Power Grid Safety Control via Fine-Grained Multi-Persona Programmable Logic Controllers

Gabriel Salles-Loustau, Luis Garcia, Pengfei Sun, Maryam Dehnavi, Saman Zonouz
Electrical and Computer Engineering Department
Rutgers University
{lag266, gs643, ps807, maryam.dehnavi, saman.zonouz}@rutgers.edu

***Abstract*—Trustworthy and safe operation of the power grid critical infrastructures relies on secure execution of low-level substation controller devices such as programmable logic controllers (PLCs). Currently, there are very few security protection solutions deployed on these devices to ensure provenance control: to execute controller code on the device that is developed by trusted parties and complies with safety/security policies that are defined by the code developer as well as the power grid operators. Resource-limited PLC controllers have been becoming increasingly popular among not only legitimate system operators, but also malicious adversaries such as the most recent Stuxnet and BlackEnergy malware that caused various damages such as unauthorized infrastructural safety and integrity violations. We present PLCtrust, a domain-specific solution that deploys virtual micro security-perimeters, so-called *capsules*, and the corresponding device-level runtime power system-safety policy enforcement dynamically. PLCtrust makes use of data taint analysis to monitor and control data flow among the capsules based on data owner-defined policies. PLCtrust provides the operators with a transparent and lightweight solution to address various safety-critical data protection requirements. PLCtrust also provides the legitimate third-party controller code developers with a taint-aware programming interface to develop applications in compliance with the dynamic power system safety/security policies. Our experimental results on real-world settings show that PLCtrust is transparent to the end-users while ensuring the power grid safety maintenance with minimal performance overhead.**

I. Introduction

Traditionally, critical infrastructure security has been addressed through perimeter protection solutions such as rule-based gateway firewalls and network-based intrusion detection systems such as Snort and Bro. The regulatory guidelines such as critical infrastructure protection (CIP) [13] recommendations by North American Electric Reliability Corporation (NERC) also mandates enforcement of electronic security perimeter [13] protection. However, recent increasing number and complexity of attacks against controller devices within control system networks and power system substations indicate the insufficiency of network-level security monitoring solutions. Specifically, Stuxnet bypassed the network-based monitors through USB-based substation computer infections within Iranian nuclear enrichment plant. BlackEnergy leveraged spear phishing social engineering and use of trojan Ms. Office files to evade the perimeter-based protection techniques in Ukraine power grid distribution substation.

The essential core technical problem with perimeter-based protection solutions is their huge trusted computing base (TCB). TCB is defined as the minimum number of components that need to be working correctly in order for the whole infrastructure to accomplish its objective. Traditional reliance on network-based intrusion detection solutions attempts to leave all the adversaries out of the (usually huge) supervisory control and data acquisition networks. This turns out to be infeasible in practice due to the existing many attack vectors, such as social engineering and infected USB sticks, that necessarily cannot be monitored completely using network-based intrusion detectors.

Due to the inadequacy of the perimeter-based protection, there exists more in-depth security monitoring solutions that do not assume the whole SCADA as the power grid operational TCB. The most promising option is to protect end-point devices right before the physical power system is affected by the SCADA-issued control commands. The end-points are often the controller devices such as the programmable logic controllers (PLCs). Device-level protection attempts to ensure the security and safe operation of controller devices regardless of whether the SCADA is clean or infected. There have been past proposed solutions to protect the controller devices. TSV [8] intercepts all the controller bound code from the SCADA network, and verifies its correctness before it is allowed to reach the PLC device. TSV leverages advanced binary disassembly and programming languages techniques such as symbolic execution along with formal verification to validate the security of the PLC software in an offline fashion. Such offline solutions are useful when the controller software size does not exceed a fairly small threshold [8]; those solutions often have scalability issues when dealing with large code bases. Consequently, online device-level security monitoring and policy enforcement solutions to ensure the infrastructural safety are needed to address scalability concerns sufficiently in practice.

We Propose PLCtrust, a device-level online security monitoring and policy enforcement solution that guarantees safety of the underlying power system and scales up to large-scale power grid infrastructures. PLCtrust is a PLC-based isolation engine that isolates controller code with different protection requirements while still allowing operators to use any remote programming base using cryptographic developer signatures. Through an information flow architecture, PLCtrust provides fine-grained, policy-based, domain-specific controller code protection as a first class primitive in the PLC firmware itself and eliminates the need for multiple environments to access code with different protection requirements. With a reasonably low performance overhead, PLCtrust allows individual controller codes to be contained in micro-security perimeters called *capsules*. Capsules can be securely and dynamically installed or removed from the controller device, and are subject to policies specified by the capsule owner, i.e., the controller code developer. Secure capsule installation is performed by a PLCtrust server that handles device authentication and capsule certification. Policy rules included in each capsule define controller code access and mixing policies with respect to other capsules and can be based on contextual information such as location, time, or controller code source (developer identity signature). PLCtrust enables the PLC firmware to track the flow of data on a per-capsule basis as it is used by applications on the controller device and enforces the security policies associated with it, thus enabling various device-level security use-cases. More generally, PLCtrust enables controller security protection policies that are defined by *context*, e.g., some controller code may be sensitive until a certain conditional circumstance after which it can be "released" into a non-sensitive environment.

