Post-exploitation and Persistence Techniques Against Programmable Logic Controller

Andrei Bytes(✉) and Jianying Zhou

Singapore University of Technology and Design, Singapore, Singapore
andrei_bytes@mymail.sutd.edu.sg, jianying_zhou@sutd.edu.sg

Abstract. The rising appearance of system security threats against real-world Critical Infrastructure (CI) sites over the past years brought significant research attention into the security of Industrial Control Systems (ICS). Academic institutions and major industrial appliance vendors have since increased efforts on effective vulnerability discovery in these systems. However, from the investigation of the major recent ICS incidents, it is evident that a targeted post-exploitation chain plays a crucial role for an attack to succeed. After the initial access to the system is gained, typically through a previously unknown (zero-day) or unpatched vulnerability, weak credentials or insider assistance, a specific knowledge on the system architecture is applied to achieve stealthy and persistent presence in the system before the physical process is disrupted. In this work, we propose a set of post-exploitation and persistence techniques against WAGO PFC200 Series Programmable Logic Controller (PLC). It will help to raise the awareness of stealthy and persistent threats to PLCs built on top of the variations of CODESYS runtime.

Keywords: Programmable Logic Controller · Vulnerability discovery · Industrial control system security

1 Introduction

Industrial Control Systems (ICS) are widely deployed to control and supervise the safe operation of nation-wide critical infrastructure: electric power grid, water treatment and distribution, transportation. Numerous domains of modern life and economy rely on real-time stability, safety and security of ICS. Programmable Logic Controllers (PLC), as well as sensors and actuators, play a key role in ICS as field devices.

The importance of security research in ICS domain has become especially evident after a series of major security incidents which relied on the exploitation of PLCs and other industrial appliance to corrupt the control logic and therefore affect the physical process [7,22,63]. The IEC61131 standard [15] defines the programming languages and system operation requirements to be followed by PLCs, to fulfill highest safety and security standards in the physical process

J. Zhou et al. (Eds.): ACNS 2020 Workshops, LNCS 12418, pp. 255–273, 2020.
https://doi.org/10.1007/978-3-030-61638-0_15

implementation. A key difference of PLCs from traditional embedded systems is their real-time operation design, extended fault tolerance and redundancy capabilities [27]. A single corruption of physical process control logic or availability shortage, even on the software level, can have serious consequences, ranging from immediate financial loss to man-made environmental disaster.

In this work, we analyze the software internals of widely used WAGO PFC200 Series PLCs and provide a set of potentially applicable techniques which allow for post-exploitation and persistence in these devices. While the methodology is provided in relation to WAGO PLCs, a similar approach is potentially helpful for the vulnerability discovery research in other ICS products, which are built on top of the variations of CODESYS runtime. This includes more than 360 devices from Hitachi, Advantech, Schneider Automation, ABB, Bosch Rexroth and other vendors [19,42].

Contributions: We perform a study of the popular WAGO PFC200 Series (750-8202/025-001) controller and identify a set of targeted post-exploitation techniques which can be applied on the firmware components to support the attack payload persistence. We also discuss the options of detection and defence, which take into account the internal components of the above-mentioned system.

Organization: The remainder of this paper is organized as follows. Section 2 introduces the typical ICS architecture and PLC components. Section 3 formulates the threat model of the system being studied. Section 4 provides an overview of recent exploits which assist in unauthorized access to the controller, and propose a set of techniques for post-exploitation and persistence. Section 5 gives two examples of attack scenarios that leverage the post-exploitation and persistence techniques, and Sect. 6 discusses the options of detection and defence. Section 7 reviews the related work, and Sect. 8 concludes the paper.

2 Background

2.1 WAGO PFC200 Series PLC

The PFC200 Controller, produced by WAGO Kontakttechnik is built on relatively powerful hardware (ARM Cortex CPU, 256 Mbytes RAM) and a modern software stack (real-time operating system (RTOS) with a modified Linux kernel). The controller implements a variety of well-known and vendor-proprietary network protocols for remote management, `I/O` connectivity and general networking.

This hardware and software design allows for rich connectivity (two embedded webservers, on-board Java HMI, extensions, integration with "cloud" backends, interconnection with mobile applications over TCP) and relatively high security standards (SSL, SSH, OpenVPN, firewall) offered out-of-the-box.

Software: A key advantage of WAGO PFC200 Series controller for security research is the access to the privileged user and unrestricted access to the kernel space. A typical identification string of WAGO PFC200 is as follows: `Linux PFC200-450B5E 4.9.115-rt93-w02.03.00_02+2 #2 PREEMPT RT armv7l GNU/Linux`.

The target architecture is ARMv7. The key modifications to the Linux 4.9 kernel are provided by the real-time kernel patchset [51] (Preempt RT). In particular, it extends the mainline kernel with additional preemption models, changes the task scheduling and priority policies. This turns the PFC200 system into a real-time OS (RTOS).

2.2 Firmware Availability

A significant advantage of WAGO controllers from a research perspective is the public availability of the Board Support Package (BSP) for its firmware [43]. In embedded systems, BSP is a common way to provide the developers with essential tools to cross-compile the firmware for a given hardware platform [28,31].

While there are many closed-source components (such as, system daemons and runtime software), the BSP contains a root file system, a compiler toolchain for ARM instruction set, hardware-specific drivers, kernel modules, configuration utilities and documentation. A native customization utility also allows to include the components and routines which facilitate the dynamic analysis of the firmware, such as tracing and debugging tools. The build process consists of the following chain:

- PTXdist, a build system for creating Embedded Linux distributions
- Pengutronix build environment, optimized for PFC controllers
- OSELAS toolchain for ARM cross-compilation
- Other utilities and components to build the firmware image
- WAGO rule-sets and configuration scenarios for CODESYS runtime

As compared to firmware analysis routine for ICS products from other major vendors, such as Rockwell Automation Allen-Bradley [16] or Siemens SIMATIC [17,18], the availability of BSP for WAGO PFC200 significantly extends the opportunity for the analysis of exploitation context.

