

Detecting Safety and Security Faults in PLC Systems with Data Provenance

Abdullah Al Farooq Jessica Marquard, Kripa George, Thomas Moyer

UNC Charlotte

{afarooq,jmarqua1,kgeorg16,tom.moyer}@uncc.edu

Abstract—Programmable Logic Controllers are an integral component for managing many different industrial processes (e.g., smart building management, power generation, water and wastewater management, and traffic control systems), and manufacturing and control industries (e.g., oil and natural gas, chemical, pharmaceutical, pulp and paper, food and beverage, automotive, and aerospace). Despite being used widely in many critical infrastructures, PLCs use protocols which make these control systems vulnerable to many common attacks, including man-in-the-middle attacks, denial of service attacks, and memory corruption attacks (e.g., array, stack, and heap overflows, integer overflows, and pointer corruption). In this paper, we propose *PLC-PROV*, a system for tracking the inputs and outputs of the control system to detect violations in the safety and security policies of the system. We consider a smart building as an example of a PLC-based system and show how *PLC-PROV* can be applied to ensure that the inputs and outputs are consistent with the intended safety and security policies.

I. INTRODUCTION

Industrial control systems rely on automation to ensure safe and efficient operation of many industrial processes including power generation, water and wastewater management, traffic control systems, petroleum and manufacturing industries (e.g., oil and natural gas pipelines, chemical and pharmaceutical production, pulp and paper manufacturing, food and beverage production, and automotive and aerospace assembly). These industries rely heavily on automation to reduce cost and ensure safety of potentially dangerous operations. In the past, these control networks have been isolated from other networks, such as the internet, but we are seeing more and more of these types of networks being connected to the internet [1], [2], [3]. Often, the goal is to further reduce impact on humans as this new connection enables remote management. However, often the implications of connecting these safety critical control systems is not carefully considered. In many instances, attacks on these systems can lead to disruption of critical services and even loss of human life [4], [5], [6], [7], [8], [9].

One of the biggest problems with this scenario is that the security of these control networks is limited at best. The communication protocols used in these control networks lack authentication and integrity checking for messages [10]. These weaknesses make it possible to initiate many commonly-known attacks such as man-in-the-middle attacks, denial of service attacks, memory corruption attacks, replay attacks, and spoofing attacks. While enterprise networks can rely on a wide-range of security mechanisms including IPsec, transport layer security (TLS), and virtual private networks (VPNs) to

secure their communications, such mechanisms are difficult to deploy on these control networks, leaving them vulnerable to network-based attackers. Furthermore, many of the commonly applied mitigations fail to cover PLC-based systems [11].

What is needed are mechanisms that can monitor the inputs and outputs of the ICS and ensure that critical safety properties are not violated. This requires an understanding of the desired safety properties, a way to track inputs and outputs, and a mechanism to model the evolution of the system from inputs to outputs. With these mechanisms in place, it becomes possible to ensure that the PLCs do not send commands to actuators that violate the safety and security policies of the system.

In this paper, we propose *PLC-PROV*, a mechanism to track the inputs and outputs of the system and compare them against the specified safety and security properties. *PLC-PROV* relies on tracking *data provenance* for the PLCs and using that provenance to determine if a violation has occurred. Provenance, in short, is the “history of data transformed by a system”, and has been proposed as a building block for systems that require the ability to reason about the *context* in which an action is taken. Since PLCs are entirely event-driven, context is vitally important, and as such provenance is a natural fit for this sort of analysis.

The rest of this paper is organized as follows. Section II provides relevant background information on PLCs and data provenance. Section III presents the design of *PLC-PROV* and walks through a simple example to highlight how the system works. Finally, Sections IV and V detail related work and conclude.

