consider the impact limited to T0815 – denial of view or T0827 – loss of control since the module's direct functionality is restricted to communications. But when one thinks of devices themselves as essentially being networks (organized around backplane buses, inter-chip buses, on-chip interconnects, etc.), it becomes evident that this vulnerability might constitute a foothold allowing an attacker to pivot to parts of the system where far more catastrophic impacts (such as T0879 – damage to property or T0837 – loss of protection) become possible.

3.4. Vendors and standards guidance

A review of several standards (such as IEC 61508, NIST SP 800-82, HSE OG-0086) and vendor guidance [32] [33] [34] [35] [36] shows a general acceptance of the *integrated, interfaced,* and *common* BPCS/SIS architectures over the increasingly rare *separated* architecture.

With respect to *interfaced* architectures, point-to-point communication channels carrying non-routable traffic between level 1 devices seem generally considered "secure conduits." Some guidance explicitly refers to such channels as being a sufficient means of segmentation by themselves while others mention the need for firewalls on the Ethernet variants. Interestingly, none of the standards and vendor guidance discusses hardening serial links despite their very similar susceptibility to exploitation as discussed in Section 4.

The security of nested *device networks* is typically reduced to hardening of the master PLC, with little to no attention to possible external perimeters *within* those device networks. In addition, while the *common* architecture ought to result in a shared security zone between BPCS and SIS, this is not always emphasized. When control and safety functionality exist in separate modules on the same backplane, or as separate tasks within the same CPU module, no discussion is found on the backplane or intra-CPU module buses effectively constituting a security perimeter. In some cases, vendors seem to consider the fact that normal system functionality does not allow unfettered access to a safety PLC CPU module through a communication module as sufficient segmentation for hooking up the communication module to the BPCS, without considering the possibility of communication module compromise.

3.5. Deep lateral movement in other domains

We describe these adversarial tactics discussed in this report using the term "*deep lateral movement,*" to emphasize their technical and tactical distinction from more conventional lateral movement at level 2 and above, but analogous TTPs have been deployed in other domains, for instance:

- In the last few years, exposed IoT devices such as IP cameras, NAS devices, VoIP phones and printers have become a serious vector for initial access and pivoting into internal networks. Forescout's own research into the security of Building Automation Systems (BAS) and proof-of-concept IoT ransomware has demonstrated similar strategies of moving through IoT devices. In a similar vein, the "*SOHO Smash-Up*" category, at the yearly Pwn2Own hacking contest, challenges contestants to compromise WAN-exposed devices and subsequently pivot to, and compromise, embedded devices on the internal LAN.

- Attackers in enterprise IT networks have been compromising servers and workstations and subsequently moved to embedded components *within* those systems for a more strategic and persistent positioning. This includes:1) Several examples of malware targeting UEFI, designed to persist even if hard drives are replaced; 2) A firmware-level rootkit in HP Integrated Lights Out (iLO) – a product for out-of-band remote server management – designed to repeatedly wipe infected system; 3) The Conti ransomware gang discussing leveraging attacks on Intel's Management Engine (ME) – a microcontroller within the chipset of Intel systems – to drop additional payloads.

- In automotive cyber-security research (as demonstrated against Ford, Tesla and VAG), attack chains have typically moved through multiple embedded ECUs, through gateway modules, and across inter-chip busses in order reach the right CAN busses and manipulate internal functionality. A typical example is the need, after initially compromising a multimedia unit, to gain control over an on-board CAN controller chip to remove CAN message filters, then compromise a gateway module sitting at the intersection of different domain CAN busses, before being able to target ECUs of choice on the right bus.

- In mobile device security research (as demonstrated against cellular basebands and Wi-Fi chips by Broadcom, Mediatek and Marvell on Android, iOS and Linux-based devices), an initial compromise of the communication processor (CP) is almost always followed up by lateral movement to the application processor (AP) through abusing direct memory access (DMA) capabilities or vulnerabilities in chip drivers. This type of CP-AP escalation is not restricted to mobile devices and applies equally to any sort of device that integrates a CP separately from its AP, including cellular and wireless routers and communications modules found in OT environments. In fact, similar attack chains have been constructed for cellular modules in automotive telematics units.

These examples from other domains show that defenders **should understand lateral movement as possibly occurring beyond the scope of just workstations and servers** and consisting of more than the conventional "*Mimikatz-PsExec-RDP-ZeroLogon*" TTPs.

4. L1 lateral movement TTPs

The TTPs presented in this section are far from exhaustive but are based on those most frequently discussed in research or demonstrated in analogous work in other domains, as discussed above.

4.1. Protocol routing and tunneling

It is well-known that many OT protocols are highly routable by design due to complex, distributed and nested system architectures. Communications need to be carried across different protocols and over different types of communication media, sometimes aided by protocol converters while in other cases native protocol capabilities offer sufficient flexibility. Protocols like CIP, S7comm, EtherCAT and Schneider Electric's Transparent Device Access (TDA) allow attackers to route messages across different nested device networks as illustrated in Figure 11. It is also worth mentioning that many PLC backplanes are based on (variants of) routable OT protocols (such as Logix' CIP-based ControlBus, S7's Profibus-based K-Bus and Modicon's TDA-based X-Bus), which allows for easy extension into multi-rack, remote IO and nested cell setups.

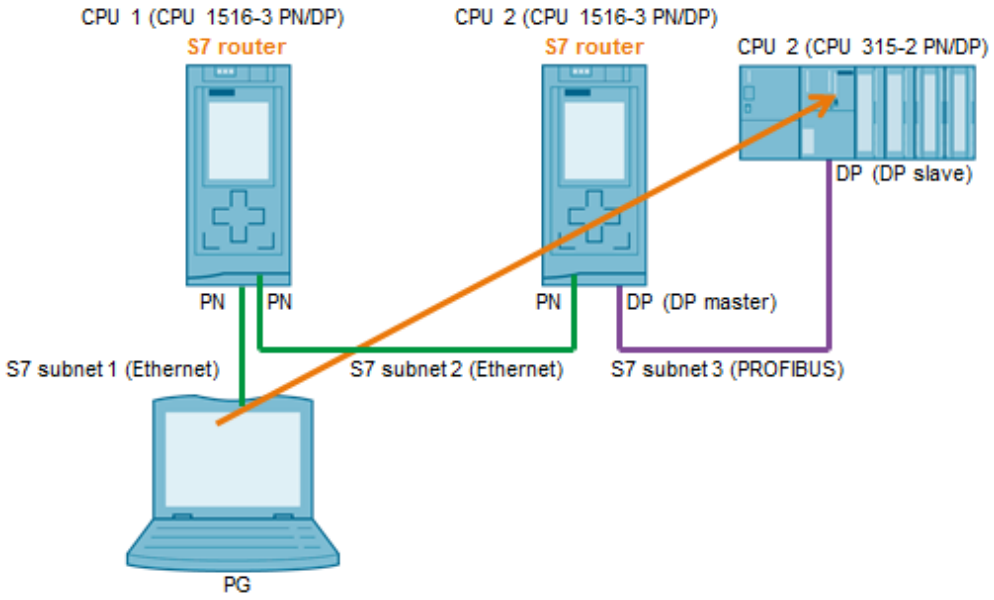


Figure 11 - Siemens S7comm routing across Ethernet and PROFIBUS – from Siemens

While such routability often trivially enables attackers to abuse insecure-by-design functionality on nested device networks, some routing schemes have security features (e.g. as part of CIP Security or through inherent upstream segmentation) and in other cases routing information might need to be explicitly configured in all devices along a route preventing completely arbitrary lateral movement.

In addition to routable protocols, many OT protocols have encapsulation and passthrough functionality. Features such as Modbus Encapsulated Interface Transport (MEI) using function code 43, HART pass-through and certain reserved FL-net transaction codes (TCD) allow completely different protocols to be encapsulated within the PDUs of the top-level protocol and effectively tunnel them to devices deep within the control or device networks. In contrast to inherently routable protocols, it is often not immediately clear to system architects that non-routable protocol traffic might still reach certain devices from far-away points in the system by being encapsulated and carried by protocols.

While this type of functionality is intentional and well-understood by many engineers, it is still too frequently overlooked from a security perspective, especially considering many firewalls and deep packet inspection (DPI) solutions cannot always enforce rules at a protocol routing level or properly decapsulate and inspect tunneled OT protocols.

4.2. In-band code downloads

Code download functionality (T0843 – Program Download), primarily in the form of control logic and firmware, is one of the most obvious ways to gain a foothold on a level 1 device in preparation for further lateral movement. While this type of functionality continues to suffer from being insecure-by-design, as recently discussed in our OT:ICEFALL research, the primary factor as to whether this is a risk for a nested device network is whether this functionality occurs in-band or not. That is, whether the code is downloaded over the same communications link as its regular communications. In addition, it is relevant to consider whether such downloads can be carried out in *hot* fashion, i.e., without first moving the device to a dedicated update mode requiring physical interaction.

To illustrate this, consider fieldbus couplers as discussed in Section 3.2.2. We have observed several cases where asset owners have deployed such couplers as perimeter devices with external parties. Let's take the case of Siemens DP/DP couplers which link two PROFIBUS DP devices or networks together. The older model (6ES7158-0AD00-0XA0) is a straight-forward 'dumb' device. However, the newer model (6ES7158-0AD01-0XA0)

started out as a similarly 'dumb' device but as of hardware revision FS:05, it started supporting (unauthenticated, unsigned) hot firmware updates via PROFIBUS DPV1. **This means that depending on your coupler's model and hardware revision, it might or might not be possible for an attacker to rewrite the coupler firmware and breach through an otherwise relatively robust perimeter and gain direct access to level 1 systems from the outside**. By contrast, Helmholz's DP/DP couplers perform firmware updates via an out-of-band USB interface. This example illustrates how in-depth knowledge of (latent) capabilities of an exact device model and revision is required in order to make sound assessments of their risk posture.

Another example of using in-band code download mechanisms to move between level 1 devices is some of the standardized CAN-in-Automation (CiA) functionality for CAN(open) bus communications. CANopen is a popular automation protocol level 1 and field device communications, especially in motion-oriented machine control, maritime, railway, and vehicular systems. CANopen nodes organize their communication and application parameters as internal 'objects' organized in an Object Dictionary. These objects can be accessed for reading or writing through unique 24-bit addresses and further 16-bit and 8-bit indices and sub-indices respectively. The Service Data Object (SDO) protocol establishes peer-to-peer client-server communications between two CANopen nodes, allowing a client to read or write objects for non-real-time data transfer purposes such as parameterization. CiA standards 301, 302-2, and 302-3, specify several SDOs which allow potential code downloads as shown in Table 3.

Table 3 – CAN-in-Automation (CiA) Service Data Objects (SDO) with code download potential

Index	Name	Description
0x1023	OS CMD	ASCII or binary manufacturer-specific command interface
0x1024	OS CMD Mode	Specify buffering or immediate execution mode
0x1026	OS Prompt	Character-driven, manufacturer-specific command interface
0x1F50	Download Program	Download (parts of a) program in manufacturer-specific format to manufacturer- and device-profile specific area
0x1F51	Program Control	Control execution of program stored in specific area

Program downloads can consist of either firmware or a user program and typically require rebooting the device into bootloader mode (or halting program execution). Sometimes this is done through the *Program Control* SDO (0x1F51) but in other cases it is preferred to use the Network Management (NMT) *reset node* (0x81) function as described in CiA 302-2. After switching the CANopen node to the proper mode, the *Download Program* SDO (0x1F50) is used to transfer the program code in question. After downloading, *Program Control* (0x1F51) is used to switch back to application run mode.

The *OS CMD* (0x1023) and *OS Prompt* (0x1026) SDOs, meanwhile, seem to be less widely supported and very manufacturer-specific in implementation. In some cases, these SDOs will offer access to an RTOS command shell while in other cases they offer access to a proprietary protocol interface (for instance, for interacting with a logic runtime). Both options typically allow for downloading malicious code to the device.

We reviewed the manuals of more than a dozen CANopen-based encoders, drives, valve controllers, and bus couplers as well as several widespread CANopen bootloaders and **found only three instances where this functionality was authenticated**. In each of these instances, **authentication consisted of simply writing a static 32-bit value to a pre-defined 'password' / 'unlock' SDO** and was likely designed to prevent unintended errors rather than unauthorized access. While by no means exhaustive, this sample is indicative of and in line with insecure-by-design expectations.

