Stealth Low-Level Manipulation of Programmable Logic Controllers I/O by Pin Control Exploitation

Ali Abbasi[1], Majid Hashemi[2], Emmanuele Zambon[1,4] *, and Sandro Etalle[1,3]

[1] Services, Cyber Security and Safety Group, University of Twente, The Netherlands,
{a.abbasi, emmanuele.zambon, sandro.etalle}@utwente.nl
[2] Quarkslab, France
mhashemi@quarkslab.com
[3] Eindhoven University of Technology, The Netherlands,
s.etalle@tue.nl
[4] SecurityMatters BV, The Netherlands
emmanuele.zambon@secmatters.com

Abstract. Input/Output is the mechanism through which Programmable Logic Controllers (PLCs) interact with and control the outside world. Particularly when employed in critical infrastructures, the I/O of PLCs has to be both reliable and secure. PLCs I/O like other embedded devices are controlled by a pin based approach. In this paper, we investigate the security implications of the PLC pin control system. In particular, we show how an attacker can tamper with the integrity and availability of PLCs I/O by exploiting certain pin control operations and the lack of hardware interrupts associated to them.

Keywords: PLC · Exploiting · SoC · ICS

1 Introduction

Programmable Logic Controllers (PLCs) are widely used today in various industries including mission critical networks that have to be both reliable and secure. One of the main purposes of the PLCs is to control the physical world equipment such as sensors, actuators or drivers within the context of an industrial process. PLCs communicate with this equipment by means of their I/O, which therefore need to be secure. Digging into their architecture, we know that the I/O interfaces of PLCs (e.g., GPIO, SCI, JTAG, etc.), are usually controlled by a so-called System on a Chip (SoC), an integrated circuit that combines multiple I/O interfaces. In turn, the pins in a SoC are managed by a pin controller, a subsystem of SoC, through which one can configure pin configurations such as

* The work of the first, third and fourth authors has been partially supported by the European Commission through project FP7-SEC-607093-PREEMPTIVE funded by the 7th Framework Program.

the input or output mode of pins. One of the most peculiar aspects of a pin controller is that its behavior is determined by a set of registers: by altering these registers one can change the behavior of the chip in a dramatic way. This feature is exploitable by attackers, who can tamper with the integrity or the availability of legitimate I/O operations, factually changing how a PLC interacts with the outside world. Based on these observations, in this paper, we introduce a novel attack technique against PLCs, which we call pin control attack. As we will demonstrate in the paper, the salient features of this new class of attacks are: (a) it is intrinsically stealth. The alteration of the pin configuration does not generate any interrupt, preventing the Operating System (OS) to react to it. (b) it is entirely different in execution from traditional techniques such as manipulation of kernel structures or system call hooking, which are typically monitored by anti-rootkit protection systems. (c) it is viable. It is possible to employ it to mount actual attacks against process control systems.

To substantiate these points, we demonstrate the capabilities offered by Pin Control attack, together with technical details and the minimal requirements for carrying out the attack in Section 3.

To demonstrate the practical feasibility of our attack technique, in Section 4 we describe the practical implementation of an attack against a PLC environment by exploiting the runtime configuration of the I/O pins used by the PLC to control a physical process. The attack allows one to reliably take control of the physical process normally managed by the PLC, while remaining stealth to both the PLC runtime and operators monitoring the process through a Human Machine Interface, a goal much more challenging than simply disabling the process control capabilities of the PLC, which would anyway lead to potentially catastrophic consequences. The attack does not require modification of the PLC logic (as proposed in other publications [19, 20]) or traditional kernel tampering or hooking techniques, which are normally monitored by Host-based Intrusion Detection systems. In Section 5.1 we discuss types of industrial network and utilities which can be affected by our attack. We also describe the consequences of the pin control attack on a specific utility. Additionally, In Section 5.2 we discuss potential mechanisms to detect/prevent Pin Control exploitation. However, because the pin configuration happens legitimately at runtime and the lack of proper hardware interrupt notifications from the SoC, it seems non-trivial to devise monitoring techniques that are both reliable and sufficiently lightweight to be employed in embedded systems. In Section 6 we discuss about similar works on attacking the I/Os. Finally we conclude our work in Section 7.

