Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit

Luis A. Garcia
Rutgers University
l.garcia2@rutgers.edu

Ferdinand Brasser
Technische Universität Darmstadt
ferdinand.brasser@trust.tu-darmstadt.de

Mehmet H. Cintuglu
Florida International University
mcint015@fiu.edu

Ahmad-Reza Sadeghi
Technische Universität Darmstadt
ahmad.sadeghi@trust.tu-darmstadt.de

Osama Mohammed
Florida International University
mohammed@fiu.edu

Saman A. Zonouz
Rutgers University
saman.zonouz@rutgers.edu

***Abstract*—Trustworthy operation of industrial control systems (ICS) depends on secure code execution on the embedded programmable logic controllers (PLCs). The controllers monitor and control the underlying physical plants such as electric power grids and continuously report back the system status to human operators.**

We present HARVEY, [1] a PLC rootkit that implements a physics-aware stealthy attack against cyberphysical power grid control systems. HARVEY sits within the PLC's firmware below the control logic and modifies control commands before they are sent out by the PLC's output modules to the physical plant's actuators. HARVEY replaces legitimate control commands with malicious, adversary-optimal commands to maximize the damage to the physical power equipment and cause large-scale failures. To ensure system safety, the operators observe the status of the power system by fetching system parameter values from PLC devices. To conceal the maliciously caused anomalous behavior from operators, HARVEY intercepts the sensor measurement inputs to the PLC device. HARVEY simulates the power system with the legitimate control commands (which were intercepted/replaced with malicious ones), and calculates/injects the sensor measurements that operators would expect to see. We implemented HARVEY on the widely spread Allen Bradley PLC and evaluated it on a real-world electric power grid test-bed. The results empirically prove HARVEY's deployment feasibility in practice nowadays.

I. INTRODUCTION

Industrial control systems (ICS) interconnect, control and monitor industrial environments such as electrical power generation, transmission and distribution, chemical production, oil and gas refining and transport, and water treatment and distribution. In recent years, ICS have received considerable attention due to security concerns originated by the trend to connect ICS to the Internet [22]. In particular, critical infrastructures connected to and controlled by ICS substantiate these security concerns. Nevertheless, the ICS market is expected to grow to $10.33 billion by 2018 [47].

Nation-state ICS malware such as the Stuxnet worm [24] against Iranian nuclear uranium enrichment facilities and BlackEnergy crimeware [23] against the Ukranian train railway and electricity power industries show that targeted attacks on critical infrastructures can evade traditional cybersecurity detection and cause catastrophic failures with substantive impact. The discovery of Duqu [14] and Havex [43] show that such attacks are not isolated cases as they infected ICS in more than eight countries.

ICS security has been traditionally handled using network security and information technology (IT) practices [52]. ICS security goals, however, differ from traditional IT security goals due to additional requirements and conditions of operation. The interconnection of the physical world and virtual world, bridged by cyberphysical systems (CPS), is a unique feature of ICS compared to traditional IT infrastructures. And unlike most traditional IT systems, high availability is critical for ICS. A process failure can have fatal consequences threatening human lives and resulting in immense financial loss.[2]

ICS are monitored and operated in a centralized fashion: embedded CPS, known as programmable logic controllers (PLC), are connected to a central control terminal (human-machine interface HMI) through which a human operator can supervise the system. PLCs are digital computing devices used for automating industrial electromechanical processes. They control the state of the output ports based on signals received from the input ports and stored programs, and operate typically under hard environmental conditions, such as excessive vibration and high noise [9], [20]. PLCs control standalone equipment and discrete manufacturing processes. Their logical behavior with regard to inputs and outputs can be programmed by the operator.

The main goal of sophisticated attackers is to remain stealthy from ICS operators. In particular, the HMI's view of the system should not indicate any effect caused by attacks. For this, the adversary can either compromise and manipulate the HMI itself, or launch a more sophisticated attack on PLCs. A prominent example of HMI related attacks is the infamous Stuxnet [24]. However, HMIs are often based on commodity computer systems for which a wide variety of security solutions exist, including anti-virus software, security enhanced operating systems, run-time attack protections, and many more. This makes the HMI an unattractive attack target. On the other hand, although many PLC related attacks have

[1] Harvey Dent (Two-Face) is a fictional super-villain adversary of the superhero Batman. The right half of his face looks normal/benign, unlike the hideously scary left side of his face.

NDSS '17, 26 February - 1 March 2017, San Diego, CA, USA

http://dx.doi.org/10.14722/ndss.2017.23313

[2] ICS are also not what is usually considered Internet of Things (IoT) as there are substantial differences (cf. Section VIII-A).

been published in recent years [8], [12], [32], [44], they have limitations with regard to stealthiness and result in obvious effects, such as disrupting the operation of PLCs [44]. Other attacks operate on the PLC's application level (called *control logic*), which allows the operator to detect their presence through the PLC's remote management capabilities [12], [32].

Goals and Contributions. In this paper we present a novel class of *stealthy* PLC attacks that we refer to as Man-in-the-PLC. Our exploit intercepts the PLC's input and output values, provides an arbitrary view of the system to the control logic (i.e., the program running on the PLC), and simulates a semantically correct system state towards the central control unit while changing the actual system state. We provide the following main contributions:

- We present a novel attack class on industrial control systems: a cyber-physical attack which is completely invisible to the control center of an ICS.
- We reverse engineered the central control loop mechanism of a widely deployed Allen Bradley 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B PLC.
- We developed a prototype implementation of HARVEY, and tested and evaluated it on an Allen Bradley PLC. Allen Bradley is one of the most used ICS suppliers in the United States.
- We evaluate our attack in a real power grid test environment.

We would like to stress that our main contribution and novelty of our rootkit lies in its physics-awareness. This makes our solution more general than all solutions published before, including real world attacks like Stuxnet [24]. To be able to implement and evaluate our prototype, we had to reverse engineer the PLC to gain the required insight into its inner working, in particular, the PLC's control of input and output lines, and the connection between the firmware and control logic programs.

Following the standard responsible disclosure policies, we have taken necessary steps and have contacted the vendor, Allen Bradley, informing them about the possibility of such malicious exploits against their controllers. Allen Bradley gave us clearance to publish our findings.

The remainder of the paper is structured as follows. First, we provide the reader with a background on industrial control systems in general and programmable logic controllers in particular, and present our system model and adversary model in Section II. In Section III, we explain the general concept of HARVEY before providing details on the physics-aware data manipulations of our attack in Section IV. In Section V, we describe how we reverse engineered the firmware of an Allen Bradley PLC and implemented our rootkit. In Section VI, we provide an extensive evaluation for our physics-aware PLC rootkit against a real power grid test-bed. Section VII provides a review of related work in the area of ICS security. We discuss our findings and possible mitigation strategies in Section VIII, and conclude in Section IX.

II. BACKGROUND AND SYSTEM MODEL

In this section, we provide basic knowledge for the rest of the paper. We provide detailed information about industrial control systems (ICS) and programmable logic controllers (PLC), and we define the system model and adversary model we will consider throughout the paper.

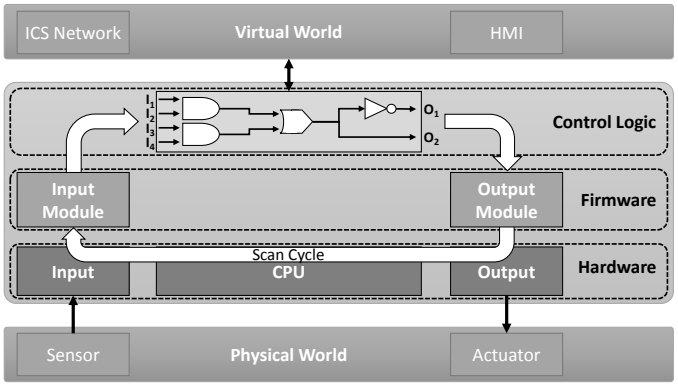


Fig. 1: PLC Architecture

A. Background

An ICS is a distributed system which is composed of physical components, like sensors and actuators, which interact with the physical system (e.g., power grid) and cyber components (e.g., cyber-networks and servers). Although ICS are largely self-contained, interfaces exist through which external components can interact with the systems. For instance, a human operator can monitor the systems state and influence it through a human-machine interface (HMI). Most PLCs are connected to the ICS via an Ethernet network, and hence, often indirectly connected to the Internet. It is also quite common that PLCs are directly connected to the Internet [32].

Centralized operation and maintenance is an essential feature of ICS. An operator can program and monitor PLCs and the applications running on them remotely, i.e., retrieve the status of a PLC and re-program it over the network. The information which can be retrieved from the PLC contains, among others, the control logic applications on the PLC. All applications, including their source code and further meta information, can be loaded from a PLC.[3]

Programmable Logic Controllers (PLC). Programmable logic controllers (PLC) are cyber-physical systems that are used to control industrial appliances. PLCs have input and output modules to interact with the physical world. They can translate physical inputs, mostly current on a wire, into digital values and vice versa. Connected to physical appliances such as sensors and actuators, PLCs can convert sensor readings into digital values, process the readings with the built-in computing unit, and forward the outputs to the physical world. The logical behavior of PLCs, i.e., the processing of the input data, is programmable.

Such control loops can be either local, i.e., the inputs and outputs are handled by a single PLC, or distributed, i.e., the inputs are read by one PLC and forwarded over the network to another PLC.

The two main software components of a PLC, control logic and firmware, are shown in Figure 1. The firmware is acting as a kind of operating system (OS). The firmware contains, among other functionality, modules to read and write the inputs and outputs of the PLC from/to the physical world. These modules can be seen as drivers for the I/O hardware. Control logic programs can be considered the PLC's application layer. The firmware provides services to read from input lines and write to output lines of the PLC. They are used by the control

[3] This enables the operator to detect malicious modification of the PLC on the control logic layer.

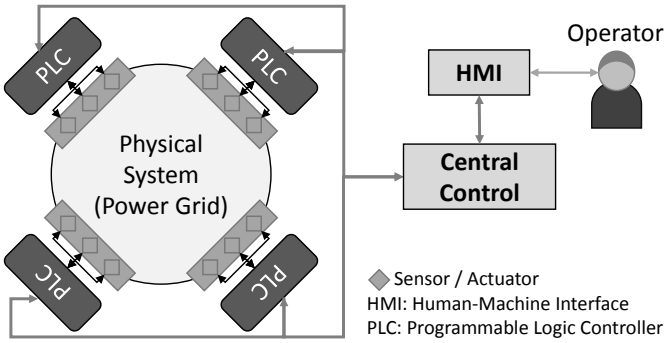


Fig. 2: System Model

logic to program the behavior of the PLC. The control logic programs are executed repeatedly in fixed intervals, called scan cycles. The control logic program reads input values from memory and stores the output values to memory. The underlying firmware is responsible for the interchange of these updated values to and from the PLC's general purpose input/output (GPIO) ports, i.e., the interface to the physical system, as well as the reporting mechanisms of the PLC, i.e., the LED display on the device and the HMI. The scan cycle is illustrated by the white arrows in Figure 1.[4]

For instance, a pump can increase the pressure in a pipe. In order to have a constant pressure in the pipe the pump must be active until the predefined pressure is reached. Whether the desired level has been reached is determined based on sensor readings. However, the sensor and the pump are not directly connected. The sensor measurements are read by the PLC through its input lines. The value is then processed by the control logic, and the result is translated back into the physical system to steer the pump.