II. BACKGROUND

A. Programmable Logic Controllers

Figure 1 shows a notional Industrial Control System, comprised of several components that together provide the ability to automate industrial processes. At the heart of this system is the Programmable Logic Controller, or PLC. The PLC takes input from the sensors and determines the appropriate commands for the actuators to adjust the environment. The logic for the PLC comes from an engineering workstation that contains the IDE used by the programmer to develop the application logic for the ICS. Additionally, an ICS has one or more Human Machine Interfaces, or HMIs, that enable operators to view current and historical data from the ICS. The historical data is stored in the data historian and is often used for post-facto analysis of events.

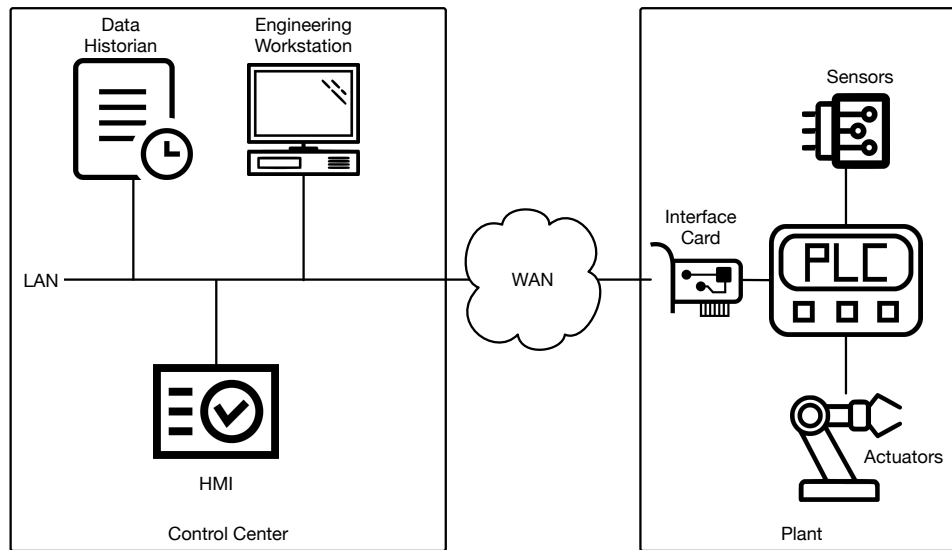


Fig. 1. A simple PLC-based control system with the basic components of any industrial control system.

The PLC applications are written in one of several programming languages including Instruction Lists (IL), Structured Text (ST), Functional Block Diagram (FBD), and Ladder Logic (LL). Regardless of the programming language used the instructions control the features of the PLC including I/O control, communication, logical decisions, timing, counting, three mode proportional-integral-derivative (PID) control, arithmetic, and data and file processing. The inputs to the PLC come from a wide array of sensors such as temperature sensors, motion detectors, smoke detectors, water leak detectors, and surveillance cameras. The outputs of the PLC go to actuators that adjust the current environment. These actuators include thermostats, humidifiers, speakers, security cameras, video doorbells, door locks, and window blinds.

The topology of an ICS network can be broadly divided into two subnets. The first is the control network where the sensors and actuators interface with the PLC. In more complex ICS environments, there may also be a Master Terminal Unit, or MTU, that provides the control programs for the PLCs. This control network uses non-IP-based protocols such as Modbus [12]. The second network is the corporate network, where the historian, HMI, and engineering workstation are located. This is a traditional enterprise network, using standard IP protocols to communicate. In order to link the corporate and control networks, interface cards are used to provide a bridge between IP-based protocols and the Modbus protocol. The PLC uses a Modbus/TCP protocol to send data to the historian. The historian also provides an HTTP interface for devices like the historian to access the stored historical data.

For a distributed system like SCADA (Supervisory control and data acquisition) or DCS (Distributed Control System), a group of PLCs are assigned to different subsystems. This is mainly done to handle long distance communication among geographically disperse assets (e.g., power grids, natural gas pipelines, water distribution, wastewater collection systems,

railway transportation systems). The far-reaching nature of these systems necessitates numerous control systems responsible for controlling local operations, but working in concert to ensure global functioning of the system. While these systems are more complex, they rely on many of the same basic components of a smaller-scale ICS.

