This Hacker Knows Physics: Device Physics Aware Mimicry Attacks in Cyber-Physical Systems

Qinchen Gu, David Formby, Shouling Ji, Brendan Saltaformaggio, Anu Bourgeois, and Raheem Beyah

Abstract—Recent work proposed to improve the security of CPSs by authenticating the CPS devices through the device operation times in the response packets from the devices, due to the strong correlation between the timing fingerprints and the physics of the devices. Although such a technique may be effective in defending against naive attackers, an advanced attacker may monitor the operation of the CPS before launching a device physics aware mimicry attack. In this paper, we show how the spoofed response packets can be crafted by an attacker to deceive the CPS device authentication method based on the device operation times. Specifically, we use the timing and physical measurements embedded in the packets to reconstruct the devices in the physical system, which can be used to spoof response packets corresponding to the actual model and configuration of the devices in the CPS. We demonstrate the performance of our technique in realistic testbeds with real devices. Finally, we propose an upgraded defense mechanism that may be used against such mimicry attacks.

1 Introduction

With the increased proliferation of Cyber-Physical Systems (CPSs), there have also been more frequent attacks on CPSs. While some attacks are wide-spread similar to computer malware that aims for better coverage, the most devastating attacks tend to be targeted. The most well-known such attack on a CPS is Stuxnet, which is a malicious worm targeting the Supervisory Control And Data Acquisition (SCADA) systems, specifically infecting and reprogramming Programmable Logic Controllers (PLCs). It was responsible for causing tremendous damage to Iran's nuclear program, by driving the fast-spinning centrifuges in Iran's nuclear facilities to a failed state. Although there have been no official conclusion as to who is responsible for this attack, the size and sophistication of the worm have led researchers to believe that at least one nation-state was involved [19]. Noticeably, a dossier published by Symantec suggested that the attackers were most likely to have conducted a significant amount of **reconnaissance** [13]. As each PLC is configured in a unique manner, the attackers would first need the schematics of the industrial control system (ICS) [7]. An attacker would then need to know details of the individual device's physical behavior in order to maximize the damage in the following targeted attack. For example, Stuxnet checks a Profibus identification number corresponding to two different models of variable frequency drive (VFD) [1], which are used to control the motors. Two different attack sequences are chosen depending on the type of VFD found. A similar incident where attackers retrieved information about the target system before mounting the actual attack occurred in December 2015. A piece of malware specially crafted to attack a Ukrainian electric utility caused a blackout in a portion of its capital equivalent to a fifth of its total power capacity [2]. It even sabotaged power distribution equipment, complicating the restoration of power [4]. A critical step in this attack was to seize control of the SCADA system and to remotely shut down substations. The attack was found to be a premeditated multi-level invasion. The attackers were thought to have hidden in the IT network of a utility company for six months, **collecting data to figure out the inner-workings of the system** before performing the actual attacks.

The key difference between the attacks on CPSs and those on traditional IT systems is the *physical* nature of CPSs. While the goal of attackers in traditional IT systems may be stealing users' private information, those who target CPSs can cause serious damage to the real world. An attack on critical infrastructures may directly threaten people's daily lives, leaving millions of dollars and even human lives at risk. However, the physical nature of CPSs can be a double-edged sword. For example, many studies have proposed to leverage the physical domain as a channel to secure CPSs [8], [10], [15], [20], [24], [25]. Specifically in [15], Formby et al. modeled the physics of ICS devices and demonstrated that the operation times of these devices can be used to generate fingerprints, which are capable of verifying the integrity of the devices' response packets subject to a false data/response injection attack. Their technique uses a high

- *Qinchen Gu is with Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.*
- *David Formby is with Fortiphyd Logic, Atlanta, GA, USA.*
- *Shouling Ji is with Zhejiang University, Hangzhou, Zhejiang, China.*
- *Brendan Saltaformaggio is with Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.*
- *Anu Bourgeois is with Department of Computer Science, Georgia State University, Atlanta, GA, USA.*
- *Raheem Beyah is with Department of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA.*

1. Part number KFC750V3 manufactured by Fararo Paya in Tehran, Iran. Vacon NX VFD manufactured by Vacon in Finland.

fidelity intrusion detection system (IDS) to detect the minute differences between the network packets sent by authenticated devices and those spoofed by the attacker. Although such a technique may detect naive attackers who are unaware of the physics-related response packets or lack proper equipment to craft accurately timed packets, we show in this paper that an advanced attacker may bypass the defense by launching a **mimicry attack** via a compromised PLC. Existing work have demonstrated the vulnerability of the PLC. For example, Garcia et al. showed that a PLC rootkit can be implemented to launch a stealthy attack in CPS [16]. Such exploits provide a basis for attackers to directly access and manipulate the commands for actuators and sensory data, as well as means to produce packets with accurate timing. **Hence, the defense proposed in [15] may be defeated under such circumstances.**

This work intends to study how an advanced attacker who is aware of the detection system can spoof packets that correspond to the actual device model and configuration (DMC) to avoid being detected by the techniques as described in [15]. More specifically, we illustrate that an advanced attacker who can perform reconnaissance when attacking a targeted system and obtain the DMC in the CPSs can better evade detection, and hence calls for a more in-depth defense. Note that for the rest of this paper, "device" refers to the "actuator" in the CPSs. To give a brief description of our method presented in this paper, we start with modeling different types of devices based on their construction and the physical process in their operation. Mathematical equations are derived to describe the operation of the devices and used to characterize the devices in the network domain. Using real testbeds that we built, we demonstrate the process of inferring the models and configurations of the devices from their response traffic, which in turn are used to forge the responses of the same devices. With the results from the testbeds, we show that the forged responses are much more difficult to be detected using device physics fingerprinting methods.

1.1 Observation

In CPSs, physical devices either take commands from the *cyber* side of the CPS and operate objects in the physical world (i.e., actuators), or provide digitized information of the physical objects to the controller or monitoring system (i.e., sensors). In [15], Formby et al. discussed how the actuators have to obey the laws of physics when executing commands sent by the controllers. During our experiments, we found that these actuators exhibit different temporal features which are correlated with the device physics when carrying out the actions.

For example, two motors given the same command of *set to full speed* may take a different amount of time to accelerate to full speed, depending on the torque generated by the motor and the magnitude of its load. Likewise, a valve given a command of *close* or *open* can take an interval of time defined by its specification set according to the mechanical and electrical properties, which include the physical composition of the valve (e.g., torque characteristics of the motor, power rating, gear ratio, size of the fluid passage, etc.). In some cases, such responses may be recorded and used as replay attack. However, there are two problems that must be addressed for such an attack to succeed. First, the responses for certain types of devices are not constant over time, and may be a function of the run-time condition of the devices. Second, the accuracy required to reproduce such responses may exceed the capability of many embedded devices as shown in [15].

We also noticed that an actuator can generate feedback information during the execution or after the completion of a command, either actively or passively. The feedback signal carries the information about the actuator's physical attributes, which can then be used to infer the model and configuration of the device. Taking the valve again as an example, all valves used in our experiment are capable of outputting signals to indicate its real-time status, such as open/closed or the percentage of opening. The response can be used as a fingerprint that uniquely identifies the device.

In this work, we leverage the device responses to infer the knowledge of the devices in the CPS. Without loss of generality, we look at a typical set of devices that adequately represent the common types of actuators in CPS. We find that the response packets contain useful information that reflects the model and configurations of the device. Finally, we apply the obtained information to forge response packets corresponding to the actual devices.

1.2 Contributions

The contributions we make in this paper are summarized as follows:

- Our work incorporates the physical domain of the CPSs, and extends the idea of reconnaissance in an attack down to the physical level of CPSs.
- We exploit the timing-based fingerprinting techniques to infer the models and configurations of the devices in CPSs. Our method not only can defeat the device-physics based defenses (e.g., [15]), but also provides the information for the attacker to launch more targeted attacks against the physical processes.
- We build testbeds that emulate common industrial systems and collect data from multiple types of real CPS devices to verify the effectiveness of our technique. The results are promising and show that both the devices' models and configurations can be inferred through the response packets, and responses can be forged using the inferred values.
- We discuss several possible defenses and propose a challenge-response based defense mechanism for the device physics aware response-spoofing attacks.