PLCtrust facilitates online device-level controller security monitoring through enhancements in both the control logic

978-1-5386-4055-5/17/$31.00 ©2017 IEEE

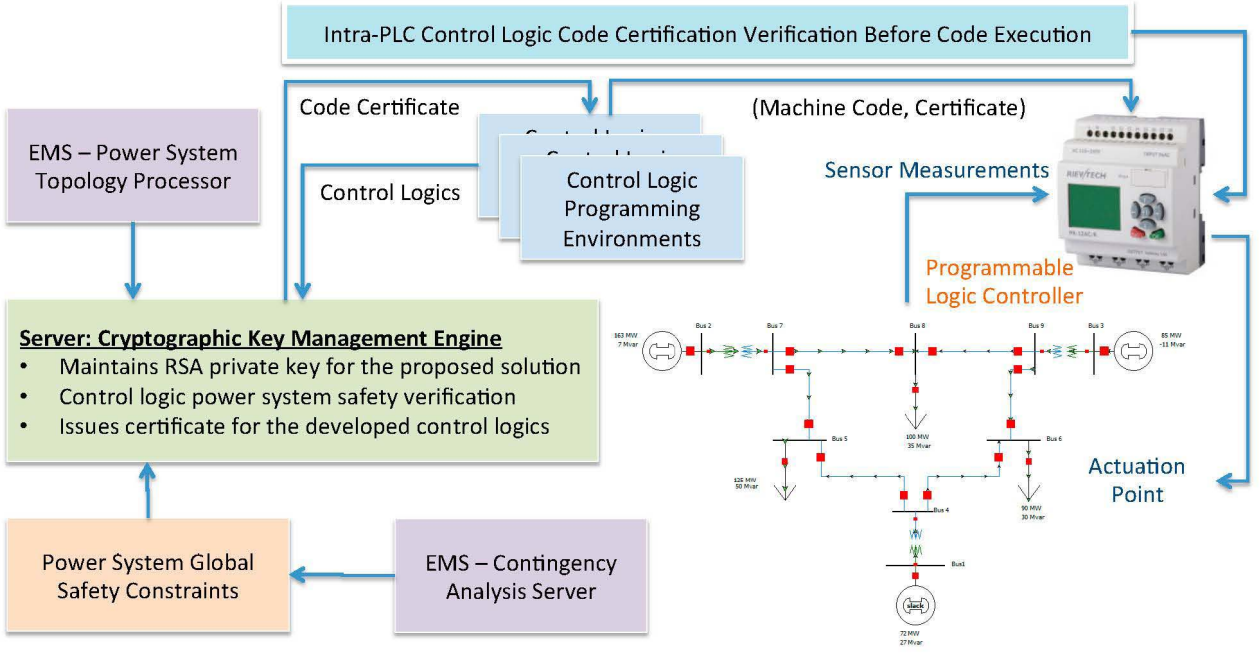

Fig. 1: PLCtrust's High-Level Architecture, Its Components and their Logical Interconnections

binary and its underlying firmware stack to ensure its efficient and secure services. PLCtrust implements a secure information flow tracking and control enforcement solution that takes into account all possible data sources, such as network sockets and SD storage card on the PLC and data propagation through control logic instruction executions. To that end, PLCtrust employs dynamic information flow analysis and control. PLCtrust labels data (including controller code) coming from various sources (different developers based on their cryptographic certificates) and provides variable-level, controller code function-level, and message-level tracking of data information flow. This enables PLCtrust to allow only *i)* the execution controller code that are from authentic sources, and *ii)* the data accesses by the controller codes that comply with the policies defined by their corresponding developers. To keep track of persistent tainted data information, PLCtrust deploys a lightweight database that maintains tainted file lists across device reboots and makes use of a LFU cache to minimize the database queries during the runtime policy enforcement. Furthermore, PLCtrust facilitates dynamic definitions of capsules and policies as well as their distribution through a PLCtrust server that implements a secure protocol where the data owner signs an encrypts the capsule before it can be downloaded and installed by the user. Once the capsule is installed, PLCtrust loads the policy in memory to control the flow of the data corresponding to the installed capsule.