Generic Runtime System: CODESYS (Controller Development System) [35] is widely adopted by ICS device vendors as a generic, portable third-party runtime software component responsible for control program execution [42]. In addition to the actual execution environment on field devices, CODESYS includes its own IDE (Integrated Development Environment) to construct IEC61131 compatible control logic applications. Control programs use proprietary binary format produced by a built-in compiler. CODESYS also ships with multiple emulation tools and supports a broad set of extensions which can be called from the control project as external libraries [23,35].

Numerous variations of CODESYS runtime are reportedly being used in at least 20% of PLCs worldwide [19]. The device directory list includes more than 360 devices from Hitachi, Advantech, Schneider Automation, ABB, Bosch Rexroth, Owen, Berghof Automation [19,42]. Typically, ICS vendors use it as a white-label platform for their own-branded IDE and device firmware. This implies an extensive amount of customization with additional software components to be introduced in the end products.

Adoption in WAGO PLC: The WAGO PFC200 firmware embeds the CODESYS runtime for control program deployment and execution on top of its real-time operating system (RTOS) and a customized Linux kernel [44].

The newest generation of CODESYS-based runtime, named WAGO e!RUNTIME, supports the latest generation of WAGO PFC200 Series controllers. The documentation clearly specifies that this runtime system is based on the original CODESYS v3, with the extended functionality introduced to it by WAGO [37]. In addition to the conventional CODESYS IDE, PFC100 and PFC200 controllers can be programmed and configured with WAGO e!COCKPIT - the desktop software also provided by WAGO, which is CODESYS v3 compatible and potentially can be used to work with non-WAGO controllers which have the non-customized revision of v3 runtime system in their firmware.

Compatibility Mode: The above mentioned PFC100 and PFC200 series of WAGO controllers can be switched to using a previous generation of the WAGO runtime, WAGO-I/O-SYSTEM 750. The documentation states that this system is based on CODESYS v2.3 and is incompatible with the newer IDE [37]. For WAGO-I/O-SYSTEM 750 runtime, an older development from WAGO should be used, named WAGO-IO-PRO. The latter has significantly lower memory and computational power requirements and is also used with low-end, non-ARM controller series [37,39].

2.3 Vendor-Specific Components

To support and tweak the runtime WAGO system includes a set of additional configuration interfaces and remote controller management tools. These also allow for physical process visualization, native integration of remote back-ends, and recently introduced "cloud" IoT connectivity [41,45,46].

One of the key toolsets, "WAGO CBM", provides a Command-Line Interface (CLI) which is capable for essential hardware and network configuration on the controller.

An embedded Human Machine Interface (HMI) system, "WebVisu", can execute Java applets, downloaded to the controller by the IDE to allow the developers to build dynamic HMI screens straight from the control application programming environment and expose them as web services on the PLC.

A separate embedded web server on the PLC runs WAGO Web-Based Management System (WBM) - a configuration and remote control wrapper built on top of the CBM toolset mentioned earlier.

Such a wide variety of custom pre-installed components motivate us to perform research on how to find opportunities for post-exploitation of the PLC and how the persistent payload can be hidden among these services to evade detection.

3 Threat Model

For the exploitation cycle analysis, we consider a threat model which is described below.

A remote attacker has a generic exploit in possession which utilises a previously unknown or unpatched vulnerability of the controller. However, preliminary knowledge about the particular deployed system and the physical process is limited. The threat actor obtains access to the system through one of the remotely exposed interfaces and seeks for long-term stealthy persistence in the system which would allow gathering a sufficient set of operational information to proceed with a targeted attack against the physical process.

4 Methodology

Objectives: A threat actor who has established the one-time unauthorized access to the system is motivated to ensure the long-term stealthy presence of the malicious payload on the PLC. This allows for passive observation on the system and its state changes, fingerprinting of the deployed configuration and ensuring the remote access in future.

Techniques: A common way to support these attack objectives at the PLC in the operational state is to place a persistent backdoor into one of its software components. In this section, we provide an overview of the attack cycle and system components which can perform invocation of the malicious payload and are suitable for placement of the backdoor executable.

4.1 Obtaining the Remote Access

At the time of writing, the most recent firmware release for PFC200 revision is FW16. This version introduced several of patches for critical security vulnerabilities. Supporting each other in a chain, these vulnerabilities facilitate a remote attacker to gain remote unauthorized access to the controller. We analyze multiple vulnerability chains below to provide an overview of unauthorized access scenarios.

Vulnerabilities in Management Interfaces: In [47], an exposed Web-Based Management (WBM) component demonstrates a serious authentication flaw which can be exploited to reduce the password trial entropy and obtain access to the configuration interface. A related authentication bypass [40] was also demonstrated in the older revision of the firmware (FW10) through the CODESYS remote control component exposed via TCP port.

Authentication Flaws: Similarly, in [52], a vulnerable encryption mechanism caused user credentials leakage in packets sent between the WAGO e!Cockpit IDE and the PLC runtime. Due to the hard-coded encryption key, it was possible to derive login credentials for any user and perform the unauthorised access to the controller. Another attack vector against the PFC200 update mechanism from the operator workstation [53], allows burning the incorrect version of WAGO update package (WUP) with false metadata in order to downgrade the firmware revision, which has known unpatched vulnerabilities for further access bypass. Previous, older implementations of the WBM component and its companion visualisation application, WebVisu, have also been proven to have authentication bypass flaws [29,38] in older firmware revisions.