2 Background

2.1 Overview of CPSs

A CPS is composed of four major types of components, **plants** (also known as processes), **controllers**, **actuators**, and **sensors**. Figure 1 shows the interaction among these components in the simplest form of CPS, using the control of a robotic arm as an example. In reality, a CPS can be a cascaded structure, where one or more control loops may be embedded in the plant of a higher level loop.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

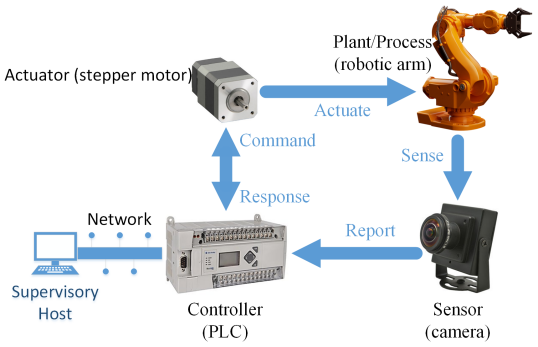


Fig. 1: Interconnections among different elements in a CPS.

The **plant** is a physical process that often entails the primary objective of the CPS, such as a room where the temperature must be maintained within a certain range, or a robotic arm that needs to be moved through space in certain sequences. The plant is a combinatorial result of the control commands and the laws of physics.

The **controller** functions as the center of computation in a CPS, and also generates the control commands. Historically, the controller has evolved from mechanical switches and valves manually operated by human operators, to simple circuits that are hard-wired to follow a routine, and finally digital hardware that are controlled by software, such as a PLC, Remote Terminal Units (RTU), or other microprocessor based embedded systems typically found in a household environment. These controllers are capable of performing network communication and thus may be connected to a computer network for the ease of centralized management. In particular, the PLC as shown in Figure 1 can take commands from a host device (e.g., supervisory computers in a SCADA system) and translate the commands embedded in the network packets into electrical signals that drive the actuator. It can also act upon its local information, such as data read by the sensors to maintain the control objective of the plant.

The **actuator** implements the control commands sent from the controller to the plant. For the software controlled controller, the actuator bridges the gap between the controller in the *cyber* domain and the plant in the *physical* domain. There are various forms of actuators including motor, valve, relay, pump, etc. Unlike the speed of information propagation in a computer network, which is predominantly determined by the processing power of the devices and the speed of light, actuators are physical devices bounded by the laws of physics. Thus, there is usually a delay in the control action carried out by the actuators, given the command from the controller. Many actuators are equipped with feedback mechanisms that report back their real-time status to the controller.

The **sensor** is a one-way interface which converts the physical quantities into electrical signals that can be read by the controller.

2.2 Physics-based Defense Techniques in CPSs

There are two types of physics-based defense techniques in CPSs: 1) one that uses models of the physics of the process (system); and 2) one that uses models of the device physics. In this paper, we seek to defeat the device-physics based defense techniques.

System-Modeling. This type of CPS defense technique attempt to model the behavior of the physical system in a CPS, usually composed of multiple devices (sensors, actuators, etc.) and processes. Such models usually leverage the knowledge about the system specifications and system and control theory to seek the detection of potential hazardous states [8], [20], [25]. For example, Cárdenas et al. proposed to use linear system models to detect attacks on networked control systems [9], [10]. They were able to detect stealthy attacks on these systems with linear system models. Urbina et al. studied if physics-based attack detection can limit the impact of stealthy attacks in the ICS and showed that the impact of such attacks can be mitigated by the proper combination and configuration of detection schemes, including a stateful model of the physical system [25]. More recently, McParland et al. proposed a framework to monitor physical constraint violations by leveraging specification-based intrusion detection [20].

Device-Modeling. While system-modeling based techniques focus on the system-level behavior on a CPS, device-modeling techniques can become quite useful in characterizing benign versus malicious operations inside CPSs as well. Such techniques focus on the correct operation of individual devices in a CPS. Similar to a physical system composed of various actuators, controllers, sensors and processes, each device in this system can also be modeled using deterministic equations, per the laws of physics. For example, Formby et al. proposed to tackle the false data injection issued during control command requests to the field devices [15]. The idea was to help ensure the authenticity of the responses by analyzing the observed response against the fingerprints developed by the operation time associated with each device in an ICS, which is a large division in CPS. They claimed to achieve an accuracy as high as 99% using this fingerprinting technique to differentiate authentic mechanical relays from spoofed ones. The authors also showed that it would require a highly knowledgeable and skilled adversary to perform a forgery attack on the fingerprinting technique.

3 Problem Description

We motivate our work with a realistic attack scenario as depicted in Figure 2, where a centrifuge (the plant) and a small part of the SCADA system are shown. The objective of the system is to set the power of the centrifuge and monitor its safe operation from the supervisory host. The control command is sent from the supervisory host over the local area network (e.g., Ethernet) using industrial standard communication protocols (①), such as Modbus or Ethernet/IP. The PLC starts the motor (②) which drives the centrifuge (③) upon receiving the command from the host (e.g., set the centrifuge to run at 25% power). In the meantime, the motor sends back its real-time speeds which are timestamped and encapsulated by the PLC to the supervisory host (④). The PLC also receives various measurements (e.g., temperature, vibration, etc.) from the sensors connected to the centrifuge (⑤(⑥)) and adjusts the control output accordingly.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

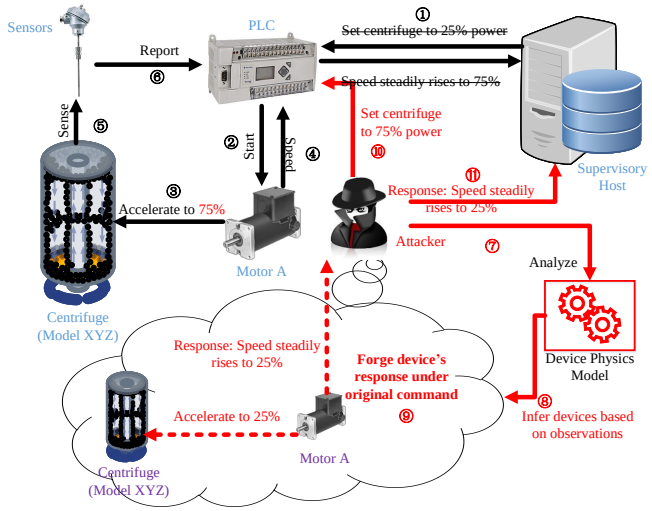


Fig. 2: Attack model used in this paper, where the attacker injects a false command to the PLC and a forged response to the supervisory host. The attacker needs to first observe the legitimate traffic to infer the actual devices' models and configurations before spoofing the response.

To bound the problem, we focus on attacking the system by spoofing the falsified commands to control the devices (⑩), while sending back responses to the original commands that can deceive the device fingerprint detection mechanism (⑪). The attacker's objectives are two-fold: a. to obtain the models and configurations of the devices (e.g., details related to the motors or centrifuges such as their weight and capacity) (⑦(⑧)); b. to forge responses based on such information (⑨). It is worthy noting that the information related to their physical properties is not explicitly transmitted in the network. Thus, the attacker can only learn such information through passive observation of the network traffic in order to minimize disturbance to the system before mounting the actual attack.

3.1 Attack Model

In this work, an attacker is assumed to have gained access to the corporate network and can spoof the control command sent by the PLC to the actuators. He can modify the network traffic between the PLC and the supervisory host by intercepting and forging network packets, but does not have access to the supervisory host. Such an assumption is reasonable, as the communication between the PLC and the supervisory host is usually unencrypted, while the supervisory host is usually a relatively powerful computer system that is equipped with modernized defense techniques. The attacker is also assumed to have only network access but no physical access to the target CPS, hence can only observe and modify the network traffic. Such network traffic does **not** include direct information of the models (e.g., manufacturer's name, model number) and configurations (e.g., load percentage) of the sensors and actuators, as these devices are controlled with electrical signals by the PLC, and do not directly transmit data/packets on the network. The attacker can carry out reconnaissance first to collect data on the general system architecture, but does not have information on the specific model and configuration of each device. For example, an attacker who targets a critical infrastructure such as a thermal power plant may be able to get information on its commission date and capacity [3]. Finally, the attacker also has access to the specifications for all models of the target device types, and can acquire specific models of the devices to perform experiment and build a catalog of their signatures.