While this inherent insecurity at level 1 should not come as a surprise, we do think it is worth pointing out that very similar functionality in CAN-connected automotive ECUs has had slightly better security for quite some time. Many ECUs expose a so-called Secondary Bootloader (SBL) which downloads target application or configuration data via the Unified Diagnostic Services (UDS) protocol and reprograms flash memory. These SBLs themselves

can be downloaded to the ECU via UDS after unlocking the bootloader, granting attackers a code execution vector. Unlocking ECU bootloaders via UDS generally requires passing a challenge-response algorithm called a *seed/key sequence*. While many seed/key algorithms are cryptographically weak homebrew solutions by OEMs, they at least posit a challenge beyond reading the manual and have the potential to be implemented in a more robust fashion.

4.3. Direct memory manipulation

Controllers and field devices typically have a memory map that is divided into distinct areas for holding I/O images, timers, counters, status values, user-defined parameters, and user programs. Parts of this memory map are frequently exposed for read and/or write access through object or register mappings in protocols like Modbus or EtherNet/IP or through similar exposure in undocumented engineering protocols. While sensitive memory areas (like those holding executable code) are typically not directly writable through such interfaces, data areas holding codeflow-relevant information (such as function pointers or dispatch tables) might be, as shown in CVE-2019-6829 affecting the Schneider Electric UMAS *WritePhysicalAddress* message (function code 0x29). In addition, improper bounds, permission, and logic checks might allow write operations to unprivileged areas to influence privileged ones in order to achieve code execution as shown in CVE-2020-45788 (discussed in Section 6.3) affecting Schneider Electric Modicon *Unity* PLCs.

Another example of direct memory manipulation (T0871 – Execution through API) enabling RCE on level 1 devices can be found in CANopen's *OS Debugger* SDO (0x1025) as defined in CiA 301. This SDO specifies a binary, manufacturer-specific debugger interface without authentication requirements. OEMs can use this SDO to expose an RTOS debug agent (such as VxWorks WDB, QNX qconn, or ENEA OSE Illuminator) or implement their own minimal interface. Such debug functionality typically allows arbitrary read and write access to the device's entire memory space rendering RCE trivial.

Direct memory manipulation differs from the code download tactics discussed in Section 4.2. **Code download mechanisms require the device to halt execution or even reboot before loading the code through a bootloader or runtime executive, something which might disrupt the controlled process and trigger operator alarms, while direct memory manipulation allows attackers to smoothly hot-patch devices**.

4.4. Protocol stack vulnerabilities

As a result of the dominance of memory-unsafe programming languages such as C(++), protocol stack vulnerabilities (T0866 – Exploitation of Remote Services) continue to haunt embedded devices. The last few years have seen several batches of (RCE) vulnerabilities in dozens of embedded TCP/IP stacks, as well as periodic discoveries of such issues in stacks for popular OT protocols such as EtherNet/IP and DNP3. As of 2017, C and C++ were still the primary programming languages of choice for 93% of then-current embedded software projects and while memory-safe languages like Rust are gaining momentum, including for OT protocol libraries, the sheer weight of legacy C(++) code will be with us for a long time.

Contrary to surprisingly popular belief, non-IP protocol stacks written in memory-unsafe programming languages are just as susceptible to these kind of vulnerabilities as their IP-based siblings. This has been demonstrated for everything from OT serial protocols and embedded USB Device Firmware Upgrade (DFU) stacks [37] [38] [39] to serial interfaces between printer cartridges and printers. While it is true that serial protocol interfaces might have a smaller attack surface than IP-based ones due to elimination of the underlying TCP/IP stacks, this represents only a marginally smaller risk considering vulnerabilities tend to be far more common in the more complex application layer that *is* present. It is also important to keep in mind that while serial interfaces might not be directly accessible by attackers, malicious traffic can be delivered through IP-based protocols (e.g., CIP-EtherNet/IP, HART-HART/IP), encapsulated within IP-based protocols (as discussed in Section 4.1), or come from gateway translation vulnerabilities.

The impact of this kind of vulnerabilities in embedded devices is compounded by the fact that they typically lack proper internal security segmentation (between modules on the same backplane, processors on the same

module, tasks on the same processor, or user and kernel mode) as well as typically lacking any sort of exploit mitigations (e.g. W^X, ASLR, stack canaries, CFI, etc.) or even the prerequisite hardware support.

To assess the potential risk of a protocol stack vulnerability, one should take the ease of this kind of deep lateral movement into account. Consider, for example, a vulnerability in a CIP parser. If the vulnerability can only be used to cause a denial-of-service, it matters greatly where the parsing happens. After all, the difference between a DoS on an Ethernet module and a CPU module is the difference between loss of communications and loss of protection or control. But if the vulnerability can be used to achieve code execution, there still is a risk of loss of protection or control even if the parsing happens on the communication module simply because of the potential for the attacker to use that module as a pivot to the CPU module.

5. Proof-of-concept scenario

To illustrate the use-cases for *deep lateral movement* and some of the tactics involved, we present a hypothetical scenario modelled after a distributed but centrally managed movable bridge infrastructure. In our scenario, an attacker seeks to physically damage a bascule bridge. In a bascule bridge design, as illustrated in Figure 12, an open bridge leaf is held in place by a counterweight resting in the bascule pit.

A typical bridge closing sequence, as shown in Figure 13, consists of the motor systems driving the bridge leaf down up to the point where the leaf makes contact with *near closed* limit switches (as illustrated in Figure 14) instructing the drive to decelerate to creep speed for safety reasons. When the leaf makes contact with the *full closed* limit switches, the drive directional command is removed in order to halt movement and the lock bar actuator is driven to safely lock the leaf in place. Before a closing sequence can be initiated, bridge sensors will need to ensure no traffic is passing and gates and lights will have to be in the correct position.

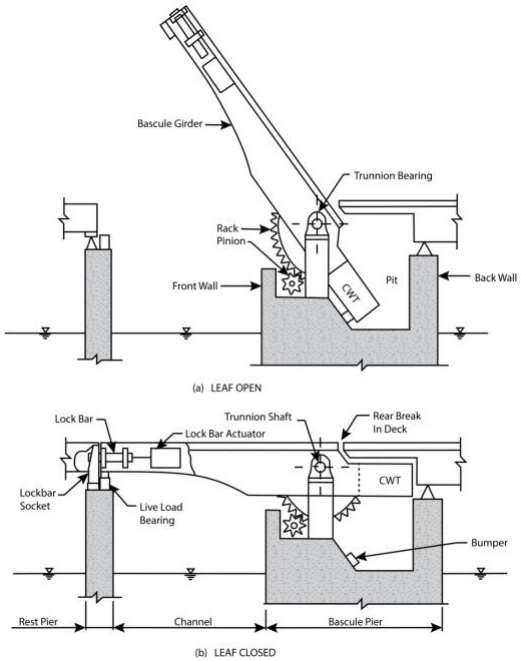


Figure 12 – Movable bascule bridge illustration – from FDOT's Bridge Maintenance Reference Manual

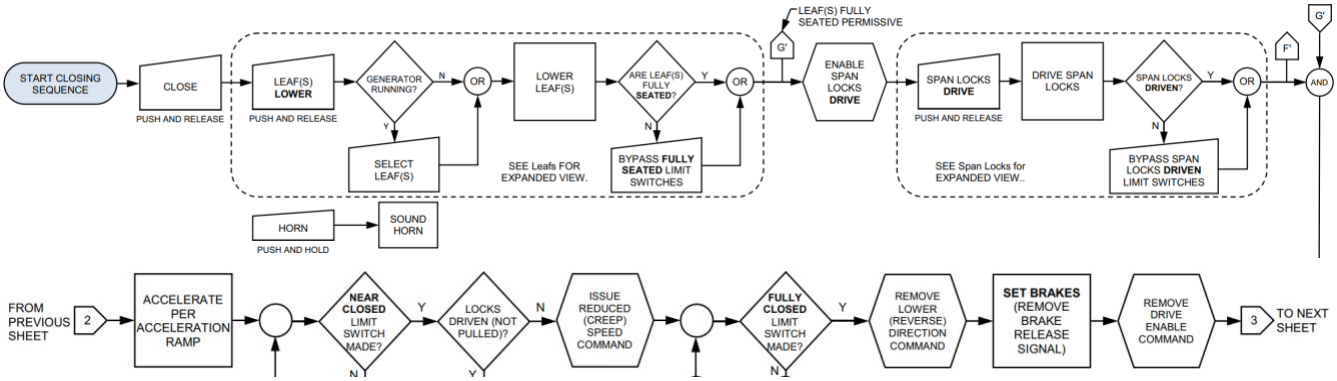


Figure 13 – Partial bridge closing sequence flowcharts – from FDOT's Bridge Maintenance Reference Manual

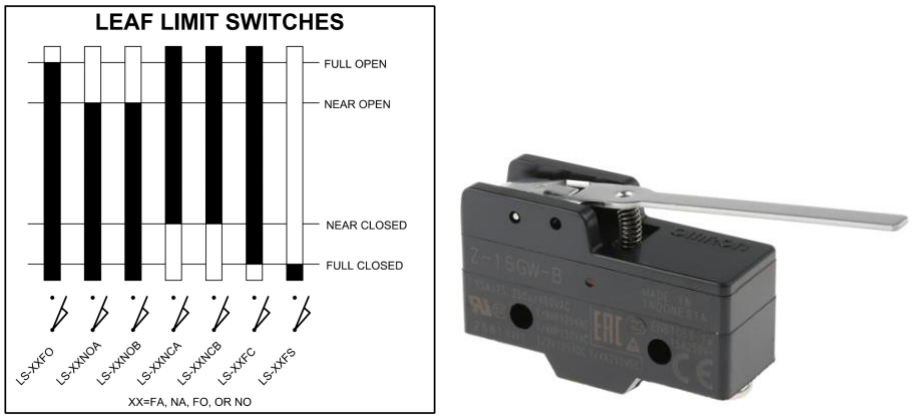


Figure 14 – Bridge leaf limit switches illustration (from FDOT's Bridge Maintenance Reference Manual) and small-scale limit switch

There are multiple cyber-physical attack scenarios possible against movable bridges – including unexpected movement sequences during the passage of dangerous cargo like chemical trains or liquefied natural gas (LNG) ships – but for the purpose of this report we focus on the following:

1. Opening the bridge with the lock bar still driven is the simplest scenario which requires the attacker to manipulate lock limit switch readings and wait for a natural opening sequence. As shown in Figure 15, this will cause the lock bar to bend and require replacement, obstructing vehicle and vessel traffic due to partially opened bridge spans. While rapid response to the incident depicted in Figure 15 meant disruption lasted for 3 days until a temporary replacement lock bar could be installed, suspicions of a cybersecurity root cause will likely take the bridge out of commission for far longer.

2. Closing the bridge at full speed and hitting the bearings with the lock bar driven. If the attacker can make the bridge leaf hit the bearings at full speed, without first decelerating to creep speed, with the lock bar driven on forehand this is likely to significantly damage the lock bar, the bridge leaf, the bearings, and the connecting road surface. This will require the attacker to bypass creep speed deceleration as well limit switch logic allowing for span lock actuation, the possibility of which depends on how much of this is regulated in controller logic. In addition, the attacker needs to suppress safety system interventions, including manual emergency stops.

3. An alternative to scenario 2 would be to initiate a closing sequence and wait until the bridge leaf has achieved maximum velocity before triggering an *emergency stop (E-STOP)*. Such a high-speed stop with a short break duration will cause large forces to move through the bridge because of the inertia of the counterweight. Whether this is sufficient to damage the bridge in a single try or would require multiple wear-and-tear attempts depends on many physical design factors specific to an individual bridge (such as bridge type, materials choices, push-pull rod specifications, drive design and wind load) that are beyond

the scope of this report. However, safety studies into emergency stops and power failures indicate that such scenarios can be extremely stressing and potentially damaging. To achieve such a scenario, an attacker will need to bypass creep speed deceleration as well as obtain the ability to trigger an emergency stop at a moment precisely synchronized with bridge leaf movement.

Figure 15 - Lock bar damage on Brickell Avenue Bridge in Miami after lock limit switch failure – via FDI Services

The scenarios described above are generally very difficult or even impossible to achieve with simple control over the bridge SCADA interface. After all, the bridge typically functions as a black box to the SCADA system which can initiate bridge movement but is bound by functional and safety constraints implemented in the logic of the bridge control system. There's no "*destroy my bridge please*" button.

Instead, the attacker will need to move from the level 2 systems into the bridge control system itself to perform detailed manipulation of its control and safety logic in order to bypass limit switches, directly control motor drives, encoders, and brakes, and trigger or suppress emergency stops.

