Cyber-Physical Testbed: Case Study to Evaluate Anti-Reconnaissance Approaches on Power Grids' Cyber-Physical Infrastructures

Hui Lin   Bibek Shrestha
University of Nevada, Reno
{hlin2, bibek.shrestha}@{unr, nevada.unr}.edu

Yih-Chun Hu
University of Illinois, Urbana-Champaign
yihchun@illinois.edu

***Abstract*—*Background.* In recent work, we proposed an anti-reconnaissance approach, known as DefRec, to disrupt and mislead adversaries' attacks causing physical damage in industrial control systems (ICS) like smart power grids. We propose this approach on top of an original physical function virtualization (PFV) that "hooks" network interactions with real physical devices and builds lightweight virtual nodes that follow the actual implementation of network stacks, system invariants, and physical state variations in the real devices.**

***Aim.* To evaluate the efficacy and efficiency of DefRec and PFV, we have developed a cyber-physical testbed, including real physical devices, hardware switches, and power-system simulations. In this paper, we present additional details on our explorations of various implementation and experiments leading to the current setups, which are scalable, repeatable, and able to resemble realistic environments.**

***Method.* After briefly presenting the design principle of PFV and DefRec, we present three implementation options of communications networks, four implementation aspects of power-system simulations, and intelligent electronic devices (IED) from three different vendors. We compare the pros and cons of different options and discuss their impacts on experiments in terms of scalability, repeatability, and efficacy to resemble realistic environments.**

***Result.* We present results regarding different implementations of communications networks, network traffic controllers based on software-defined networking, and real IED devices. Based on these results, we provide discussions that may help other researchers to select appropriate experimental methods.**

***Discussion.* In the current implementation, the testbed still lacks dynamic coupling between cyber and physical infrastructures, making it challenging to reflect transient behavior occurring in real power grids. We will leave such implementation in future development.**

I. Introduction

Reconnaissance is crucial to an adversary's preparation for an attack on industrial control systems like smart power grids (ICS) [57]. By obtaining in-depth knowledge of physical processes, adversaries can determine "attack-concept" operations to cause devastating physical disruptions without raising alarms [17]. For the attack on a Ukrainian power plant that caused a blackout affecting 225,000 residents [31], [32], security analysis directly indicates that "the strongest capability of the attackers was ... to perform reconnaissance operations required to learn the environment." Reconnaissance allows adversaries to design attack strategies that cause physical damage (e.g., compromising measurement data or maliciously turning off circuit breakers).

To disrupt reconnaissance in a general-purpose computing environment, current moving target defense (MTD) techniques [10], [12], [56] rely on mimicking and simulation of system behaviors and thus, can be easily identified [25]. To overcome these drawbacks, we proposed in our recent work the design of physical function virtualization (PFV) to build lightweight virtual nodes that follow the actual implementation of network stacks, physical state variations, and system invariants of real physical devices in power grids. PFV leverages a network control application based on software-defined networking (SDN) to "hook" network interactions with real devices and use them as network flows of virtual nodes. Based on PFV, we presented DefRec, a specific defense mechanism to *significantly increase the reconnaissance effort required to infer the knowledge of power grids' cyber-physical infrastructures* without affecting legitimate applications that already know the actual power grid configurations (e.g., the identities of real physical devices). DefRec specifies and implements two security policies to: (i) obfuscate communications by adding random interactions with virtual nodes, introducing significant overhead for adversaries to identify real devices; and (ii) mix decoy data (from virtual nodes) with real data (from physical devices), based on which adversaries would design ineffective and easy-to-detect attacks (e.g., activities that access virtual nodes).

To the best of our knowledge, PFV is the first design to build virtual nodes following the actual implementation of network stacks, system invariants, and physical state variations of real physical devices and DefRec is the first anti-reconnaissance approach targeting on power grids' cyber-physical infrastructures. Evaluating the effectiveness and efficiency of PFV, DefRec, and other security mechanisms for power grids raises new implementation requirements.

- ***Close to Realistic Environments.*** Even though there are

Learning from Authoritative Security Experiment Results (LASER) 2020
23 February 2020, San Diego, CA, USA
ISBN 1-891562-65-7
https://dx.doi.org/10.14722/laser.2020.23001
www.ndss-symposium.org

few research works performing small-scale evaluations in a real environment, such as an in-use communications network or a utility power grid, we believe that a laboratory implementation close to a realistic environment is necessary, for better *repeatability* and *portability*.