This approach allows for high flexibility, e.g., the sensor and actuator (pump) might not operate in close proximity and the sensor measurements have to be sent to the PLC controlling the actuator. This can be done cost-efficiently through existing computer networks, even over the internet. Furthermore, processing data in the cyber network allows for more complex relations between sensor measurements and actions.

B. System Model

In this paper, we mainly consider large distributed ICSs that are operated in a centralized manner. Figure 2 shows an abstract view of our system model in which a physical system is operated from a central control terminal. The control terminal provides a HMI that allows an operator to monitor the system and interact with the system (by sending control commands). The connection between the control terminal in the cyber world of the ICS and the physical world is provided by PLCs. PLCs capture the physical system's state by reading measurements from sensors. Additionally, PLCs control the system's actuators, based on both the control actions generated by their local control logic and the control commands sent from the operator.

[4]PLCs can be programmed with multiple independent control logic applications which are executed sequentially within in each scan cycle.

C. Adversary Model and Assumptions

Stealthiness. The main goal of the adversary is to launch a stealthy attack on an industrial control system (ICS).[5] Stealthy means that the attack does not cause unintentionally observable effects. For instance, sensor readings analyzed by a system operator or automated tools should align with what they are expected to be. Real world examples like Stuxnet [24] have shown that stealthy attacks have a more enduring impact on a system than a short attack which will rapidly break down a system.

The attack exploits the circumstance that in real world systems the operator's view of the entire system is limited to the information provided by the HMI, i.e., he cannot directly observe the physical system and detect attack effects through an out-of-band channel like visual contact. This limitation can be due to different reasons, e.g., in large and distributed systems the operator is physically not capable of observing the entire system, or the system operates in a hazardous environment and for safety reasons the operator only has remote access to the system.

PLC-only Attack. We assume that the adversary compromises only PLCs and no other components of the ICS, hence the name Man-in-the-PLC. In particular, we do not assume that the adversary has manipulated the human-machine interface (HMI), e.g., to hide suspicious activities from the operator.[6] Besides an operator observing the HMI, the ICS might include security mechanisms like SCADA-specific[7] intrusion detection systems (IDS) that monitor the system [52]. We assume the adversary cannot compromise (all) monitoring entities (i.e., IDS systems) in an ICS in order to hide an attack.

Furthermore, ICS components like HMI's are usually based on commodity hardware and software, e.g., a workstation PC running Windows operating system. Security solutions for those systems already exist, e.g., anti-virus software and automated software update solutions, increasing the probability for detection of an attack.

Physical Model Extraction. We assume the adversary has knowledge about the inner workings of his target and uses this information to build a model of the correct behavior of the target to be able to hide suspicious effects of the attack. The adversary can get the required information, for instance, through an insider, or through preceding information gathering attacks [32]. Although physical model extraction is outside of the scope of this paper, it is worth noting that monitoring and management systems of the ICS can be leveraged to extract information about the physical model. For instance, in power systems, energy management systems (EMS) are used to control the power grid infrastructure. An EMS is a collection of computer-aided tools used by operators of electric utility grids to monitor, control and optimize the performance of generation and transmission systems. A typical suite of EMS applications includes several components that feed sensor measurements into state estimation systems, contingency analysis software, optimal power flow control analysis software, as well as an HMI. Hence, the power system topology information can be extracted through insider intruders (e.g., Stuxnet [24]) or EMS compromises. Additionally, unlike in purely-cyber settings, the physical power system and its topology is often exposed to outside world; therefore, physical system reconnaissance is relatively simpler.

[5]Obviously the attacker could also use HARVEY's capabilities to cause very visible attacks if he chooses to.

[6]Stuxnet for instance utilized a compromised HMI to hide itself from the operator [24].

[7]SCADA: supervisory control and data acquisition.

III. HARVEY: MODEL-AWARE ROOTKIT

The central property of our rootkit HARVEY is the fact that it takes into account the physical topology of the industrial control system (ICS) it infects. This gives HARVEY unique capabilities. Most importantly, it allows our rootkit to be stealthy. HARVEY is completely invisible to the ICS's virtual world. This means that the effects of attacks launched by HARVEY can neither be detected by human operators monitoring system measurements nor by security tools like intrusion detection systems (IDS) monitoring the ICS network. This makes HARVEY uniquely powerful and goes beyond attacks known today.

The idea of our model-aware rootkit is to infect the firmware of a programmable logic controller (PLC), allowing HARVEY to control all inputs and outputs of the PLC. The control logic program gets access to the PLC's input values through the firmware from the physical world, processes them, and then provides outputs that are forwarded to the physical world through the firmware. The control logic can also interact with other cyber components in the industrial control system (ICS) over network.

Because HARVEY lives in the firmware layer and intercepts the control and information flow of the firmware, it is completely transparent to the control logic. Each output which is passed from the control logic to the firmware is captured by HARVEY and can be changed, e.g., to maximize the effects of the attack. Similarly, HARVEY can change the input values seen by the control logic arbitrarily, e.g., to hide effects of its attack.

HARVEY only compromises the PLC's firmware, hence, it cannot be detected by the operator's PLC management tools. In contrast, malware that operates on the control logic level can be detected through the PLC's remote management capabilities. Control logic malwares need to rely on additional techniques or assumptions to hide themselves from the operator. Stuxnet compromised the operators workstation to hide itself [24]; Brüggemann and Spenneberg rely on the observation that they can cause the operator's remote management software to crash by manipulating meta-data stored on the PLC [12].

Although our Man-in-the-PLC rootkit cannot be detected directly by the operator, it could still be detected indirectly through the effects it causes. To prevent unintended, possibly suspicious effects in the ICS, HARVEY does not change input and output values arbitrarily. Instead, our rootkit acts according to a model of the target system which ensures that the operator's view of the system stays consistent with his expectations. This means that if the malware's goal is to increase the pressure in a pipe to damage it, it is not sufficient to activate the pump by setting the output of the PLC accordingly. A sensor would measure the increasing pressure and would alert the operator or trigger an automatic safety mechanism. Hence, for the attack to be successful, the malware must also ensure that sensor readings presented to the operator are not suspicious, e.g., hide the increasing pressure.

Since our rootkit uses a model to manipulate inputs *and* outputs of a PLC in a coordinated fashion, it can present a normal operation view towards the cyber world while manipulating the physical world.

Model. The model according to which our malware is operating can be created and/or obtained in different ways. HARVEY essentially makes use of the same models that are used to control the underlying physical platform legitimately. However, the malware optimizes the control commands for an adversarial objective function.

For attacks on simple systems the model would be simple, too, and can be created manually. For more complex systems the model can be created with automated tools.[8] The attacker's advantage is that he does not need to have a comprehensive model of the entire system. She only needs a model for parts of the system her attack will operate in. For instance, if the attacker aims to damage a specific pipe in a large plant, she only needs a model covering those components that her attack will effect, which might be as few as a single pump and a single sensor. Additionally, the model can also incorporate the deployed (if any) intrusion detection sensors to ensure the malicious control commands do not trigger the alerts.

In Section VI, we will present our evaluation results of our model-aware malware on a real power grid. It shows that manipulations in the physical world can be hidden from the cyber world.

PLC Infection. Our rootkit works by compromising a PLC's firmware, which the attacker can achieve in different ways. The attack can use the built-in firmware update mechanism to replace the firmware on a PLC. Depending on the PLC model, firmware updates are not secure against manipulations.[9] Firmware updates can be deployed over the network for most PLCs. Hence, PLCs which are reachable by the attacker over the network can be compromised directly.

The attacker can also compromise PLCs locally. Many PLCs allow firmware updates from local media such as SD cards. Additionally, the attacker can use hardware interfaces like JTAG[10] to connect to the PLC and manipulate its firmware. An attack has recently been presented that shows the ease of JTAG implantation [25]. In Section V we describe in detail our attack on a PLC through its JTAG interface.[11]

Lastly, if the previous attack methods are not available, the adversary could facilitate run-time attacks, e.g., network exploits. To the best of our knowledge, there are no PLCs available that have run-time attack mitigation techniques deployed. This means that attacks like code injection are likely to succeed on PLCs. Remote code execution vulnerabilities on PLCs have been issued just this past year, e.g., CVE-2016-0868, CVE-2016-5645 and all vulnerabilities associated with the FrostyURL vulnerability, such as CVE-2015-6490, CVE-2015-6492, CVE-2015-6491, CVE-2015-6488, and CVE-2015-6486. However, even in the presence of protection mechanisms, the developments in the desktop and server world have shown that attackers will find new means of circumventing these protection mechanisms. Code reuse attacks, like return oriented programming (ROP), have already been applied on embedded systems [27].

IV. PHYSICS-AWARENESS

This section explains how HARVEY manipulates the control actuation actions and sensor measurements within a PLC.

Figure 3 shows how HARVEY manipulates data streams to damage the physical plant while ensuring the operators see what they would expect to see based on their inputs to the system. HARVEY performs the attack through manipulation

[8] Related work focusing on modeling industrial control systems is described in Section VII.

[9] We will elaborate on our findings on firmware update security in Section V.

[10] Joint Test Action Group (JTAG) is an IEEE standard for testing and debugging integrated circuits.

[11] Note that the goal of our work is to show the effectiveness of physics-aware malware, providing novel infection vectors were not the scope of our efforts.

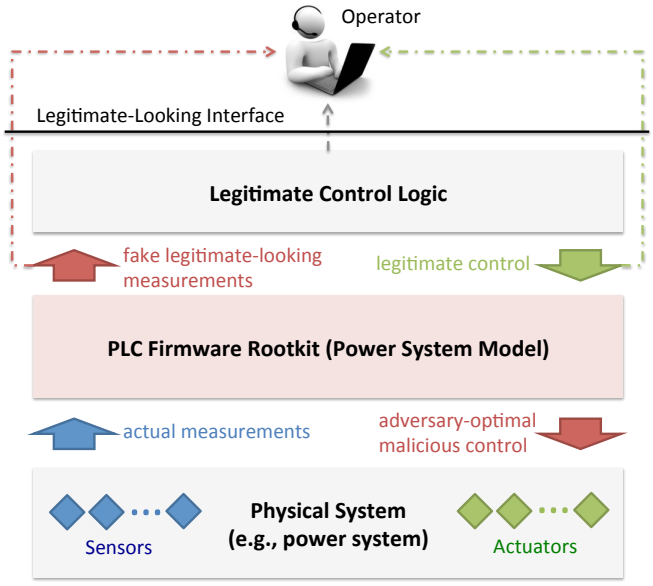


Fig. 3: HARVEY Two-Way Data Manipulation Attack

of data in both directions: *i)* control commands sent by the operators and/or the control logic on the PLC to the actuators deployed on the physical plant; and *ii)* sensor measurements from the deployed sensors to the operators. The control commands are corrupted to damage the physical system and cause system failures, whereas the sensor measurements are corrupted to evade the operator detection of the caused failures.