The attacker's objective is to sabotage the physical process of the CPS by injecting a false command to the PLC. However, the attacker is also required to deceive the IDS by injecting the forged response corresponding to the original command, as the IDS ensures the system is intact and normally operating by checking the response from the PLC against the expected response from the underlying physical devices. Note that a naive replay attack may result in a failure as mentioned in Section 1.1. Because of the imperfection in the timing of the replay attack, the threshold for mounting a successful replay attack is high [15]. Hence he must achieve enough precise timing control of the spoofed packets, whether the spoofed packets were pre-recorded or dynamically generated.

3.2 Formal Definition of the Device Response Mimicry Attack Problem

The primary goal of our work is to infer the models and configurations of the devices in CPSs. The assumptions we use in this study include: 1) There exist $\mathcal{N}$ product models $\mathcal{D}_i$ ($i \in [1, \mathcal{N}]$) for a given device type. Each $\mathcal{D}_i$ has $\mathcal{M}$ configurations $\mathcal{D}_{i,j}$ ($j \in [1, \mathcal{M}]$). 2) The attacker initially does not have knowledge of the value for i and j. By observing the legitimate responses $\mathcal{R}_{i,j}$ sent by a device $\mathcal{D}$, the attacker classifies $\mathcal{D}$ into $i_a \in [1, \mathcal{N}]$ and $j_a \in [1, \mathcal{M}]$. 3) The attacker injects false command $\mathcal{C}_a$ to alter the original command $\mathcal{C}$ sent to the the device $\mathcal{D}_{i,j}$, while spoofing the finite-time response $\mathcal{R}_{i_a,j_a}$ of the device $\mathcal{D}_{i_a,j_a}$ under command $\mathcal{C}$.

The problem can be formally defined as a two parts. The first part is a classification task, where the product model i_a and configuration j_a needs to be determined based on the observation of $\mathcal{R}_{i,j}$ and $\mathcal{C}$. The second part is an attack on the binary classifier trained with $\mathcal{R}_{i,j}$, where goal is for the spoofed response $\mathcal{R}_{i_a,j_a}$ to be classified as legitimate.

4 Methodology

As stated in Section 3, our method focuses on inferring the information of a device in a CPS in two aspects: which specific **product model** a certain device corresponds to, and what **run-time configuration** it runs in. For example: a motor in the schematic of a CPS (e.g., a conveyor system in a factory controlled with SCADA) can be implemented with the product selected from various brands and model numbers. The specific **product model** number used during the construction of the system can be chosen from a variety of available products, as long as it satisfies the required constraints. The parameters of each model may also vary within a reasonable range, e.g., power rating at $500W$ versus $600W$. The **run-time configuration** refers to the configurable

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

TABLE 1: List of CPS devices and their physical properties

Device Type	Load Dependent	Input Type	Output Type	Motion Type
Relay	No	Binary	Binary	N/A
Valve	Slightly	Binary/Analog	Binary/Analog	Linear
Pump	Yes	Binary	Analog	N/A
Stepper Motor	Yes	Analog	Analog	Rotary
Solenoid	No	Binary	Binary	Linear
Electric Motor	Yes	Analog	Analog	Rotary
Hydraulic Cylinder	Yes	Analog	Analog	Linear

states with which the device directly interfaces in the CPS, e.g., the speed setting of the motor, and the load attached to the motor's output shaft.

4.1 Device Physics Modeling

To have an understanding of how the DMC inference technique can be applied among different devices, we first need to build the physical models of various devices. In this section, we build the model of one device, namely the electric motor. Table 1 lists the comparison of seven common devices in a CPS and their physical properties. *Load Dependent* refers to whether the output behavior of the device depends on its load. *Input/Output Type* means whether the device takes/generates binary (e.g., on/off, closed/open) or analog (e.g., continuously variable speed) signal/states. Note that some devices such as valves can have both binary and analog types of input/output, depending on its model and application. Without loss of generality, we leverage three types of devices which cover the most variations in each property dimension to demonstrate our method, namely electric motor, relay, and valve. We take the electric motor as a running example and begin with the mathematical modeling of its device physics. The processes for building three other types of devices can be found in the appendix.

A physical model connecting the electrical domain to the mechanical domain is illustrated in Figure 3. A circuit loop is formed between the positive and negative leads (i.e., brushes) connected by the rotor coils, where E_s is the source DC voltage, i_{emf} is the armature current, R is the armature resistance, and E_o is the induced *counter-electromotive force* (CEMF), as its polarity always acts against E_s. E_o is generated on the abstract component *EMF*, which produces a torque τ on the shaft connected to the load through a bearing and can be expressed as

$$E_o = ZnF/60, \tag{1}$$

where Z is the winding coefficient, n is the rotor's rotation speed, and F is the flux per pole. Using *Ohm's Law* and *Newton's second law for rotation*, the equation which governs the dynamics of the motor is thus

$$\tau = \frac{ZF(E_s - ZnF/60)}{2\pi R}. \tag{2}$$

Consider the case where the load is initially at reset and accelerated by applying a constant voltage E_s to the motor. The angular velocity ω over time t satisfies the equation

$$\omega(t) = \int \alpha dt = \int \frac{ZF(E_s - ZnF/60)}{2\pi R I_{load}} dt - \int \frac{\hat{\tau}}{I_{load}} dt, \tag{3}$$

under the boundary condition $\omega(0) = 0$. Solving the differential Equation 3 and substituting $n = \frac{30}{\pi}\omega$, we get an exponential decay function

$$\omega(t) = -Ae^{-t/B} + A, \tag{4}$$

where $A = (\frac{ZF}{2\pi R}E_s - \hat{\tau})\frac{4\pi^2 R}{Z^2F^2}$ and $B = \frac{4\pi^2 R}{Z^2F^2}I_{load}$. Recall that Z, F, R are constants determined by the specific construction of a motor, hence are **product model** related. $\hat{\tau}$ is independent of ω and can be assumed to be constant too. Therefore, A is linearly correlated to the source voltage E_s and B is proportional to I_{load}. A is **product model** related parameter, while B is a **configuration** related parameter given a fixed product model.

Recall that in Figure 1, the controller sends a command to the actuator and receives response when the actuator executes the command. In this case, the command can be *start motor*, and the response is the timestamped rotation speed of the motor. As network packets containing the timestamps and speed values are sent back from the controller to the other host(s), a time series can be extracted from the packets that correspond to Equation 4. Such time series satisfy the equation when the proper device physics parameters are plugged in.

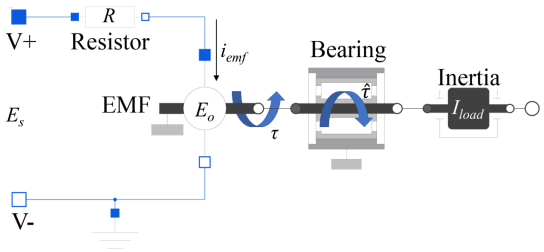


Fig. 3: Physics model of permanent magnet DC motor.

4.2 Characterization

The output type of each device in Table 1 is classified as either binary or analog. Combining this with the physics models of each device, we find that each device has a deterministic response which can be observed and measured. We define two types of response from a CPS device.

Operation Curve. For those that have an analog output, e.g., speed of a motor, position of a modulating valve, a time series data can be obtained by continuously sampling the output value with a timestamp. We refer to this type of data as the operation curve.

Operation Time. Devices with binary output, such as the open/closed states of a relay or a two-position valve can be characterized based on the time difference between the applied signal and the desired state change. This time difference is called the operation time

The operation curve is a direct result of the signal given to the device, as well as the specific physical construction of the device itself. Similar to the principle behind the operation curve, the operation time is also dependent on the physics process inside the device. Therefore, both the operation curve and the operation time are a function of the device physics, defined by its governing equations and the parameters. Whether the reverse is also true is the key to infer the device physics from its observable output, and will be discussed in the following sections.

4.3 Device Model and Configuration Inference

We now identify the DMC from its response by going backwards in time, i.e., find the product model and configuration

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

related parameters in the mathematical model of the device, that will generate the response of the legitimate device. We use the non-linear least squares method. It is a form of least squares analysis used to fit a set of m observations with a model that is non-linear in n unknown parameters ($m > n$).