Figure 16 shows a simplified architecture for this scenario based on observed real world systems, while Figure 17 shows the corresponding demo setup. Here, the bridge objects are delivered as packaged units managed by a third party – such as a municipality – while overall management through the SCADA is done by a central government agency or regional utility. The SCADA and associated level 2 systems communicate with the bridges through a *coupler* device – in our demo setup represented by a Wago 750-852. While we could have chosen any type of fieldbus coupler device, we had one of these in our lab and the Wago 750 series is a popular line of internally similar fieldbus couplers supporting many different OT protocols and fieldbuses. The Wago 750-852 has an internal Ethernet switch with two ports. On one port, we have a link going to our simulated SCADA environment where the attacker is positioned. On the other side, we have a separate, restricted subnet for bridge communications. It is not possible for the SCADA to talk directly to the bridges, instead the SCADA controls the bridges through a set of mapped Modbus registers on the coupler which internally translates these to Modbus registers on the bridge control system interface.

Each of the individual bridge packaged units is controlled by an *object PLC* – in our case a Schneider Electric M340. The M340 in our demo setup consists of a BMXP3420302 CPU module, a BMXNOR0200H RTU module, and IO modules. The bridge system field devices are controlled by the M340 CANopen interface on the CPU module and IO modules. The RTU module exposes a limited set of variables and setpoints to the *coupler*, allowing for initiating open and close sequences but offering no granular control over the bridge. In order to connect the control and safety parts of the system, the Ethernet interface on the CPU module is connected to the Ethernet module on the safety PLC by means of a non-routable point-to-point Ethernet link (though this could also have been a serial link with a similar profile).

The safety PLC is an Allen-Bradley GuardLogix setup consisting of a 1756-L61S safety CPU module and 1756-LSP safety partner, a 1756-L61 helper CPU module,and a 1756-EN2T/D Ethernet module. While the CPU modules are older EOL parts we had in the lab for testing purposes, this is immaterial to the demo scenario since

the vulnerabilities we will be exploiting concern the Ethernet module and functionality that is similar on newer CPU modules as well. The Ethernet module only exposes a limited number of status values over the point-to-point link and does not allow the M340 on the other side to send EtherNet/IP commands to it or the rest of the safety system. The safety PLC is in turn connected to remote IO handling SIS bypass enabling and emergency stop activation.

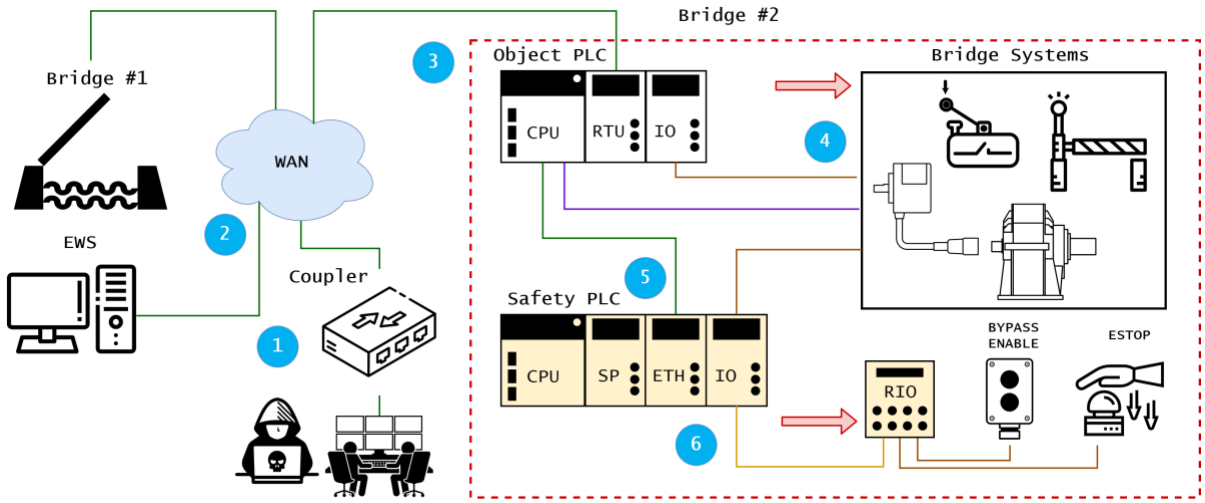


Figure 16 – Attack path

An attacker coming in from the central SCADA who wishes to carry out one of the attack scenarios we described will need very granular control over both the *object* and *safety* PLCs as the systems nested behind them. In order to achieve this, they will need to first get past the coupler in order to have unrestricted communications with the object PLC. Next, they will need to get RCE on the object PLC to have unrestricted access to the CANopen fieldbus as well as a way to move across the point-to-point link to the safety PLC. In the next section, we will describe this process step-by-step.

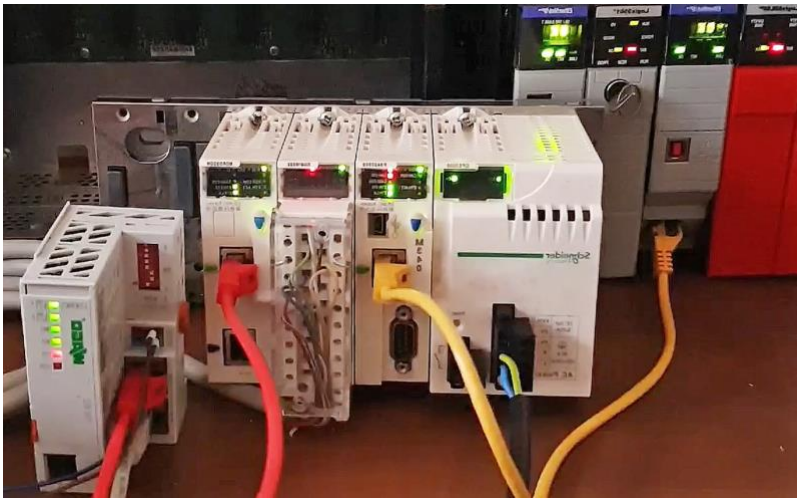


Figure 17 – Demo setup – from our PoC video

6. Proof-of-concept attack chain

We implemented a proof-of-concept for the scenarios described in the previous section with the following steps:

1. Obtain RCE on the Wago coupler to enable unrestricted communications with the M340 object PLC.
2. Bypass UMAS service authentication on the M340 object PLC.

3. Obtain RCE on the M340 object PLC to move to the bridge control system's internals.
4. Manipulate field devices connected to the M340 object PLC.
5. Obtain RCE on the GuardLogix safety PLC Ethernet module to cross the point-to-point link.
6. Manipulate wide safety systems across the GuardLogix backplane.

These steps illustrate deep lateral movement across a fieldbus coupler, into package unit internals, via a PTP link to the SIS, illustrating the kind of perimeter crossing and granular control goals discussed in Section 3.

6.1. Wago 750 coupler RCE

To obtain code execution on the Wago coupler, we will use a pre-authentication stack buffer overflow in the FTP daemon that Vedere Labs discovered as part of NUCLEUS:13: CVE-2021-31886 (T0866). The NUCLEUS:13 research report presents a detailed proof-of-concept exploit for that vulnerability. Exploitation is straightforward due to the lack of exploit mitigations on the underlying Nucleus RTOS.

After gaining initial code execution, no further privilege escalation is required since our payload will run in supervisor mode with no significant separation between different RTOS tasks, allowing us to repurpose parts of the system firmware as shown in Figure 18. The next step is to install an *implant* (T0857) into the firmware by hooking (T0874) the Wago's Modbus handler functionality so that traffic with function code 0x5A (more details about UMAS below) are tunneled transparently through the coupler.

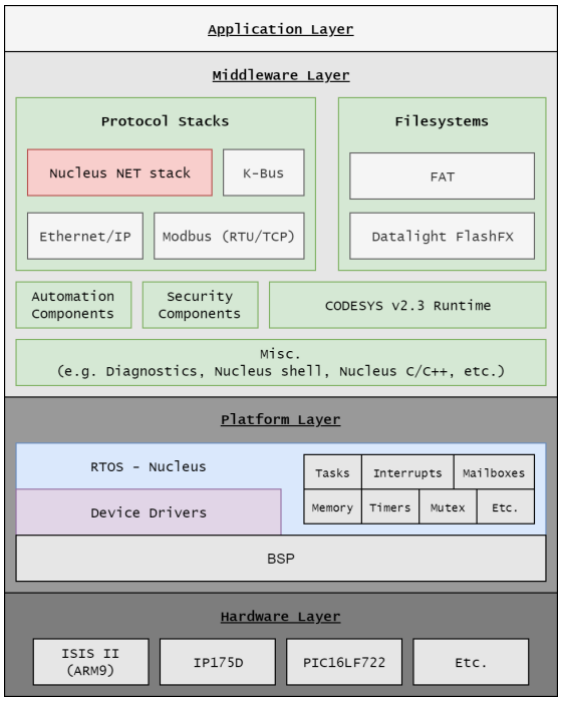


Figure 18 – Wago 750-852 architecture (as of FW v01.09.25 (16))

6.2. Schneider Electric UMAS authentication bypass

Now that we are able to talk to the M340 PLC, we plan on gaining code execution through a vulnerable UMAS feature as discussed in Section 6.3. But first we will need to bypass the authentication mechanism.

Unified Messaging Application Services (UMAS) is the proprietary engineering protocol for Schneider Modicon PLCs. It is one of the best understood and reverse-engineered proprietary OT protocols and allows all the typical engineering functions such as starting and stopping the PLC, downloading logic, forcing IO values, etc. Some basic familiarity with UMAS is assumed for this section, for which we refer to the wealth of prior work [4] [1] [6] [3].

Historically, UMAS was unauthenticated and used the so-called `SimpleReserve` mechanism illustrated in Figure 19 in order for a Control Expert engineering workstation (EWS) to go online with the PLC and take out a "reservation," which ensures exclusive engineering access. Eventually, Schneider Electric introduced the `SimpleCyberReserve` illustrated in Figure 20, where reservations now require supplying a salted-and-hashed **application password**.