A. Control: Malicious Plant Actuation

In real-world control systems, e.g., power systems, the system dynamics continuously change due to various factors, e.g., electricity consumption changes by the civilians. Such changes require updated control of the physical system to ensure it maintains its core functionalities. For instance, in the power grid control systems, the generation set-points (control commands) are updated dynamically by the controllers to ensure the amount of power that is consumed equals the generated amount by the generators.

The control of the physical plant is often performed based on the models of the underlying physical dynamics, e.g., fluid dynamics for the water networks and Kirchhoff laws for the power systems. The models encode the correlations between the system parameters due to the physics laws caused by inter-component connections and dependencies. The most popular generic modeling follows linear dynamical state-based formalism $\dot{x} = f(x,u)$, where

$$f(x,u) = Ax + Bu + \omega \quad (1)$$

$$y = Cx + \varepsilon. \quad (2)$$

The matrices A,B,C represent the dynamics of the physical system, x is the state vector of the system, y is the sensor measurements sent to the PLC, and u are the control outputs from the PLC to the actuators. ω and ε encode the noise in system dynamics and the sensor measurements, respectively. The PLC control logic receives the sensor measurements y and calculates the corresponding optimal control commands u^* to maximize a domain-specific objective function $u^* = \arg\max_u \text{obj}(x,f)$, e.g., the minimum total generation cost in power systems. The calculated control commands are sent to the PLC output modules through its firmware facilities.

HARVEY sits within the firmware and intercepts the output module write requests, and replaces them with malicious control output u_m^*. It calculates the malicious control outputs through similar logic as the legitimate controller with only one difference: it calculates the control outputs that minimize the corresponding objective function $u^* = \arg\min_u \text{obj}(x,f)$ to maximize the damage on the physical plant. HARVEY writes the calculated commands to the corresponding PLC output ports that are connected to the physical plant actuators. The corrupted control commands drive the plant towards unsafe states leading to potential physical failures, e.g., power system blackouts.

It is noteworthy that HARVEY's malicious control output calculation can also incorporate additional constraints that may be introduced to evade the detection of the attack. For instance, in a water plant, the constraints may avoid an extreme increase of the pump rotation speed because of its potentially noticeable noise. Additionally, without the loss of generality, the linear model mentioned above can be replaced with either simpler or more complicated models used by the PLC control logic depending on the specific control system domain (see Section VI for an empirical power system case study.)

B. Monitoring: Sensor Data Corruption

The control command manipulation attack, discussed above, causes unsafe physical plant states, but it does not contain the necessary amount of stealth for practical real-world deployment. Hence, it can be easily detected by the control system operators, who continuously monitor the real-time physical plant state on the human-machine interface (HMI) screens. The HMI screens are frequently updated automatically by fetching the PLC's memory for the most recent physical system sensor measurements that have been delivered to the PLC's input module.

To evade the detection, HARVEY intercepts the sensor measurements on the PLC and replaces them with fake values that would make the underlying physical system status look normal to the operators. Stuxnet implemented an innovative system status record-and-replay attack. The worm would record the plant dynamics for 13 days before it injected malicious control logic on the PLC. Once the attack started on the plant (malicious control logic execution on the PLC), Stuxnet would replay the recorded data stream back to the operator screens.

Such record-and-replay attacks would work in specific control system plants with static and stationary low-pace dynamics such as uranium enrichment, which was Stuxnet's target. However, the sole record-and-replay attack would not be practical and can be easily detected if used in typical control systems with high-pace dynamics such as a power grid. In power grids, the operators constantly change the system configuration/topology and parameter values as a result of many external unpredictable factors, such as real-time demands by the end-point electricity consumers and real-time climate for renewable energy sources such as solar and wind generation plants. Therefore, a replay of a previously recorded sensor measurement stream back to the operators is very unlikely to match exactly the expected status of the power system following the operator's most recent control commands.

HARVEY addresses the challenge above in highly dynamic control system environments through real-time and lightweight physical plant simulation within the PLC firmware. HARVEY takes the legitimate controller commands that either the operator or the legitimate controller logic on the PLC issues to be sent to the output modules and actuators. HARVEY then simulates the physical plant dynamics by solving the corresponding plant models (e.g., Equation (1)). Through the simulation of the

physical system, HARVEY essentially calculates how the power system would "look" if the legitimate control commands would really be sent to the deployed actuators. HARVEY replaces the actual sensor measurements with their corresponding simulated fake values before they are written to the PLC memory. The following PLC memory reads by the operators' HMI software would be receiving the fake measurements. Hence, the HMI screens would show a legitimate-looking physical system state to the operators.

The fabricated PLC memory values are used as sensor measurements by the legitimate control logic that is developed by the operators and runs on top of the malicious PLC firmware. Consequently, the legitimate control logic will calculate control commands that satisfy the operators' expectations on the HMI screens. It is noteworthy that HARVEY does not replace the legitimate control logic execution. Instead, it runs its malicious code in parallel to the legitimate control logic, and hence the outputs from both executions are calculated and used for different purposes (i.e., for faking the physical system appearance and damaging its actual components).

C. Distributed Monitoring and Control

In practice, a large-scale control system is often maintained by a set of distributed PLCs, each in charge of their assigned local "zone". As an example, in power systems, the electricity grid is typically partitioned into several sub-areas each maintained by a separate controller [28]. In water plant facilities, the water treatment is often performed in a sequence of several serial phases such as chlorination, pH control, filtration, and disinfection. Individual phases are usually operated by separate control logic programs either all on the same PLC or each sitting on a separate PLC. For real-time monitoring and control, each PLC takes the monitoring and control responsibility of its associated zone independently such that its local sensor measurements suffice for its zonal control operations.

The distributed monitoring and control paradigm is traditionally employed to ensure real-time and reliable operation; however, it can be leveraged by the PLC firmware attacks such as HARVEY to ensure its stealthiness against large-scale control systems even when one or just a few of PLCs are infected. Put in other words, to perform an attack against a large-scale platform, the adversaries do not have to compromise all the controllers to maintain stealth.

V. HARVEY IMPLEMENTATION

This section describes our rootkit implementation for an Allen Bradley 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B. It is noteworthy that the attack implementation is explained specifically for the PLC model above. The PLC software/hardware architectures are fundamentally similar across various vendors. Hence, the proposed techniques can be generalized to other widely-used industrial PLCs.

HARVEY deeply interferes with the core functionality of the PLC's firmware. This interference allows complex behavior manipulations of the PLC, which is required for stealthy control system attacks as described in the previous section. Since the firmware of PLC is not openly available, we had to reverse engineer it as the first step of our prototype implementation. Most techniques in this multi-step process are known but nonetheless technically challenging and needed.

Device Selection and Specification. Before we get into the analysis details of the Allen Bradley 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B, we shortly explain which criteria we used to select the target device for our implementation. On one hand, groundwork on reverse engineering Allen Bradley PLCs had been done before by the Air Force Institute of Technology [7], [44]. On the other hand, unlike Siemens PLCs–which have received a lot attention in recent years–Allen Bradley PLCs are mostly uncharted. Nevertheless, Allen Bradley is one of the largest vendors for industrial control systems internationally. In particular, the CompactLogix L1 series is widely used in several safety-critical infrastructure applications such as power grids, water plants, oil&gas refineries, and medical devices (e.g., the Adept Viper S650 surgical assistant robot).

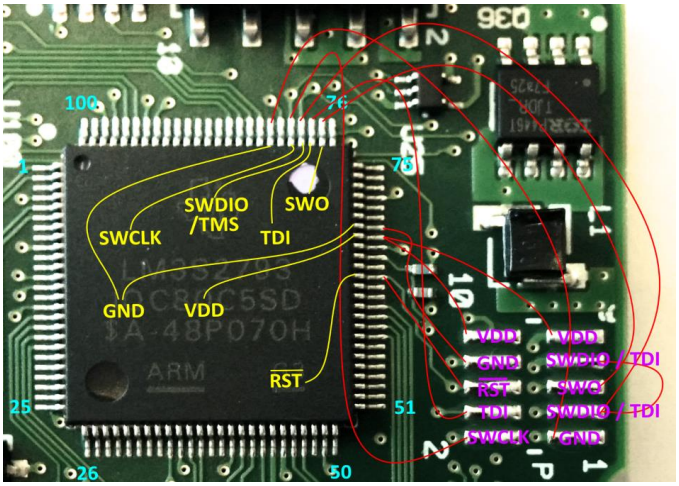


Fig. 4: LM3S2793 JTAG Pins and Their Connections to the 10-pin ARM-JTAG Connector.

The 1769-L18ER-BB1B CompactLogix 5370 L1 Rev. B is based on a Texas Instruments Stellaris LM3S2793 Microcontroller,[12] which uses the ARM Cortex-M3 instruction set architecture (ISA).

Two sets of pins from the processor's pin configuration were of relevance to our work: (1) the processors pins associated with the JTAG interface, and (2) the input/output port pins of the processor.

Joint Test Action Group (JTAG) is commonly used to refer to IEEE Standard 1149.1 [1]. JTAG can be used as a hardware debugging interface in the processor.[13] We used JTAG to develop our HARVEY prototype, as the JTAG connection allowed us to read out the CPU's memory, including flash memory, read-only memory (ROM) and static random access memory (SRAM). Although HARVEY is not limited to JTAG as method to infect a PLC exploring further infection paths is out of scope in this work.

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.

A. Preparation

The ultimate goal of our work was to modify the firmware of the PLC to manipulate input and output values of the PLC without changing the PLC's control logic. To accomplish that

[12]The processor's data sheet can be found online and reveals important details about the processor [49].

[13]Although the JTAG header was physically compatible to typical ARM Cortex-M3 10-pin JTAG connection, the pin configuration was different.

TABLE I: TI LM3S2793 Memory Map

Start	End	Description
0x00000000	0x0001FFFF	On-chip Flash
0x00020000	0x00FFFFFF	Reserved
0x01000000	0x1FFFFFFF	ROM
0x20000000	0x2000FFFF	On-chip SRAM
0x20010000	0x21FFFFFF	Reserved
0x22000000	0x221FFFFF	Bit-band alias of SRAM
...	...	
0x4005C000	0x4005CFFF	GPIO Port E (AHB)
0x4005D000	0x4005DFFF	GPIO Port F (AHB)
0x4005E000	0x4005EFFF	GPIO Port H (AHB)
0x4005F000	0x4005FFFF	GPIO Port G (AHB)
...	...	

we had to find the firmware functions which are responsible for handling the PLC's inputs and outputs.