Consider device's response to be a set of m data points, $(t_1, \mathbf{V}_1)$, $(t_2, \mathbf{V}_2)$, ..., $(t_m, \mathbf{V}_m)$. Without loss of generality, $\mathbf{V}_i$ is a vector of all the values in the response at time t_i, where $i \in [1, m]$. Assume that the function mapping all the product model and run-time configuration related parameters $\mathbf{P}$ to $\mathbf{V}$ is $\mathbf{V} = f(t, \mathbf{P})$, where $\mathbf{P} = (P_1, P_2, ..., P_n)$ ($m > n$). The goal is to find the vector $\mathbf{P}$ such that the curve best fits the given data in the sense of the least sum of squares, i.e.,

$$S = \sum_{i=1}^{m} r_i \cdot r_i \tag{5}$$

is minimized, where the residuals $\mathbf{R}_i$ are given by

$$r_i = \mathbf{V} - f(t, \mathbf{P}). \tag{6}$$

The Gauss-Newton algorithm can be used to solve this optimization problem. Given an initial value of $\mathbf{P}_0$, an iterative search updates the parameters by

$$\mathbf{P}_{k+1} = \mathbf{P}_k - (\mathbf{J_r}^\intercal \mathbf{J_r})^{-1} \mathbf{J_r}^\intercal \mathbf{r} \mathbf{P}_k, \tag{7}$$

where k is the iteration number, and the Jacobian is defined as

$$(\mathbf{J_r})_{ij} = \frac{\partial r_i \mathbf{P}_k}{\partial P_j}. \tag{8}$$

The best-fitting $\mathbf{P}$ is found when the algorithm converges.

The final step is to map the values in $\mathbf{P}$ to the device's product model and run-time configuration. While the latter can be interpreted numerically, the former one can be found using the specification sheets of a set of candidate devices commonly found in CPSs.

4.4 Device Response Packets Synthesis

The last step is to synthesize the device response packets containing the timestamped measurements. The most straightforward method is to plug the parameter values found in Section 4.3 back into the device physics modeling equations in Section 4.1 using the values corresponding to the device models and configurations. However, the parameter values obtained using the non-linear least square method can vary across different observations of the device's responses, even when the DMC is constant. This may arise either due to the measurement errors or slight change in these device parameters themselves (e.g., the resistance varies when the temperature changes). Therefore, a sampling process is added to randomly choose an observed value for each parameter before plugging in.

5 Experiments

It comes with extreme difficulty to obtain real data in CPS due to the cost associated with interfering the normal operation of a potentially valuable system. To address it, we set up two testbeds using real industrial standard devices that simulate corresponding systems that have realistic objectives, such as balancing the liquid level in a container or stirring materials during a chemical reaction in a chemical factory. Each of the testbeds could be a potentially valuable target for the attacker.

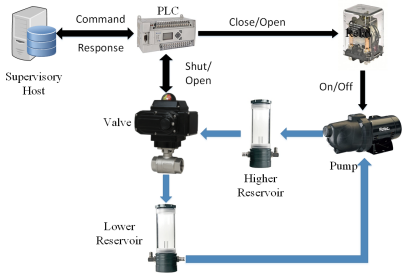


Fig. 4: Block diagram of Testbed 1 setup. The valve and relay are of specific interest in this study.

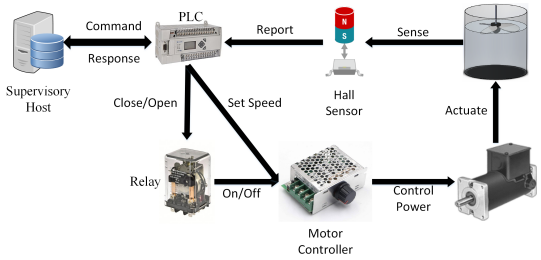


Fig. 5: Block diagram of Testbed 2 setup. The motor and relay are of specific interest in this study.

Among the devices used to build the testbeds, we focus on three types to infer the DMC of each of the devices, namely electric motor, relay, and valve. Leveraging the three types of devices, two physical testbeds were constructed that appropriately mirror a real-world CPS environment, as shown in Figure 4 and 5. In this section, we explain how each type of chosen device operates in the testbed, as well as the experiment procedures and parameter settings.

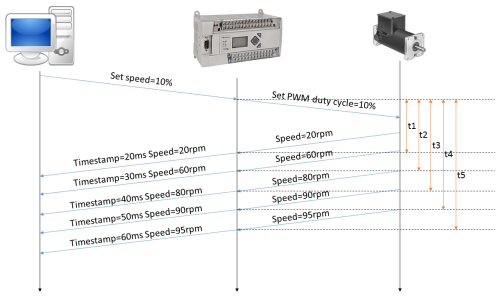
In general, we collected the responses of each type of device, and extracted the operation curves or operation times of each device labeled with the actual model $i \in [1, \mathcal{N}]$ and configuration settings $j \in [1, \mathcal{M}]$. These operation curves and operation times are then used as the input to the classifiers to generate the predicted model $i_a \in [1, \mathcal{N}]$ and configuration $j_a \in [1, \mathcal{M}]$.

In this section, we first state the common setup across each testbed. Then we discuss the methodology used to attack each of the three types of devices used in the testbeds, namely electric motors, relays and valves, followed by explaining the technique used to craft precisely timed response packets. Finally, we present and interpret the results of attacking the three types of devices.

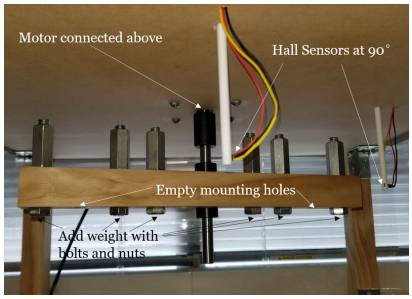
5.1 Timestamps and Protocols

As can be seen in Figure 6a, a supervisory host sends high-level command to the controller, and the controller processes the command in the form of network packets and directly controls the field device (motor in this case) to perform the action. This setup corresponds to the Level 0 to 2 in a SCADA architecture. To closely mimic an industrial environment, we chose a PLC for the controller among other available options (e.g., a PC or embedded platform). In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing



(a) Network packet timing diagram of the experiment setup.



(b) Motor device showing sensor positions and load adjustment.

Fig. 6: Electric motor testbed setup. (a) also applies to other types of devices.

addition to adding authenticity to the testbed, the PLC also comes with an important feature that may be absent from other controllers, i.e., a high precision clock that is used to *timestamp* the events. The PLC we used was an Allen-Bradley Micrologix 1400 series. It has a 32-bit high speed clock which provides a timing resolution of $9.92063492\mu s$. Due to the simple program flow and predictable program scan cycles of a PLC, the timing for an event at the PLC can almost always occur immediately after it. In comparison, neither a PC nor an embedded controller such as a Raspberry Pi is able to achieve a constant timing resolution like a PLC, because of their non-real time operating system or much slower clock frequency. The timestamps can either be embedded natively in the packet if supported by the protocol (e.g., distributed network protocol (DNP3)), or taken by the program running on a PLC, if the protocol does not have native support for timestamps (e.g., Modbus). Without loss of generality, we chose the latter method in all experiments. Note that the timestamps are transmitted to the supervisory host as network packets.

For the command and response sent between the supervisory host and the PLC, we used Modbus as the communication protocol, as it has become a de facto standard communication protocol and is now a commonly available means of connecting industrial devices [11]. The supervisory host ran a Python script leveraging the *modbus-tk* library and acted as an HMI running on the engineering workstation. It was responsible for initiating the communication and sending command to the PLC acting as a Modbus slave, as well as constantly checking for new responses. Each response contained an event, such as a new speed reading in the case of the motor testbed. The event was always accompanied with a timestamp taken by the high speed clock on the PLC.

5.2 Electric Motor

The testbed setup of the electric motor can be seen in Figure 5. Note that although the command sent from the host contained both the power setting and a start command, we only applied the power setting at the motor controller as different power output, which was set prior to the start of the motor. This is because we intended to emulate the scenario where the motor is of a certain model that operates under the given power ratings. Thus, the power setting would be obscured from an attacker who can only observe the network traffic. The shaft of the motor was connected to a load with variable MOI as shown in Figure 6b. Two hall sensors were placed to enable the PLC to calculate the angular speed of the load. A timestamp relative to the command received from the host was taken and read by the host together with the angular speed of the motor, which formed an operation curve.