2 Background

For an attacker the ultimate objective when attacking an industrial control network is to manipulate the physical process without being detected [1] by advanced intrusion detection systems (IDS) or plant operators. Before the highly publicized Stuxnet malware, most of the attacks were trivial intrusions against the IT equipment of industrial control network. However, the Stuxnet malware

has intensified a race to the bottom where low-level attacks have a tactical advantage [2] and are therefore preferred [23]. PLCs play a significant role in the industry since they control and monitor industrial processes in critical infrastructures [13]. Successful low-level exploitation of a PLC can affect the physical world and, as a result, can have serious consequences [14]. There are few low-level techniques against PLCs which can help an attacker to reach his malicious objective. We can distinguish these techniques into the following major groups:

- *Configuration manipulation attacks:* these attacks allow an adversary to modify critical configuration parameters of a PLC (e.g. Logic) to alter the controlled process. Various research shown the feasibility of such attacks [8], [19], [20] against PLCs. To defeat this attack, the industry is using logic checksums and IDS [18], [21], [2].
- *Control-flow attacks:* in general, this category of attacks is achieved by exploiting a memory corruption vulnerability (e.g. buffer overflow), which allows the execution of arbitrary code by an adversary. Recent research has shown the possibility of control-flow attacks against PLCs [10], [27], [9]. Although several techniques have been proposed to detect or prevent control-flow attacks on general IT systems, effective countermeasures that are simultaneously applicable to the PLCs have yet to be developed.
- *Malicious code within the PLC:* possibility of remote code execution [26], [9] in a PLC paves the way for an attacker to install malicious software. To counter this problem Reeves et al. [23] proposed Autoscopy Jr. Autoscopy Jr protects the PLC kernel from malicious software which modify or hooks the functions within the OS. Autoscopy Jr limits the attacker capability to target the PLCs kernel since almost all existing malicious attack needs a function hooking to operate stealthily. Our attack does not hook any function and therefore, will be invisible to Autoscopy Jr or similar techniques.

2.1 Pin Control Subsystem

In SoCs that are used in PLCs, pins are bases that are connected to the silicon chip. Each pin individually and within the group is controlled by a specific electrical logic with a particular physical address called a register. For example, "Output Enabled" logic means that the pin is an output pin and "Input Enabled" logic means that the pin is an input pin. In PLCs these logic registers are connected to "register maps" within the PLCs SoC and can be referenced by the OS. These "Register maps" are a mere translation of physical register addresses in the SoC to reference-able virtual addresses in the PLCs operating system. The concept of controlling these mapped registers with a software is called Pin Control. Pin control mainly consists of two subsystems namely Pin Multiplexing and Pin Configuration. Pin Multiplexing is a way to connect multiple peripherals within the SoC to each individual pin. Pin configuration is a process in which the OS or an application must prepare the I/O pins before using it.

2.2 How PLCs Control the Pins

The main component of a PLC firmware is a software called *runtime*. The runtime interprets or executes process control code known as *logic*. The logic is a compiled form of the PLC's programming language, such as function blocks or ladder logic. Ladder logic and function block diagrams are graphical programming languages that describe the control process. A plant operator programs the logic and can change it when required. The purpose of a PLC is to control field equipment (i.e., sensors and actuators). To do so, the PLC runtime interacts with its I/O. The first requirement for I/O interaction is to map the physical I/O addresses (including pin configuration registers) into virtual memory. The mapping process is usually carried by the OS or PLC runtime.

After mapping pin configuration registers, the PLC runtime executes the instructions within the logic in a loop (the so-called program scan). In a typical scenario, the PLC runtime prepares for executing the logic at every loop by scanning its inputs (e.g., the I/O channels defined as inputs in the logic) and storing the value of each input in the variable table. The variable table is a virtual table that contains all the variables needed by the logic: setpoints, counters, timers, inputs and outputs. During the execution, the instructions in the logic changes only values in the variable table: every change in the I/O interfaces is ignored until the next program scan. At the end of the program scan, the PLC runtime writes output variables to the mapped I/O virtual memory that eventually is written to the physical I/O memory by the OS Kernel. Figure 1 depicts the PLC runtime operation, the execution of the logic, and its interaction with the I/O. For example, the PLC runtime will put some pins of the PLC into input mode (for inputs and reading values) and some other pins into output pins (to control the equipments).