The first step in analyzing the firmware of our prototyping platform was to obtain images of the firmware. We used two approaches: (1) We downloaded firmware update packages from the vendor's website and extracted the firmware images from them.[14] (2) We extracted the firmware from the PLC's memory using the JTAG interface of the PLC's main processor.

Firmware Images. Allen Bradley PLCs (at least CompactLogix L1 and ControlLogix L61 models) have two firmwares. (1) A base firmware which is shipped with the PLC which provides a minimal function set of the PLC, and (2) a full flashed firmware which provides all functionality for operation. The latter will be referred to as *full firmware* for the rest of the paper, while the first we call *base firmware*.

The base firmware can only be extracted from the PLC's memory as it is not contained in the firmware update package. The base firmware is intended to have one central functionality: to recover the PLC in case an update of the full firmware did not succeed. Hence, the base firmware should not be updated.

The full firmware of the PLC can be updated, with several versions are available for download from the vendor's website [42]. It can be updated remotely over Ethernet or locally via USB or SD card.

Memory Layout and I/O mapping. The TI LM3S2793 maps all flash memory, ROM, SRAM as well as peripheral devices into one contiguous memory address space. Table I shows parts of the memory map of the CPU [49].

B. I/O Interception

To recover the functionality of the firmware we use offline as well as online analysis of the firmware. Offline analysis was done with standard reverse engineering tools such as hex editors and dis-assemblers. The online analysis was possible due to the JTAG connection we could establish with the PLC.

Offline Firmware Analysis. We disassembled the downloaded and extracted firmware binary files from the firmware update file. However, only a small portion of the code was initially disassembled correctly, to improve the results we utilized techniques presented in [7]. This code provided a greater insight of higher level functionality of the firmware, but the memory we extracted directly from the CPU through the JTAG interface proved to be more fruitful. Using the JTAG-extracted memory files, we first aimed to get a general understanding of the functionality of the firmware. Using the ARM Cortex-M3 documentation, we were able to identify the nested vector interrupt controller (NVIC) table. This table contained the address of the reset handler, i.e., where the device starts execution after a reset. Using this address, we were able to follow the boot sequence and disassemble the core functionality in the PLC's flash memory as well as all functions called in SRAM and ROM.

More importantly, we used the ROM data sheets for Stellaris LM3S devices to identify calls to the microcontroller's built-in functions [48]. This helped to identify when important calls to functions that interacted with system peripherals were executed. Identifying these functions gave us a basic understanding as to how the firmware was configuring the controller. Specifically, the functions to control the CPU's general purpose input/output (GPIO) ports, such as the ROM_GPIOPinTypeGPIOInput, ROM_GPIOPinTypeGPIOOutput, ROM_GPIOPinWrite and ROM_GPIOPinRead functions, provided us with functions to look out for as we disassembled the firmware.

One other detail worth mentioning is that we were able to find where the NVIC table was being re-based after the initial boot sequence. Typically the NVIC is re-based to address 0x4000 in ARM Cortex-M3 processors. This was confirmed in our dis-assembly as the vector table offset (located at address 0xE000.ED08) was set to 0x4000 at the end of the boot sequence. By knowing where the NVIC is, we knew where the addresses of the interrupt service routines (ISR) specified in the LM3S2793 documentation were located. In the following section, we will see the significance of this detail.

Online Firmware Analysis. The JTAG connection of the PLC allowed us to analyze and debug firmware during its execution. We could set break points, step through functions, read and write memory and modify registers of the CPU while it is executing. We used this to follow the control flow of the firmware and discover reachable code sections.

Through the analysis we could identify the *main loop* of the firmware. We coupled our online analysis with our offline analysis to step through functions and follow along the disassembled paths. By knowing the boundaries of the main loop, we could investigate where the interaction between the LEDs/HMI and the GPIO Ports occurred. The PLC is equipped with LEDs, one per I/O pin. Similarly, the input values sent to the HMI allow the operator to observe the system state.

When the PLC is power-cycled, an LED sequence is generated where each LED associated to the embedded I/O is sequentially blinked, starting from the Input Ports to the Output Ports. This sequence is relatively slow and involves a sleep period. This allowed us to halt the processor in between two LEDs being blinked. After stepping through the LED sequence, we were able to determine the subroutine associated with sending the LED values over I^2C. Additionally, we identified the address where the I/O values are stored before they are sent to the LEDs. We confirmed this by modifying the associated registers (while the CPU was halted) and stepping through to force arbitrary values different from the typical LED sequence. Although this confirmed that we could control what values were being sent to the LEDs, we still needed to take control of the interchange between the LEDs/HMI and the GPIO Ports.

[14] Vendors like Siemens encrypt their firmware images in update packages. However, the key to decrypt them have been published [8]. The firmware images associated with our PLC are not encrypted.

We found that the main loop has one reference to this LED update function. We noticed that before this update function, a few timer interrupts were being disabled. Using the information from the re-based NVIC table as well as the data sheet, we were able to find the associated ISRs located in SRAM. In particular we found that the Timer 0A ISR was responsible for the interchange between the GPIO Pins and the LEDs/HMI.

As described in Section II-A PLCs operate on the basis of so-called scan cycles, i.e., in fixed intervals inputs are read, the control logic is executed, and the results are written back. We are careful to identify this main loop as being directly correlated to the scan cycle. The Timer 0A ISR seems to be independent of the scan cycle as it is set to run every 0.25 µs. It is only interrupted when updating the LED values. Because this process has not been fully reverse-engineered, we cannot make much stronger inferences than those already mentioned.

C. I/O Interception Code Modifications

As described in the previous section, we identified the exact two subroutines where the values from GPIO Ports E and F are being forwarded to the PLC memory associated with the input values, i.e., the input values sent to the control logic program, the LEDs and the HMI, as well as where the output values from the PLC memory are forwarded to the associated output memory for the LEDs, HMI, and GPIO Ports G and H.

For the output update routine, HARVEY uses the physical models to send legitimate-looking data to the LEDs/HMI, and sends malicious values to GPIO Ports G and H. For the input update routine, HARVEY uses the legitimate input data from GPIO Ports E and F to update the malicious model and possibly coordinate with the output modifications. HARVEY again uses the physical models to report legitimate-looking input values to the LEDs/HMI. The input values, which the control logic might report, e.g., to the system operator, can be crafted such that they correspond to the observation HARVEY makes on the control logic's output, as described in Section IV.

I/O Interception Details. In order to implement our attacks, we modified two subroutines within the Timer 0A ISR that are responsible for the interchange of values to and from the GPIO Ports. Figure 5 shows the aforementioned first attack scenario where we reported false output values to the LEDs and HMI.

The figure shows the original subroutine that was responsible for forwarding an updated output value from memory to the GPIO Ports G and H. The subroutine first updates the address corresponding to the LED and HMI output, `=LED_Output`, and then calculates the correct values to send to the GPIO Ports. For our attack, we first modified an arbitrary location of usable (or re-usable) memory in SRAM and injected our malicious assembly code. Once the malicious code was written, we modified the subroutine to branch to our malicious code. The malicious code would then branch back to the subsequent instructions once the appropriate values were modified. In our attack scenario, a safe system state with Output Port 0 high would have a "0" at the least significant bit, representing the 0 port, and the rest of the bits would be set to 1, i.e., a value of `0xFFFFFFFE`. In our attack, we want to set Output Port 0 to low and Output Port 1 to high, i.e., write a value of `0xFFFFFFFD` to the associated memory address. We chose to branch to an arbitrary memory location to prove that we can make use of the available memory to implement more complex attacks. Figure 6 shows the second attack implemented in a similar fashion.

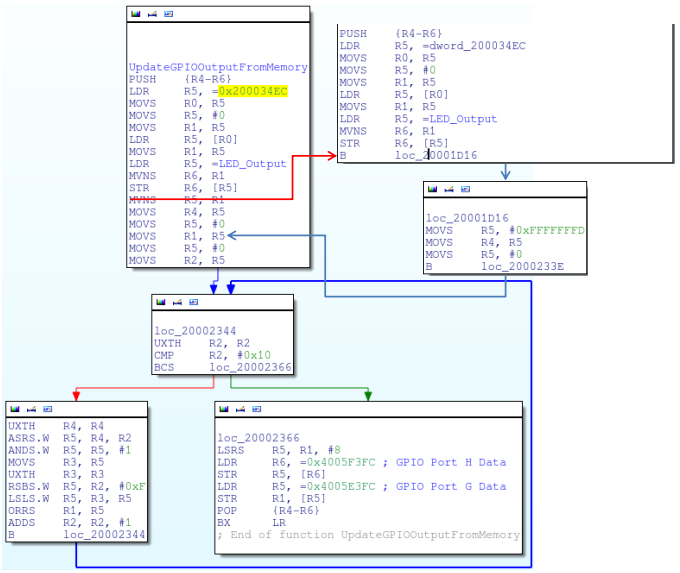


Fig. 5: Original GPIO-output update ISR assembly code compared to modified subroutine with branch to malicious code.

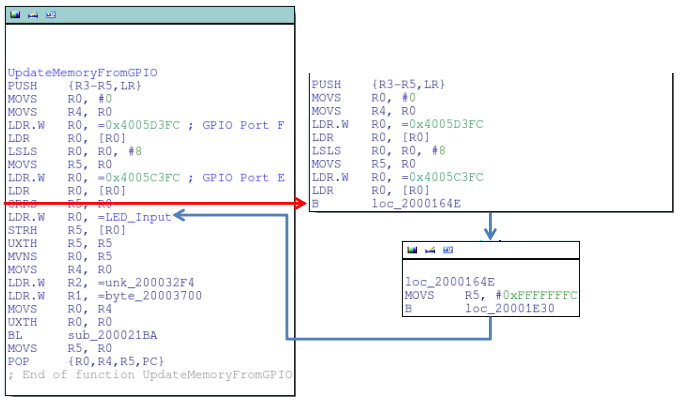


Fig. 6: Original GPIO-input update ISR assembly code compared to modified subroutine with branch to malicious code.

In this case, the goal was to fake the input values being sent to the LED's/HMI as well as the actual ladder logic program running on the PLC. With no inputs, the expected value would be `0xFFFFFFFF`. In our attack, we disregard the values read from the GPIO Ports E and F and simply wrote a value of `0xFFFFFFFC` to the input LEDs, setting Input Port 0 and Input Port 1 to high.