Both the product model and the run-time configuration related parameters were variable in this testbed. For each start command sent from the host, we collected the responses from the motor while varying its product model and run-time configurations. In summary, 100 trials were performed for each of the 80 experiment settings, and all $8,000$ sets of data were collected. Each trial took 30 seconds to ensure the speed of the motor stabilizes, resulting in roughly 67 hours of data collection.

TABLE 2: Key Parameters of the Relays Taken from Their Specifications

Model	Close Time	Open Time
Schneider 785XBXCD-24D	20ms	20ms
Omron G2RV-SR500 DC24	20ms	20ms
TE K10P-11D15-24	10ms	13ms
TE KUEP-11D15-24	10ms	15ms
TE KUIP-14D15-24	15ms	20ms
TE KUL-11D15D-24	25ms	25ms
TE KUP-14D15-24	15ms	20ms
Omron MKS3PI DC24	20ms	30ms
TE MT221024	10ms	15ms
Schneider RSLZVA1	5ms	12ms

5.3 Relay

The relay testbed setup shared a similar structure as that of the electric motor, except that the we focus on different models of the relay and hence swap different models of relay used in the testbed. Because the operation of the relay is hardly affected by the load it controls (i.e., run-time configuration), we therefore did not connect any load as we did in the electric motor testbed.

In reality, the selection of the specific relay model at a single point in a system depends on the application requirements, e.g., the control circuit voltage, socket type, rated operating voltage and current, etc. To set up a realistic experiment, we selected the set of relays using common industrial settings, i.e., $24VDC$ control voltage and DIN rail mounted relays. A total of 10 different relays were found as

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

TABLE 3: Key Parameters of the Valves Taken from Their Specifications

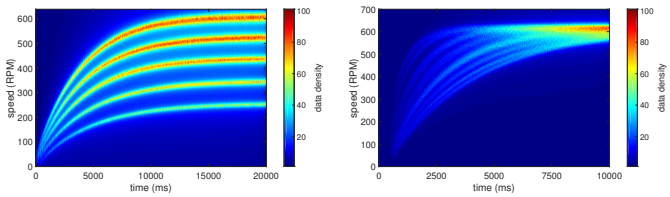
Model	Type	Open/Close Time
Dwyer WE01-CTD01-A	Two-Position	4s
Dwyer WE01-CMD01-A	Modulating	10s
Dwyer WE01-GTD02-A	Two-Position	20s

listed in Table 2. The supervisory host sent either a *close* or *open* command to the PLC depending on the last status of the relay. The PLC recorded the time when the command was received, and either energized or de-energized the coil of the relay accordingly. It then polled its input pin connected one contact of the relay output, while the other contact was connected to a logic high voltage. When the PLC first detected a signal level change, it again recorded the command completion time. The difference of the two timestamps were calculated and stored with a flag, which can be read by the host as the operation time. For each relay, we conducted the close/open cycle for $1,000$ operations. Each operation was allocated 5 seconds, therefore the entire data collection process took around 14 hours.

5.4 Valve

Similar to the relay testbed, we kept the components from the supervisory host to the PLC, while replacing the field device with three different valve models listed in Table 3.

This setup studies the correlation between the operation of each valve and its specification sheet. In each experiment, we connected only a single valve to the PLC, and swapped different models. Without loss of generality, we used two types of valves in our experiment, namely the two-position valve and the modulating valve. The two-position valve operates in a binary manner, i.e., the PLC outputs a binary signal to fully *close* or *open* input to the valve. When the valve finished executing the action, a limit switch in the valve would be triggered, and thus the event could be detected by the PLC. A time difference was calculated between the reception of the command and detection of the completion, and was read by the host as an operation time. The modulating valve operates in an analog manner, i.e., the PLC outputs a $4-20mA$ current loop to the valve that linearly translates to a valve position. The valve compares the current loop input with its current position, and executes accordingly to adjust for the difference. Meanwhile, its physical position is continuously translated to another $4-20mA$ current loop, which can be read by the PLC. The PLC kept polling the readings and stores in its memory with timestamps, which in turn would be read by the host as an operation curve. To compare this operation curve of the valve with its open/close time listed in Table 3, we converted the operation curve to an operation time by defining a cutoff position that the valve reaches at the end of each actuation, and computed the difference between the command time and the time of the cutoff position. Another reason which called for the conversion was that the modulating valves were changing their positions at a constant speed, unlike the electric motor. Hence converting their operation curve to the operation time resulted in negligible loss of information.



(a) 5 Different Power Ratings. (b) 16 Different Load MOI.

Fig. 7: Heat map plot of the electric motor's operation curves from different models and under various run-time configurations. Each curve is aggregated over 100 runs.

In total, $1,000$ opening closing operations were performed on each model, taking around 19 hours for data collection.

5.5 Implementing Timestamped Forged Response Packets

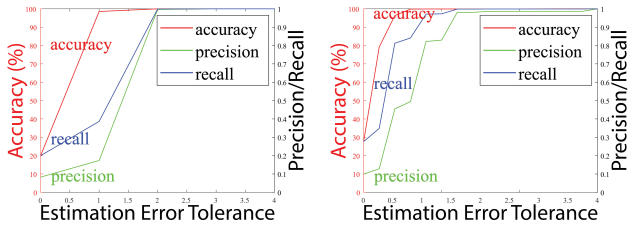
As mentioned in 1.1, an attacker with the forged response packets faces the challenge of sending them with accurate timing. In [15], the authors mentioned that an attacker who has a device with limited capability can be detected due to low clock precision and clock drifting. To overcome this issue, we leverage the PLC's real-time program execution feature. During our experiment, the timestamps and measurement values in the generated responses are stored in the PLC's memory table and loaded by the ladder logic diagram sequentially. Specifically, starting from the first stored timestamp/measurement value pair in the table, the time t since a command has been received is taken using the high-speed clock's value in the PLC. A pointer index p is initialized to 0 following each command. The elapsed time is compared with the timestamp t_p stored in the table t_i. If $t \geq t_i$, the pth measurement value (if the device has an operation curve) or the flag value (if the device has an operation time) is copied to the Modbus memory for the supervisory host to read, and p is incremented by 1. The spoofing process ends when all responses are sent.

5.6 Results

This section shows the results of inferring each device's product model and configurations, as well as the attacker's forged responses. The performance of the inferring technique is measured with three standard metrics in classification tasks, namely accuracy, precision and recall. Let TP, TN, FP and FN be true positive, true negative, false positive and false negative, respectively. **Accuracy** is defined as $\frac{TP+TN}{TP+TN+FP+FN}$. **Precision** is defined as $\frac{TP}{TP+FP}$. **Recall** is defined as $\frac{TP}{TP+FN}$. We also introduce another metric named **estimation error tolerance** when evaluating the performance of device model and configuration inference. The estimation error tolerance is defined as the number of values between the estimated and the correct values, when all possible values for the given parameter are sorted.

Electric motor. The authentic responses from the electric motor were gathered by varying both its models and run-time configurations, namely the power ratings and the MOI of the load. For better visibility, the aggregated operation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing



(a) Accuracy, Precision and Recall of Power Rating Inference.

(b) Accuracy, Precision and Recall of Load MOI Inference.

Fig. 8: Performance of the run-time configuration inference of the electric motor. The estimation error tolerance is the allowed distance between the estimated value and the attacker's assumed value.