Code Execution via File Uploads: The above can be chained with [54] for arbitrary code execution through package upload to the controller. The packaging system of the PFC200 through `ipk` archives provides no integrity checks on its content and is passed to opkg activation utility which executes the injected payload with superuser privileges.

Vulnerable Extension Packages: The newly added extension for Cloud Connectivity functionality of WAGO PFC200 was exploited in [48,56,57]. A manipulated remote firmware update command string is interpreted on the controller site and passed to the CBM utility without validation. As the former runs with superuser privileges, this results in a high-privileged remote code execution.

Exploitation of Network Services: A direct exploitation of the privileged services is one of the most dangerous attack vectors for the remote attacker. In [58–62], the "I/O-Check" service which implements the WAGO service protocol and is reachable through TCP port 6626 of the PFC200 controller allowed for a heap buffer overflow with a potential code execution. The above-mentioned service protocol provides the capability to read and write data to the EEPROM of the controller, which can lead to the controlled memory corruption. Notably, this vulnerable behaviour does not require any authentication and can be invoked by an anonymous client.

In an unpatched system, a vulnerability chain similar to the scenarios described above can provide an attacker with an unauthorised access to the affected PFC200 system. Once the one-time access is established, a more specific knowledge of the system is required to perform post-exploitation operations and prepare the attack against the physical process.

4.2 Privilege Escalation Techniques

Access Control System: The access control in WAGO PFC200 implements customized, vendor-built procedures which apply in multiple contexts of the controller configuration invoked by CBM and WBM utilities. As mentioned in the documentation, user management in the custom access control system is isolated from system user groups for security reasons [36]. In practice, this means that

the services which run CLI and Web configuration applications run themselves with root privileges and perform the access control validation on the application layer. This design bypasses more robust access control mechanisms which are provided by the Linux kernel, replacing it with vendor-added validation logic. Since the impact of potential vulnerabilities in CBM and WBM is no longer mitigated by system user isolation, this builds up a major privilege escalation vector.

Vulnerabilities in Privileged Services: The controller also runs a set of high-privileged system services which are not always reachable through the network. However, there are known scenarios in which such services process input files which can be tampered with by unprivileged user. Successful exploitation of file processing vulnerability in one of the privileged services bypasses the vendor-built access control gives an extensive opportunity for privilege escalation into a superuser. Thus, in [49,50], a vulnerable "WAGO IO-Check" privileged service can be exploited through a low privilege user-writable cache file in the controller filesystem. The cache file parser does not fully sanitize the retrieved arguments and allows for command injection with root privileges. Similarly, in [50], it demonstrated that the privileged process could also deliver the payload from fields in the tampered cache into `sprintf()` call without validation, resulting in a stack buffer overflow and command execution in superuser context. This introduces another vector for code execution and privilege escalation on the controller.

4.3 Gathering System Information

Logging: By default, the PFC200 controller is configured with multiple logging services. A wide range of debug information in PFC200 is populated into log files in different locations. The data retrieved from log files can be used in a post-exploitation stage to determine the controller runtime state, network events, date and time patterns of operator assistance and firmware updates. Types of information and log file locations are summarized in Table 1.

Runtime Configuration: The CODESYS runtime on the controller writes its state information into multiple configuration files on the controller filesystem. From the `[SysFileMap]` of the `eRUNTIME.cfg` configuration file located under the home path of the `codesys_root` user, a list of useful state file mappings can be determined. From `Project.xml`, it is possible to identify the state of configured modules of the control program project, initialized names and values of local and global variables. The timestamps contained in `ProjectCfg.txt` allow to identify when the latest configuration update for the control program was performed.

Thus, the `PlcLogic` path under `codesys_root` u hosts multiple status files related to the currently uploaded control program.

Current mapping of the hardware indicators on the front panel of the controller is written to `/tmp/led.xml` and `/var/www/wbm/led.xml`.

More information about the exposed management interfaces of the controller, mode of authentication, MODBUS and serial port initialization can be obtained at `/etc/rts3s.cfg`.

The configuration of the embedded webserver, embedded into the controller runtime, can be found at `CODESYSControl.cfg,CmpWSServer.cfg,/etc/webserver_conf.xml`.

In [55], the abuse of the concurrent process pool limitation set by these configuration files was demonstrated, leading to a denial of service attack against the controller management interface.

Table 1. Logged information of PFC200

Useful information	Location
booted runtime mode	`/var/run/runtime`
runtime init log	`/tmp/runtime_state.log`
hardware port mapping	`/var/run/ifstate`
Per-thread trace log: OPC UA, MODBUS events	`/var/log/runtime`
WAGO events and diagnostic information	`/var/log/wago/wagolog.log`
Booted firmware revision	`/etc/REVISIONS`
Firmware update log	`/log/fwupdate.log`
latest executed privileged commands	`/var/log/sudo.log`
WAGO CBM calls, firewall rules and state transitions	`/var/log/messages`
PLC boot events	`/home/check-system/events.log`

Analysis of the Control Program: The control logic, compiled by the WAGO e!Cockpit IDE can be retrieved from `PlcLogic/Application/Application.app` binary in the `codesys_root` user home path in the controller. Alternatively, in the older firmware, the binary is deployed as `DEFAULT.PRG` to the home path of `codesys_root` on the controller. If such option is enabled, the IDE can include the full source code of the control program which can be retrieved from `source.dat` binary file in the same path.

If the source file is lacking due to the project deployment configuration, an analysis of the compiled program binary can be done based on the techniques and file layout described in [19]. Re-construction of the control flow allows for context-aware post-exploitation payload generation.