D. Firmware Update

We believe that the built-in remote firmware update functionality of PLCs is the most likely method for an attacker to compromise a device. This is based on observations in related work that firmware updates are not protected against malicious modifications [7], [44]. For our PLC model, the situation is different. Firmware updates are protected by cryptographic means. Firmware updates are delivered together with certificates in the X.509 standard [16]. The certificate contains a SHA-1 [37] hash of the firmware binary file and is signed with 1024 bit RSA [41]. Although the certificate is self-signed, the PLC will abort the update process when provided with a self-signed certificate using a different key than the original one. This makes it practically impossible for an attacker to change the firmware and install it on the PLC. To succeed,

the attacker has two options, (1) he could attempt to find a pre-image hash collision for the SHA-1 hash of the benign firmware binary, or (2) he could factorize the public key used to sign the certificate.

However, an attacker can always compromise a device through the JTAG interface, like we did.

VI. Evaluations

We evaluated several aspects of HARVEY. On one hand, we evaluated the effects of HARVEY on the individual programmable logic controllers (PLC), its influence on execution times and its memory consumption. On the other hand, we evaluated HARVEY in a real-world power system to empirically prove that HARVEY can (1) maximize the effects on the physical system, and (2) hide its malicious effects from the operator.

A. PLC Evaluation

Our PLC model is equipped with an ARM Cortex-M3 processor with 64 KB RAM [49]. It has 512 KB memory for user programs (control logic), as well as 16 DC digital inputs and 16 DC digital outputs.

Experimental Setup. To evaluate the effects of HARVEY on a PLC we set it up with a typical control logic. We installed a control logic program for a PID (proportional–integral–derivative) controller which is shipped by the vendor of our PLC as a standard control logic instruction used in many environments.

In order to model fake input and output values, we used a custom implementation of a PID controller. Figure 9 shows an extract of its assembler code. The code represents a sample PID update function that takes in the current system error and the difference in time since the last iteration and updates the control output based on the summation of the scaling terms. These scaling terms are determined by the type of PID controller. In this case, we defined proportional, derivative and integration error terms to be summed for the control output. A windup guard is used to set a maximum value for the integration term. We compiled this code using a pre-built GNU toolchain for ARM Cortex-M3 processors as well as the StellarisWare libraries for our processor.

To validate the modifications of HARVEY we had to compare the physical outputs of the PLC with the information provided to the operator. To measure the physical output of the PLC we wired it to a voltmeter. To determine the operator's view of the system state, we used the built-in LEDs of the PLC as well as the online monitoring provided by the vendor's control logic development suite, which we will refer to as our HMI. The PLC has a dedicated LED per input and output pin which lights up according the logical state of the pin, i.e., when there is current on the pin the LED will turn on, otherwise, the LED will be off. On the HMI side, online updates of the downloaded control logic program are displayed in real-time.

We used HARVEY to break the relationship between the displayed system state and the actual inputs and outputs on the physical pins of the PLC.

Attack. We were able to change the LED and HMI states of the PLC arbitrarily and independently of the state of the input and output pins. Similarly, we were able to set the output values of pins regardless of the commands sent by the control logic.

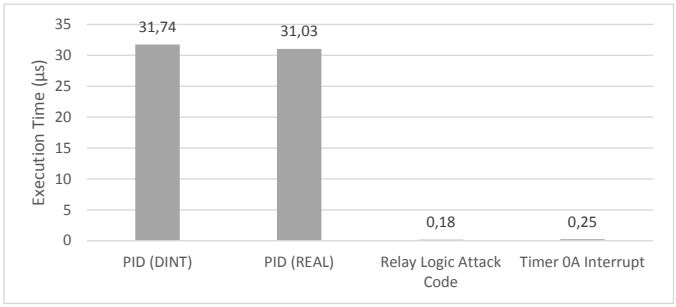


Fig. 7: Feasibility Analysis: Performance Overhead

Execution Time. To evaluate the performance of HARVEY, we measured its execution time and compared it to the execution time of the PID control logic. The measurements are provided in Figure 7, depending on the mode the PID control logic takes between 31.03 μs and 31.74 μs. The timer A0 interrupt handler, which controls the input/output interchange between the GPIO ports and the LEDs/HMI, takes only 0.25μs, which is two orders of magnitude faster. Our attack code, which implements a simple relay logic, takes only 0.18μs.

As described in Section II-A PLCs work in scan cycles. This means inputs are read and outputs are written at a fixed rate, while the control logic gets executed in between. The control logic execution time may not exceed the scan cycle length, otherwise it gets interrupted. Usually, the execution time of control logic is well below the length of a scan cycle. This means that HARVEY can utilize the time difference between control logic execution length and scan cycle length.

If the unused time of a scan cycle is not sufficient for HARVEY, there are several potential solutions that can be implemented to influence the length of a scan cycle. For instance, HARVEY's periodic execution depends on a timer interrupt configuration. Because HARVEY is executing within the firmware, the timer interrupt configuration can be modified to better suit the attacks needs. Similarly, the reporting mechanisms for the control logic/HMI also depend on the timer interrupt configurations that are implemented within the firmware. Therefore, there are several permutations of an attack vector that would allow HARVEY to execute in a timely fashion with a reasonable amount of independence from the scan cycle duration.

Memory Consumption. Our PLC has a finite amount of memory which must be shared between the benign firmware and HARVEY. Although the firmware initially occupies most parts of the memory, large parts are never used. These sections were determined during our online analysis. Furthermore, we examined subroutines that were no longer called after the initial boot sequence. Once the PLC reached the aforementioned main loop, we were able to identify the subset of subroutines that were called by the PLC during the boot sequence but were no longer referenced within the main loop. We refer to this memory as reusable memory. We verified that these functions were no longer used by setting breakpoints at the function addresses as well as any referenced locations within the subroutine. We consider unused memory as memory parts which contain regular patterns that indicate that the memory is not used, for instance, memory sections filled with all 0x00000000 or all 0xFFFFFFFF. Additionally, we found large chunks of memory that contain what seems to be garbage code that is never referenced throughout the firmware execution.

For the practical feasibility of HARVEY, Figure 8 lists

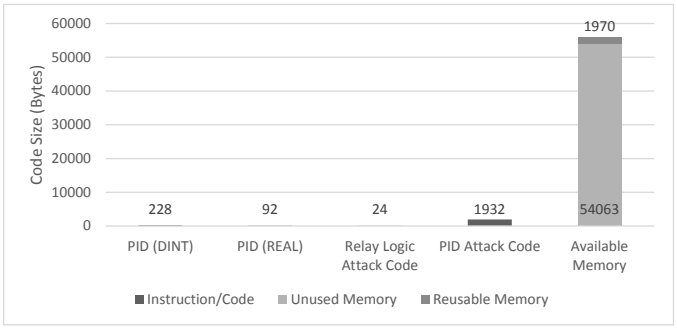


Fig. 8: Available Memory vs. Malware Size

the memory consumption of the PID control logic as well as reusable and unused memory in the firmware. It also shows the memory consumption of the custom PID update function we used as the adversary's system model in our setup.

HARVEY can utilize the unused memory as well as the reusable memory parts, which is significant portion of the PLC's memory (56.033 Bytes out of 65.536 Bytes).

B. Real-World Power System Case Study

We evaluated HARVEY on a real-world power system test-bed, where distributed PLCs with installed PID modules along with more complicated control algorithms (discussed below) maintain safe power system operation.

The electricity grid is modeled using the mathematical power flow equations (physical Kirchhoff laws):

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

which mandate how the sensor measurements (e.g., real/reactive power values on i-th power node (bus) P_i/Q_i, power bus voltages V_i, inter-bus phase angles θ_{ij}, and the admittance (inverse resistance) parameters (G_{ij}, B_{ij}) on the transmission line between the buses i and j correlate due to well-known physics Kirchhoff laws. P_i^g represents the amount of power that is injected to the i-th power bus by a generator, and P_i^l is the amount that is consumed by the end-users at that bus.

Optimal power flow (OPF) is the most widely used control algorithm that is used in practice nowadays to calculate the optimal control commands continuously. In power systems, OPF finds an optimal power generation set-point that minimizes total cost c while meeting operational safety constraints [2]. The control commands typically include power output (set-points) of generators [19]. The OPF's equality constraints are the power balance equations at each bus in the system. Its inequality constraints are the network operating safety limits such as line flow capacities and generator power output limits:

$$\begin{aligned} \min_{u} \quad & c(x,u) \\ \text{s.t.} \quad & P_i^g - P_i^l = \sum_{k} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \\ & Q_i^g - Q_i^l = \sum_{k \in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \\ & P_l^g \le P_l^{gmax} \\ & \forall i,j \in N,\ \forall l \in G,\ \forall k \in C \end{aligned} \quad (4)$$

where u denotes the controls commands to be calculated; x represents dependent variables; V and θ denote the bus voltage magnitudes and angles, respectively.

```
pid_update
PUSH     {R4-R6}
(collapsed code)
STRD.W   R3, R4, [R7,#0x30]
(collapsed code)                ;integration with windup guarding
BEQ      loc_81D0

LDR      R3,=int_error     ;int_error >= windup_guard
(collapsed code)
B        loc_81F2

LDR      R3,=int_error  ;int_error-=windup_guard
(collapsed code)
B        loc_81F2

LDR      R3,=windup_guard    ;int_error>windup_guard
                             ;int_error=windup_guard
(collapsed code)

LDR      R3,=prev_error          ;differentiation
(collapsed code)
STRD.W   R3,R4,[R7#0x28]
LDR      R3,=proportional_gain ;scaling
(collapsed code)
BL       _muldf3
(collapsed code)
LDR      R3,=integral_gain
LDRD.W   R0,R1,[R3]
LDR      R3,=int_error
(collapsed code)
BL       _muldf3
(collapsed code)
LDR      R3,=derivative_gain
(collapsed code)
BL       _muldf3
(collapsed code)
LDR      R2,=control             ;summation of terms(control=p+i+d)
(collapsed code)
LDR      R2,=prev_error          ;prev_error=curr_error
(collapsed code)
POP      {R4-R7,PC}
```

Fig. 9: Injected Malicious PID Controller

The legitimate OPF's objective is to minimize the cost while ensuring the system operates safely. HARVEY implements a modified version of the algorithm, malicious optimal power flow (mOPF), to maximize the cost without the need for compliance with safety constraints:

$$\begin{aligned} \max_{u} \quad & c(x,u) \\ \text{s.t.} \quad & P_i^g - P_i^l = \sum_{k} |V_i||V_k|(G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik}) \\ & Q_i^g - Q_i^l = \sum_{k \in C} |V_i||V_k|(G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik}) \\ & \forall i,j \in N,\ \forall l \in G,\ \forall k \in C \end{aligned} \quad (5)$$

where the calculated control commands would *maximize* the amount of possible damage to the power system. The calculated commands are used as set-points to be maintained by the inner-loop PID controllers. Please note that the specific objective function for different malicious goals can be simply used instead in the formulation above.