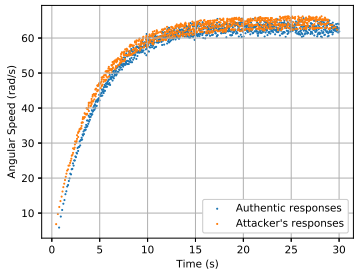


Fig. 9: Comparison of the authentic and spoofed responses from an electric motor.

curves are shown as heat maps in Figure 7. Each heat map was generated by plotting the density of the 100 operation curves for every model or configuration setting. It can be seen that the operation curves are clearly distinguishable and are highly correlated with the parameter values. In the experiment, we used 5 different power ratings (to emulate five different models of electric motors, corresponding to the product model related parameter A from Section 4.1) and 16 different load settings (corresponding to the configuration related parameter B from Section 4.1). Thus there are 80 different combinations and $8,000$ operation curves in total. Each operation curve was taken as a response from this device and its model and configuration were inferred. The result is shown in Figure 8, which shows that our method was able to correctly infer 98.6% of the power ratings (device model) within a tolerance of 1 value, and 99.9% of the load values (device configuration) within 3 values. The non-zero error tolerance values exist because the operation of physical devices could not be perfectly modeled with the equations. For example, there was always energy loss in the form of heat or vibration, which caused the inferred power ratings to be lower than the actual power consumed. We also note that under the same estimation error tolerance, an increase in the number of possible values for a parameter may adversely affect the performance of the inference results. This is expected from the method we used, as a larger number of possible values for a parameter typically means these values are more densely distributed over a range. Nevertheless, we found the differences between the inferred values and the actual ones corresponding to the DMC to be systematic errors, which can trivially be calculated either by knowing at least one actual values or a prior experiment conducted by the attacker using testbeds. Moreover, when we used these DMC parameter values to generate the forged responses, the difference with the authentic responses was negligible, as shown in Figure 9. The DMC inference method was applied to the forged responses and found all of the responses correspond to the actual DMC. Hence, the device physics fingerprinting method in [15] cannot detect our forgery attack.

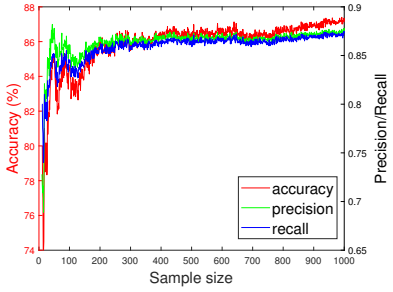
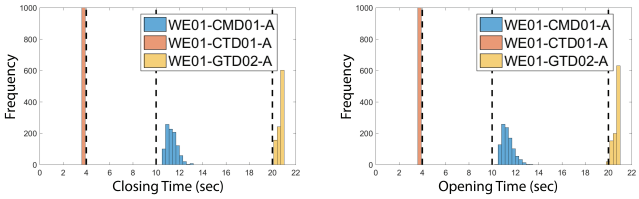


Fig. 10: Classification performance using relays' operation time.

Relay. Similarly, the authentic responses of relays were collected first. Each relay model exhibited a densely distributed closing/opening time around a mean value, with some overlap among several models. Because the operation time of each relay has been explicitly noted in its specifications (which is an important factor in choosing the right relay for any time-critical application for safety), the logical next step was to test the correlation between the relays' specified operation time values with their experimental values. However, we found only 0.54 and 0.58 correlation coefficients for closing and opening time, respectively, using product-moment correlation coefficient (PPMCC). Thus, we focused on each model and ranked each model according to the distance between its experimental operation time and the specification values of all 10 models. Since the operation time is a 2-dimensional vector, we used a number of distancing metrics including Euclidean, Chebychev and Manhattan, and found that the Euclidean metric performed the best among all. In summary, when using the Euclidean distance metric, six relays were ranked top 3, which means that when inferring the model of the relays, the attacker is guaranteed to find the correct model within three trials (which can be compared with the estimation error tolerance in the electric motor's results). However, if the reference operation time of each candidate relay model can be experimentally measured (the attacker may acquire every model of a device) instead of being taken from their specifications, the classification performance can be greatly improved as shown in Figure 10. Because the cost of data collection increases with the number of times the measurements are taken, we vary the sample size from 10 to $1,000$ to show the relationship between the classification performance and the resourcefulness of the attackers. The classification was carried out by training a Nearest Neighbor classifier in 10-Fold under each sample size. Because the forged responses were crafted based on the inferred model of relay, the accuracy of classifying them into the DMC (a correct classification means a successful attack) is not shown as it followed a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

similar curve as the accuracy curve in Figure 10.

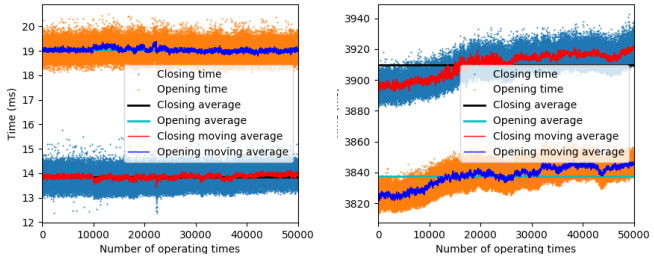


(a) Closing Operation Time. (b) Opening Operation Time.

Fig. 11: Histogram of the valves' operation time. Dashed lines indicates the specification values.

Valve. In contrary to the relays, the valves' actual operation time values were very close to their specification values, and thus can all be correctly inferred without separate measurements. As shown in Figure 11, the distribution of the closing and opening time of each valve was adjacent to the corresponding values listed in Table 3. The only misalignment is the modulating valve, with model number WE01-CMD01-A, which has a slightly larger operation time than expected. Since it did not have a binary output as the other two valves did, it was not obvious to define an exact position of the valve as fully closed or open. As explained in Section 5.4, heuristic values were used to derive the operation time from the its operation curves, which lead to the minor offset. Nevertheless, this did not influence the performance of our inference method, as all three valves' operation time was clearly distinguishable. For all three valves used in our experiment, the accuracy of inference is 100% and both precision and recall remain 1.0. All of the forged responses were also identified as the intended model of valve by the device physics fingerprinting method in [15].

Impact from time-variant factors. The physical nature of the devices means that they may suffer from wearing and aging effects in the long term, especially in an industrial environment where the devices are used frequently. In such case, the parameters of the device physics model could deviate from its ideal values. For example, the operation time of a mechanical relay depends partially on the electromagnetic force generated and the force in its spring. As the number of operations increases, the spring may suffer from fatigue, which causes the deviation of the relay's operation time. Similarly, an operation time of a valve may deviate as it keeps operating throughout its lifetime. To have a preliminary understanding of how this affects our technique, we performed extended tests with both testbeds used in the experiments, and observed different changes in their operation curves or operation time. Only the valve shows a deviation in its operation time. For example, Figure 12a shows the operation time recorded over $50,000$ times of open/close operation of a relay (Omron MKS3PI DC24), which is rated to be operated at $18,000 ops./hour$ and with a mechanical endurance of 5×10^6 operations. As it is challenging to perform measurements over its entire lifetime, we have used its maximum operating frequency to accelerate the wearing and aging effects. We use moving average as the metric, with the window size set to 50. However, we did not find evidence that the operation time changes during this test. On the other hand, the valve used in the wearing and aging



(a) Relay's operation time over $50,000$ operations. (b) Valve's operation time over $50,000$ operations.

Fig. 12: Wearing and aging test of relay and valve.

test (Dwyer WE01-CTD01-A) does exhibit a gradual increase in its operation times as shown in Figure 12b. Namely, the moving average of its closing time increases from $3883ms$ to $3943ms$, and that of its opening time increase from $3814ms$ to $3860ms$. The increase over the $50,000$ operations is 1.5% and 1.2%, respectively. At this rate, it would take approximately 3×10^6 operations before its operation time becomes indistinguishable from the other models used in our experiments. Apparently, the performance of our DMC inference technique is insignificantly affected throughout the extended tests. However, it is worth noting that the number of cycles performed in our test is still far away from rated life of the devices. Therefore, the conclusion drawn from this test is only preliminary. We plan to perform more extensive experiments to find out the long-term impact, as well as other factors such as temperature and humidity on the performance of our technique in future works.