Visualisation Applet Extraction: The PFC200 supports embedded rendering of visualisation applets, assembled and deployed to the controller by a CODESYS-based IDE. Extraction of `webvisu.jar` is possible from the home path of `codesys` user. Further decompilation and analysis of this applet give additional information about the control components and prioritized metrics, performed on the physical process.

4.4 Persistence

One of the primary objectives for the malicious code, deployed into the PLC is persistence - an ability to re-execute the payload in the affected system regardless of possible reboots, control switches to and from other PLCs by the fault-tolerance logic, planned and unexpected power cycles.

To support this operation, the malicious payload performs a set of modifications to the selected components system. A number of system components in PFC200 provide an opportunity to establish execution persistence.

Aiming to the long-term, passive presence of a deployed payload, modification of default but vendor-specific components of the PLC firmware places lower detection risk as compared to generic Linux persistence techniques.

Injection to CBM Modules: WAGO CBM is a vendor-added set of command-line interface (CLI) utilities which play a key role in the controller setup, monitoring, management of its hardware and the state of CODESYS runtime. These contain a set of scripts which obey the custom access permission system [44]. Many of these utilities are not only meant to be manually invoked by the operator but provide a call interface for other software on the controller.

Table 2 lists some of the frequently invoked CBM scripts with the location path related to `/etc/config-tools/`. The systematic invocation of high-privileged CBM scripts makes them a reliable target for payload injection and system persistence.

Table 2. Frequently invoked WAGO CBM scripts

Utility name	Trigger condition
`cbm-script-modules/*`	Multiple: configuration protocol
`events/*`	Multiple: power cycle and interface `up/down`
`start_reboot`	Power cycle
`firmware_restore`	PLC boot and firmware update

Web Components: The PFC200 controller runs two groups of web applications via separate embedded web servers. Running as root, the `/usr/sbin/webserver`

is responsible for serving web components of the pure CODESYS distribution (including WebVisu visualisation applets). The `/usr/sbin/lighttpd` webserver with `/usr/bin/php-cgi` interpreter is running as www user and serves the WAGO WBM remote management utility. The deployed configuration of lighttpd also turns it into a reverse proxy, which redirects its certain routes to the CODESYS server.

This can be observed by inspecting `/etc/webserver_conf.xml`, `redirect_wbm.conf`, and host configurations under `/etc/lighttpd/`.

The important property which makes these web components attractive for persistence hook execution is extensive non-traditional privileges of the www user. With a sudoers record, a list of additional commands is granted to it. These include powerful actions like hardware devices access, firmware replacement and service configuration.

To achieve the payload persistence though the ability of its remote invocation from WBM, `/var/www/wbm/page_elements`, `/var/www/wbm/fs_utils` are suitable injection points for a malicious payload hook. To ensure that a given component can be accessed from WBM with an authenticated request, a reference can be checked against a permission rules file `wbm/paperm.inc.php`.

Process Migration Candidates: Once the superuser privileges are gained, an efficient option would be to host the persistent payload stealthily among the running processes.

A reason behind this method is an additional detection countermeasure effect. While these processes will be likely common for every PFC200 series controller which runs in a given mode of operation (CODESYS 2.5 and 3 have significant difference and are supported by PFC200 as separate modes), a generic Linux rootkit detection tool would likely not be able to attest the integrity of these binaries.

A good candidate for this is the `codesys3` process itself. In a typical configuration, it spawns 36 named threads for different jobs of its execution cycle. In particular, this includes I\O operations, MODBUS and networking, visualization thread, cycle scheduling, webserver threads, load monitoring and other systematic tasks. Many of these threads happen to be dormant, judging by consumed CPU time as their functionality or target network interface is always enabled. However, even in this case, the same number of named threads is still spawned by the runtime.

For a stealthy backdoor on the controller, this makes the `codesys3` process a right candidate for process migration.

Another potential target could be the custom vendor-built `oms.d` service which is responsible for handling the hardware button events in the controller and triggers call-backs for power on, soft and factory reset actions with root privileges.

The downside of this method is the unavailability of debugging and function hooking tools in the typical firmware build configuration. However, the board

support package [43] provides rules to include these utilities to the firmware distribution for debugging purposes.

Generic Techniques: In addition to the targeted techniques which leverage vendor-specific components in PFC200, the persistence opportunities are extended by a number of generic assets present in the firmware. These techniques are widely used by Linux malware [9] to obtain persistence in desktop platforms. Despite being very limited in available kernel modules and user-space utilities, the real-time operating system (RTOS) of WAGO PFC200 embeds a Linux kernel and multiple generic system services. This makes such techniques applicable for persistence purposes on the PLC.

Crontab Records: PFC200 actively uses Cron daemon for purposes of auxiliary system operations. If Cron daemon is available in the system, it is often possible to achieve persistence by adding a record to a given user's crontab [9] with access privileges of this user.

To reduce the risk of detection, an attacker can append the malicious payload to one of known script invocation records or forge the process name with one of the default cron jobs, preserving same invocation frequency to reflect in system logs. A suitable candidate for this in WAGO PFC200 can be the default crontab record for `logrotate` service, which is systematically called to perform the management of multiple system and event logs on the controller filesystem.

Terminal Sessions: One of the conventional persistence techniques in Linux systems is backdooring the terminal session initialization file to invoke additional commands when the user initiates the session [9].

For the purposes of local and remote in-network configuration, CLI capabilities are provided by PFC200 out-of-the-box. By default, pre-configured users of the system are also enabled to initiate Bash terminal sessions. When such session is opened, typically through the built-in Dropbear SSH server or serial interface, multiple configuration files are invoked. In particular, the following scripts are part of the terminal session invocation chain:

- `/etc/wago-screen-prompt.sh`
- `/etc/profile.passwd`
- `/etc/config-tools/get_user_info`

Altering these scripts give an additional opportunity to invoke the backdoor when a session is opened for a given user.