Our power system test-bed implements IEEE nine-node (bus) benchmark topology [15] including three synchronous power generators and controlled by nine distributed PLCs. Figure 10 shows the test-bed (top right), its cyber network topology (top middle), power system topology (top left), LabView control diagram (bottom right), supervisory control and data acquisition device interconnections (bottom middle), and monitoring and control operations (bottom left). The model has three substations and corresponding loads (which consume power). The power nodes are connected through power transmission lines. To follow real world implementations, we equipped each substation with protection functions such as over-current, voltage and frequency, i.e., the substation will open a transmission line if they carry current beyond its physi-

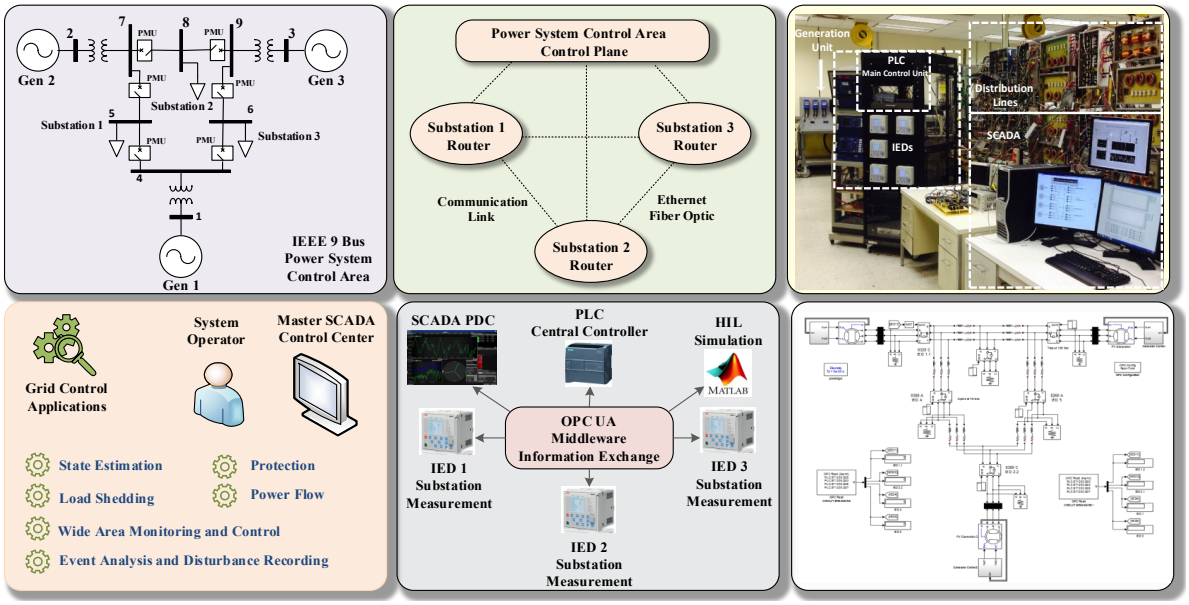


Fig. 10: The Evaluation Smart Grid Test-Bed

cal capacity or cause over-voltage or over-frequency situations. To monitor the power system, the voltage and current sensors (phasor measurement units PMUs) send their measurements to PLC controllers that act as monitoring/control agents and are responsible for all operational functions in the system.

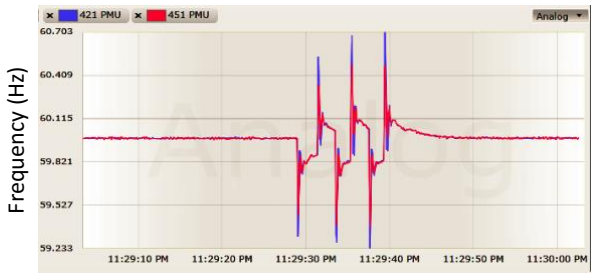
On the cyber end, the testbed includes a human-machine interface (HMI) server to provide the system status to the operators through its connections to the PLC. The data exchange between different field devices is established by open platform communications (OPC) Client I/O servers [38]. Kepware OPC Server provides embedded drivers to connect to the PLC. The testbed employs a ReLab device driver to connect to and obtain measurements (IEEE C37.118) from PMU sensors. Briefly, using Kepware's IEC 61850 MMS clients, the KEPServerEX OPC Server drivers create an interface for any of the OPC clients running in the network.

We evaluated HARVEY for two attack scenarios.

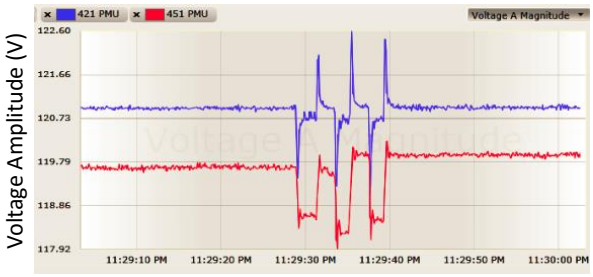
Steady-state system malicious attack: Repeated heavy load circuit breaker open/close triggering without loss of power system stability. In this scenario, the malicious PLC firmware randomly (blindly) selects a circuit breaker to attack and triggers the opening/closing of the breaker several times, i.e., a transmission line opened and closed repeatedly. The power system was able to withstand this attack scenario without losing the stability since the target circuit breaker load was in the limits of power system generation reserve capacity. The SEL-451 PMU is located on generator 1 bus, and the 421-PMU is located at generator 2 bus. Figure 11 shows the power system status during the attack that starts at 11.29.30 PM. The circuit breaker was opened and closed three times sequentially within ten seconds. The heavy loading in the system deteriorated the system frequency (Figure 11a) and voltage (Figure 11b). The AC phase angle difference between generator 1 and generator 2 exceeded permissible limits (Figure 11c). The power flow magnitudes (Figure 11d) also violated safety thresholds temporarily. As shown, although the instant voltage and frequency of the system exceeded permissible limits, the power system was able to withstand this type of attack. During the attack, HARVEY was able to run the power system model on the PLC in parallel and generate fake legitimate-looking sensor measurements to be viewed by the operators. Figure 12 shows the results (before the noise was added for the presentation clarity). As the attack on the physical plant would result in noticeable side effects such as equipment operational noise, HARVEY's outputs show a minor system perturbation within safety limits that is normally observed on daily dynamic power system operations. From the operators' viewpoint, the system acts safely and no corrective action is needed.

Adversary–optimal control attack: optimal malicious attack using real-world control algorithms. In this attack scenario, HARVEY implements a real-world power system controller algorithm, called optimal power flow (OPF) [10], that is widely used in power system control centers internationally. OPF is implemented as a linear programming function: it typically finds the optimal power system control strategy that minimizes the overall cost while ensuring the system safety. The system safety is usually defined by a set of lower and upper bound thresholds for various system parameters such as power transmission line current capacities, and minimum/-maximum allowed system frequency 59.5-61Hz (60Hz is the nominal power grid frequency in USA). The control strategy is essentially a set of control commands that the PLC sends to the actuators, e.g., generation set-points to the generators that mandate how much power each generator should generate. HARVEY implements the same control algorithm on the PLC after making three modifications to the algorithm (we call it malicious OPF - mOPF): *i)* it removes the condition that ensures the system is within safety margins; *ii)* it replaces the cost minimization function with maximization so that the adversarial impact becomes maximum; and *iii)* HARVEY adds predefined stealthy conditions to ensure its malicious control actions do not get noticed/detected by the local operators on site due to the noise the actions generate. Example conditions are "no power generator disconnect from the rest of the power grid" in large power plants, since such disconnects cause a noticeable sound noise to the potential local operators. In practice, there are typically few or no operators present on remote power system substations. This gives HARVEY more freedom in terms of what malicious actions it can carry out.

In this attack scenario, HARVEY's objective was to im-

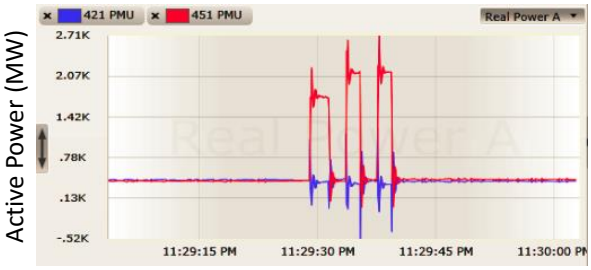


(a) Frequency



(b) Voltage Magnitude

(c) AC Voltage Phase Angle



(d) Power

Fig. 11: Actual Power System Measurements

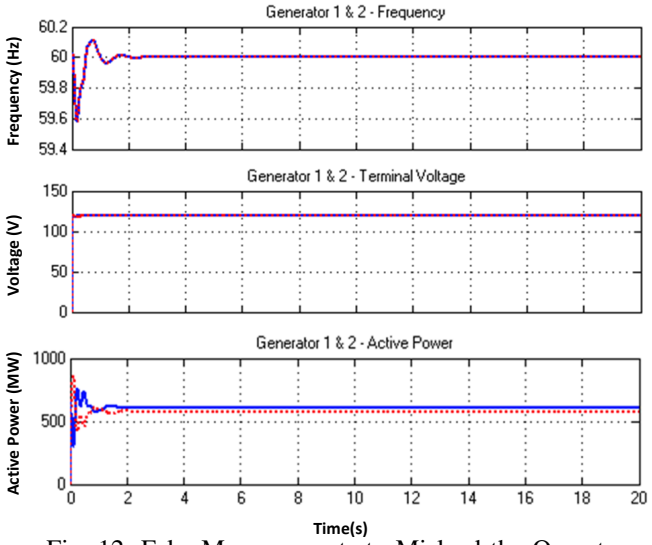


Fig. 12: Fake Measurements to Mislead the Operator

plement mOPF on the PLC to calculate adversary-optimal control strategy for the power plant. Using the power system's safety constraints, HARVEY intercepts the legitimate control action outputs and instead sends out its optimally-calculated malicious control commands to the power actuators at specific time points. HARVEY sets the nominal frequency reference to 62 Hz, and its malicious controller calculates and sends out control commands accordingly.