Even in localized ICS environments (e.g., smart buildings), it is common to rely on several PLCs that work in concert to provide a range of functions. Consider for example that there might be a PLC that controls the heating, ventilation, and cooling, or HVAC, system, one PLC that controls the elevators, one PLC that controls the door and window locks, and finally one PLC that monitors for hazard conditions (e.g., smoke, carbon monoxide, water and chemical leaks, etc.). While these systems can be implemented independently, there are often dependencies that need to be accounted for. Consider a case where the hazard monitoring PLC detects smoke in the building, it must send notifications to the elevator and door/window lock PLCs that this condition is present so that appropriate actions can be taken. Those actions might be to open the locks on the doors, and move the elevator to the ground floor and then lock out use of the elevator. These dependencies ensure safety and efficiency in these automated systems.

B. Data Provenance

Data provenance is defined as the “history of data transformed by a system” [13]. The provenance of a piece of data describes what the inputs and outputs of each process are, what processes were executed, and who had control of those processes during execution. Provenance is often represented as a directed acyclic graph (DAG) with nodes representing the data, processes, and controlling entities. The edges represent causal relationships between these nodes.

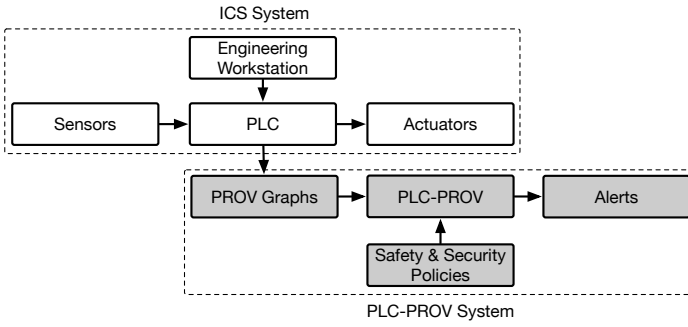


Fig. 2. PLC-PROV Architecture

Provenance was first introduced in databases and computational sciences for tracing and debugging. However, more recently it has been proposed as a primitive for building secure and resilient systems [14] that can “fight through” attacks. In order to provide such capabilities, novel collection, storage, and analysis mechanisms have been proposed to enable near-real-time analysis of provenance to support security and resilience decisions [15]. These mechanisms are being used to provide forensic analysis and intrusion detection capabilities [16].

III. DESIGN

Due to the distributed nature of PLC systems an attacker can trigger an event that leads to conflicting actions for the same object or feature of the plant/environment. Let us consider a smart building as an example where PLC is used [17], [18], [19], [20], [21]. An attacker can compromise a carbon monoxide detector and trigger a false alarm. This sends a command to the smart windows to open allowing fresh air into the building. Occupants will evacuate the building and a thief can use the open windows to enter the building. Another example can be creating multiple events that trigger a thermostat to increase and decrease the temperature of a room at the same time. Sending two different commands in the thermostat at the same time continuously can damage it, by artificially shortening the device’s lifespan. In this way, the attacker not only damages an asset but also may drive the occupants of the room to leave due to fluctuations in the comfort level of the room. In addition to the attacks described above, an attacker can create a series of attacks (cascading attack) [22]. Moreover, misconfiguration in the smart building operation is possible as there are numerous rules or policies for taking actions by the controllers after events have occurred.

A formal methods approach for detecting conflicts in IoT system is presented in [23]. A PLC-based system (e.g., smart home, power supply, water supply, wastewater management, and traffic control system) works on the same sensor-actuator-controller functionalities. Therefore, we adopted the safety policies defined in [23] as the basic policies to analyze using the collected provenance graphs. PLC-PROV will check whether there is any violation of the defined safety and security properties. If found, it is reported as an anomaly for the

system, which also provides the administrator with the detailed traces that are needed to understand the impact of the anomaly. The basic architecture is given in Figure 2.