In summary, the contributions of this paper are as follows:

- We introduce a runtime capsule (virtual security perimeter) definition, distribution and installation framework for power grid embedded controllers.
- We present an efficient and dynamically reconfigurable inter- and intra-process data flow control enforcement engine.
- We introduce a easy-to-use lightweight and transparent support for multi-persona controller device usage experience using cryptographic certificates.
- We propose a new firmware-level interface for safety-critical controller program developers to develop multi-context-aware controllers compliant with a given data flow policy.
- We validate the PLCtrust framework with a working prototype and on a real-world test-beds.

This paper is organized as follows: Section II reviews the threat model and the assumptions about the adversary capabilities. **??** and Section IV presents PLCtrust s formal power system safety description and power system analyses in details. Section VI provides the details of how PLCtrust performs on a realistic power grid setup. We review the most related past work and how they fall short to achieve PLCtrust s objectives in Section VII. Finally, Section VIII concludes the paper.

II. Threat Model

PLCtrust's adversary model is consistent with practical settings in real-world critical infrastructures nowadays. Its assumptions about the attacker includes the following: the attacker can have access to all the devices within the power grid control center, substations and supervisory control and data acquisition platforms except the following two privilege domains: *i)* either the root-level access on the programmable logic controller or the firmware-level access on the controller if its software stack does not include an operating system; and *ii)* the programming environment, where the control logic programs for the PLC controller are developed by the operators. The PLC's root level trust has to be maintained, because PLCtrust's information flow tracking and policy enforcement engines run as the PLC's root-level user and the root-level attack can possibly corrupt PLCtrust's core functionalities. Additionally, the programming environment is where PLCtrust creates the cryptographic certificates for the developed control logic programs. Any malicious control gain over the programming environment would allow the adversaries to create fake certificates for malicious control logic programs that will pass PLCtrust's on-PLC code integrity verification checks. This can result in malicious code execution on the PLC controller, while it is monitoring and controlling an underlying safety-critical process, such as a generator set-point control. Malicious code execution on the PLC can cause catastrophic failures and large-scale physical damages.

III. PLCtrust Design

Figure 1 shows a high-level overview of the PLCtrust's architecture and how its components are logically interconnected. To describe how PLCtrust achieves its objectives, let's

consider the following scenario. The power system operator Bob would like to use the PLCtrust-enabled controller for different purposes such as checking a local circuit breaker status continuously and controlling it (through open/close commands) whenever needed for the power system stability. PLCtrust creates an infrastructure to facilitate convenient and secure embedded PLC controller device usage experience for Bob who does not need to worry about whether a malicious controller logic program may access and/or manipulate confidentiality/safety-sensitive power substation actuation points in an authorized manner. PLCtrust deploys a dynamic information flow tracking and a system-wide mandatory access control policy enforcement engine to keep track of Bob's sensitive data, and terminate unauthorized data access efforts. PLCtrust deploys its information flow tracking engines across different layers of the PLC's VxWorks operation system and its Linux kernel to ensure data tracking across the intra-device entities. PLCtrust enforces access control policies at three points. PLCtrust monitors and controls data accesses within the Linux kernel to make sure that operating system processes do not propagate tainted data illegitimately. File accesses and inter-process communication (IPC) calls among the processes are monitored at this level. Access control policies are enforced in the Linux IPC to monitor and stop unauthorized inter-control logic program communications within Bob's PLC that cause PLCtrust's capsule policy violations.

A. Capsules

PLCtrust allows dynamic definition and enforcement of virtual micro security perimeters within the programmable logic controllers. Each micro perimeter is defined using a capsule by the data owners who could be *i)* power system operators and PLC users (e.g., who would like to prevent their sensitive control system-sensitive data file to leave the device); *ii)* control logic program developers (e.g., proportional-integral-derivative control algorithm developers who would not like any uncertified parties to access their file on the device); and *iii)* third-parties (e.g., the substation owners who may not wish to have the sensor measurements obtained at their places to be posted online). It is important to note the threat model variation for the cases above. In particular, in the first case, the controller users protect their sensitive data against outsiders such as malicious applications and network exploits, whereas in the second and third cases, the external parties (i.e., developers and third-parties) attempt to protect against curious (and potentially malicious) controller device users.

For each micro security perimeter, PLCtrust gets the corresponding perimeter's virtual boundary definition as a part of a signed capsule, which contain the policy ruleset file and additional optional files that are pushed onto the controller device during the capsule installation process. In particular, capsules can be downloaded, installed, and enforced dynamically while the device is running. At installation time, PLCtrust assigns each capsule with a local label that is unique according to the capsule's signature. This label is used to mark the corresponding sink data objects and network connections at runtime and the following data propagations (Figure 2). By extension, a capsule represents a set of controller logic programs, data and connections that correspond to a specific use, e.g., entities and activities related to the user's work environment as opposed to those of his/her personal matters. By default, capsules are isolated the ones from the others according to a policy defined per capsule or per group of capsules. PLCtrust allows the capsule objects to overlap, enabling the user to use a single controller logic program in different contexts transparently.