5 Attack Scenarios

In this section, we provide two examples of attack chains which leverage the post-exploitation and persistent techniques proposed in Sect. 4.

Example 1. A WAGO PFC200 controller runs firmware revision 03.01.07(13) which contains unpatched vulnerabilities known to the attacker. A remote attacker accesses the WBM service through the exposed TCP port and uses the authentication flaw [47] to derive login credentials (Fig. 1). The attacker crafts a malicious `ipk` package and achieves its execution in the system using the remote code execution flaw [54], resulting in privilege escalation to the superuser. Using the same vulnerability, the attacker uploads a crafted backdoor, compiled for ARMv7 instruction set [5] to the persistent partition on the PLC (Fig. 2). During the post-exploitation, the attacker analyses the logfiles located at `var/log/wago/wagolog.log`, and `/var/log/messages`. She observes that systematic maintenance is done on the controller, through command-line sessions over the serial interface. To achieve the persistence of the malicious backdoor, the attacker adds an additional record to `/etc/wago-screen-prompt.sh` to invoke the previously uploaded binary every time the operator logins (Fig. 3). The payload potentially preserves its dormant state until the second stage of the attack is activated, affecting the physical process of the plant. In the system logs and historian server data, the actions placed by the malicious implant will conform timestamps of legitimate operator actions.

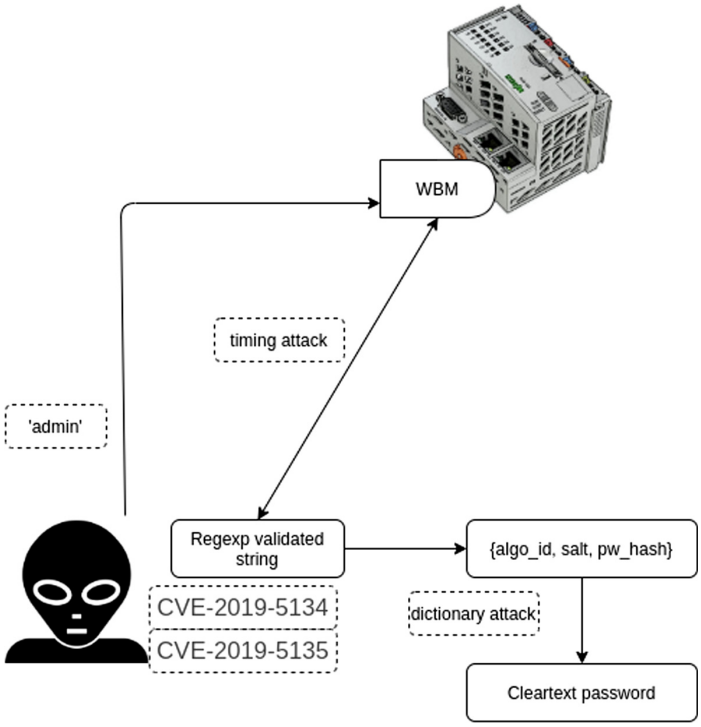

Fig. 1. E1 Stage I: Authentication bypass

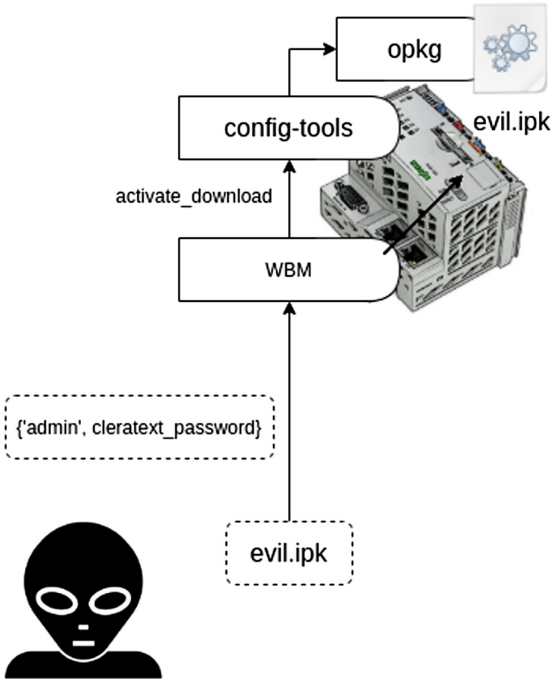

Fig. 2. E1 Stage II: Code execution

Example 2. A WAGO PFC200 controller runs firmware revision 03.01.07(13) which contains unpatched vulnerabilities known to the attacker. The attacker exploits a buffer overflow flaw [59] in WAGO service protocol reachable through TCP port 6626 on the controller. This results in arbitrary code execution with superuser privileges. The attacker crafts a malicious executable and writes it into the persistent partition on the controller. To secure the re-execution of the payload, an attacker appends the malicious executable invocation hook to the `/etc/config-tools/events/networking/update_config` event rule. This results in persistence on the system after power cycle of the PLC or its network interfaces re-configuration.

6 Discussion

We have to note that the research is significantly facilitated in the case of CODESYS runtime by having direct access to the device filesystem through a number of control interfaces and protocols. The ability to have shell access to the PLC and process monitoring utilities in the embedded OS plays an important role to understand the architecture.