The PLC contains the core rules/logic for controlling the plant/environment. The controller issues command to the actuators based on these sensor measurements. A PLC has to be operated with software that provides interaction with the sensors and actuators. These addresses are mapped as variables in the source program. In this project, we use CODESYS¹ which is a development system for PLC applications. To start, our framework traces the variables designated for sensors and actuators. These values of these variables are collected with timestamps into traces of system execution.

These traces are the input for a provenance management tool *Curator* [24]. Curator is a lightweight library which minimizes integration complexity for the application developers. It is also capable of integrating provenance from multiple levels of abstraction, a feature that enables reasoning about provenance both at the sensor reading level (micro) and at the environment/plant level (macro). As Curator is targeted for microservice based system, it fits well for our case where we collect traces from disparate components of the system, similar to microservices. From there, the provenance graph is generated, showing the evolution of the system from sensor readings through the PLC and controller, to the actuators. Figure 3 is a notional example of the type of graph that is generated.

The provenance graphs collected by PLC-PROV enable an administrator to answer the following questions:

- Has an actuator been actuated more than once at the same time?
- If an actuator receives multiple commands at the same time, are these same or different?
- What are the reasons behind conflicting action?
- Which are the sensors influencing conflicting commands?
- Has any sensor measurement gone beyond normal range? If yes, how many times did that happen and how long did it last?
- Are there more than two actions affecting the same environment feature?

A. Example

In order to illustrate how PLC-PROV will operate, we provide a short example of an attack scenario. Consider the floorplan shown in Figure 4 which includes a secure, access-controlled area in the lower-left corner. In this scenario, we consider the policy that security doors should only be unlocked when presented with a valid access card and in an emergency when human life is at stake. In our scenario, we have PLCs controlling various aspects of our smart building. The first we call our *safety PLC* which contains a smoke detector as an input. The output of this PLC is connected to an alarm system and to our *security PLC*. The security PLC has a card reader as input and a lock control on its output. Finally, we have the