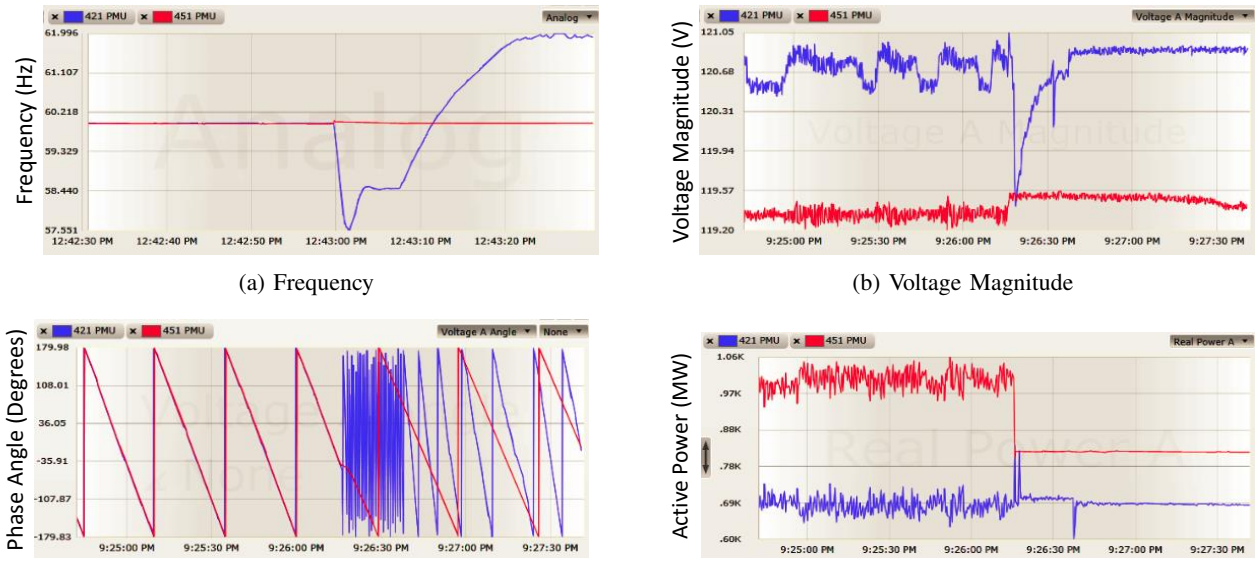
Figure 13 shows the actual power system measurements. HARVEY makes the power system frequency exceed its safety margins through its malicious commands (Figure 13a). The system's voltage magnitude (Figure 13b), AC voltage phase angle (Figure 13c), and electric power values (Figure 13d) experience serious instability as well. However, in order to mislead the operator, HARVEY implements a legitimate OPF algorithm in the background to simulate the power system and calculate individual system parameters assuming that the legitimate OPF control commands were carried out on the power system. The fabricated fake sensor measurements (Figure 14) are sent back to the operators' HMI screens. Consequently, from the operators' viewpoint, the underlying power system follows their expectation, while in reality, the system goes through serious instability situations facing potential large-scale failures. An experienced operator might get suspicious of small disturbances visible in the graph. However, such disturbances can also occur in normal operation. Similarly, an automated tool monitoring the ICS must be tolerant to small disturbances to reduce the number of false positive alarms.

VII. RELATED WORK

We discuss related work on ICS security in terms of proposed defense mechanisms and possible attacks.

Defense mechanisms have been proposed on network and host/device levels. SOCCA [53] generates network-level attack graphs based on Markov decision processes considering the impact of the adversarial actions on the physical power system. CPMA [17] uses the ICS attack graphs to perform security-oriented risk analysis, so-called contingency analysis, regarding potential threats against the power grid. Both solutions consider PLCs as the interface between cyber and physical assets of the infrastructures, and identify them as potential targets by the adversaries. SCPSE [54] and CPAC [21] present a stateful detection mechanism to detect attacks against control systems based on the received sensor measurements by the operators. HARVEY evades such detectors completely through replacing them with legitimate-looking fake measurements.

Unlike traditional IT cyber networks, ICS networks often follow well-defined behavioral patterns. Therefore, online ICS intrusion detection solutions monitor the runtime operation for anomalous behaviors as opposed to the signature-based paradigm [13], [31], [51]. Formby et al. [26] employ the behavioral profiles for device fingerprinting and access control. Security solutions in ICS has to be non-intrusive against safety-critical operations with real-time constraints that run mostly on resource-limited embedded devices/controllers [52]. Such anomaly-based solution cannot identify HARVEY, since it uses



(a) Frequency

(b) Voltage Magnitude

(c) AC Voltage Phase Angle

(d) Power

Fig. 13: Actual Power System Measurements

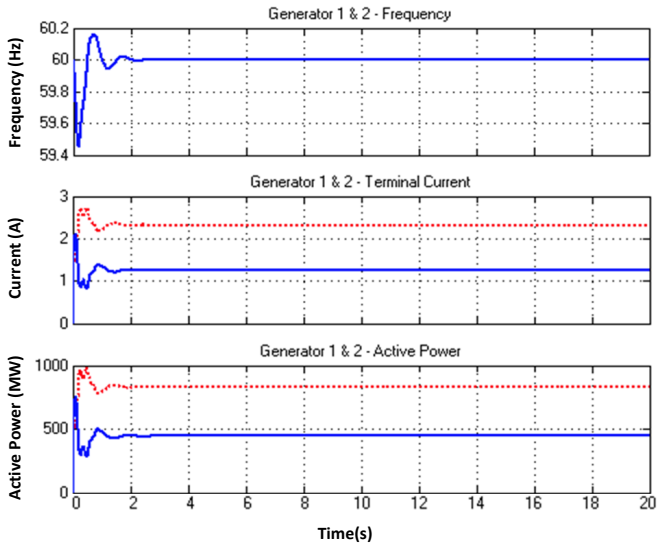


Fig. 14: Fake Measurements to Mislead the Operator

the same power model to fake sensor measurements and make them look normal.

TSV [34] and [55] provide a bump-in-the-wire solution between the HMI and PLC device to intercept and analyze the control logic downloads on the PLC by the HMI server. TSV implements formal methods to verify the safety of the code regarding the physical plant safety requirements, and drops the control logic if a counterexample is found. TSV is unable to detect HARVEY as it targets control logic updates and does not support firmware updates or network-level firmware exploits.

The Weaselboard [36] is a PLC backplane analysis that captures backplane communications between modules with the intention of preventing zero-day exploits on PLCs. The inter-module traffic is forwarded to an external analysis system that detects changes to process control settings, sensor values, module configuration information, firmware updates, and process control program (logic) updates. Because the board monitors backplane communication, HARVEY would remain undetected as HARVEY would feed fake-legitimate-looking measurements to the PLCs backplane, i.e., HARVEY resides between the the I/O modules and the backplane. However, a Weaselboard implementation can prevent arbitrary firmware updates over the network. Therefore, an attacker would need to convince the operator to run a firmware update with a compromised firmware binary file or use a JTAG implantation to modify the firmware.

There have been several security solutions focused on the detection of firmware modifications that could prevent an attack like HARVEY, e.g., control-flow monitoring solutions [40]. However, our contribution is not in the evasion of firmware modification attacks, but rather the evasion of intrusion detection solutions that sit outside of the PLC. Attacks such as Ghost in the PLC [3] have shown that these firmware modification monitoring solutions can be circumvented.

Attacks on ICS and PLCs have grown significantly since their seminal example emergence, the Stuxnet worm [24] that targeted the Iranian Natanz nuclear enrichment plant. Stuxnet is categorized as a malicious control command injection attack. Through four Windows zero-days, Stuxnet compromised HMI and sent malicious control logic to the PLC. Stuxnet attacks would be identified using TSV [34] as violating the plant's safety requirements and blocked from the PLC execution. Similar to Stuxnet, PLC-Blaster worm [12] injects a malicious control logic on the vulnerable PLCs (Siemens S7-1200v3) after a network scan. For stealth, PLC-Blaster manipulates the meta-data of its control logic which will cause the HMI software (Siemens Step7) to crash when the operator tries to retrieve information from an infected PLC. This would raise the operator's suspicion leading to potential detection. Ghost in the PLC [3] provided a PLC rootkit that exploited I/O pin control operations to provide a cyber framework for an undetectable rootkit. However, the rootkit does not provide a means of stealthiness with respect to the monitoring entity overseeing the physical evolution of the system.

On the sensing side, false data injection attacks [33], [35], [50] have been shown to be capable of misleading the operators. The attackers in control of a subset of sensors would send corrupted measurements to control centers to mislead the state estimation and controller servers. False data injection

attacks do not consider the operator's control commands to the plant, and hence their fabricated system state may not satisfy the operators' expectation, and hence can be detected simply.

On the network side, Beresford [8] discovered vulnerabilities in Siemens S7 series communication protocol for replay attacks leading to a remote shell access. The attack was specific to that PLC model number of Siemens only. The similar attacks on firmware vulnerabilities (e.g., insecure checksum validation during the update process [7], DDoS attacks against common industry protocol CIP package handler functions [44]) are orthogonal to HARVEY since HARVEY's core contribution is to inject and run a power system model as a rootkit (after the firmware is compromised) to damage the physical plant while evading the operator detection. The above-mentioned PLC attacks did not leverage the domain-specific features to *i)* maximize their destructive physical impact using adversary-optimal control algorithms, and *ii)* simulate the physical plant model to fabricate legitimate-looking measurements to the operators.

Klick et al. [32] show that internet-facing controllers can be compromised, act as a SNMP scanner or SOCKS proxy, and be misused by an adversary to attack devices that are not directly connected to the internet. This technique can be used to extend HARVEY's compromised devices. Additionally, there have been theoretical attack frameworks proposed against water SCADA [4]. The attack drives the underlying physical plant towards unsafe states by solving the partial differential equations that model the water plant dynamics. The authors do not discuss details of how such attacks can be implemented in practical real-world settings and the involved challenges. Similarly, system-theoretic models [39] [5] have been proposed to identify stealthy cyber-physical attacks against the power grid either through compromised measurements or dynamic load-altering attacks. However, the models assume that attacker has already compromised the required assets. These models could be utilized by HARVEY to model the spoofed measurements being sent to the HMI.

VIII. DISCUSSIONS AND MITIGATIONS

We discuss the generality of HARVEY, and describe potential mitigation mechanisms that could be deployed to protect critical infrastructures against similar attacks.

A. Unique Challenges in ICS

In this section we elaborate on some of the unique challenges in the ICS space by comparing our PLC rootkit against rootkits known from workstations and servers. We also elaborate on the relation of industrial control systems and the Internet of Things (IoT).

Rootkits. The concept of rootkits, or more generally, compromising the privileged software of a computing system with the goal of hiding has been known for decades [29]. However, most known rootkits target commodity operating systems like Windows or Linux which have vast amounts of resources that can be (mis-)used by the rootkit to hide itself and perform its malicious actions. PLCs, on the contrary, have a significantly different software design (cf. Section II-A) which requires new techniques for rootkits in the domain of ICS.

HARVEY, in particular, aims at manipulating the input and output of a PLC, i.e., interaction with the physical world. "Classical" rootkits operate only within the deterministic cyber world, which makes hiding and other actions simpler. For instance, a typical hiding method of Windows rootkits is to remove themselves from a list structure maintained by the OS [46]. HARVEY, however, has to compensate for the changes to its physical world which will eventually feed back as input readings.

Internet of Things (IoT). In recent years, the term IoT is used over-extensively even though (or because) there is no widely accepted and precise definition of the term. While there are definitions that include ICS (e.g., every computing device with a network connection) we would argue that ICS are not part of the IoT. Many IoT devices are based on commodity hardware and software (e.g., ARM Cortex-A processors and Linux based OS) and thus not much different from systems like smartphones. Hence, attacks on IoT devices, such as the compromise of the Google nest thermostat [30], are not applicable to ICS. Another significant difference between consumer IoT devices and ICS is the real-time operation of PLCs, reflected by the control loop design of the PLC's firmware. IoT devices rarely have strict real-time requirements. Further differences include software deployment schemes (IoT device's software is controlled by the device manufacturer, PLC control logic is installed by the programmer/operator), or device lifetime (ICS might be operated for decades). Lastly, the operation of IoT devices (and their effects on the environment) are often unsupervised, hence, an IoT rootkit does not need to provide a manipulated system state view to an operator.