Capsule contents. Each capsule contains access control policy rules to be enforced and includes the list of entities that PLCtrust links to that specific capsule. In particular, each policy file contains seven possibly empty sections:

1) A system-wide unique capsule ID.
2) The set of rules that define how the meeting points among this capsule data and data tainted (linked) with other capsules on the PLC device should be handled. The policy rule could ask PLCtrust to *i)* allow a request, i.e., data can flow and/or be merged between capsules; *ii)* deny the request[1], i.e., no merging allowed; *iii)* audit and log the taint mixing incident; *iv)* delete a tainted data object, e.g., a sensor measurement file.
3) The list of temporal and geographical contexts, i.e., time intervals and geographic locations[2], where capsule objects have to be considered valid[3], as well as an action to be taken when the context is invalidated, e.g., deletion or encryption of tainted data objects or releasing the object as taint-free. The temporal and geographical contexts for power system areas can be defined for when/where the sensors and actuators measurements and control commands, respectively, come from and are send to.
4) The list of data files and on-PLC directories that should be linked with the capsule label. Among several cases, this entry is specifically required when an application comes with its own sensitive data files and directories that need to be protected. Consequently, PLCtrust taints any data read from the listed files and directories with the associated taint label. PLCtrust additionally updates each file taint if it is written to by a tainted entity, e.g., an application process.
5) The set of control logic programs that are included in this capsule. PLCtrust will monitor the execution of those controller programs and will not allow them to access and process tainted data unless no policy rules is violated. For control logics that are not tagged by any installed capsule, PLCtrust maintains them as taint-free initially; however, the controller program inherits the label from the sending source object as soon as it receives tainted data from other processes or via reading labeled objects.
6) The connections, e.g., SSL certificates, that PLCtrust has to consider as taint sources by labeling controller programs that receive data from them. As a case in point, capsules related to the control system substation "A" may define the sockets that are connected to IPs associated with "A" as a taint source so that the data coming from those connections should be labeled as "A" data.

PLCtrust implements an easy-to-use capsule distribution and installation interface for nontechnical users, power operators, and capsule owners. The capsules are distributed via PLCtrust's server that is also in charge of certificate issuance as well as controller program vetting regarding the physical plant's safety constraints (Figure 1). Our distribution scheme aims to allow capsule owners to create capsules and define policies they would like to enforce on their data as well as to allow users to aggregate multiple capsules in a persona, e.g., substation "A". The PLCtrust's capsule distribution framework includes a platform verification procedure to verify the genuineness of PLCtrust on the embedded PLC controllers to make sure that policies will be correctly enforced by the system on the corresponding data capsules.

IV. On-Device Information Flow Tracking

Dynamic information flow tracking is a programming language analysis techniques with applications for cyber security. The objective is to track data flow within the system as the

[1]PLCtrust prioritizes *deny* over *allow* in the case of conflicting capsule policies. Alternatively, the policies could be analyzed in a offline manner for consistency that is outside the scope of this paper.

[2]The geographic locations of power system areas in PLCtrust are defined as circles (the center point's latitude and longitude information along with a radius).

[3]At each time instant, PLCtrust enforces policies on capsule objects that are valid at that moment.