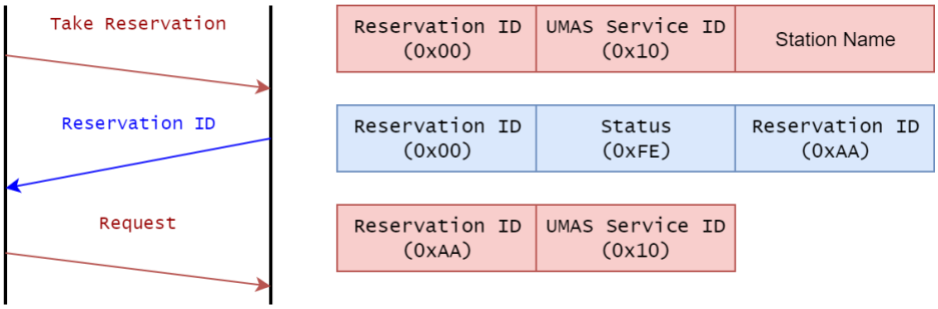


Figure 19 – Schneider Electric UMAS SimpleReserve mechanism

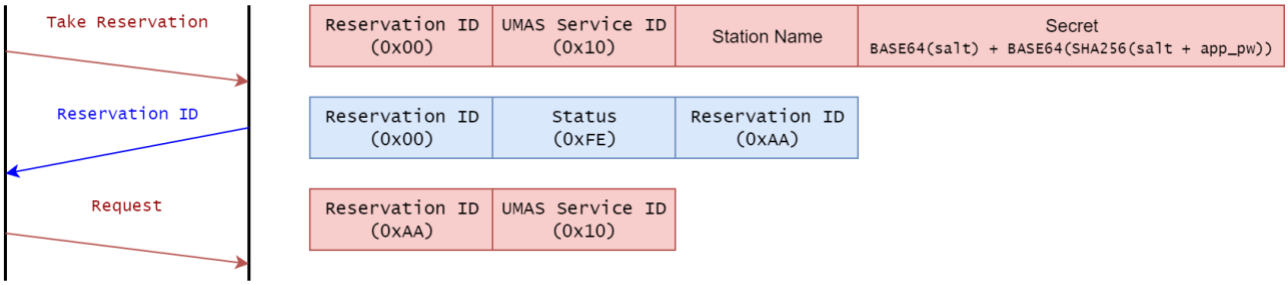


Figure 20 – Schneider Electric UMAS SimpleCyberReserve mechanism

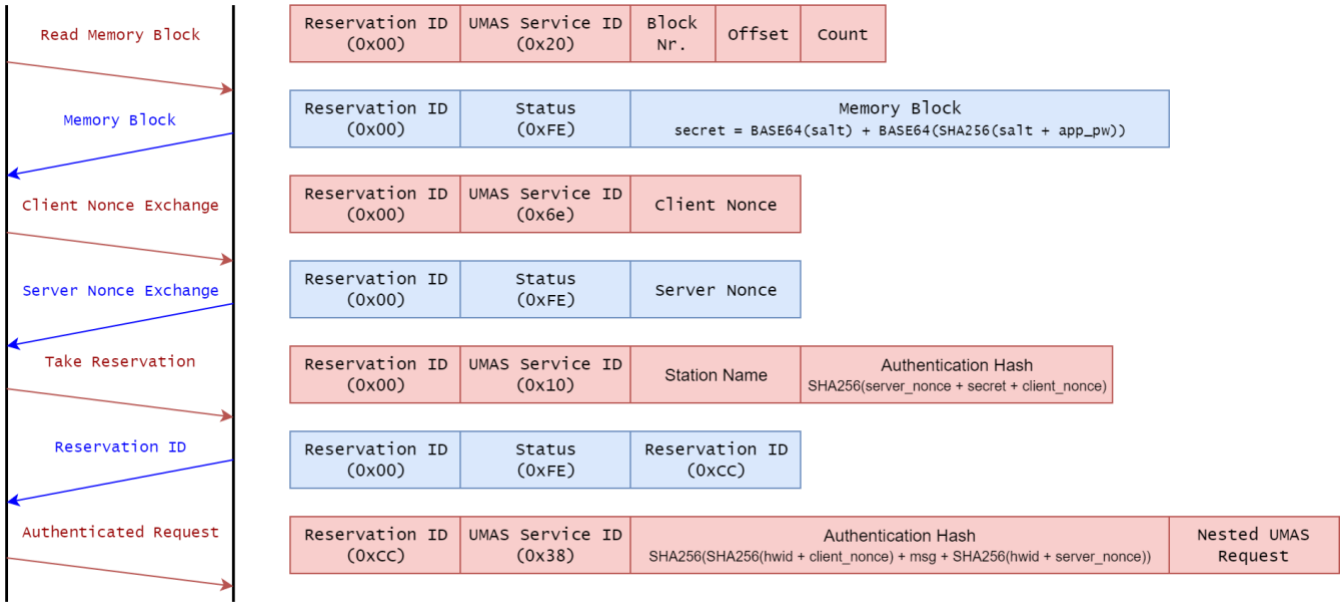


Figure 21 – Schneider Electric UMAS EnhancedCyberReserve mechanism

This mechanism was again modified into the current `EnhancedCyberReserve` mechanism illustrated in Figure 21. Here, the EWS first reads out a memory block from the PLC holding the salted-and-hashed application password. Then 32-byte nonces are exchanged between EWS and PLC, after which an authentication hash is constructed based on the derived application password and nonces. After this has been validated by the PLC, subsequent UMAS messages are wrapped in an "authenticated requested" message, where they are accompanied by an authentication hash derived from the message and nonces.

Since the application password is simply requested from the PLC in the first step, and this retrieved hash can just be sent back to the PLC again, this mechanism was trivially broken as reported in CVE-2021-22779, which was independently discovered by multiple different parties [1] [6] [3] [7]. Schneider Electric addressed this issue in SEVD-2021-194-01 and as of Control Expert V15.1 and M340 CPU firmware V3.50 this issue has been remediated by no longer allowing for unauthenticated retrieval of the hashed application password. However, as of February 1, 2023, firmware SV4.02 for the M580 CPU has been retracted for quality issues leaving this issue unaddressed.

In addition to the application password, a program/safety password has existed for Modicon PLCs for quite some time but this is basically irrelevant from a security perspective and functions mostly to prevent unintentional engineering errors. Finally, Control Expert offers the option to encrypt project files (using AES-CBC-256) but this feature is not related to UMAS authentication or the subsequent RCE vector we describe below.

We designed our setup to run with a fully up-to-date Control Expert and M340, at then-current versions 15.1 (HF001) and 3.50 respectively, with application password, program/safety password, and file encryption enabled.

When investigating the patched `EnhancedCyberReserve` mechanism, we observed that it is still fundamentally broken. First, nonces are maintained by the PLC as *globals* without any explicit ties to a particular client or reservation ID as illustrated in the M340 firmware excerpt shown in Figure 22. In addition, these nonces only renew when a session ends or an explicit UMAS nonce exchange request is received.

```
MNGT_NEW_RESV();
dword_2043FD34 = a1[2] | (a1[3] << 8) | (a1[4] << 16) | (a1[5] << 24);
memcpy_s((int)&client_nonce, 32, (int)(a1 + 6), 32);
byte_2043FD30 = 1;
memcpy_s(a3 + 3, 32, (int)&server_nonce, 32);
if ( byte_2043FAD4 == 3 && pu_IsCyberSecurityAppPwd() && pu_IsSdaResvMechanismApp() && !byte_2043FD38 )
  umas_computeSecretsWithNonces();
timeoutSession = 500;
```

Figure 22 - Schneider Electric UMAS EnhancedCyberReserve management (BMXP34 FW v3.50)*

As such, the PLC's reference nonces will correspond to those of the last nonce exchange operation. When sniffing legitimate authentication traffic, an attacker can simple extract those nonces and the corresponding authentication hash. Since there is no *freshness* involved in the authentication request itself and nonce exchanges are not mandatory, the attacker can replay the legitimate authentication request (without doing a new nonce exchange operation) and the PLC will accept this and initiate a new reservation (locking out the current legitimate EWS connection) as illustrated in the **reservation replay** scenario in Figure 23.

More subtly, however, an attacker can simply **forge authenticated requests without locking out the legitimate engineering connection** (T0856) due to the fact that the 'authentication hash' in the authenticated requests **does not actually include a secret** nor any *freshness*. In addition, reservation IDs handed out by the PLC are not connected to any particular connection or IP which means there is nothing binding authenticated requests to the actual authentication procedure. So, not only are these requests replayable, an attacker would only to know the nonces to forge their own arbitrary, malicious authenticated requests.

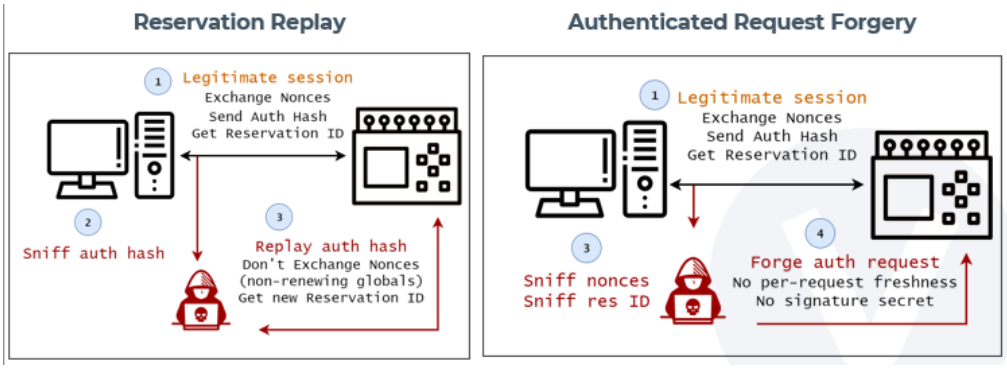


Figure 23 – CVE-2022-45789

We reported these issues to Schneider Electric in July 2022 and they were assigned CVE-2022-45789 with corresponding advisory SEVD-2023-010-06.

6.3. Schneider Electric Modicon Unity RCE

Now that the attacker can authenticate to the object PLC, they want to get code execution on it to move deeper into the bridge system. There have been different approaches to achieve RCE on Modicon PLCs in prior work, ranging from downloading logic blocks that have unsigned ARM code in them [40] [2] to vulnerabilities in UMAS messages allowing for memory read/write access [6] [7], and TCP/IP stacks.

On a fully patched M340 PLC, prior vulnerabilities in UMAS messages have been patched however, and contrary to the M580 (which uses the IPnet tcp/ip stack), the M340 uses the older *WindNet tcp/ip stack* which is not affected by the recent TCP/IP stack vulnerability disclosures (though we would be surprised if those couldn't be found).

Furthermore, we want a method that allows for smooth hot-patching of running code without disrupting the PLC's logic in a way that might cause process upsets or alarms to go off. In addition, it would be preferable for an attacker to avoid clear markers of compromise visible in the EWS such as changed project checksums and source-code. As such, we avoided going for the straightforward route of manipulating the executable code within a regular logic download.

Instead, we opted for an approach that requires a bit of background on the internal organization of the Modicon PLCs. Modicon PLCs are programmed and configured via *project files* created in the Control Expert engineering software (previously Unity Pro). These project files (with varying incarnations over the years), are basically archives holding several texual and binary files. One of these files is called the *application binary file* (APX) the undocumented internals of which have received little public attention (though there is some prior mention with varying degrees of accuracy). This format, as illustrated in Figure 24, is parsed by the Modicon runtime executive in order to organize the internal PLC memory according to various functional areas. Each of these areas has a header with index, number of sections, load address, attributes, etc. Within an area, there are several different sections. Each of these sections has a header and a data part that contains *blocks*. These can hold data, code, source-code, constants, and all other parts that make up PLC logic and configuration. These blocks are internally referenced through the *relocation table* (RT), where each *table entry (RTE)* describes a block, where it is located, its size, attributes, etc. The Modicon runtime executive links these blocks and their internal relocated references together for proper execution of its logic tasks.

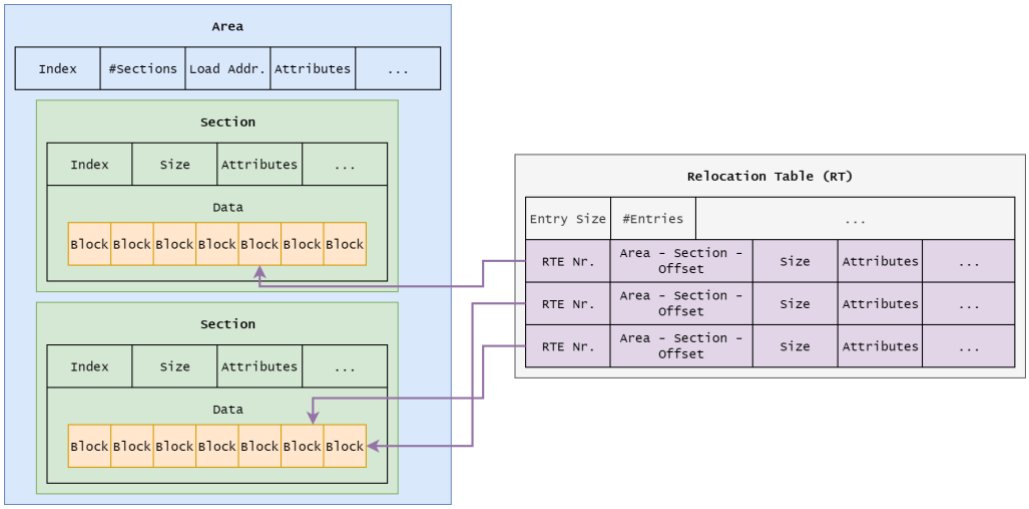


Figure 24 – APX, Application Binary File

There is a UMAS message which has so far, to the best of our knowledge, remained unexplored in public. This is the so-called CSA request (service code 0x50), which exposes a very complex subsystem. This subsystem allows the client to create and manage what are essentially *'virtual pages'* on the PLC as illustrated in Figure 25. After allocation, the client can write a proprietary command set to these pages and schedule them for execution. After execution, any operational results can be read from the pages. This subsystem seems to be used for a variety of operations, including so-called *online modifications* where some dynamic changes to the PLC logic can be made without requiring a full logic stop. Such modifications are restricted in what they can achieve however so we decided to dig under the hood to turn this mechanism into a vector for arbitrary RCE.