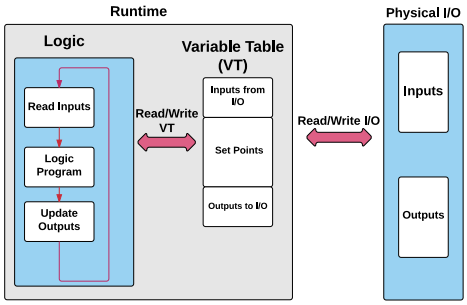

Fig. 1. Overview of PLC runtime operation, the PLC logic and its interaction with the I/O

3 Pin Control Attack

3.1 Security Concerns Regarding Pin Control

The Pin Control subsystem lacks of any hardware interrupt support. This raises a security concern about how the OS knows about the modification of pin configuration. Lack of interrupts when the configuration of a pin changes means neither the driver/kernel nor the runtime will notice about it when an attacker directly write to the pin registers. For example, in a PLC, if a pin (e.g. GPIO) that is set in Output mode gets reconfigured to Input mode by a malicious application, since there is no interrupt to alert the PLC OS about the change in pin configuration, the driver or kernel will assume that the pin is still in output mode and will attempt the write operation without reporting any error. However in reality the PLC SoC ignores the write operation (since the pin is in input mode) but will not give any feedback to the OS that the write operation was failed (because no interrupt is implemented). Therefore, the OS will assume that the write operation was successful while exactly opposite case happened. The OS then informs the PLC runtime that the operation was successful (which is wrong).

We have brought this security concern to the attention of the Linux Kernel Pin Control Subsystem group, which has confirmed our findings and its dangerous consequences for PLCs, but could not suggest a viable solution that did not require expensive artifacts such as ARM TrustZone or TPMs (Trusted Platform Module) in a PLC.

3.2 Pin Control Attack Details

A pin control attack basically consists of misusing the pin configuration functionalities of the PLCs SoC at runtime. An attacker can manipulate the values read or written to/from a peripheral by a legitimate process in the PLC without the PLC even noticing about those changes. As described in Section 3.1 if the I/O is in input mode and one tries to write to it, the I/O pin will ignore this request without raising an exception/error. Even if one change the I/O input/output status there will be no alert from the hardware for this change. Therefore, an attacker can manipulate the PLCs read and write operations to its I/Os by leveraging the configuration of pins as follows:

1. *For write operations*: if the PLC software is attempting to write a value to an I/O pin that is configured as output, the attacker reconfigures the I/O pin as input. The PLC runtime write to the input mode pin register in mapped virtual address. The PLC runtime write operation will not succeed, but the runtime will be unaware of the failure and return success since the runtime could carry out write operation in virtual memory. However, as mentioned earlier the PLCs SoC will ignore the write operation, since the configuration of the pin is in input mode.

2. *For read operations*: if the PLC software is attempting to read a value from an I/O pin that is configured as input, the attacker can reconfigure the I/O pin as output and write the value that he wishes to feed to the PLC software in the reconfigured pin.

This attack can have significant consequences due to two reasons: first, because it can be used to alter the way a PLC interacts (and possibly controls) the physical process. Second, because the attack is stealth in its nature since the PLC runtime will never notice about changes in the configuration due to lack of hardware interrupts for Pin Control system within PLCs SoC. To substantiate the feasibility of such attack we provide a practical implementation in Section 4.