6 Discussion

6.1 Applicability to Other Field Protocols

In our experiment, we have employed Modbus as the communication protocol used between the PLC and the host. In reality, other protocols may be used depending on the specific application, such as DNP3, Common Industrial Protocol (CIP), BACNet, or ProfiNET. Most of the protocols were designed without security features and transmit application layer data (e.g., the timestamped values sent by the actuators) in clear text. However, some protocols have now been modified or designed with security in mind (albeit rarely used in practice), and use encryption to protect data privacy and data integrity, such as secure DNP3. In such cases, the system still may not be exempt from an attacker who has access to the PLC's firmware or program. As have been discussed in Section 5.6, the application layer data is decrypted in the PLC if 1) the protocol natively supports timestamping or 2) timestamping function is added to the PLC program. In the last case where timestamps can only be obtained at the tapping point, an extra step need to be employed to correlate the timestamp information at the tapping point with the value-only data decrypted at the PLC, in order to produce the accurate time series values generate by the actuators. The attack's performance can be affected if the attacker's source of time is different from the one used at the tapping point.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2021.3089163, IEEE Transactions on Dependable and Secure Computing

6.2 Applicability to Other Device Types

Although we only demonstrated the methodology using three types of CPS devices, we believe it can be generalized to many other devices that share similar properties. Because the majority of actuators in CPSs are made of mechanical or electro-mechanical components that obey the laws of physics, and rarely include programmable components, their operations can be very stable in every actuation. Additionally, given a specific command of actuation, these actuators' response signals in the temporal domain are tightly correlated with their models and configurations. The motion of most actuators (such as electric motor, relay, valve, solenoid, and stepper motor, etc.) can be described with first- or second-order differential equations, which enables the mathematical model inversion that leads to the inference of the DMC. The method can be applied as long as these devices can be modeled with equations, which are fitted with the operation curves or operation time of the devices. Depending on the type of device, our method may be able to infer either its model, configuration or both. The cost associated with the method comes mostly from modeling the devices. However, some device types may take more effort to model, such as the turbine in a thermal power plant, due to the complicated computations involved with thermal and fluid dynamics.

6.3 Defending Against CPS Mimicry Attacks

Most of the existing defenses against mimicry attacks [26] focus on the IT domain. For example, some researchers proposed to use system call trace to enforce the correct program execution [14], [22]. Another paper used control-flow integrity (CFI) checks to prevent attacks from arbitrarily controlling program behavior [5]. However, such solutions can hardly be applied in the CPS environment, due to the difficulty of instrumenting code execution in the PLC. Moreover, our attack can be implemented on the PLC program level, which doesn't change the control flow of the firmware. A related work [6] attempted to design a PLC-compatible CFI mechanism. However, the authors mentioned that their framework could not be implemented on a real PLC and used an open source software instead. Therefore, we propose a new defense technique.

It is worth noting that although Modbus - a communication protocol with no encryption - has been used in this experiment, however, even when using a protocol which supports encryption, our mimicry attack will still be successful. This is because the encryption only exists **between** the PLC and the supervisory host, and the data has to be plain text in PLC's memory. An attacker who compromises the PLC's firmware or program can access the PLC's memory regardless of the encryption used in the communication protocol.

An intuitive countermeasure is to inject noise into the operation curve or operation time when sending the response packets from the PLC, or at the sensor which is measuring the physical signal. While this may prevent an attacker from obtaining an accurate DMC and hence generate the correct response, there are two drawbacks of this approach. First, the control algorithm of certain types of devices may leverage these sensory signal inputs, then adjust the output signal to the actuator to achieve optimal operation of the device. Noise in the sensory input may interfere with such control algorithm and degrade the safe and smooth operation of the CPS. Second, the noise-injected responses may interfere with the device fingerprint based defense system, and inadvertently increase the false positive rate. Therefore, adding noise is not a feasible option.

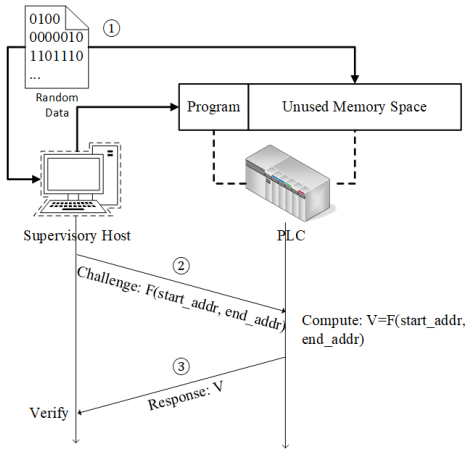


Fig. 13: Challenge-response framework to defend against device physics mimicry attacks.

Based on our observation in this study, we find that in order for the responses to be generated and spoofed in time when a command is received, the attacker needs to pre-compute the timestamps and measurements of a device and store the data in PLC's memory for fast access. Storing such data over another network device may not meet the real-time performance requirement of sending the responses. Also, unlike a computer program which uses the stack and heap which are dynamic in memory, a PLC program accesses its memory via assigned blocks arranged in table files, which almost always uses a known amount of memory in fixed locations. Therefore, we propose a challenge-response defense framework as shown in Figure 13. The steps are described as follows:

1) In the initial setup, the (potentially large) unused memory in a PLC is filled with pseudo-random data known to the supervisory host. Essentially, the supervisory host keeps a copy of all unused memory of every PLC in its network. Typically, the user memory in PLC is less than 100MB (e.g., [1], [12], which leaves even less space after the control program has been loaded.

2) During normal operation, the supervisory host periodically sends a request to the PLC to apply a function F over the data in a randomly chosen region of its unused memory $[start\_addr, end\_addr]$. Such function can be XOR, hash, etc.

3) The PLC computes and sends the result $V = F(start\_addr, end\_addr)$ back to the supervisory host. If there is a mismatch or a certain time threshold is exceeded, an alarm can be raised.

The attacker may choose to minimize memory usage and dynamically generate the responses from a few parameters instead of storing a large amount of pre-computed data in the PLC's memory. However, this will add a substantial amount of delay to the response packets.

6.4 Limitations

Modeling Accuracy. Because the accuracy of our DMC inference technique is highly dependent on the estimation of the parameters in the physics model of the device, such estimation can be less accurate when there is a mismatch between the speculated model and the real device. Thus, a comprehensive analysis may be required to understand the anatomy of the device. The model (and parameters) may also deviate from its original values due to wearing and aging of the physical components in the long term, although the extent to which this may affect the performance of our technique varies among different types of devices. In such case, the operation time/curve may be re-collected and machine learning models need to be re-trained. The equations used in this work may not be 100% accurate in modeling the underlying physics of the devices operations. For example, the aging of the devices and external variables such as temperature, humidity may play a role in the mechanical and physical properties of the materials in the devices, which ultimately affect the response from the device. Despite the possible omission of these modeling errors, our methods worked well as demonstrated by the experimental results in this work, which proved the equations used to be accurate enough.

Signal Availability. Our method assumes that the device must be able to send either corresponding responses upon certain events (e.g., completion of a command), or contain observable state variables to recover the state of the device. Such assumption may often be valid because a closed control is often used in an industrial environment to ensure stability.

System level information. In this paper, we have focused on inferring the information of the *devices* in CPSs. An advanced attacker may step up and attempt to infer the system level information of CPSs. We intend to leave this problem and the corresponding defense technique study as a future work.

7 Related Work

Many existing work in securing CPSs focus on the information technology (IT) domain, such as examining the network traffic in order to look for abnormal packets similar to what a traditional IDS does [21]. For this category of solutions, it is possible to provide a level of security of the control system's network, by treating it as an instance of IT networks and applying mature secure access technologies (e.g., Virtual Private Network (VPN), Firewall, etc.). Neilson proposed to secure the control system from cyber attacks with traditional IT solutions such as VPN [21]. Although the author listed pros and cons of each solution, none of them took the physical system being controlled into account. He et al. designed a novel access control and authentication scheme for the home IoT devices, which focuses on device capabilities instead of a per-device granularity [18]. According to the authors, such scheme may provide finer control over the authorization of the IoT devices. Similarly, Schuster et al. applied the situational access control approach used in smartphone frameworks to the IoT domain, and claimed to reduce over-privileging issue which may be used in an attack [23]. However, these solutions which only focus on the IT domain may fail to defend against attacks in a CPS, where an attack originated from inside the network (e.g., an insider, malicious device firmware) can hardly be detected.