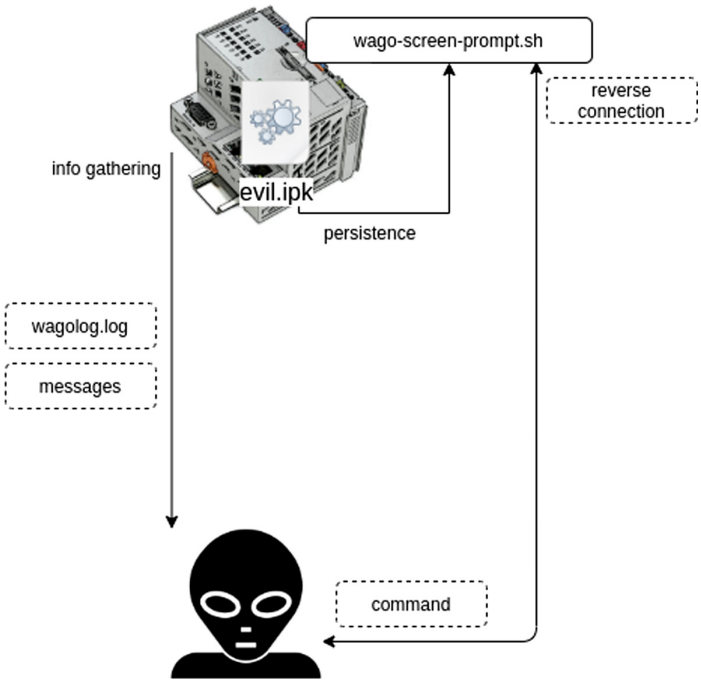

Fig. 3. E1 Stage III: Post-exploitation

Defence and Detection: The objective of a persistent backdoor placed on the affected PLC is to minimize detection risk in a long-term perspective. The assumed detection mechanisms can be categorised in the following categories:

- Generic Linux rootkit detection utilities
- Network activity anomaly detection
- System behaviour analysis
- Manual in-system investigation

In relation to the techniques described in Sect. 4, the customization to specific PLC firmware components is assumed as an advantage against detection by generic detection algorithms. Use of a generic Linux rootkit detection software would likely not be able to verify the integrity of modified vendor-specific utilities.

Assumed that the attack payload is in a dormant state but persists in the PLC system long-term, the in-network behaviour analysis mechanisms similar to [13] and [2] will not apply as there is no immediate deviation from the historical data or change of physical process state.

Applicability: In this work, we have studied the methodology in relation to RTLinux-based WAGO PLCs, which introduces vendor-specific system services to manage the CODESYS runtime. A similar approach is potentially helpful

for attack cycle research in other ICS products, which are built on top of the variations of CODESYS runtime. This includes more than 360 devices from Hitachi, Advantech, Schneider Automation, ABB, Bosch Rexroth and other vendors [19,42].

7 Related Work

From analysis of the major security ICS incidents [7,22,63], a key difference from traditional IT systems can be observed in the crucial importance of a post-exploitation chain for a successful attack. This motivates us to specifically focus on the post-exploitation stage in this work, to research its practical implication against PLCs.

Similarly, multiple works on the security of PLCs and ICS field devices [3, 10,14,20,33,34] focus on the attacks which could enable remote unauthorised access. In [27], authors survey the hardware components used in most common field devices. This provides a view on the share of ARM platforms among other hardware platforms in ICS at the time of writing. Since then, the growth of WAGO PFC200 on the market further altered this proportion.

A wider view on the internals of the CODESYS runtime used in WAGO PFC200 is given in [23]. In [1], WAGO 750-8202 controllers are used as a test target for the proposed `I/O`-aware rootkit to facilitate a stealthy attack against the physical process.

Extensive research is done with a focus on the security of the control code, executed by PLCs [6,8,11,24–26,30,32,64]. The security design challenges for PLC and other field devices were studied in [4,21].

To achieve the persistence of a malicious payload, Govil et al. demonstrate the PLC "Logic Bombs" [12]. Written in Ladder Logic or other PLC programming language, compiled and deployed, e.g. to CODESYS runtime, such code is hard to be detected in the controller operation. The trigger condition of such an implant can be constructed as a pre-defined set of physical process events, which pass the control to the malicious payload. In [9], a study is done on the effective persistence techniques used by real-world Linux malware samples.

From the defence perspective, Hsio et al. [13] have proposed an ICS security monitoring solution to reveal anomalies which can be applied to the detection of malicious rootkit activity on the PLC. In [2], authors propose to detect the attacks against the physical process using noise analysis of the field devices.

A significant contribution was made recently in the domain of reverse engineering of WAGO PFC200 CODESYS-compiled binaries. In [19], authors propose a structured way of reverse-engineering the CODESYS-compiled binaries. The proposed open-source framework is aware of the proprietary binary format and canto reconstruct the Control Flow Graph from the given binaries automatically. In the post-exploitation context, a fully automated, targeted attack generation was demonstrated against WAGO PLCs.

8 Conclusion

In this work, we proposed a set of post-exploitation and persistence techniques for WAGO PFC200 Series PLC and crafted two examples of attack scenarios. We further analyzed detection and defence options, taking into account the internal system components utilized by the persistence chain.

We highlighted that in the ICS domain, in addition to the initial vulnerabilities which provide a way to penetrate the system, the targeted post-exploitation techniques play a crucial role in the attack to succeed. Apart from attacks against PLCs, this is also relevant to a wide range of other ICS devices.

Acknowledgement. This work was partly supported by the SUTD start-up research grant SRG-ISTD-2017-124.