3.3 Threat Model

In a critical physical process even randomly modifying the I/O can have a malicious consequence to the critical physical process. But when an attacker wants to remain stealth while maximizing the damage to the physical process then the pin control attack can be the most effective option. For pin control attack we consider three requirements which an attacker must satisfy. The requirements are the followings:

- *PLC runtime privilege:* we can envision an attacker with an equal privilege as the PLC runtime process which gives her the possibility to modify the pin configuration registers. Since the PLC runtime can modify such configuration registers, having equal privilege means that an attacker can also modify those registers. In recent years, multiple research has shown that the PLCs from multiple vendors such as Siemens [26], [3], ABB [9], Schneider Electric [11], [10], Rockwell Automation [12], and WAGO [6] are vulnerable to system-level code execution via the memory corruption vulnerabilities. Therefore, we can argue that getting equal privilege as PLC runtime is not a farreaching assumption.
- *Knowledge of the physical process:* we also assume that the attacker is aware of the physical process on the plant. Attacking critical infrastructure is usually carried out by state-sponsored attackers and usually such actors study their targets before launching their attack. For example, as reported [15] in Stuxnet [8] case, the attackers were very well aware of the physical process in their target plant. Therefore, we can assume that it is feasible for other state-sponsored attackers to study their target physical process and be aware of it.
- *Knowledge of mapping between I/O pins and the logic:* we assume that the attacker is aware of the mapping between the I/O pins and the logic. The PLC logic might use various inputs and outputs to control its process; thus, the attacker must know which input or output must be modified to affect the process as desired. The mapping between I/O pins and logic is already available in the PLC logic and therefore, an attacker can access to it within the PLC without any limitation. Additionally, the works presented by McLaughlin et al. [20], [19] can be used to discover the mapping between the different

I/O variables and the physical world. Thus we can argue that it is reasonable to assume the attacker can be aware of the mapping between I/O pins and the logic.

4 A Pin Control Attack in Practice

4.1 Environment Setup

Target device and runtime To mimic a PLC environment we choose a Raspberry Pi 1 model B as our hardware, because of the similarity in CPU architecture, available memory, and CPU power to a real PLC. For PLC runtime we use the Codesys platform. Codesys is a PLC runtime that can execute ladder logic or function block languages on proprietary hardware and provides support for industrial protocols such as Modbus and DNP3. The operating system of the Raspberry Pi is a Linux with real-time patch identical to the Wago PLC families. Currently, more than 260 PLC vendors use Codesys as the runtime software for their PLCs [6]. The combination of features offered by the Raspberry Pi and the Codesys runtime make such a system an alternative to low-end PLCs. The Raspberry Pi includes 32 general-purpose I/O pins, which represent the PLC's digital I/Os. These digital I/Os can also control analog devices by means of various electrical communication buses (e.g., SPI, I2C, and Serial) available for the Raspberry Pi.

The logic and the physical process We use pins 22 and 24 of the Raspberry Pi to control our physical process. In our logic, we declare pin 22 as the output pin and pin 24 as the input pin. In the physical layout, our output pin is connected to an LED and our input pin is connected to a button. According to our logic, the LED turns on and off every five seconds. If someone is pressing the button, the LED simply maintains its most recent state until the button is released, at which time the LED begins again to turn on and off every five seconds.

4.2 Attack Implementation

In this implementation we assume that the attacker has the same privileges as the PLC runtime. This is achievable for example by exploiting a memory corruption vulnerability that allows remote code execution, such as a buffer overflow [12], [27], [4]. A remote code execution vulnerability of such kind is known to be affecting the Codesys runtime [28]. The implementation consists of an application written in C that can be converted and used in an remote code execution exploit against a PLC runtime (in our example Codesys). The application uses /dev/mem, sysfs, or a legitimate driver call to access and configure the pins. In our target platform the Codesys runtime uses the /dev/mem for I/O access, therefore, our attack uses the same I/O interface. The application checks whether the processor I/O configuration addresses are mapped in the PLC runtime. The

list of all mapped addresses is system wide available in various locations for any user space application (e.g. via /proc/modules, or /proc/$pid/maps).