B. Generality

HARVEY involved reverse engineering of a real-world commercial PLC device and binary software modules. Although we worked on a specific model, the techniques we used, such as JTAG debugging and binary analysis, can be generalized to PLC and controllers from other vendors, because they generally follow similar technical approaches such as scan-cycle-based execution paradigm followed by periodic I/O interrupts and memory updates. Additionally, the proposed two-way data manipulation attack can be implemented on other (not necessarily power grid) control system settings, where controller devices are used to monitor and control underlying physical plants.

HARVEY can be protected against using three major mitigation solutions: *i)* remote attestation allows a verifier to check the software integrity of a system. A trusted component provides an authenticated measurement of the memory of the device to be attested. Different approaches specifically for embedded systems have been developed and could be applied to PLCs [11], [18], [45]; *ii)* with secure boot, the integrity of a device's configuration is not verified by an external entity but by the device itself possibly using a trusted platform module [6]. Secure boot ensures that only a known and trustworthy software can be loaded on a device. Secure boot could be used to ensure the integrity of PLC firmware; and *iii)* an external bump-in-the-wire device between the PLC controller and the physical plant could be monitoring the two-way sensor-to-PLC and PLC-to-actuator data streams (unlike TSV [34] that would sit between the HMI and PLC). The solution could possibly check whether the control commands issued by the PLC satisfy the plant's essential safety requirements that must be defined by the operators. Additionally, the solution could implement coarse-grained control consistency checks to validate whether sensor measurements and actuation commands are consistent in terms of how the plant should be controlled.

IX. CONCLUSIONS

We presented HARVEY, a PLC rootkit that implements a physics-aware man-in-the-middle attack against cyber-physical

control systems. HARVEY damages the underlying physical system, while providing the operators with the exact view of the system that they would expect to see following their issued control commands. Our experimental results with a commercial PLC controller on a real-world power system testbed demonstrates the feasibility of HARVEY in practice.

REFERENCES

[1] "IEEE standard test access port and boundary scan architecture," *IEEE Std. 1149.1-2001*, 2001.

[2] (2010) Federal energy regulatory commission (ferc): Optimal power flow and formulation papers. http://www.ferc.gov/industries/electric/indus-act/market-planning/opf-papers.asp.

[3] A. Abbasi and M. Hashemi, "Ghost in the plc: Designing an undetectable programmable logic controller rootkit via pin control attack," 2016.

[4] S. Amin, X. Litrico, S. S. Sastry, and A. M. Bayen, "Stealthy deception attacks on water scada systems," in *Proceedings of the 13th ACM International Conference on Hybrid Systems: Computation and Control*, ser. HSCC, 2010.

[5] S. Amini, H. Mohsenian-Rad, and F. Pasqualetti, "Dynamic load altering attacks in smart grid," in *Innovative Smart Grid Technologies Conference (ISGT), 2015 IEEE Power & Energy Society*. IEEE, 2015.

[6] W. Arbaugh, D. Farber, and J. Smith, "A secure and reliable bootstrap architecture," in *IEEE Symposium on Security and Privacy*, 1997.

[7] Z. Basnight, J. Butts, J. Lopez, and T. Dube, "Firmware modification attacks on programmable logic controllers," *International Journal of Critical Infrastructure Protection*, 2013.

[8] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," in *Black Hat USA 2011*, ser. Black Hat USA '11.

[9] W. Bolton, *Programmable logic controllers*. Newnes, 2015.

[10] S. Bose, S. H. Low, T. Teeraratkul, and B. Hassibi, "Equivalent relaxations of optimal power flow," *Automatic Control, IEEE Transactions on*, 2015.

[11] F. Brasser, B. El Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "TyTAN: Tiny Trust Anchor for Tiny Devices," in *Proceedings of the 52nd Annual Design Automation Conference*, ser. DAC, 2015.

[12] M. Brüggemann and R. Spenneberg, "Plc-blaster der virus im industrienetz," https://events.ccc.de/congress/2015/Fahrplan/events/7229.html, 2015.

[13] 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*, Miami Beach, Florida, jan 2007.

[14] E. Chien, L. OMurchu, and N. Falliere, "W32.Duqu - The precursor to the next Stuxnet," Symantic Security Response, Tech. Rep., 2011.

[15] R. Christie, "Power systems test case archive," *Electrical Engineering dept., University of Washington*, 2000.

[16] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk, "Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile," https://tools.ietf.org/html/rfc5280, 2008.

[17] K. R. Davis, C. M. Davis, S. Zonouz, R. B. Bobba, R. Berthier, L. Garcia, P. W. Sauer *et al.*, "A cyber-physical modeling and assessment framework for power grid infrastructures," 2015.

[18] K. E. Defrawy, A. Francillon, D. Perito, and G. Tsudik, "SMART: Secure and Minimal Architecture for (Establishing Dynamic) Root of Trust," in *NDSS*, 2012.

[19] H. Dommel and W. Tinney, "Optimal power flow solutions," *IEEE Transactions on Power Apparatus and Systems*, 1968.

[20] K. T. Erickson, "Programmable logic controllers." Institute of Electrical and Electronics Engineers, 1996.

[21] S. Etigowni, D. Tian, G. Hernandez, K. Butler, and S. Zonouz, "Cpac: Mitigating attacks against critical infrastructure with cyber-physical access control," in *Annual Computer Security Applications Conference (ACSAC)*, 2016.

[22] European Network and Information Security Agency (ENISA). (2011) Protecting industrial control systems – recommendations for Europe and Member States. https://www.enisa.europa.eu/.

[23] F-Secure Labs, "BLACKENERGY and QUEDAGH: The convergence of crimeware and APT attacks," 2016.

[24] N. Falliere, L. O. Murchu, and E. Chien, "W32.Stuxnet Dossier," Symantic Security Response, Tech. Rep., 2010.

[25] J. FitzPatrick and M. King, "Nsa playset: Jtag implants," 2015.

[26] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? device fingerprinting for cyber-physical systems." in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2016.

[27] A. Francillon and C. Castelluccia, "Code injection attacks on harvard-architecture devices," in *Proceedings of the 15th ACM Conference on Computer and Communications Security*, ser. CCS '08, 2008.

[28] A. Gómez-Expósito, A. J. Conejo, and C. Cañizares, *Electric energy systems: analysis and operation*. CRC Press, 2016.

[29] A. Hay, D. Cid, and R. Bray, *OSSEC Host-Based Intrusion Detection Guide*. Syngress, 2008.

[30] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart nest thermostat: A smart spy in your home," in *BlackHat USA*, 2014.

[31] 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*, 2014.

[32] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing plcs - a new back orifice," in *Black Hat USA*, 2015.

[33] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security (TISSEC)*, 2011.

[34] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A trusted safety verifier for process controller code." in *Proceedings of the Network and Distributed System Security (NDSS) Symposium*, 2014.

[35] S. McLaughlin and S. Zonouz, "Controller-aware false data injection against programmable logic controllers," in *IEEE SmartGridComm*, 2014.

[36] J. Mulder, M. Schwartz, M. Berg, J. R. Van Houten, J. Mario, M. A. K. Urrea, A. A. Clements, and J. Jacob, "Weaselboard: Zero-day exploit detection for programmable logic controllers," tech. report SAND2013-8274, Sandia Nat'l Laboratories, Tech. Rep., 2013.

[37] N. I. of Standards and Technology, "FIPS 180-4, Secure Hash Standard, Federal Information Processing Standard (FIPS)," http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf, Tech. Rep., 2012.

[38] OPC Foundation. (2015) Open platform communication foundation. https://opcfoundation.org/.

[39] F. Pasqualetti, F. Dörfler, and F. Bullo, "Cyber-physical attacks in power networks: Models, fundamental limitations and monitor design," in *IEEE CDC and ECC*. IEEE, 2011.

[40] J. Reeves, "Autoscopy jr.: Intrusion detection for embedded control systems dartmouth computer science technical report tr2011-704 a thesis," Ph.D. dissertation, DARTMOUTH COLLEGE Hanover, New Hampshire, 2011.

[41] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, 1983.

[42] Rockwell Automation. (2015) Rockwell automation download center. http://compatibility.rockwellautomation.com/Pages/MultiProductDownload.aspx?famID=4&Keyword=Controller&crumb=112.

[43] J. Rrushi, H. Farhangi, C. Howey, K. Carmichael, and J. Dabell, "A quantitative evaluation of the target selection of havex ics malware plugin."

[44] C. D. Schuett, "Programmable logic controller modification attacks for use in detection analysis," DTIC Document, Tech. Rep., 2014.

[45] R. Strackx, F. Piessens, and B. Preneel, "Efficient isolation of trusted subsystems in embedded systems," in *Security and Privacy in Communication Networks*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering. Springer, 2010, vol. 50.

[46] Symantic, "Windows Rootkit Overview," Symantic Security Response, Tech. Rep., 2005.

[47] TechNavio. (2014) Global Industrial Control Systems (ICS) Security Market 2014-2018. http://www.technavio.com/report/global-industrial-control-systems-ics-security-market%C2%A02014-2018.

[48] Texas Instruments . (2011-2013) Stellaris LM4F120H5QR ROM User's Guide. www.ti.com/lit/ug/spmu245a/spmu245a.

[49] Texas Instruments. (2007-2014) Stellaris LM3S2793 Microcontroller Data Sheet. www.ti.com/lit/gpn/lm3s2793.

[50] L. Xie, Y. Mo, and B. Sinopoli, "False data injection attacks in electricity markets," in *Smart Grid Communications (SmartGridComm), 2010 First IEEE International Conference on*. IEEE, 2010.

[51] D. Yang, A. Usynin, and J. W. Hines, "Anomaly-based intrusion detection for scada systems," 2006.

[52] B. Zhu and S. Sastry, "SCADA-specific intrusion detection/prevention systems: A survey and taxonomy," in *Proceedings of the First Workshop on Secure Control Systems*, 2010.

[53] S. Zonouz, C. M. Davis, K. R. Davis, R. Berthier, R. B. Bobba, and W. H. Sanders, "SOCCA: A security-oriented cyber-physical contingency analysis in power infrastructures," *Smart Grid, IEEE Transactions on*, 2014.

[54] S. Zonouz, K. M. Rogers, R. Berthier, R. B. Bobba, W. H. Sanders, and T. J. Overbye, "Scpse: Security-oriented cyber-physical state estimation for power grid critical infrastructures," *IEEE Transactions on Smart Grid*, 2012.

[55] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated plc code analytics," *Security & Privacy, IEEE*, 2014.