References

1. Abbasi, A., Hashemi, M.: Ghost in the PLC: designing an undetectable programmable logic controller rootkit via pin control attack, pp. 1–35. Black Hat, November 2016. https://research.utwente.nl/en/publications/ghost-in-the-plc-designing-an-undetectable-programmable-logic-con
2. Ahmed, C.M., et al.: Noiseprint: attack detection using sensor and process noise fingerprint in cyber physical systems. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security. ASIACCS 2018, pp. 483–497. Association for Computing Machinery, New York (2018). https://doi.org/10.1145/3196494.3196532
3. Bytes, A., Adepu, S., Zhou, J.: Towards semantic sensitive feature profiling of IoT devices. IEEE Internet Things J., (2019). https://doi.org/10.1109/JIOT.2019.2903739
4. Cardenas, A., Amin, S., Sastry, S.: Secure control: towards survivable cyber-physical systems. In: 2008 28th International Conference on Distributed Computing Systems Workshops. ICDCS 2008, pp. 495–500, June 2008
5. Casey, P., Topor, M., Hennessy, E., Alrabaee, S., Aloqaily, M., Boukerche, A.: Applied comparative evaluation of the metasploit evasion module. In: 2019 IEEE Symposium on Computers and Communications (ISCC), pp. 1–6 (2019)
6. Castellanos, J.H., Ochoa, M., Zhou, J.: Finding dependencies between cyber-physical domains for security testing of industrial control systems. ACM, December 2018. https://doi.org/10.1145/3274694.3274745
7. Cobb, P.: German steel mill meltdown: rising stakes in the internet of things (2015). https://securityintelligence.com/german-steel-mill-meltdown-rising-stakes-in-the-internet-of-things/
8. Costin, A., Zaddach, J.: Embedded Devices Security and Firmware Reverse Engineering. ResearchGate, July 2013. https://www.researchgate.net/publication/259642928_Embedded_Devices_Security_and_Firmware_Reverse_Engineering
9. Cozzi, E., Graziano, M., Fratantonio, Y., Balzarotti, D.: Understanding Linux malware. In: 2018 IEEE Symposium on Security and Privacy (SP), pp. 161–175 (2018)
10. Dragos: CRASHOVERRIDE: Analyzing the Malware that Attacks Power Grids—Dragos, April 2019. https://dragos.com/resource/crashoverride-analyzing-the-malware-that-attacks-power-grids. Accessed 14 Apr 2019

11. Garcia, L.A., Brasser, F., Cintuglu, M.H., Sadeghi, A.R., Zonouz, S.A.: Hey, my malware knows physics! Attacking PLCs with physical model aware rootkit. ResearchGate, January 2017. https://doi.org/10.14722/ndss.2017.23313
12. Govil, N., Agrawal, A., Tippenhauer, N.O.: On ladder logic bombs in industrial control systems. In: Katsikas, S.K., et al. (eds.) CyberICPS/SECPRE -2017. LNCS, vol. 10683, pp. 110–126. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-72817-9_8
13. Hsiao, S.W., Sun, Y.S., Chen, M.C., Zhang, H.: Cross-level behavioral analysis for robust early intrusion detection. In: 2010 IEEE International Conference on Intelligence and Security Informatics (ISI), pp. 95–100. IEEE (2010)
14. Huang, T., Zhou, J., Bytes, A.: ATG: an attack traffic generation tool for security testing of in-vehicle CAN bus. ResearchGate, pp. 1–6, August 2018. https://doi.org/10.1145/3230833.3230843
15. IEC 61131–3 industrial control programming standard. https://www.isa.org/standards-publications/isa-publications/intech-magazine/2012/october
16. Firmware from Rockwell Automation - Software Download, April 2019. https://www.rockwellautomation.com/rockwellsoftware/support/firmware.page. Accessed 15 Apr 2019
17. Operating System Update for SIMATIC S7–1200 CPU Firmware V3 - ID: 64789124 - Industry Support Siemens, April 2019. https://support.industry.siemens.com/cs/document/64789124/operating-system-update-for-simatic-s7-1200-cpu-firmware-v3?dti=0&pnid=13615&lc=en-WW. Accessed 15 Apr 2019
18. Support Packages for the hardware catalog in the TIA Portal (HSP) - ID: 72341852 - Industry Support Siemens, April 2019. https://support.industry.siemens.com/cs/document/72341852/support-packages-for-the-hardware-catalog-in-the-tia-portal-(hsp)?dti=0&pnid=13615&lc=en-US. Accessed 15 Apr 2019
19. Keliris, A., Maniatakos, M.: ICSREF: a framework for automated reverse engineering of industrial control systems binaries. In: The Network and Distributed System Security Symposium (NDSS) (2019)
20. Krotofil, M., Gollmann, D.: Industrial control systems security: what is happening?, pp. 664–669. ResearchGate, July 2013. https://doi.org/10.1109/INDIN.2013.6622963
21. Lee, E.A.: cyber-physical systems: design challenges. Technical report UCB/EECS-2008-8, EECS Department, University of California, Berkeley, January 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-8.html
22. Lipovsky, R.: New wave of cyber attacks against Ukrainian power industry, January 2016. http://www.welivesecurity.com/2016/01/11
23. Lufkin, D.: Programmable Logic Controllers: A Practical Approach to IEC 61131–3 using CoDeSys (12 2015)
24. McLaughlin, S.: On dynamic malware payloads aimed at programmable logic controllers, p. 10. ResearchGate, August 2011. https://www.researchgate.net/publication/262355936_On_dynamic_malware_payloads_aimed_at_programmable_logic_controllers
25. McLaughlin, S., McDaniel, P.: SABOT: specification-based payload generation for Programmable Logic Controllers, pp. 439–449. ResearchGate, October 2012. https://doi.org/10.1145/2382196.2382244
26. McLaughlin, S., Zonouz, S., Pohly, D., McDaniel, P.: A trusted safety verifier for process controller code. ResearchGate, January 2014. https://doi.org/10.14722/ndss.2014.23043