¹<https://www.codesys.com/>

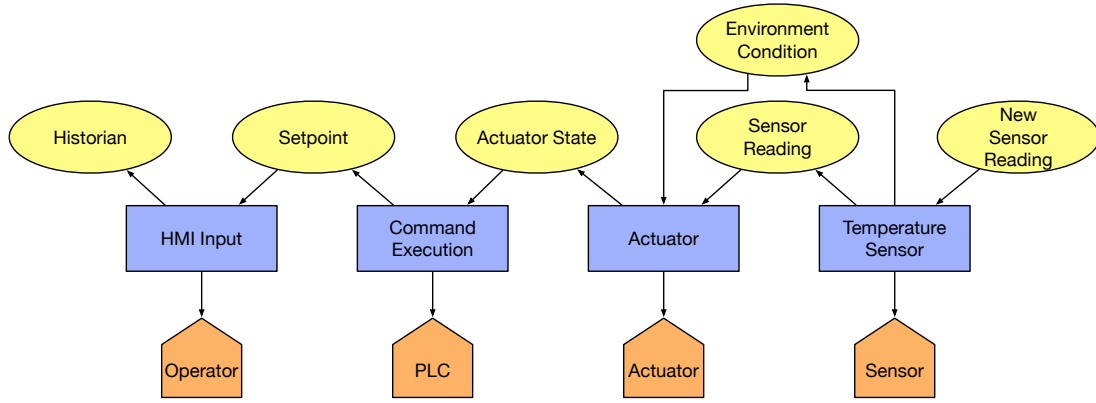


Fig. 3. Provenance Model for PLC-based System

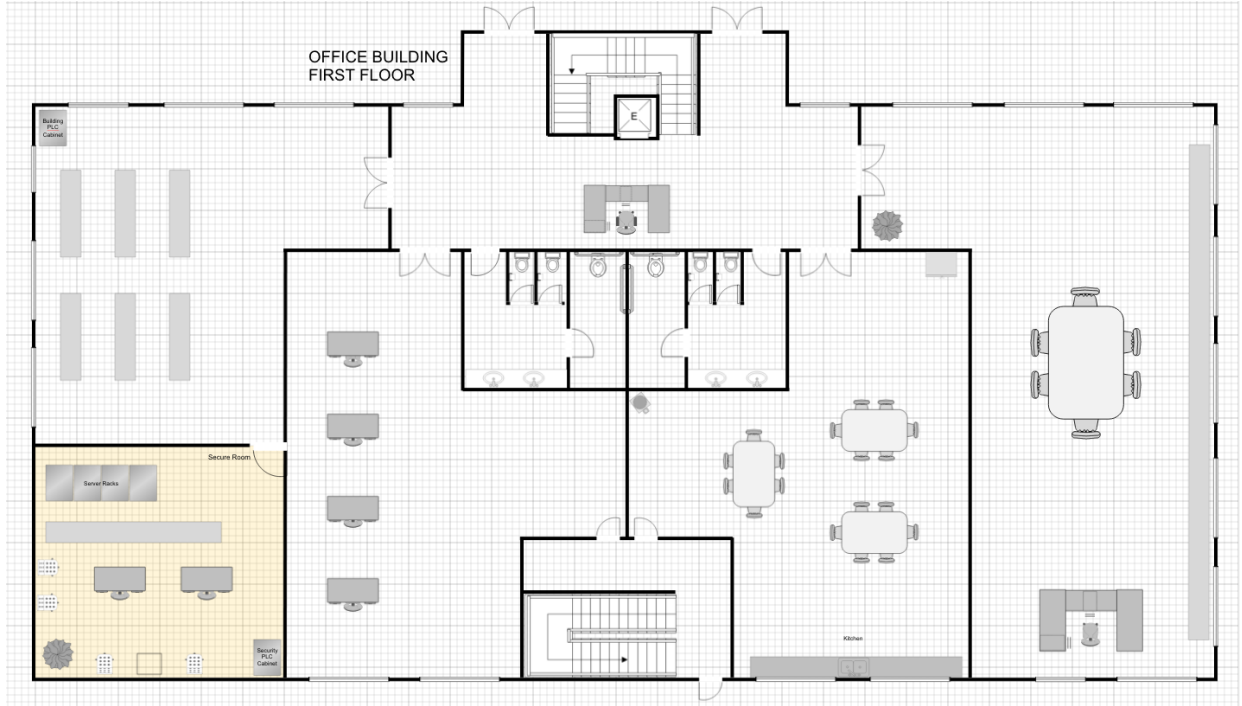


Fig. 4. Notional office floorplan with a secure area that is access controlled and environmentally monitored.

environmental PLC which monitors environmental conditions (such as temperature and humidity). As a safety mechanism, when hazards are detected that require evacuation, security doors should be unlocked to ensure no one is locked in the building, also allowing first responders to clear all areas.

Given this setup, we now consider an attacker that wishes to gain physical access to the secured areas within the building. In order to gain access, the attacker first poses as a maintenance technician to gain access to the common areas of the building. From here, he connects to the PLC network and forges a sensor reading from the smoke alarm indicating a hazard condition in the secure area. This hazard condition triggers the alarm to evacuate the secure area, which also causes the doors to unlock. In this scenario, the attacker can use the unlock and evacuate behavior to gain access to the secure area.

PLC-PROV aids the administrators in detecting this attack

in the following way. As it monitors the flow of inputs and outputs of the system, it would take into account context. There are several indicators that something is likely wrong. One, smoke detector alarms will typically correspond to a rise in temperature as well, which won't be present in the attack scenario. If the attacker is smart enough, he can simulate this behavior. However, another indication of malicious behavior is that the smoke detector signal and temperature readings would have to be present in the secure space and that means that the smoke detector reading at the PLC would be on a specific input line. However, since the attacker doesn't have access to the space, they have to fake the measurement from another point within the control network. Here we assume that the PLCs controlling the secured space are located within the secured space itself, as is typical practice in building secure areas, while the environmental PLCs are located in the common areas

of the building to allow technicians to gain access. This is depicted in Figure 4 with the upper-left block representing the environmental PLC cabinet and the lower-right block in the secure area representing the security PLC cabinet. It is also assumed that in order for the alarm from the environmental sensors to reach the secure area that there is a connection between the two cabinets. By monitoring the flow of inputs and outputs, PLC-PROV would be able to detect this deviation in known good behavior and flag it as anomalous, further alerting building personnel to the potential issue. In this way, we can ensure that safety and security policies are not violated.