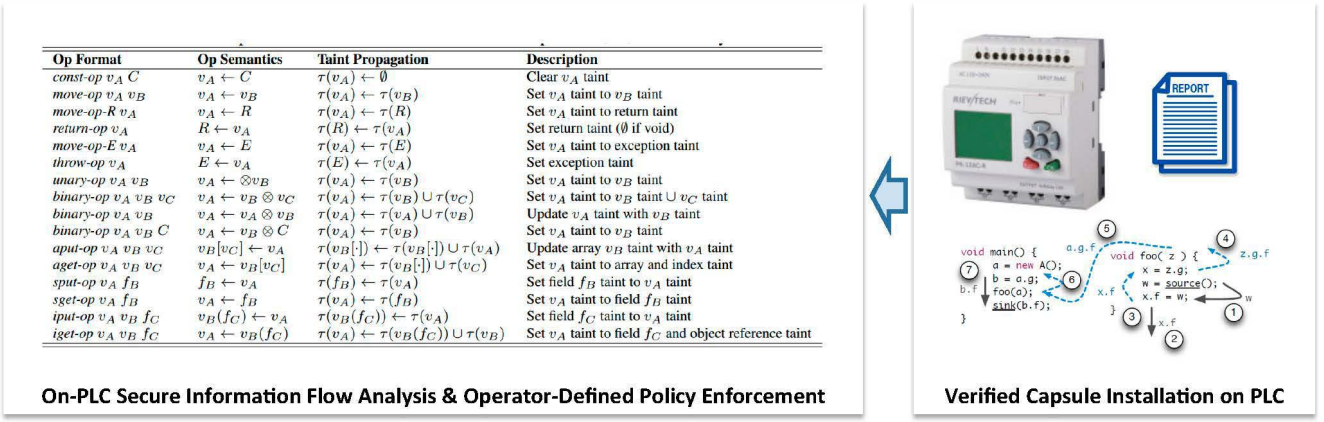

Fig. 2: Dynamic Information Flow Analysis

result of binary code instruction execution at runtime. The data sources such as files, network sockets, sensor inputs to the system are initially marked as sources initially. The information flow tracking solution associates a *taint* ID with each data source, and maintains a logical shadow memory of the system memory and storage during its runtime operation. The shadow memory contains an updated version of meta data (taint ID set) for each memory address. For instance, let us assume the following scenario: a process reads data from a network SSL socket marked as a sensitive data source, and sends it to another processing through inter-process communication channel; the target process uses the received data to create a file on disk before exiting. The underlying dynamic information flow tracking should track the propagation of the sensitive data throughout the system-level activities above, and mark the destination file as *tainted* with the sensitive data ultimately.

PLCtrust performs similar information flow tracking on the PLC device for the control logic program executions on its firmware. The goal is to determine the set of inputs to the device and sensitive data sources that affect the value of each output value from the PLC to the underlying plant actuators. Real-time maintenance of such fine-grained information flow allows PLCtrust to facilitate security access control policies within the system, whenever unauthorized data accesses and modifications are about to happen. For instance, a power system-safety policy may allow modification of certain output ports or read access of a certain confidentiality-sensitive input sensor port on the PLC by only the control logic programs that are developed by certain authorized operators. Dynamic information flow tracking allows PLCtrust to detect and block whenever a control logic program developed by a certain developer tries to access a sensitive input port, or write to a certain output port. The verification of the control logic program developers can be done by PLCtrust's server and through a cryptographic public/private-key certificate-based validation (Figure 1). The dynamic information flow tracking is implemented following a sequence of data flow ruleset as shown on left side of Figure 2. Individual instruction execution may result in a low-level data flow between memory addresses and registers, and hence requires an update to the taint propagation shadow memory.

V. Automated Power Safety Policy Generation

PLCtrust can enforce any given set of security data access control policies on the PLC controllers dynamically. For concreteness, we discuss how a set of power system safety policies can be generated automatically for PLCtrust's runtime enforcement. PLCtrust uses power system contingency analysis algorithms to implement a speculative what-if analysis of potential security incidents on the power system's global safety status.

Safety and security operation of real-world power grid critical infrastructures calls for accurate contingency analysis based on modeling and analysis of the power system dynamics. The core functionalities of the smart grid [4] compared to traditional power systems are the following: *i)* monitoring that requires instrumentation of the power system components with many sensors to measure various system parameters such as transmission line current magnitudes; *ii)* state estimation and contingency analysis that receive the noisy sensor measurements, calculate noiseless current state of the system using algorithms such as weighted least squares [1], and perform speculative risk analysis of potential upcoming system component failures. To enable such analyses, critical infrastructure operators and utilities employ power system modeling. The most common type of models used in power system operations are steady state power flow models and sensitivities. These models are used to monitor the state of the system and predict the effects of changes on the system.

PLCtrust performs steady-state security-oriented contingency analysis of the power system to determine the potential incidents that the target PLC controller can drive the underlying power system towards unsafe system states. Unlike traditional contingency analysis, security-oriented contingency analysis of PLCtrust does not consider all possible failures, and instead focuses on failures that can caused maliciously by the target PLC controller given its outputs and controllable power system actuators. Steady state power system modeling consists of enforcing the conservation of power. Given a set of power injections and withdrawals, the power flow finds the set of voltages and angles that satisfy the conservation of power. The system state may be written as

$$\mathbf{x} = [\mathbf{V}, \theta] \tag{1}$$

where $\mathbf{V}$ is a vector of voltages, θ is a vector of voltage angles. The vector of real power loads is $\mathbf{P_l}$ and the vector of reactive power loads is $\mathbf{Q_l}$. Since generator outputs are controllable (within limits), they are collected separately in a vector of controls, $\mathbf{u}$.