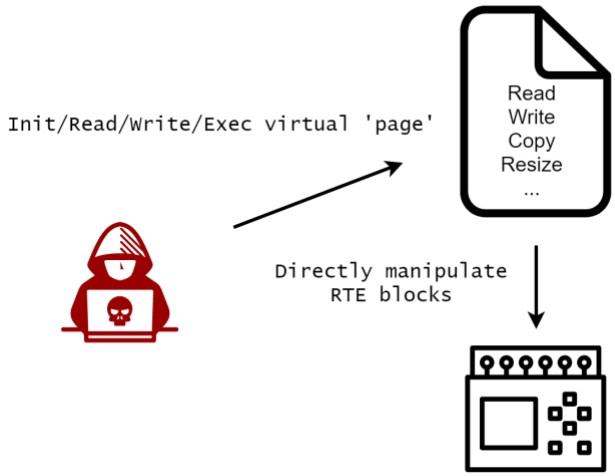


Figure 25 – Schneider Electric UMAS CSA message feature

The command set operates directly on the internal memory blocks described above, referencing them by RTE index, allowing an attacker to read, write, copy, resize, allocate and deallocate blocks. Initially, the impact seemed limited because direct write operations to code blocks are prohibited through an attribute-based permission check. But as shown in Figure 26, this check is *ignored* for copy operations, allowing an attacker to write their payload to a data block first and then copying it to the code block. There are many ways in which to approach this, including expanding existing blocks to append the payload or finding unused "caves" to hide the payload in. Either way, the payload will execute as part of the regular execution of the targeted code block (T0866). The complete flow is illustrated in Figure 27, where we show the message flow and structure but leave out technical details like command set values to prevent easy replication by attackers in light of both the sensitivity of these systems and the timelines involved in hardening them.

We reported this issue to Schneider Electric in April 2022 and it was assigned CVE-2022-45788 with corresponding advisory SEVD-2023-010-05.

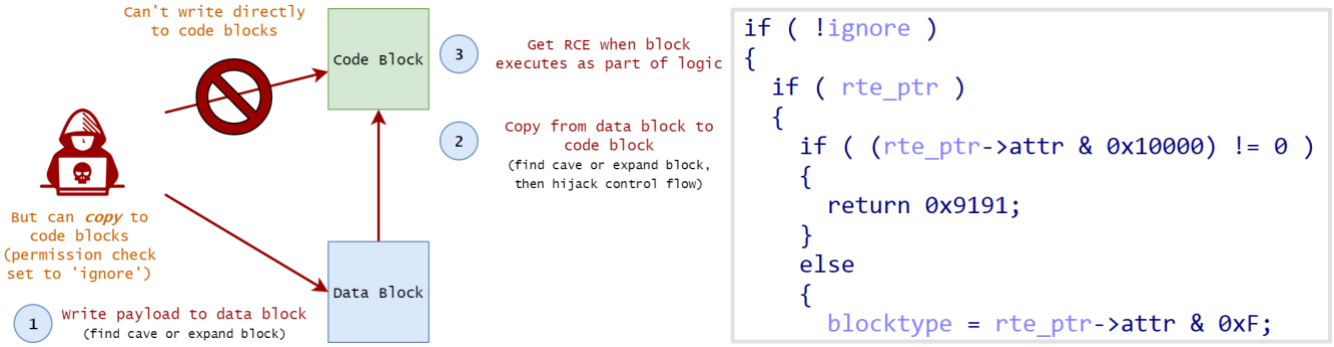


Figure 26 – Manipulating live code blocks via CVE-2022-45788

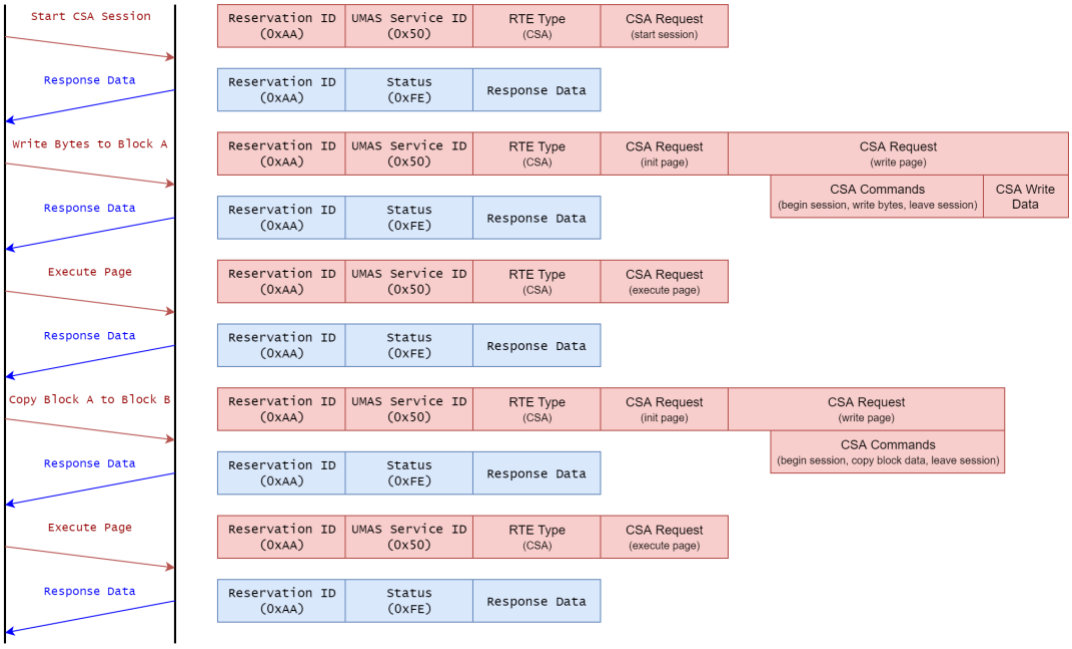


Figure 27 - Schneider Electric UMAS CSA message flow for CVE-2022-45788

This vulnerability exposes a very powerful primitive to an attacker for stealthy and smooth injection of arbitrary payloads. As shown in Figure 28, the Control Expert engineering software will indicate everything is fine and nothing has changed to the logic despite our payload injection since no project checksums have changed. Similarly, if an engineer were to perform a full project upload of the compromised PLC for digital forensics and incident response (DFIR) purposes, none of the source-code sections would differ in any way. An attacker could incorporate further anti-DFIR tactics by relocating from the manipulated code block to another part of memory and subsequently restoring the manipulated code and data blocks to their original state (T0872), thus also removing indicators that might come up during in-depth forensics investigation of the blocks present in the uploaded APX.

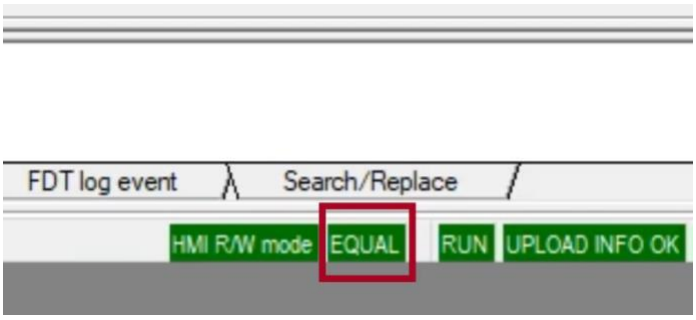


Figure 28 - Control Expert indicating project is unmodified despite payload injection – from our PoC video

When our payload executes as part of the code block, we do so not within the same memory segment as the system firmware but in a dedicated user code segment as discussed in prior work. What has not been discussed previously, however, is that code executed by the `sas_UserCodeExec` routine as part of the logic cannot immediately reach out to arbitrary firmware functions and is bound by an interrupt-driven *watchdog* timer which will trigger an error handler if payload execution takes too long.

To bypass these restrictions, our payload will first hook a globally writable function handler (T0874) and redirect its execution to another part of our payload so that when the hooked handler executes, our payload will execute in a context not restricted by the constraints of the runtime executive. Once this is achieved, there are no more obstacles to installing our *implant* as a reliable VxWorks task using the OS API (T0857) since we run in supervisor mode, with no exploit mitigations being present, and no real separation between tasks as shown in Figure 29.

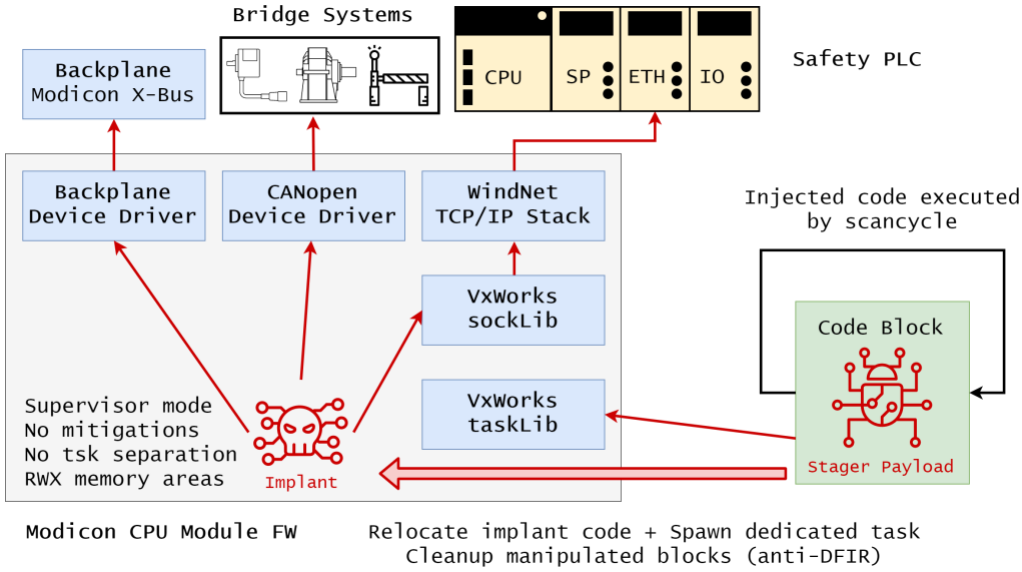


Figure 29 - M340 stager and implant

6.4. CANopen interaction from the M340 PLC

Now that we have free reign within the M340's CPU module, we can repurpose parts of its firmware functionality for additional deep lateral movement goals as illustrated in Figure 31 and Figure 32. Payloads can be written in C as long as the prototypes of internal PLC functions are defined and invoked from statically specified (and firmware-version specific) addresses and the code is compiled in position-independent fashion.

An easy way to demonstrate this arbitrary low-level access is by manipulating the M340's status LEDs, which can be done either through directly interacting with the memory-mapped peripherals or by invoking the convenient wrapper functions as shown in Figure 30, similar to what prior work has demonstrated against the M580.

```
ledRUNblinking                                  ;
                MOV     R12, SP
                PUSH    {R11,R12,LR,PC}
                MOV     R0, #4
                SUB     R11, R12, #4
                MOV     R1, #3
                BL      BlinkingLed
                MOV     R0, #2
                MOV     R1, #3
                BL      BlinkingLed
                MOV     R0, #0
                MOV     R1, #1
                LDMFD   SP, {R11,SP,LR}
                B       BlinkingLed
```

Figure 30 – Schneider Electric BMXP3420302 LED driving functionality (as of FW v3.50)

More interesting, however, is that we can talk directly to the CANopen device driver to interact with connected field devices such as motor drives and encoders, not only sending arbitrary (and potentially malformed) CANopen traffic but also writing granular payloads timing message spoofing with Network Management (NMT) commands to drive certain devices offline (T0816) in order to deal with message confliction issues during spoofing. Depending on the attackers' needs and the field devices in question, they could also potentially gain code execution on those devices through the TTPs outlined in Sections 4.2 and 4.3. To generalize this approach, one would need to take into account how the fieldbus stack is implemented and how device drivers interact with it. Certain types of attacks might require a kind of granular control that requires the attacker to bypass device drivers and interact directly with the fieldbus controller or even manipulate its firmware in similar fashion to the obstacles CAN controllers can pose in automotive cyber-security.

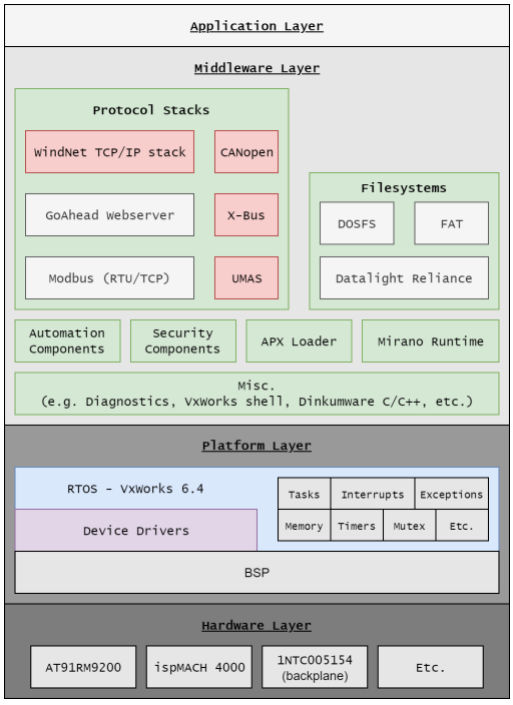


Figure 31 – Schneider Electric BMXP3420302 architecture (as of FW v3.50)

```
can_SWrite_SDO(ND, 0x1F51, 1, START_BOOT,
can_SWrite_SDO(ND, 0x1F51, 1, ERASE_FLASH,
...
can_SWrite_SDO(ND, 0x1F50, 1, block[i],
```

Figure 32 – Example invocation of CANopen functionality from within firmware

6.5. RCE on GuardLogix Ethernet module to cross the restricted PTP link to SIS

To suppress safety system obstacles and responses to their attack, the attacker will need to at least be able to communicate in unrestricted fashion with the GuardLogix CPU. But since the M340 is connected to the GuardLogix Ethernet module through a restricted point-to-point link, the attacker will need to move across that first.