IV. RELATED WORK

The closest work to PLC-PROV is PROV-CPS [25] where provenance was collected from resource-constrained embedded devices of the cyber-physical system. However, this research collects provenance from sensors only to identify anomalous measurements. On the contrary, apart from collecting provenance from the sensor measurement, our work covers the actions of the actuators and the dependencies among the PLCs in finding malicious activities. Our approach is complete in expressing the causality and dependencies among the data objects through the provenance graph. Another notable work in this area is ProvThings [26] where a provenance collection framework is proposed for IoT apps and devices. ProvThings presents an automated instrumentation mechanism for IoT apps and device APIs. The collected provenance is then used to generate explanations for “why” a particular action occurred. Our work captures provenance data for all sensors and actuators in order to detect safety and security policy violations.

There have been several other attempts to deploy security policies with static verification, dynamic verification, and the hybrid of these two approaches. Static verification (model checking) is proposed in TSV [27] where a middleware ensures the safety of a PLC-based system sitting between PLC and the devices. TSV verifies the safety behavior of the code executed on PLC before commands reach the actuators. The safety properties are written in temporal logic which is verified using model checking. While this work verifies the system’s behavior, there are some other works that started the verification from PLCs’ source program [28], [29]. Later, others proposed mechanism to automatically generate formal models from PLC programs [30], [31], [32], [33], [34].

While static analysis performs the verification before the PLC program is released for operation (i.e. compile-time), dynamic analysis ensures that policies are not violated at run-time. C^2 [35] introduced an enforcement mechanism for safety policies in PLC-based system. When a PLC issues a command to an actuator, the current states of the system are checked and then decisions are made whether or not the command should be issued through their enforcement mechanism, C^2 . In this work, concerns about the size of the trusted computing base (TCB) and state explosion in the model checking were expressed. [36] addresses reduces the size of the TCB considerably, and merges the static and dynamic analysis of TSV

and C^2 . The works by McLaughlin, et. al. focus specifically on safety properties. This was subsequently extended in [37] with an effort to find malicious PLC programs. Another approach to dynamic analysis is proposed in [38] using Interval Temporal Logic (ITL) and the Tempura framework, which aims to provide early alerts in PLC-based systems.

Later, this work was extended in [39] where an ITL/Tempura definition of a Siemens S7-1200 PLC ladder logic was presented. Their developed monitoring methodology captures a snapshot of the current state (with values for markers, input, output, counters, and timers) of the PLC. Tempura was implemented to execute on an Arduino Uno connected to the PLC, ensuring that the PLC does not need a powerful computing node to perform the computations.

While static analysis has proven promising, the number of possible inputs and outputs for a PLC system can lead to a state explosion. Furthermore, dynamic analysis suffers from a coverage problem, where only executed code paths are verified. Symbolic execution can minimize the state space, but cannot guarantee complete verification of outputs (actuation command) from input sets (sensor measurement). For these reasons, what is needed is a mechanism that can provide high-level safety and security policy descriptions that can be enforced at run-time where the appropriate context can be considered.

V. CONCLUSION

This project focuses on the integration of data provenance in PLC controlled system in order to detect safety policy violation there. We have modeled data provenance that considers operator (through HMI) input, command execution, actuators’ state, and sensors’ measurement. Therefore, we claim that our model is complete in expressing the causality and dependencies among the data objects in a PLC-controlled system. We evaluated our model with smart building environment. It turns out that data provenance has great potential applicability in PLC controlled system where the change of sensor measurement and actuator actions take place very frequently. Despite being used in critical infrastructures, PLCs have little or almost no security. The integration of plc-prov is capable of enforcing adequate safety and security policies.