For manipulating write operations, the application needs to know a reference starting time. This is the relative time where the PLC runtime writes to the pin. While the application knows the logic and is aware that every five seconds there is a write operation to pin 22, it does not know at what second the last write operation happened. This can be easily found by monitoring the value of pin 22. Once the application intercepts the correct reference starting time, for every write operation in the logic it will carry out two tasks. First, right before the reference starting time (which is when the PLC runtime will start writing its desired original value to the I/O) the application reconfigures the pin to input mode. The Codesys runtime then attempts to write to the pin. However, the write operation will be ineffective, since the pin mode is set to input. Our application then switches the pin mode to output and writes the desired value to it. Manipulating read operations are almost identical to the write manipulation, except that the application changes the state of the pin from input to output and writes it constantly with the desired value. With this implementation, we can successfully manipulate the process. The LED turns on and off every ten seconds instead of five. Additionally, we can completely control input pin 24 and make its value 0 or 1 whenever we wanted, while the Codesys runtime was reading our desired value. Also, Codesys never notices about the failures of write operations (turning on or off the LED) whenever we modify pin configuration registers due to the fact that there is no interrupt to alert the Codesys about write operation failure. Beside that both the Codesys runtime and the PLC logic is untouched in our attack, therefore any mechanisms that checks the integrity of the PLC key components, applications and configurations would never notice about the attack.

This implementation is significantly lightweight and only causes a two percent CPU overhead. There is, however, a small chance that a race condition happens during read manipulation. However, in our tests, this race condition never happened.

5 Discussion

5.1 Implications of Attack on the ICS

The pin control attack can be used against PLCs in various utilities, but it can be considerably more powerful and stealthier when it is used against specific type of industrial processes where rapid control is not essential. The reason to choose slower processes is to reduce the chance of race conditions in the pin control attack. To understand which industrial processes mounts better for our attack we studied three different utilities namely electrical, gas, and water utilities. We created a reference taxonomy for each individual utility [22]. We identified their individual sub processes and the field equipment they most commonly used in the plant (including types of PLCs). We then confirmed our reference taxonomy by having interviews with different utilities and control engineers. According the

reference taxonomy we concluded that one of the suitable targets for pin control attack would be water utilities. Based on our interview sessions with water utility experts we identified the water distribution network as one of the most likely target for attackers who want to disrupt the water delivery service.

Disrupting the water distribution pipes The water distribution network consists of pipes, water tanks and pumps, and is instrumented with multiple sensors, such as pressure sensors and meters. Several components of the water distribution network are controlled by the water distribution automation system. One possible technique to disrupt water distribution pipes is based on creating a so-called water hammer effect. A water hammer effect is a pressure surge or wave caused when a fluid in motion is forced to stop or change direction suddenly. The water hammer effect can cause major problems, ranging from noise and vibration to pipe collapse. To mitigate the water hammer effect, water utilities equip the pipeline with shock absorbers and air traps. In addition, besides the physical countermeasures against the water hammer effect, the water distribution automation system has built-in safety logic to prevent actions that could cause a water hammer effect. However, both the physical countermeasures and the automation system are not designed to counter the effects of an intentional attack. In case safety controls are lost during a cyber-attack, valves can be controlled to cause a water hammer effect that can break the pipes in the distribution network.

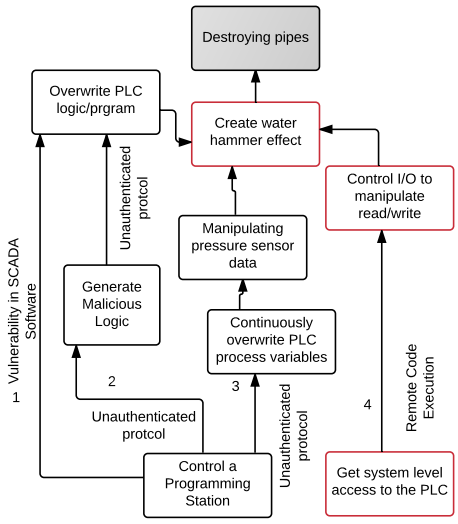

Fig. 2. Attack graph describing the water distribution pipe destruction scenario.

Figure 2 depicts the attack graph that describes this attack scenario. In the first three paths, the attacker needs to exploit the PLC programming stations (similarly to Stuxnet) to alter the process and cause the damage. Regarding the stealthiness of the attack, all of those attacks rely on attacking the SCADA station infection. To remain hidden attacker needs to suppress various protection mechanism [17], [29] in an SCADA station. Contrary, the attacker in the fourth path does not need to modify PLC logic, firmware or suppress any protection system. Also, even those few techniques exist to detect malicious behavior within the PLC [23], [5] are not effective against our attack. Also due to little performance overhead imposed by the Pin Control attack it is difficult to tell whether the PLC is targeted by the attack or not. What actually the attacker does with the pin control attack will be based on what described by Larson [16] where the PLC will suddenly close the valves of the water distribution pipes with network of infected PLCs while the operator will still sees the fake profile of the valve speed or even can be tricked to see the state of the valves are open while in reality the valves are opening and closing. The reading of other sensors regarding the pipe pressure and heat level of the pipes (due to water hammer generated heat) can be altered with the same network of the infected PLC using pin control attack.