The power flow problem can now be written as

$$\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0} \tag{2}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{0}$ is a complex vector representing the injection at each node in the system. The function $\mathbf{f}(\mathbf{x}, \mathbf{u})$ represents the system model. It encapsulates factors like line impedances and system topology. Breaking $\mathbf{f}(\mathbf{x}, \mathbf{u})$ into real and reactive parts gives

$$f_i^p = -P_i^g + P_i^l + \sum_{k \in C} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \tag{3}$$

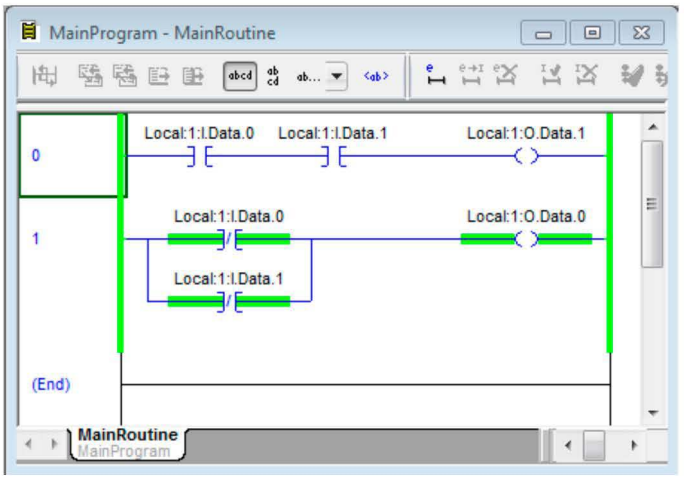

Fig. 3: Ladder Logic program developed in the PLC Vendor's Software Suite.

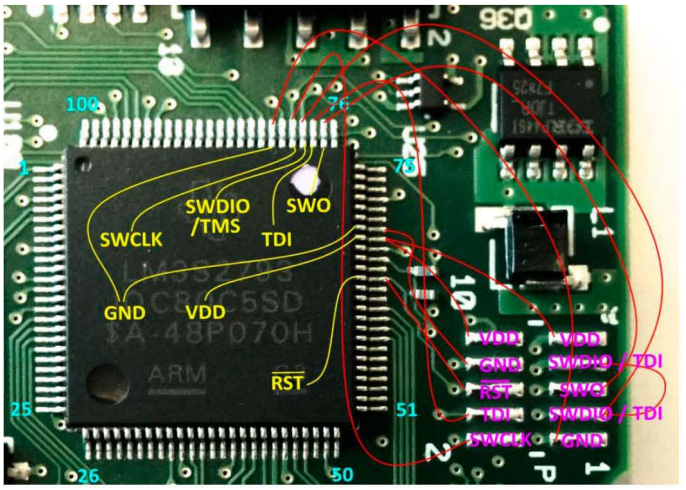

Fig. 4: LM3S2793 JTAG Pins and Their Connections to the 10-pin ARM-JTAG Connector.

$$f_i^q = -Q_i^g + Q_i^l + \sum_{k \in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \quad (4)$$

These equations represent the nonlinear problem that is commonly called the power flow in power systems literature. The power flow is at the heart of most power systems analysis. It provides the basis for many tools and sensitives that are used to predict the state of the system in the event of an outage.

PLCtrust enumerates all possible outputs of the target PLC controller and analyzes their impact against the power system's current state (based on the most recent sensor measurements and energy management system's state estimation server outputs) and its global safety. The safety of the power system is defined by the contingency analysis constraints such as the power line current capacities and generators' maximum set-points given their physical limitations. Given each security-oriented incident by the target PLC controller, PLCtrust solves the power flow equations above and estimates the system parameters for the components for which the contingency constraints are defined, e.g., line current phasors. Consequently, PLCtrust determines the incidents that violate power system safety requirements and adds them to PLC's internal safety policies.

Given the policies, PLCtrust's runtime information flow analysis and policy enforcement ensures that the control logic program execution on the device does not violate the power system safety requirements. Any output value write request to the actuators by the control logic programs that cause system unsafety is blocked by PLCtrust's real-time policy enforcement.

VI. Evaluations

We developed a simple ladder logic program to demonstrate how PLCtrust works in practice. Figure 3 shows a screenshot of our ladder logic program developed using the PLC vendor's (Rockwell's) Software Suite.

The program monitors two inputs: Input Port 0 and Input Port 1. The safety specification of this system requires that those two inputs cannot be high at the same time. If both inputs are high at the same time, we raise an "alarm" by setting Output Port 1 high. Otherwise, Output Port 0 is high, representing that the system is in a safe state.