27. Mulder, J., Schwartz, M., Berg, M., Van Houten, J., Urrea, J.M., Pease, A.: Analysis of field devices used in industrial control systems. In: Butts, J., Shenoi, S. (eds.) ICCIP 2012. IAICT, vol. 390, pp. 45–57. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35764-0_4
28. Noergaard, T.: Embedded Systems Architecture: A Comprehensive Guide for Engineers and Programmers. Newnes (2013). https://books.google.com.sg/books/about/Embedded_Systems_Architecture.html?id=piGhuAAACAAJ&source=kp_book_description&redir_esc=y
29. Online: Wago-i/o-system codesys 2.3 webvisu password extraction (2019). https://packetstormsecurity.com/files/127438/WAGO-I-O-SYSTEM-CODESYS-2.3-WebVisu-Password-Extraction.html
30. Siddiqi, A., Tippenhauer, N.O., Mashima, D., Chen, B.: On practical threat scenario testing in an electric power ICS testbed. In: Proceedings of the Cyber-Physical System Security Workshop (CPSS), co-located with ASIACCS, June 2018. https://doi.org/10.1145/3198458.3198461
31. Toolchains - eLinux.org, April 2019. https://elinux.org/Toolchains. Accessed 15 Apr 2019
32. Giraldo, J., Urbina, D., Cardenas, A.A., Tippenhauer, N.O.: Hide and seek: an architecture for improving attack-visibility in industrial control systems. In: Deng, R.H., Gauthier-Umaña, V., Ochoa, M., Yung, M. (eds.) ACNS 2019. LNCS, vol. 11464, pp. 175–195. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-21568-2_9
33. Urias, V., Van Leeuwen, B., Richardson, B.: Supervisory Command and Data Acquisition (SCADA) system cyber security analysis using a live, virtual, and constructive (LVC) testbed. In: 2012 Military Communications Conference - MILCOM 2012, pp. 1–8 (2012)
34. Valentine, S., Farkas, C.: Software security: application-level vulnerabilities in SCADA systems, pp. 498–499. ResearchGate, August 2011. https://doi.org/10.1109/IRI.2011.6009603
35. Codesys. The system. https://www.codesys.com/the-system.html
36. Security for controller pfc100/pfc200 v 1.1.0, 5 December 2018. https://www.wago.com/medias/mxxxxxxxx-CyberSecurity-0en.pdf
37. Wago controllers brochure. https://www.wago.com/infomaterial/pdf/60386168.pdf
38. Wago ethernet web-based management authentication bypass vulnerability. https://ics-cert.us-cert.gov/advisories/ICSA-16-357-02
39. (May 2014). https://www.wago.com/infomaterial/pdf/51236524.pdf. Accessed 15 Apr 2019
40. Vulnerabilities in WAGO PFC 200 Series (2017). https://sec-consult.com/en/blog/advisories/wago-pfc-200-series-critical-codesys-vulnerabilities/index.html
41. (Apr 2019). https://www.wago.com/sg/download/public/IoT-Brosch%25C3%25BCre/AU-NA-DE-DE-FP-180827_001%2BIoT-Box%2BBrochure_web.pdf. Accessed 15 Apr 2019
42. Codesys device directory, April 2019. https://devices.codesys.com/device-directory.html. Accessed 15 Apr 2019
43. WAGO Global—swreg_linux_c, April 2019. https://www.wago.com/global/d/swreg_linux_c. Accessed 15 Apr 2019
44. WAGO—Controllers with Embedded Linux, April 2019. https://www.wago.com/sg/embedded-linux. Accessed 15 Apr 2019

45. WAGO—IoT PLC Controllers with MQTT Protocol for Industry 4.0, April 2019. https://www.wago.com/sg/automation-technology/plc-mqtt-iot. Accessed 15 Apr 2019
46. WAGO—WebVisu, April 2019. https://www.wago.com/global/automation-technology/discover-software/webvisu. Accessed 15 Apr 2019
47. Talos Vulnerability Report 2019–0923 (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0923
48. Talos Vulnerability Report 2019–0950 (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0950
49. Talos Vulnerability Report 2019–0961 (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0961
50. Talos Vulnerability Report 2019–0962 (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0962
51. Technical basics: Preempt RT (2020). https://wiki.linuxfoundation.org/realtime/documentation/technical_basics/start
52. WAGO e!Cockpit authentication hard-coded encryption key vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0898
53. WAGO e!COCKPIT Firmware Downgrade Vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0951
54. WAGO PFC 200 Web-Based Management (WBM) Code Execution Vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2020-1010
55. WAGO PFC100/200 Web-Based Management (WBM) FastCGI configuration insufficient resource pool denial of service (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0939
56. WAGO PFC200 Cloud Connectivity Multiple Command Injection Vulnerabilities (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0948
57. WAGO PFC200 Cloud Connectivity Remote Code Execution Vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0954
58. WAGO PFC200 iocheckd service "I/O-Check" getcouplerdetails remote code execution vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0864
59. WAGO PFC200 iocheckd service "I/O-Check" ReadPCBManuNum remote code execution vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0873
60. WAGO PFC200 iocheckd service "I/O-Check" ReadPCBManuNum remote code execution vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0874
61. WAGO PFC200 iocheckd service "I/O-Check" ReadPCBManuNum remote code execution vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0863
62. WAGO PFC200 iocheckd service "I/O-Check" ReadPSN remote code execution vulnerability (2020). https://talosintelligence.com/vulnerability_reports/TALOS-2019-0871
63. Weinberger, S.: Computer security: is this the start of cyberwarfare? Nature **174**, 142–145 (2011)
64. Zonouz, S., Rrushi, J., McLaughlin, S.: Detecting industrial control malware using automated PLC code analytics. IEEE Secur. Priv. Mag. **12**(6), 40–47 (2014). https://doi.org/10.1109/MSP.2014.113