5.2 Detection of Pin Control Attack

One might believe that it is relatively simple to devise countermeasures that could detect and possibly block pin control attacks. In this section we enumerate three of these possible countermeasures and discuss their effectiveness and practical applicability to the PLCs.

1. *Monitoring the mapping of pin configuration registers*: an attacker needs to use the virtual addresses of the pin configuration registers to write to them. To do so, the attacker needs to either map the physical registers by herself or use already mapped addresses. This is not the case in PLCs since the PLC is already using the target I/O. Therefore, an attacker can use already mapped register addresses to carry the attack.
2. *Monitoring the change of pin configuration registers*: one may detect our attack by monitoring the frequency at which pin configuration registers are changed. This may be challenging for two reasons. First, since changes in configuration registers do not generate hardware interrupts, therefore, an attacker will be able to bypass monitoring mechanisms. Second, since pins get re-configured legitimately by a PLC, it may be difficult to tell with reliable accuracy whether a sequence of changes is legitimate or not.
3. *Using a trusted execution environment*: the reliable solution to prevent all pin control attacks would be running a micro kernel in a trusted zones (e.g. an ARM TrustZone) within the kernel to verify write operations on configuration pins. However, as confirmed by the Linux Kernel Pin Control Subsystem group, using TrustZone for I/O operations would cause a significant performance overhead.

6 Related Work

Few research focused on the possibility of system level attack against PLCs. A relevant stream of work has explored memory corruption vulnerabilities against PLCs [27] which is the closest thing to attacking the PLCs at operating system level. Part of our work bears some similarities with System Management Mode (SMM) rootkits [25], [7], [24] for X86 architectures. These rootkits tap the system I/O, similarly to what we did in our Pin Control attack. However, the modification of system I/O in SMM causes interrupts which need to be suppressed by SMM rootkits, typically by attacking kernel interrupt handlers. In our case, this operation is not needed due to the lack of interrupts for pin configuration.

7 Conclusion

In this paper, we first looked into the pin control subsystem of embedded systems. We found that the lack of hardware interrupts for pin control subsystem brings an opportunity for the attacker. We showed that it is practical to target the Pin Control subsystem by implementing an attack against a PLC and manipulate a process in which it controlled. The result shows that attackers can stealthy manipulate the I/O of the PLCs without using traditional attack techniques such as function hooking or OS data structure modification. We now plan to investigate possible defensive techniques against Pin Control attack. We believe that defending PLCs against this new front will pose a notable hindrance to attackers, significantly reducing their success rate in the future.

References

1. Abbasi, A., Wetzels, J., Bokslag, W., Zambon, E., Etalle, S.: On emulation-based network intrusion detection systems. In: Research in Attacks, Intrusions and Defenses, pp. 384–404. Springer (2014)
2. Basnight, Z., Butts, J., Jr., J.L., Dube, T.: Firmware modification attacks on programmable logic controllers. International Journal of Critical Infrastructure Protection 6(2), 76 – 84 (2013)
3. Beresford, D.: Exploiting Siemens Simatic S7 PLCs. In: Black Hat USA (2011)
4. Beresford, D., Abbasi, A.: Project IRUS: multifaceted approach to attacking and defending ICS. In: SCADA Security Scientific Symposium(S4) (2013)
5. Cui, A., Stolfo, S.J.: Defending embedded systems with software symbiotes. In: 14th International Symposium on Recent Advances in Intrusion Detection (RAID). pp. 358–377. Springer (2011)
6. DigitalBond: 3S CoDeSys, Project Basecamp (2012), `http://www.digitalbond.com/tools/basecamp/3s-codesys/`
7. Embleton, S., Sparks, S., Zou, C.C.: Smm rootkit: a new breed of os independent malware. Security and Communication Networks 6(12), 1590–1605 (2013)
8. Falliere, N., Murchu, L.O., Chien, E.: W32. stuxnet dossier. White paper, Symantec Corp., Security Response 5 (2011)
9. ICS-CERT: Abb ac500 plc webserver codesys vulnerability (2013), `https://ics-cert.us-cert.gov/advisories/ICSA-12-320-01`