The Ethernet module in question, the 1756-EN2T/D, is vulnerable to a pre-authentication stack buffer overflow during IP option parsing (CVE-2019-12256 – one of the Urgent/11 issues) which seems like a good way to gain a foothold on the safety system (T0866). This vulnerability has been exploited on a variant of this module (the 1756-EN2TR/C) in prior work, and after taking certain platform differences and memory organization nuances into account we could reproduce the exploit – consisting of a ROP chain redirecting code execution to an RWX memory area holding the payload – to obtain reliable code execution. Our initial payload consisted of a *stager* using the VxWorks OS API to spawn a dedicated task for our *implant* (T0839), which has unrestricted access to the Ethernet module's internals due to running in supervisor mode and there being a lack of any further mitigations or task separation. This can be demonstrated by manipulating the LED display on the Ethernet module to show an arbitrary message of our choice, done by invoking the `Z20BSP_DispDrvrWriteMsgPKci` display driver routine as shown in Figure 33.


```
int rawsocket = socket( AF_INET, SOCK_RAW, IPPROTO_TCP );

if ( setsockopt( rawsocket, IPPROTO_IP, IP_HDRINCL, (char *)&one, sizeo
{
    struct packet p;
    bzero((char*)&p, sizeof(struct packet));

    p.iph.ip_v   = 4;
    p.iph.ip_hl  = ( IPHSIZE + EXPLOIT_PAYLOAD_SIZE ) >> 2;
    p.iph.ip_tos = 0;
    p.iph.ip_len = htons( IPHSIZE + EXPLOIT_PAYLOAD_SIZE );
    p.iph.ip_id  = htons( 1 );
    p.iph.ip_off = 0;
    p.iph.ip_ttl = 64;
    p.iph.ip_p   = IPPROTO_TCP;
    inet_aton( SRC_IP, &p.iph.ip_src );
    inet_aton( DST_IP, &p.iph.ip_dst );

    bcopy( p.exploit_payload, EXPLOIT_PAYLOAD, EXPLOIT_PAYLOAD_SIZE );
    sendto( rawsocket, (caddr_t)&p, ntohs(p.iph.ip_len), 0, (struct soc
```

Figure 33 - Display manipulation on 1756-EN2T/D – from our PoC video – and raw socket exploit implementation

One aspect that complicates our attack chain is that we must launch this exploit from our implant on the M340 CPU module rather than the usual Python-and-Scapy for general purpose platforms. Luckily, VxWorks supports a UNIX BSD compatible sockets library that allows us to easily craft and send malformed IP packets through raw sockets. Keep in mind that on some embedded platforms, Ethernet device drivers may drop malformed packets instead of sending them which would require an attacker to circumvent this.

6.6. Manipulating wider safety systems across the GuardLogix backplane

Now that the attacker has unconstrained access to the Ethernet module, the firmware internals of which are shown in Figure 34, they can repurpose its internal functionality to interact with the rest of the GuardLogix in a manner not constrained by the limits of the point-to-point link.

One approach is to route regular CIP messages to GuardLogix modules or connected remote IOs over the backplane in order to abuse insecure-by-design functionality (such as manipulation of logic or sensitive tags related to SIS bypass enabling or emergency stops) (T0855). Since we cannot route such CIP messages from outside the safety PLC via EtherNet/IP, we will have to instruct the implant on the Ethernet module to talk to the backplane device driver.

The Logix backplane uses a protocol called ControlBus (which is related to ControlNet and thus CIP-based). While older Logix family modules use an Atmel ASIC backplane controller, newer ones use a custom ASIC named "APEX2". The device driver exposes a backplane API similar to the one available on the customizable ControlLogix compute modules and an attacker can interact either with this API or directly with the APEX2 ASIC in order to manipulate modules, tags, configurations, and logic on the safety PLC as shown in Figure 34. Keep in mind, however, that CIP security controls (if present) and CPU module RUN mode switch settings might still restrict attacker capabilities. In order to bypass these from the attacker's vantagepoint on the Ethernet module, they would need an additional vulnerability in the CPU module (such as a CIP parser vulnerability) and the ability to send malformed CIP traffic through the backplane (likely bypassing the drivers and interacting directly with the APEX2 ASIC). This might be an interesting area for future research.

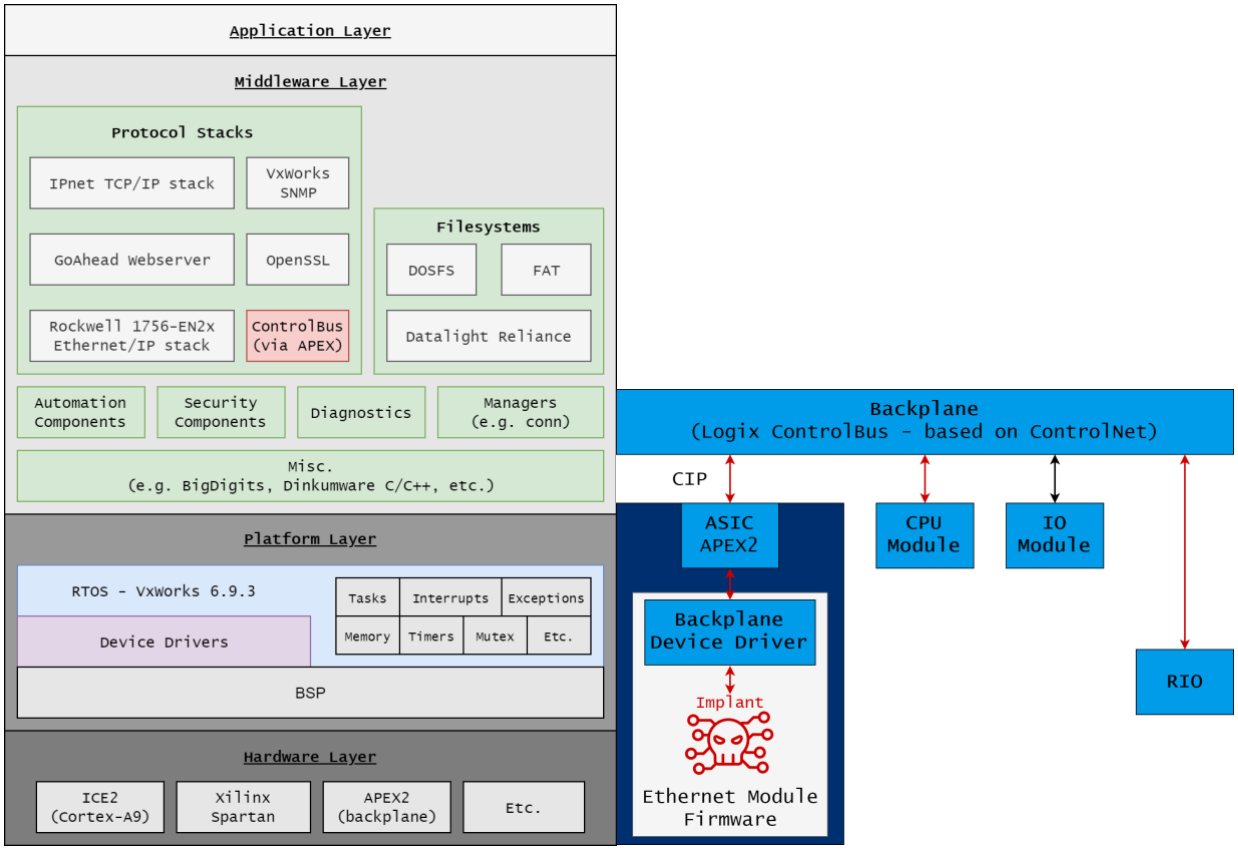


Figure 34 – Allen-Bradley 1756-EN2T/D architecture (as of FW v10.007) and implant backplane communications

7. Mitigation and detection strategies

First, we want to emphasize that this research concerns advanced attacker tactics deployed to achieve high-impact scenarios. Depending on your risk profile and current security maturity, the issues discussed should likely not be near the top of your priority list.

However, the scenario discussed can serve as a point of reference driving an investigation of incorrect assumptions around unacknowledged perimeters or a lack of hardening and visibility at level 1 within your organization. By evaluating whether similar scenarios would be possible around your most important systems, and whether one would even have the visibility and collection capabilities to detect such TTPs, you typically get a much better picture of possible high-consequence attack paths. In addition, it can be beneficial to take this kind of *deep lateral movement* into account during the architectural design phase of greenfield projects or when replacing old "dumb" perimeter devices and connectivity modules at level 1 with new "smart" ones.

7.1. Overview

In addition to the vendor advisories referred to below, the following mitigation steps can be taken to reduce the risk of the kind of deep lateral movement described in this report:

- When segmenting systems, consider not only the routability of allowed protocols but also their (sometimes undocumented) encapsulation capabilities.
- Take stock of perimeters (or what should be perimeters) at the lower Purdue levels, especially when they cross over into highly sensitive zones (such as the SIS) or interact with systems either managed by external parties or to which they might have access (such as PUs).
- Do not assess the risk of a particular link purely based on its normal operational capabilities. Just because it only carries serial traffic that passively reads some status values doesn't mean there isn't a parser somewhere that can be exploited.
- If engineering services (such as Schneider's UMAS) are not required over a particular link, restrict their accessibility or disable them. This goes even for nested devices which might not immediately be exposed to risky networks but might still be reachable through protocol routing (such as Schneider's TDA feature). Of course, there is a tradeoff to be made here with remote maintainability though in some cases dedicated out-of-band networks through interfaces separate from the control network might be an option.
- Use firewalls and IP-based ACLs to restrict sensitive flows (such as engineering traffic) between the EWS and PLCs. For firewalls to be able to do this properly, they need DPI capabilities that can set rules at an appropriate level of granularity (e.g., allowing regular Modbus traffic to a PLC from multiple hosts but restricting function codes like UMAS 0x5A to the EWS).
- If risk assessments indicate there is an unaccounted residual risk on sensitive level 1 boundaries (such as outer perimeters or BPCS/SIS links) that cannot be mitigated against, it makes sense to at least develop visibility into those links through DPI solutions. For serial links with similar risk profiles, a drop-in collector with out-of-band reporting capabilities might be justified if the collected traffic can be properly ingested and parsed by higher-level solutions.
- Where possible, seek to regularly collect level 1 device event and status logs as described below. If those cannot be properly ingested into a SIEM, at least retention aid any future DFIR purposes (whether related to an actual cyber-incident or not).
- Follow Top 20 Secure PLC Coding Practices and, in particular, monitoring anomalous PLC error flag and resource consumption rates (especially related to network communications, CPU errors, memory usage) for ingestion into SOC/SIEM.