REFERENCES

- [1] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. Mohammed, and S. A. Zonouz, “Hey, my malware knows physics attacking plcs with physical model aware rootkit,” in *Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA, 2017*, pp. 26–28.
- [2] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, “Internet-facing plcs as a network backdoor,” in *Communications and Network Security (CNS), 2015 IEEE Conference on*. IEEE, 2015, pp. 524–532.
- [3] R. Leszczyna, E. Egozcue, L. Tarrafeta, V. F. Villar, R. Estremera, and J. Alonso, “Protecting industrial control systems-recommendations for europe and member states,” European Union Agency for Network and Information Security, <https://www.enisa.europa.eu/publications/protecting-industrial-control-systems-recommendations-for-europe-and-member-states>, Tech. Rep., 2011.
- [4] K. Stouffer, J. Falco, and K. Scarfone, “Guide to industrial control systems (ics) security,” *NIST special publication*, vol. 800, no. 82, pp. 16–16, 2011.

- [5] C. Interactive, “teen hacker faces federal charges,” *viewed on*, vol. 15, 1988. [Online]. Available: <http://www.cnn.com/TECH/computing/9803/18/juvenile.hacker/index.html>
- [6] T. Smith, “Hacker jailed for revenge sewage attacks,” *The Register*, vol. 31, 2001. [Online]. Available: https://www.theregister.co.uk/2001/10/31/hacker_jailed_for_revenge_sewage/
- [7] D. Hancock, “Virus disrupts train signals,” Aug. 2003.
- [8] M. Abrams and J. Weiss, “Bellingham control system cyber security case study,” 2007.
- [9] N. Falliere, L. O. Murchu, and E. Chien, “W32. stuxnet dossier,” *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.
- [10] J. L. Rrushi, *SCADA Protocol Vulnerabilities*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 150–176. [Online]. Available: https://doi.org/10.1007/978-3-642-28920-0_8
- [11] J. Pincus and B. Baker, “Mitigations for low-level coding vulnerabilities: Incomparability and limitations.”
- [12] Modbus Organization, “Modbus Application Protocol Specification V1.1b3,” http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf.
- [13] World Wide Web Consortium and others, “PROV-Overview: an overview of the PROV family of documents,” <https://www.w3.org/TR/prov-overview/>, 2013.
- [14] T. Moyer, K. Chadha, R. Cunningham, N. Schear, W. Smith, A. Bates, K. Butler, F. Capobianco, T. Jaeger, and P. Cable, “Leveraging data provenance to enhance cyber resilience,” in *Cybersecurity Development (SecDev)*, IEEE, 2016, pp. 107–114.
- [15] X. Han, T. Pasquier, and M. Seltzer, “Provenance-based intrusion detection: Opportunities and challenges,” in *10th USENIX Workshop on the Theory and Practice of Provenance (TaPP 2018)*. London: USENIX Association, 2018. [Online]. Available: <https://www.usenix.org/conference/tapp2018/presentation/han>
- [16] Y. Xie, D. Feng, Z. Tan, and J. Zhou, “Unifying intrusion detection and forensic analysis via provenance awareness,” *Future Generation Computer Systems*, vol. 61, pp. 26–36, 2016.
- [17] W. Kastner, G. Neugschwandtner, S. Soucek, and H. M. Newman, “Communication systems for building automation and control,” *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1178–1203, 2005.
- [18] W. Granzer, F. Praus, and W. Kastner, “Security in building automation systems,” *IEEE Transactions on Industrial Electronics*, vol. 57, no. 11, pp. 3622–3630, 2010.
- [19] C. Barz, S. Deaconu, T. Latinovic, A. Berdie, A. Pop-Vadean, and M. Horgos, “Plcs used in smart home control,” in *IOP Conference Series: Materials Science and Engineering*, vol. 106, no. 1. IOP Publishing, 2016, p. 012036.