10. ICS-CERT: Schneider electric modicon quantum vulnerabilities (update b) (2014), https://ics-cert.us-cert.gov/alerts/ICS-ALERT-12-020-03B
11. ICS-CERT: Schneider electric modicon m340 buffer overflow vulnerability (2015), https://ics-cert.us-cert.gov/advisories/ICSA-15-351-01
12. ICS-CERT: Rockwell automation micrologix 1100 plc overflow vulnerability (2016), https://ics-cert.us-cert.gov/advisories/ICSA-16-026-02
13. Igure, V.M., Laughter, S.A., Williams, R.D.: Security issues in SCADA networks. Computers & Security 25(7), 498–506 (2006)
14. Koopman, P.: Embedded system security. Computer 37(7), 95–97 (2004)
15. Langner, R.: To kill a centrifuge: A technical analysis of what stuxnets creators tried to achieve. Online: http://www. langner. com/en/wp-content/uploads/2013/11/To-kill-a-centrifuge. pdf (2013)
16. Larsen, J.: Physical damage 101: Bread and butter attacks. Black Hat USA (2015)
17. Liang, Z., Yin, H., Song, D.: HookFinder: Identifying and understanding malware hooking behaviors. In: Proceeding of the 15th Annual Network and Distributed System Security Symposium (NDSS'08) (2008), http://bitblaze.cs.berkeley.edu/papers/hookfinder_ndss08.pdf
18. Maxino, T.C., Koopman, P.J.: The effectiveness of checksums for embedded control networks. IEEE Transactions on Dependable and Secure Computing 6(1), 59–72 (Jan 2009)
19. McLaughlin, S., McDaniel, P.: SABOT: Specification-based payload generation for programmable logic controllers. In: Proceedings of the 2012 ACM Conference on Computer and Communications Security. pp. 439–449. CCS '12, ACM, New York, NY, USA (2012)
20. McLaughlin, S.E.: On dynamic malware payloads aimed at programmable logic controllers. In: HotSec (2011)
21. Peck, D., Peterson, D.: Leveraging ethernet card vulnerabilities in field devices. In: SCADA Security Scientific Symposium. pp. 1–19 (2009)
22. PREEMPTIVE-Consortium: Reference taxonomy on industrial control systems networks for utilities. http://preemptive.eu/wp-content/uploads/2015/07/preemptive_deliverable-d2.3.pdf (2014)
23. Reeves, J., Ramaswamy, A., Locasto, M., Bratus, S., Smith, S.: Intrusion detection for resource-constrained embedded control systems in the power grid. International Journal of Critical Infrastructure Protection 5(2), 74–83 (2012)
24. Schiffman, J., Kaplan, D.: The smm rootkit revisited: Fun with usb. In: Availability, Reliability and Security (ARES), 9th International Conference on. pp. 279–286 (2014)
25. Sparks, S., Embleton, S., Zou, C.C.: A chipset level network backdoor: bypassing host-based firewall & ids. In: Proceedings of the 4th International Symposium on Information, Computer, and Communications Security. pp. 125–134. ACM (2009)
26. Spenneberg, R., Brüggemann, M., Schwartke, H.: Plc-blaster: A worm living solely in the plc. Black Hat Asia (2016)
27. Wightman, R.: Project basecamp at s4. SCADA Security Scientific Symposium (2012), https://www.digitalbond.com/tools/basecamp/schneider-modicon-quantum/
28. Wrightman, K.R.: Vulnerability inheritance in plcs. DEFCON 23 IoT Village (2015)
29. Yin, H., Song, D.: Hooking behavior analysis. In: Automatic Malware Analysis, pp. 43–58. Springer (2013)