Table 4 – Attack chain mapped to MITRE ATT&CK ICS TTPs and mitigations

Step	ATT&CK	Mitigations
CVE-2021-31886	T0866	- Mitigation guidance - In-depth report - IDS should alert on and firewall should block CVE-2021-31886
Install Wago 750 implant	T0857	Monitor diagnostic interfaces outlined in Section 7.2
Hook Wago Modbus handler	T0874	- OT DPI should alert on UMAS operations (Modbus FC 0x5A) - UMAS traffic to non-Modicon devices warrants inspection
Forge UMAS requests to M340	T0856	- SEVD-2023-010-06 - OT DPI should alert on UMAS operations (Modbus FC 0x5A) - PCAP DFIR showing UMAS authenticated requests (SVC 0x38) from client that did not perform nonce exchange (SVC 0x6E) or uses reservation ID requested by other host warrants inspection
Abuse UMAS CSA for RCE on M340	T0866	- SEVD-2023-010-05 - OT DPI should alert on UMAS CSA operations (Modbus FC 0x5A, SVC 0x50) - For DFIR suggestions see Section 7.3.
Anti-forensic blocks cleanup on M340	T0872	- OT DPI should alert on UMAS CSA operations (Modbus FC 0x5A, SVC 0x50)
Hook M340 protocol handler	T0874	N/A (device lacks capabilities for introspection)
Spawn dedicated implant task on M340	T0857	N/A (device lacks capabilities for introspection)
Manipulate CANopen field devices from M340	T0816 T0843	Field device disruptions or alarms should be ingested into or cross-referenced with SIEM
CVE-2019-12256	T0866	- Mitigation guidance - IDS should alert on and firewall should block CVE-2019-12256 - Monitor diagnostic interfaces outlined in Section 7.4
Spawn dedicated implant task on 1756-EN2T/D	T0839	Monitor diagnostic interfaces outlined in Section 7.4
Manipulate modules and devices across GuardLogix backplane	T0855	- Field device disruptions or alarms should be ingested into or cross-referenced with SIEM - OT DPI in safety networks should detect malicious CIP traffic from backplane to e.g. remote IOs over EtherNet/IP

7.2. Implant on Wago 750

The Wago 750 series couplers offer little native functionality to aid in detecting suspicious activity, but one feature that might provide an indicator is the data obtained from Modbus registers 0x1029 and 0x102A. These registers hold information about the Modbus TCP statistics (including message counters) and the number of connections. By permanently monitoring these values and alerting on anomalous values, the Modbus proxy functionality of the implant could be possibly detected (assuming sufficient follow-up investigation).

Table 141: Register Address 0x1029

Register Address 0x1029 (4137_{dec}) with 9 Words		
Value	MODBUS TCP statistics	
Access	Read/write	
Description	1 word SlaveDeviceFailure	→ local bus error, fieldbus error by activated watchdog
	1 word BadProtocol	→ error in the MODBUS TCP header
	1 word BadLength	→ Wrong telegram length
	1 word BadFunction	→ Invalid function code
	1 word BadAddress	→ Invalid register address
	1 word BadData	→ Invalid value
	1 word TooManyRegisters	→ Number of the registers which can be worked on is too large, Read/Write 125/100
	1 word TooManyBits	→ Number of the coils which can be worked on is too large, Read/Write 2000/800
	1 word ModTcpMessageCounter	→ Number of received MODBUS/TCP requests
	By writing 0xAA55 or 0x55AA the register is reset.	

Table 142: Register Address 0x102A

Register address 0x102A (4138_{dec}) with a word count of 1	
Value	MODBUS/TCP connections
Access	Read
Description	Number of TCP connections

Figure 35 - Wago 750 series Modbus diagnostics

7.3. CVE-2022-45788, CVE-2022-45789, and Implant on Schneider Modicon

In order to mitigate these issues, we recommend following Schneider Electric's advice outlined here on SEVD-2023-010-05 and SEVD-2023-010-06.

For CVE-2022-45788 a fix seems to be forthcoming. For CVE-2022-45789, there is currently only mitigation advice but we suggested that – short of a full protocol overhaul – a more secure authentication mechanism could be retrofitted onto the current protocol features by using the Secure Remote Password (SRP) protocol for mutual authentication and key exchange purposes. After deriving a mutually held secret key, this could be used for symmetric signing of authenticated messages using a HMAC solution provided freshness is incorporated into the authenticated parts of the protocol. While not as ideal as a purpose-built secure protocol, a solution such as this seems more feasible for rollout within a reasonable timeframe. After all, Schneider Electric has shown to be able to retrofit authentication features on an insecure-by-design protocol in the past and roll them out into production.

In addition to the vendor guidance, one could monitor specific Modicon system words for malicious indicators as shown in Table 5.

Table 5 – Schneider Electric Modicon System Words offering possible indications of malicious activity

System Word	Name	Monitoring
%SW94 %SW95	Application signature	Application signature which is updated with new project changes. While this won't detect CVE-2022-45789 or the implant described in this report, monitoring this value against a known-good reference

		provides an extra detection point for “regular” logic modifications (malicious or otherwise).
%SW125	BLKERRTYPE	Last fault detected. This word should be monitored for faults related to watchdog overflows (16#DEB0) considering these might occur if the attacker does not take this into account when modifying code blocks.
%SW126 %SW127	ERRADDR0 ERRADDR1	Application blocking error instruction. In case of watchdog stops, these values hold number and value of violating MAST tasks which can be used for investigation triaging.

If an attacker does not take anti-forensics steps to clean up the modified blocks, a project upload through the Control Expert engineering software might contain valuable DFIR artifacts.

While the uploaded project will not show any indicators of malicious activity in Control Expert, forensic investigators can proceed as follows to check for careless attackers seeking to exploit this vector:

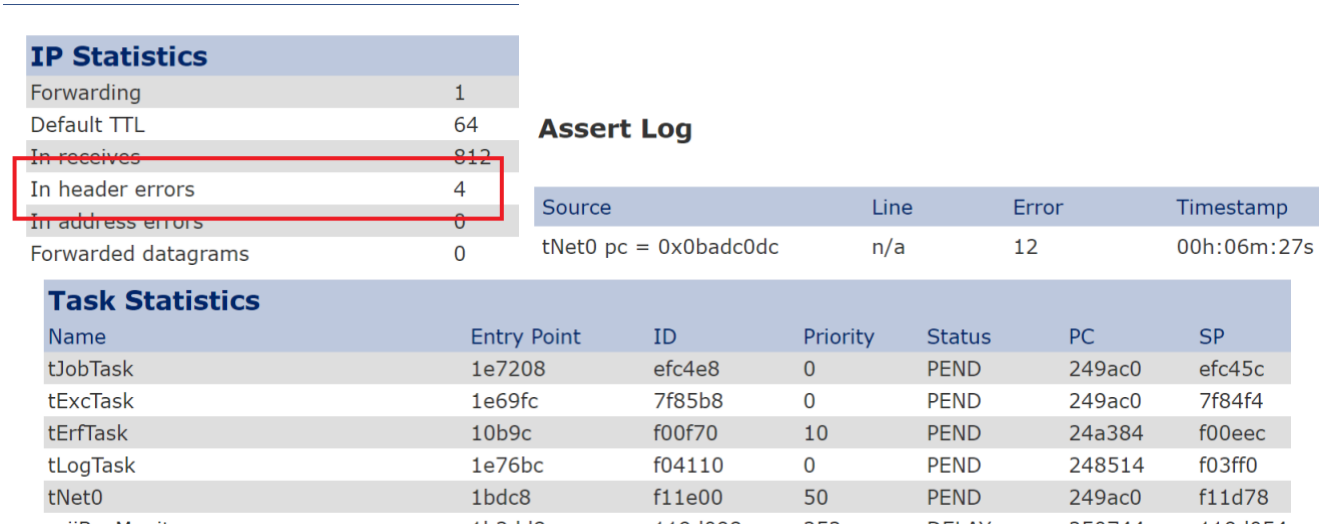
1. Unpack the .STU project file as a regular PKZip archive.
2. Inside is a `BinAppli` folder holding the APX file, extract this file.
3. While the APX format is proprietary and discussing it in-depth is beyond the scope of this report, it is sufficient to know that the code blocks are stored as plain binary ARM code in the file.
4. As such, forensic analysts can run byte-wise differential analysis between a known-good version of the APX file and a suspected malicious one, drilling down on any differences occurring in code areas.
5. Additionally, analysts could run shellcode detection tools on the APX file to detect signs of unusual or malicious ARM code. While such detectors cannot rely on signatures of regular ARM shellcode (we are dealing with specialized payloads tailored to VxWorks after all), certain patterns such as GetPC code, egghunters, and self-decoders might be present and detectable.

If the attacker has relocated their implant from the code blocks and cleaned up afterwards, however, no such forensic artifacts will be present in the uploaded data.

7.4. Implant on Allen-Bradley 1756-EN2*

The Allen-Bradley 1756-EN2* series of Ethernet modules, as well as several other Allen-Bradley communication modules and network-enabled compact PLCs, typically include a web interface which has several diagnostic overview pages, as shown in Figure 36.

Several of these pages can be monitored for indicators of exploitation or implantation as described in this report. In particular, unsuccessful exploitation attempts or buggy implant functionality resulting in crashes will show up in the **Assert Log**, while malformed packet errors will show up in the **IP, TCP and UDP statistics**. If an implant on the module wishes to establish a foothold through spawning a stable RTOS task, this task will show up in the **Task Statistics** page. All of these pages could be monitored for unusual behavior, provided the attacker implant does not implement rootkit functionality (T0851) to hide these indicators and provided the web interface is enabled (which is a security/functionality trade-off).



IP Statistics

Forwarding	1
Default TTL	64
In receives	812
In header errors	4
In address errors	0
Forwarded datagrams	0

Assert Log

Source	Line	Error	Timestamp
tNet0 pc = 0x0badc0dc	n/a	12	00h:06m:27s

Task Statistics

Name	Entry Point	ID	Priority	Status	PC	SP
tJobTask	1e7208	efc4e8	0	PEND	249ac0	efc45c
tExcTask	1e69fc	7f85b8	0	PEND	249ac0	7f84f4
tErfTask	10b9c	f00f70	10	PEND	24a384	f00eec
tLogTask	1e76bc	f04110	0	PEND	248514	f03ff0
tNet0	1bdc8	f11e00	50	PEND	249ac0	f11d78

Figure 36 – Allen-Bradley 1756-EN2T/D web interface indicators

References

[1] P. Nesterov, N. Komarov and A. Muravitsky, "The secrets of Schneider Electric's UMAS protocol," [Online]. Available: https://ics-cert.kaspersky.com/publications/reports/2022/09/29/the-secrets-of-schneider-electrics-umas-protocol/.

[2] Airbus, "Applying a Stuxnet Type Attack to a Modicon PLC," [Online]. Available: https://www.cyber.airbus.com/applying-a-stuxnet-type-attack-to-a-modicon-plc-2/.

[3] N. Miles, "Examining Crypto and Bypassing Authentication in Schneider Electric PLCs (M340/M580)," [Online]. Available: https://medium.com/tenable-techblog/examining-crypto-and-bypassing-authentication-in-schneider-electric-plcs-m340-m580-f37cf9f3ff34.

[4] Digital Bond, "Project Basecamp," [Online]. Available: https://github.com/digitalbond/Basecamp.

[5] J. Rittle, "Schneider Electric Modicon M580 UMAS Improper Authentication Vulnerability," [Online]. Available: https://talosintelligence.com/vulnerability_reports/TALOS-2018-0741.

[6] G. Jian, "Going Deeper into Schneider Modicon PAC Security," Hack in the Box, [Online]. Available: https://conference.hitb.org/hitbsecconf2021sin/materials/D1T2%20-%20Going%20Deeper%20into%20Schneider%20Modicon%20PAC%20Security%20-%20Gao%20Jian.pdf.

[7] G. Kaufmann and B. Seri, "ModiPwn," [Online]. Available: https://www.armis.com/research/modipwn/.

[8] B. Hadad, G. Kauffman and B. Seri, "Exploring and Exploiting Programmable Logic Controllers with URGENT/11 Vulnerabilities," [Online]. Available: https://info.armis.com/rs/645-PDC-047/images/Armis-URGENT11-on-OT-WP.pdf.

[9] MITRE, "Stuxnet," [Online]. Available: https://attack.mitre.org/software/S0603/.

[10] MITRE, "Triton," [Online]. Available: https://attack.mitre.org/software/S1009/.

[11] R. Spenneberg, M. Brüggemann and H. Schwartke, "PLC-Blaster: A Worm Living Solely in the PLC," Black Hat Asia, 2016. [Online]. Available: https://www.blackhat.com/docs/asia-16/materials/asia-16-Spenneberg-PLC-Blaster-A-Worm-Living-Solely-In-The-PLC-wp.pdf.

[12] N. Brubaker, K. Lunden, K. Proska, M. Umair, D. Zafra, C. Hildebrandt and R. Caldwell, "INCONTROLLER: New State-Sponsored Cyber Attack Tools Target Multiple Industrial Control Systems," Madiant, [Online]. Available: https://www.mandiant.com/resources/blog/incontroller-state-sponsored-ics-tool.

[13] D. dos Santos, C. Speybrouck and E. Costante, "Cybersecurity in Building Automation Systems," Forescout, [Online]. Available: https://www.forescout.com/resources/bas-research-report-the-current-state-of-smart-building-cybersecurity-2/.

[14] S. Brizinov, "The Race To Native Code Execution In PLCs," [Online]. Available: https://www.youtube.com/watch?v=r-dmxU1gEl0.