We implemented two practical investigation of PLC firmware binaries for the real-world PLC controller. We reverse engineered the firmware of the controller through hardware J-TAG dynamic analysis and firmware acquisition. Given the firmware binary, we disassembled the acquired executable using the TI ARM processor data sheets. Starting with the basic static function call graph tracing, we narrowed down our investigation to the functions of our interest, i.e., functions that write memory values to the output modules (connected to the actuators on the physical system) and the functions that read input module values (sensor measurements) into the memory values. These functions are where PLCtrust's policies are enforced to ensure sensor measurement reading and actuation command writes do not happen by unauthorized parties. Figure 4 shows the PLC's CPU, i.e., a Texas Instruments LM3S2793. The CPU's pin allocations are marked as well as the connection of the JTAG pins to the solder pad on the right side of the board.

Figure 5 shows the PLC's main board soldered-on JTAG header and connected cable. The pin layout of the JTAG header was not directly compatible with our JTAG debugger, so we had to rewire the individual pins.

We developed two safety attack scenarios against PLCtrust's policy enforcement engine. In our first attack scenario, the attack raised an alarm despite the system being in a safe state. Furthermore, the LED's and the HMI show that the system is in a safe state despite having an alarm raised from the actual embedded I/O module. In the second attack scenario, we modified the input to the PLC to force an alarm without changing the logic of the Output Ports. We will ignore the incoming input values from the embedded I/O module and send a different value to the LED's and HMI. This value will make the PLC believe that the two inputs are high and raise an alarm despite no actual inputs being connected to the embedded I/O module. PLCtrust's control logic program execution monitoring and runtime safety policy enforcement engine was able to identify unauthorized unsafe system state and blocked the actions by the malicious control logic program.

VII. Related Work

Research on embedded system security and taint analysis is extensive. We review the most related recent ones here.

We first discuss security and verification solutions presented for control systems. TSV [8] presented an external bump-in-the-wire verifier for process controller code downloaded to the PLC. Mohan et al. [9] introduced a monitor

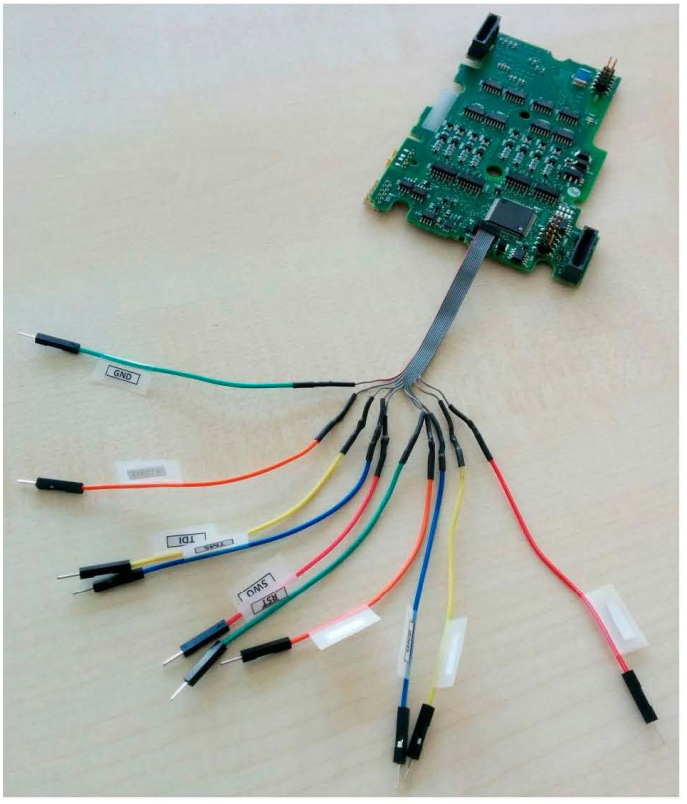

Fig. 5: 10-pin ARM-JTAG Cable Connected to PLC Main Board.

that dynamically checks the safety of plant behavior. Offline intrusion detection solutions have been proposed to model PLC traffic as a deterministic finite automaton in [6] and [7]. Another model based intrusion detection was proposed in [2]. In all cases, the security solutions were implemented as external solutions as opposed to within the PLC. Avatar [14] provides a framework to support dynamic security analysis of embedded systems firmware. However, the firmware resides below the control logic level and security/verification solutions cannot be easily integrated into the scan-cycle of the PLC. In general, our solution focuses more on application-level security solutions. [5] uses mathematical analysis techniques to evaluate various aspects, such as safety and reliability, of a given control system, but focuses on accidental failures and not malicious actions. PLC vendors themselves typically use basic security mechanisms with a single privilege level [8].