- [20] T. Sysala, M. Pospíchal, and P. Neumann, “Monitoring and control system for a smart family house controlled via programmable controller,” in *Carpathian Control Conference (ICCC), 2016 17th International*. IEEE, 2016, pp. 706–710.
- [21] N. Skeledzija, J. Cestic, E. Koco, V. Bachler, H. N. Vucemilo, and H. Dzapo, “Smart home automation system for energy efficient housing,” in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*. IEEE, 2014, pp. 166–171.
- [22] S. COBB, “10 things to know about the october 21 iot ddos attacks,” 2016. [Online]. Available: <http://www.welivesecurity.com/2016/10/24/10-things-know-october-21-iot-ddos-attacks/>
- [23] A. A. Farooq, E. Al-Shaer, T. Moyer, and K. Kant, “Iotc2: A formal method approach for detecting conflicts in large scale iot systems,” in *Integrated Network and Service Management (IM), 2017 IFIP/IEEE Symposium on*. IEEE, 2019.
- [24] W. Smith, T. Moyer, and C. Munson, “Curator: Provenance management for modern distributed systems,” *CoRR*, vol. abs/1806.02227, 2018. [Online]. Available: <http://arxiv.org/abs/1806.02227>
- [25] E. Nwafor, “Trace-based data provenance for cyber-physical systems,” Ph.D. dissertation, Howard University, 2018.
- [26] Q. Wang, W. U. Hassan, A. Bates, and C. Gunter, “Fear and logging in the internet of things,” in *ISOC NDSS*, 2018.
- [27] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, “A trusted safety verifier for process controller code,” in *NDSS*, vol. 14, 2014.
- [28] J. Sadolewski, “Automated conversion of st control programs to why for verification purposes,” in *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*. IEEE, 2011, pp. 849–854.
- [29] —, “Conversion of st control programs to ansi c for verification purposes,” *e-Informatica*, vol. 5, no. 1, pp. 65–76, 2011.
- [30] S. Biallas, J. Brauer, and S. Kowalewski, “Arcade. plc: A verification platform for programmable logic controllers,” in *Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2012, pp. 338–341.
- [31] D. Darvas, B. Fernandez Adiego, and E. Blanco, “Transforming PLC Programs into Formal Models for Verification Purposes,” CERN, Tech. Rep., Nov 2013. [Online]. Available: <http://cds.cern.ch/record/1629275>
- [32] B. F. Adiego, D. Darvas, J.-C. Tournier, E. B. Vinuela, and V. M. G. Suárez, “Bringing automated model checking to plc program development—a cern case study—,” *IFAC Proceedings Volumes*, vol. 47, no. 2, pp. 394–399, 2014.
- [33] F. Markovic, “Automated test generation for structured text language using uppaal model checker,” 2015.
- [34] E. P. Enoiu, A. Čaušević, T. J. Ostrand, E. J. Weyuker, D. Sundmark, and P. Pettersson, “Automated test generation using model checking: an industrial evaluation,” *International Journal on Software Tools for Technology Transfer*, vol. 18, no. 3, pp. 335–353, 2016.
- [35] S. McLaughlin, “Cps: Stateful policy enforcement for control system device usage,” in *Proceedings of the 29th Annual Computer Security Applications Conference*. ACM, 2013, pp. 109–118.
- [36] —, “Blocking unsafe behaviors in control systems through static and dynamic policy enforcement,” in *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE, 2015, pp. 1–6.
- [37] S. Zonouz, J. Rrushi, and S. McLaughlin, “Detecting industrial control malware using automated plc code analytics,” *IEEE Security & Privacy*, vol. 12, no. 6, pp. 40–47, 2014.
- [38] A. Nicholson, H. Janicke, and A. Cau, “Position paper: Safety and security monitoring in ics/scada systems,” in *ICS-CSR*, 2014.
- [39] H. Janicke, A. Nicholson, S. Webber, and A. Cau, “Runtime-monitoring for industrial control systems,” *Electronics*, vol. 4, no. 4, pp. 995–1017, 2015.