[15] M. Krotofil, "Evil Bubbles or How to Deliver Attack Payload via the Physics of the Process (and How to Defend against such Attacks)," Black Hat , 2017. [Online]. Available: https://www.blackhat.com/docs/us-17/wednesday/us-17-Krotofil-Evil-Bubbles-Or-How-To-Deliver-Attack-Payload-Via-The-Physics-Of-The-Process.pdf.

[16] D. Atch and G. Lashenko, "Exfiltrating Reconnaissance Data From Air-Gapped Ics/Scada Networks," Black Hat Europe, 2017. [Online]. Available: https://www.blackhat.com/docs/eu-17/materials/eu-17-Atch-Exfiltrating-Reconnaissance-Data-From-Air-Gapped-Ics-Scada-Networks.pdf.

[17] M. Krotofil and R. Derbyshire, "Greetings from the '90s: Exploiting the Design of Industrial Controllers in Modern Settings," Black Hat Europe, 2021. [Online]. Available: https://i.blackhat.com/EU-21/Wednesday/EU-21-Krotofil-Greetings-from-the-90s-Exploiting-the-Design-of-Industrial-Controllers-in-Modern-Settings.pdf.

[18] A. Abbasi and M. Hashemi, "Ghost in the PLC: Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack," Black Hat Europe, 2016. [Online]. Available: https://www.blackhat.com/docs/eu-16/materials/eu-16-Abbasi-Ghost-In-The-PLC-Designing-An-Undetectable-Programmable-Logic-Controller-Rootkit-wp.pdf.

[19] J. Wetzels and M. Krotofil, "A Diet of Poisoned Fruit: Designing Implants," TROOPERS, 2019. [Online]. Available: https://troopers.de/downloads/troopers19/TROOPERS19_NGI_IoT_diet_poisoned_fruit.pdf.

[20] L. Garcia, F. Brasser, M. Cintuglu, A. Sadeghi, O. Mohammed and S. Zonouz, "Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit," NDSS, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/hey-my-malware-knows-physics-attacking-plcs-physical-model-aware-rootkit/.

[21] M. Davis, "SmartGrid Device Security: Adventures in a new medium," Black Hat, 2009. [Online]. Available: https://www.blackhat.com/presentations/bh-usa-09/MDAVIS/BHUSA09-Davis-AMI-SLIDES.pdf.

[22] R. Santamarte, "SCADA Trojans: Attacking the Grid," RootedCON, 2011. [Online]. Available: https://www.slideshare.net/rootedcon/rubn-santamarta-scada-trojans-attacking-the-grid-rooted-con-2011-7351226.

[23] A. Bolshev and G. Cherbov, "DTM Components: Shadow Keys to the ICS Kingdom," Black Hat Europe, 2014. [Online]. Available: https://www.blackhat.com/docs/eu-14/materials/eu-14-Bolshev-DTM-Components-Shadow-Keys-To-The-ICS-Kingdom.pdf.

[24] C. Sistrunk and A. Crain, "Master Serial Killer," 2014. [Online]. Available: https://www.slideshare.net/chrissistrunk/master-serial-killer-def-con-22-ics-village.

[25] S. Brizinov, "Evil PLC Attacks - Weaponizing PLCs," DEF CON, 2022. [Online]. Available: https://www.youtube.com/watch?v=pNNOUiR8EQo.

[26] "IEC 61511," [Online]. Available: https://en.wikipedia.org/wiki/IEC_61511.

[27] "IEC 61509," [Online]. Available: https://en.wikipedia.org/wiki/IEC_61508.

[28] NIST, "SP800-82 - Guide to Industrial Control Systems (ICS) Security," [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final.

[29] Center for Chemical Process Safety, "Guidelines for Safe Automation of Chemical Processes," [Online]. Available: https://www.wiley.com/en-us/Guidelines+for+Safe+Automation+of+Chemical+Processes%2C+2nd+Edition-p-9781118949498.

[30] OT Cybersecurity, "Interfaced or integrated?," [Online]. Available: https://otcybersecurity.blog/2020/05/10/interfaced-or-integrated-2/.

[31] V. Hirsch, J. Gummersbach and T. Bartsch, "Safety & Security - Is a Physically Separation of the SIS Necessary?," [Online]. Available: https://www.aidic.it/cet/16/48/113.pdf.

[32] Schneider Electric, "Modicon Controllers Platform Cyber Security Reference Manual," [Online]. Available: https://download.schneider-electric.com/files?p_enDocType=User+guide&p_File_Name=EIO0000001999.09.pdf&p_Doc_Ref=EIO0000001999.

[33] Schneider Electric, "Safe and secure process automation with Modicon M580 Safety," [Online]. Available: https://download.schneider-electric.com/files?p_Doc_Ref=998-20424516.

[34] Emerson, "DeltaV SIS™ and Cybersecurity," [Online]. Available: https://www.emerson.com/documents/automation/white-paper-deltav-sis-cybersecurity-en-1262078.pdf.

[35] Siemens Energy, "Key cyber-security controls for reliable pipeline operation," [Online]. Available: https://assets.siemens-energy.com/siemens/assets/api/uuid:b3b1da61-d4d3-46fa-a43d-bff249b75835/pipelinetechjournal-key-cyber-security-controls-pipelines-articl.pdf.

[36] R. Prew, "Integrated but separate," ABB, [Online]. Available: https://library.e.abb.com/public/a5030e895b0a64c6c125728800508843/88-92%20%20SRAS26_72dpi.pdf.

[37] M. Szekely and K. Temkin, "Fusee Launcher," [Online]. Available: https://github.com/Qyriad/fusee-launcher.

[38] "Open-source jailbreaking tool for many iOS devices," [Online]. Available: https://github.com/axi0mX/ipwndfu.

[39] I. Zhuravlev and J. Boone, "Zephyr and MCUboot Security Analysis," [Online]. Available: https://research.nccgroup.com/wp-content/uploads/2020/05/NCC_Group_Zephyr_MCUboot_Research_Report_2020-05-26_v1.0.pdf.

[40] J. Rittle, "Schneider Electric Modicon M580 UMAS Strategy File Write Vulnerability," [Online]. Available: https://talosintelligence.com/vulnerability_reports/TALOS-2018-0742.

[41] J. Klick, S. Lau, D. Marzin, J. Malchow and V. Roth, "Internet-Facing PLCs - A New Back Orifice," Black Hat, 2015. [Online]. Available: https://www.blackhat.com/docs/us-15/materials/us-15-Klick-Internet-Facing-PLCs-A-New-Back-Orifice.pdf.

[42] R. Stark, "PLC's: Backdoors in Disguise," S4, [Online]. Available: https://www.youtube.com/watch?v=QrFzzR7HfVk.

[43] C. Miller and C. Valasek, "Remote Exploitation of an Unaltered Passenger Vehicle," Black Hat, 2015. [Online]. Available: https://packetstormsecurity.com/files/download/142307/Remote-Car-Hacking.pdf.

[44] Computest, "The Connected Car: Ways to get unauthorized access and potential implications," 2018. [Online]. Available: https://www.computest.nl/nl/knowledge-platform/rd-projects/car-hack/.

[45] Tencent Keen Security Lab, "Car Hacking Research: Remote Attack Tesla Motors," 2016. [Online]. Available: https://keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/.

[46] IEC, "IEC 61508 - Functional safety of electrical/electronic/programmable electronic safety-related systems," [Online]. Available: https://webstore.iec.ch/publication/22273.

[47] HSE, "Cyber Security for Industrial Automation and Control Systems (IACS) Edition 2," [Online]. Available: https://www.hse.gov.uk/foi/internalops/og/og-0086.pdf.

[48] NERC, "CIP-005-6 — Cyber Security – Electronic Security Perimeter(s)," [Online]. Available: https://www.nerc.com/pa/Stand/Reliability%20Standards/CIP-005-6.pdf.

[49] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams and A. Hahn, "Guide to Industrial Control Systems (ICS) Security," NIST, [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final.

[50] FDOT, "Bridge Maintenance Reference Manual," [Online]. Available: https://www.fdot.gov/maintenance/bmrm.shtm.

[51] Gartner, "Operational Technology (OT)," [Online]. Available: https://www.gartner.com/en/information-technology/glossary/operational-technology-ot.

[52] PERA, "PERA Enterprise Integration Web Site," [Online]. Available: http://www.pera.net/.

[53] CiA, "CAN in Automation," [Online]. Available: https://www.can-cia.org/.

[54] Forescout, "OT:ICEFALL," [Online]. Available: https://www.forescout.com/research-labs/ot-icefall/.

[55] Schneider Electric, "Modicon," [Online]. Available: https://www.se.com/in/en/about-us/events/modicon.jsp.

[56] Schneider Electric, "Security Notification - Modicon Controllers," [Online]. Available: https://www.se.com/ww/en/download/document/SEVD-2023-010-05/.

[57] Schneider Electric, "Security Notification - Modicon Controllers," [Online]. Available: https://www.se.com/ww/en/download/document/SEVD-2023-010-06/.

[58] Shodan.io, [Online]. Available: https://www.shodan.io/search?query=Schneider+Electric+BMX.

[59] Schneider Electric, "Modicon M340," [Online]. Available: https://www.se.com/ww/en/product-range/1468-modicon-m340/#overview.

[60] Control Global, "Modern controls, system integrator automate rock crushing," [Online]. Available: https://www.controlglobal.com/control/plcs-pacs/article/11299605/modern-controls-system-integrator-automate-rock-crushing.

[61] Forescout, "Project Memoria," [Online]. Available: https://www.forescout.com/research-labs/project-memoria/.

[62] Forescout, "NUCLEUS:13," [Online]. Available: https://www.forescout.com/research-labs/nucleus-13/.

[63] International Society of Automation, "ISA-62443-2-1-2009, Security for Industrial Automation and Control Systems Part 2-1: Establishing an Industrial Automation and Control Systems Security Program," [Online]. Available: https://www.isa.org/products/isa-62443-2-1-2009-security-for-industrial-automat.

[64] Ladder Logic World, "PLC Manufacturers: The Latest PLC Brands, Rankings & Revenues," [Online]. Available: https://ladderlogicworld.com/plc-manufacturers/.

[65] R. Kumar, "An overview of the global PLC industry and its dynamics," [Online]. Available: https://medium.com/world-of-iot/95-an-overview-of-the-global-plc-industry-and-its-dynamics-102ae2cfc0b4.

[66] T. Dawson, "Who Were the Leading Vendors of Industrial Controls in 2017?," Interact Analysis, [Online]. Available: https://web.archive.org/web/20200920044648/https://www.interactanalysis.com/who-were-the-leading-vendors-of-industrial-controls-plcs-and-dcs-in-2017/.

[67] D. Peterson, "Is The Purdue Model Dead?," [Online]. Available: https://dale-peterson.com/2019/02/11/is-the-purdue-model-dead/.

[68] U. Katz, "Top-Down and Bottom-Up: Exploiting Vulnerabilities In the OT Cloud Era," [Online]. Available: https://claroty.com/team82/research/exploiting-vulnerabilities-in-the-ot-cloud-era.

[69] CISA, "ICS Advisory (ICSA-23-012-03) - InHand Networks InRouter," [Online]. Available: https://www.cisa.gov/uscert/ics/advisories/icsa-23-012-03.

[70] B. Johnson, D. Caban, M. Krotofil, D. Scali, N. Brubaker and C. Glyer, "Attackers Deploy New ICS Attack Framework “TRITON” and Cause Operational Disruption to Critical Infrastructure," [Online]. Available: https://www.mandiant.com/resources/blog/attackers-deploy-new-ics-attack-framework-triton.

[71] A. Bolshev and J. Larsen, "A Rising Tide: Design Exploits in Industrial Control Systems," [Online]. Available: https://www.usenix.org/conference/woot16/workshop-program/presentation/bolshev.

[72] M. Krotofil and C. Sistrunk, "“MAN-IN-THE-SCADA”: Anatomy of Data Integrity Attacks in Industrial Control Systems," Black Hat, 2017. [Online]. Available: https://www.blackhat.com/docs/asia-17/materials/asia-17-Krotofil-Man-In-The-SCADA-Anatomy-Of-Data-Integrity-Attacks-In-Industrial-Control-Systems.pdf.

[73] J. Larsen, "Miniaturization," Black Hat, 2014. [Online]. Available: https://www.blackhat.com/docs/us-14/materials/us-14-Larsen-Miniturization-WP.pdf.

[74] M. Krotofil, A. Cardenas, J. Larsen and D. Gollman, "Vulnerabilities of cyber-physical systems to stale data—Determining the optimal time to launch attacks," International Journal of Critical Infrastructure Protection, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1874548214000638.