Fortunately, many researchers realized that there is also a significant component in CPS which falls into the operation technology (OT) domain. Earlier work in this category focused on modeling the system behavior and comparing the values output from the model's sensors with those from the real-world sensors. Such approach leverages the knowledge about the system specifications, thus seeking to detect potential hazardous states [8], [20], [25]. Solutions in this category (e.g., modeling of the system physics) attempt to incorporate knowledge specific to the CPSs. Some researchers propose to leverage physics of the system in order to solve this issue. Cárdenas et al. identified several challenges for the CPS security research community including new vulnerabilities, threats and consequences of potential attacks on networked control systems, and proposed to use linear system models to detect such attacks [9], [10]. The authors showed that they were able to detect stealthy attacks that change the physical behavior of the targeted control system by incorporating the knowledge of the physical system. Urbina et al. studied if physics-based attack detection can limit the impact of stealthy attacks in ICS and showed that the impact of such attacks can be mitigated by the proper combination and configuration of detection schemes, including a stateful model of the physical system [25]. More recently, Formby et al. focused on individual devices rather than the entire system behavior of CPS, and found that CPS devices can be fingerprinted due to their unique physical compositions [15]. Later, Gu et al. studied the feasibility of inferring the device models in CPSs [17]. They claimed that such fingerprint can be overlooked by an average attacker, and can be difficult to mimic due to the limited timing accuracy of embedded devices. However, their assumption of attackers using an embedded Linux development platform and PC to generate device fingerprint is not realistic in a typical ICS environment. A PLC may be a vulnerable target in this case, as demonstrated by Garcia et al. [16], who demonstrated a physics-aware attack against PLC's firmware. Their work also showed that an attacker who is aware of the physics of the CPSs can deceive the system model-based detection at the PLC level, as the disjoint is created between the control system and the physical devices.

8 Conclusion

In this paper, we studied the problem of launching a device response spoofing attack in Cyber-Physical Systems. We proposed a novel technique which bridges the gap between the physics of the CPS devices and the responses from the devices. We also built several testbeds and benchmarked our methodology with real devices used in CPSs, and achieved high accuracy in inferring the device model and configuration information, as well as forging responses that are indistinguishable from the authentic devices' responses in most cases. We then discussed the impact on the performance of our method stemmed from various factors, including the amount of available data and chronological wearing of the CPS devices. Finally, we also proposed to use a challenge-response method to defend against such attacks.

REFERENCES

[1] ControlLogix 5570 Controllers.

[2] Crash Override Malware Took Down Ukraine's Power Grid Last December — WIRED, 2018.

[3] Hacking Critical Infrastructure — OSINT Soup, 2018.

[4] Ukraine's power outage was a cyber attack: Ukrenergo, 2018.

[5] Martin Abadi, Mihai Budiu, Ulfar Erlingsson, and Jay Ligatti. Control-Flow Integrity Principles, Implementations, and Applications. In *12th ACM Conference on Computer and Communication Security*, 2005.

[6] Ali Abbasi, Thorsten Holz, Sandro Etalle, and Emmanuele Zambon. ECFI: Asynchronous Control Flow Integrity for Programmable Logic Con-trollers. 12, 2017.

[7] David Albright and Frank Pabian. It Fits! Qom Site Layout. Technical report, 2018.

[8] Alvaro A. Cárdenas, Saurabh Amin, Zong-Syun Lin, Yu-Lun Huang, Chi-Yen Huang, and Shankar Sastry. Attacks against process control systems: Risk Assessment, Detection, and Response. In *ASIACCS*, page 355, New York, New York, USA, 2011. ACM Press.

[9] Alvaro A. Cárdenas, Saurabh Amin, and Shankar Sastry. Research challenges for the security of control systems. In *HOTSEC*, page 6, San Jose, CA, 2008. USENIX Association.

[10] Alvaro A. Cardenas, Saurabh Amin, and Shankar Sastry. Secure Control: Towards Survivable Cyber-Physical Systems. In *2008 The 28th International Conference on Distributed Computing Systems Workshops*, pages 495–500. IEEE, 6 2008.

[11] Bill Drury. *Interfaces, communications and PC tools*. Institution of Engineering and Technology, 2010.

[12] Schneider Electric. Modicon M241 Logic Controller - Hardware Guide - 04/2014. Technical report, 2014.

[13] Nicolas Falliere, Liam O Murchu, and Eric Chien. W32.Stuxnet Dossier. *Symantec-Security Response*, (February 2011):1–69, 2011.

[14] H.H. Feng, O.M. Kolesnikov, P. Fogla, W. Lee, and Weibo Gong. Anomaly detection using call stack information. In *Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405)*, pages 62–75. IEEE Comput. Soc.

[15] David Formby, Preethi Srinivasan, Andrew Leonard, Jonathan Rogers, and Raheem Beyah. Who's in Control of Your Control System? Device Fingerprinting for Cyber-Physical Systems. In *NDSS*, San Diego, CA, USA, 2016. Internet Society.

[16] Luis A. Garcia, Ferdinand Brasser, Mehmet H. Cintuglu, Ahmad-Reza Sadeghi, Osama Mohammed, and Saman A. Zonouz. Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit. *NDSS*, (March 2017), 2017.

[17] Qinchen Gu, David Formby, Shouling Ji, Hasan Cam Cam, and Raheem Beyah. Fingerprinting for Cyber Physical System Security: Device Physics Matters Too. *IEEE Security & Privacy*, 2018.

[18] Weijia He, Maximilian Golla, Ruhr-university Bochum, Roshni Padhi, Jordan Ofek, Markus Dürmuth, Weijia He, Maximilian Golla, Roshni Padhi, Jordan Ofek, and D Markus. Rethinking Access Control and Authentication for the Home Internet of Things (IoT). *Proceedings of the 27th USENIX Conference on Security Symposium*, pages 255–272, 2018.

[19] David Kushner. The real story of stuxnet. 50(3):4853, 2013.

[20] Chuck McParland, Sean Peisert, and Anna Scaglione. Monitoring Security of Networked Control Systems: It's the Physics. *IEEE Security & Privacy*, 12(6):32–39, 11 2014.

[21] Carl Neilson. Securing a Control Systems Network. *ASHRAE*, 2013.

[22] Niels Provos. Improving host security with system call policies. In *Proceedings of the 12th Conference on USENIX Security Symposium - Volume 12*, SSYM'03, pages 18–18, Berkeley, CA, USA, 2003. USENIX Association.

[23] Roei Schuster, Vitaly Shmatikov, and Eran Tromer. Situational Access Control in the Internet of Things. *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security - CCS '18*, pages 1056–1073, 2018.

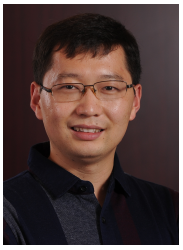
[24] P Srinivasan. Fingerprinting Cyber Physical Systems: A Physics-Based Approach. Master's thesis, 2015.

[25] David I. Urbina, Jairo A. Giraldo, Alvaro A. Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *CCS*, pages 1092–1105, New York, New York, USA, 2016. ACM Press.

[26] David Wagner and Paolo Soto. Mimicry Attacks on Host-Based Intrusion Detection Systems *. Technical report, 2002.

Qinchen Gu received his BS, MS and Ph.D. degrees in electrical and computer engineering from Georgia Institute of Technology. His current research interests include physical side-channel attack and defense and PLC binary analysis in Cyber Physical Systems.

David Formby received the M.S. and Ph.D. degrees in electrical and computer engineering from Georgia Institute of Technology (Georgia Tech) in 2014 and 2017, respectively. He was a Postdoctoral Researcher in the School of Electrical and Computer Engineering at Georgia Tech, and a member of the Communications Assurance and Performance (CAP) group. He is now CEO/CTO of Fortiphyd Logic, a startup focused on cybersecurity for industrial control systems.



Shouling Ji is a ZJU 100-Young Professor in the College of Computer Science and Technology at Zhejiang University and a Research Faculty in the School of Electrical and Computer Engineering at Georgia Institute of Technology. He received a Ph.D. in Electrical and Computer Engineering from Georgia Institute of Technology and a Ph.D. in Computer Science from Georgia State University. His current research interests include AI Security, Data-driven Security and Data Analytics. He is a member of IEEE and ACM and was the Membership Chair of the IEEE Student Branch at Georgia State (2012-2013).

Brendan Saltaformaggio is an Assistant Professor in the School of Electrical and Computer Engineering at Georgia Tech, with a courtesy appointment in the School of Computer Science. His research interests lie in computer systems security, cyber forensics, and the vetting of untrusted software.

Anu Bourgeois is an Associate Professor and Director of Undergraduate Studies in the Department of Computer Science at Georgia State University. Her research focuses on algorithm design, architecture, fault tolerance, and energy efficiency for wireless and optical networks. She is currently working on issues for wireless sensor networks.

Raheem Beyah currently holds the Motorola Foundation Professorship in the School of Electrical and Computer Engineering and serves as the Vice President for Interdisciplinary Research at Georgia Tech. His work is at the intersection of the networking and security fields.