Taint analysis solutions keep track of information accessed by the different entities on your system. HiStar [15] introduces a system using a small trusted code base and uses Asbestos [3] labels to perform information flow control between objects. cleanOS [12] introduces an approach to use information flow tracking as a method to encrypt and evict sensitive data out of the embedded devices. Access control mechanisms have also proposed to provide a separation between different components. SELinux [11] implements mechanisms at the kernel and middleware level, and provides robust and fine-grained mandatory access controls; however, their policies are defined statically and, unlike PLCtrust, do not allow execution in dynamically defined multiple contexts.

Security mechanisms used to establish a root of trust are necessary in order to ensure that a client is operating on a trusted platform. Trusted platform modules (TPM) are hardware cryptographic chips used to store cryptographic keys specific the owning device. TPM is not currently widely deployed on most embedded controller devices in the market. There have been several solutions to provide a software virtualization mechanism of a TPM chip. TPMε [10] (mini TPM emulator) emulates a TPM chip by measuring all executables loaded on a platform in order to provide a mechanism for remote attestation. However, if the PLC device is rooted, then any rogue contro logic application will have access to the loaded modules and may also compromise the integrity of the TPM ε solution.

VIII. Conclusions

PLCtrust aims at provision of a transparent solution to protect programmable logic controller users and external code developers and program owners against unauthorized data leaks, code integrity-violating and power system safety attacks.PLCtrust deploys system-wide information-flow tracking as well as intra- and inter-process level mandatory access control for two purposes. First, PLCtrust enables code developers, such as PLC users and control logic developers, to define virtual micro security perimeters, so-called capsules, to determine how their original and other data affected by their data should be treated by the operating system. Second, using a runtime solution, PLCtrust keeps track of the defined capsule boundaries at each time instant within the PLC controller. PLCtrust provides a secure code development and execution solution that is completely transparent to the operators and code developers, and facilitates dynamic definition and efficient installation of capsules in real-time.

Acknowledgement

This material is based upon work supported by the National Science Foundation under Award Number CNS 1453046.

References

[1] A. Abur and A. Expósito. *Power System State Estimation: Theory and Implementation*. Marcel Dekker, 2004.

[2] S. Cheung, B. Dutertre, M. Fong, U. Lindqvist, K. Skinner, and A. Valdes. Using model-based intrusion detection for scada networks. In *Proceedings of the SCADA security scientific symposium*, volume 46, pages 1–12. Citeseer, 2007.

[3] P. Efstathopoulos, M. Krohn, S. VanDeBogart, C. Frey, D. Ziegler, E. Kohler, D. Mazieres, F. Kaashoek, and R. Morris. Labels and event processes in the asbestos operating system. *ACM SIGOPS Operating Systems Review*, 39(5):17–30, 2005.

[4] H. Farhangi. The path of the smart grid. *Power and Energy Magazine, IEEE*, 8(1):18–28, 2010.

[5] W. M. Goble. *Control systems safety evaluation and reliability*. ISA, 2010.

[6] N. Goldenberg and A. Wool. Accurate modeling of modbus/tcp for intrusion detection in scada systems. *International Journal of Critical Infrastructure Protection*, 6(2):63–75, 2013.

[7] A. Kleinman and A. Wool. Accurate modeling of the siemens s7 scada protocol for intrusion detection and digital forensics. *The Journal of Digital Forensics, Security and Law: JDFSL*, 9(2):37, 2014.

[8] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel. A trusted safety verifier for process controller code. In *NDSS*, 2014.

[9] S. Mohan, S. Bak, E. Betti, H. Yun, L. Sha, and M. Caccamo. S3a: secure system simplex architecture for enhanced security of cyber-physical systems. *arXiv preprint arXiv:1202.5722*, 2012.

[10] M. Nauman, S. Khan, X. Zhang, and J.-P. Seifert. Beyond kernel-level integrity measurement: enabling remote attestation for the android platform. In *Trust and Trustworthy Computing*, pages 1–15. Springer, 2010.

[11] S. Smalley and R. Craig. Security enhanced (se) android: Bringing flexible mac to android. 2013.

[12] Y. Tang, P. Ames, S. Bhamidipati, A. Bijlani, R. Geambasu, and N. Sarda. Cleanos: Limiting mobile data exposure with idle eviction. OSDI, 2012.

[13] G. A. Weaver, C. Cheh, E. J. Rogers, W. H. Sanders, and D. Gammel. Toward a cyber-physical topology language: Applications to nerc cip audit. In *Proceedings of the first ACM workshop on Smart energy grid security*, pages 93–104. ACM, 2013.

[14] J. Zaddach, L. Bruno, A. Francillon, and D. Balzarotti. Avatar: A framework to support dynamic security analysis of embedded systems' firmwares. In *NDSS*, 2014.

[15] N. Zeldovich, S. Boyd-Wickizer, E. Kohler, and D. Mazières. Making information flow explicit in histar. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation*, volume 7, pages 19–19, 2006.