- ***Reasonable Scalability.*** We need an implementation that allows us to evaluate the scalability of the proposed work, e.g., regarding the number of network nodes or physical devices.
- ***Integration of both cyber and physical infrastructures.*** The implementation should reflect the domain-specific characteristics found in industrial control environments, e.g., proprietary system specifications, network stacks, and communication requirements.

To satisfy these requirements, we use this paper to discuss the implementation to evaluate the efficacy and efficiency of PFV and DefRec and discuss its potential usage in other related security research. In this paper, we present not only the ultimate implementation included in our published work, but also alternative options leading to the implementation, providing both negative and positive experiences for future research.

Even though we present the design of security evaluation in the context of power grids, the implementation can be extended to other ICSs. By adding parsers and encoders of protocols used in different ICS networks and profiling characteristics reflecting system invariants in those environments, PFV can monitor and manipulate network interactions with real devices without any proprietary instrumentation. Based on those adjustments in PFV, the first security policy in DefRec that randomizes communications with virtual nodes is also extensible. The second security policy relies on the control theoretical model of power grids. By using the model of physical processes in other ICSs, we can also implement the security policy for different utility environments.

The main contributions of the paper are:

- **Network Communication.** We discuss three different implementations that we have explored, including network emulation, cloud environments, and using hardware switches. We use the results in these environments to discuss their pros and cons.
- **Power Grid Simulation.** Despite using the state-of-the-art power-system simulation, we discuss alternative implementation options from other research literature.
- **SDN Implementation.** After discussing a few open-source SDN controllers, we present evaluations related to the unique features that ONOS (the tool that we choose in our experiments) possess.

II. Background

In this section, we first present background information on power grids. Then we describe our design principles based on the threat model considered in this work.

A. Power Grid Basics

A power grid is an ICS, in which generators supply power to load demands over a wide geographical area. The generators and load demands are connected by transmission lines in a complex topology, often referred to as a *transmission network*. In the graphical representation of a transmission network (e.g., in Figure 2), we use a *bus* to represent a substation, where generators or load demands are deployed. In each bus and transmission line, we can have physical measurement data, including voltage, current, power consumption, and generation.

In Figure 1, we show a hierarchical communications network used by power grids. A control center uses an IP-based *control network* to retrieve data from substation devices periodically; this process is also known as *data acquisition*. Based on the retrieved data, the control center uses *state estimation* to determine the physical state of power grids.

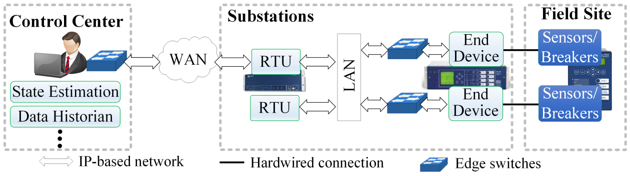


Fig. 1: **Hierarchical network infrastructure of power grids.**

For further discussion, we use the following definitions:

Definition 1. End Devices: Intelligent electronic devices (IED) located at the end of a communication path connecting the control center and substations.

End devices connect to sensors or circuit breakers through hardwired connections in their downstream communications. In their upstream communications, multiple end devices connect to a higher-level IED, e.g., RTUs (remote terminal units), which forwards information (e.g., aggregated measurements or commands) to/from the control center.

Definition 2. Edge Switches: Network switches located at the first or the last hop of a communication path that connects the control center and end devices.

B. Design Objective & Threat Model

Assumptions on Adversaries' Capability. We assume that adversaries penetrating a control network have limited knowledge of network configurations and physical data. We assume that adversaries can compromise any computing devices connected to the control network; however, they are not able to obtain knowledge of a whole power grid based on data collected by those compromised devices.

For clarity, we classify adversaries' attack capabilities into three types. ***Passive attacks*** monitor network traffic to obtain the knowledge of power grids' cyber-physical infrastructures. ***Proactive attacks*** achieve the same goal by using probing messages to trigger responses from real devices or virtual nodes. ***Active attacks*** directly manipulate network traffic, including dropping, delaying, compromising existing network packets, or injecting new packets. *Passive and proactive attacks are common techniques used by adversaries to perform reconnaissance, while active attacks are used to issue attack-concept operations to cause physical damage.*

DefRec's Objective. DefRec's objective is to disrupt and mislead adversaries' reconnaissance based on ***passive*** and ***proactive*** attacks, such that their ***active*** attacks become ineffective. Reconnaissance is a necessary step for "targeted attacks"

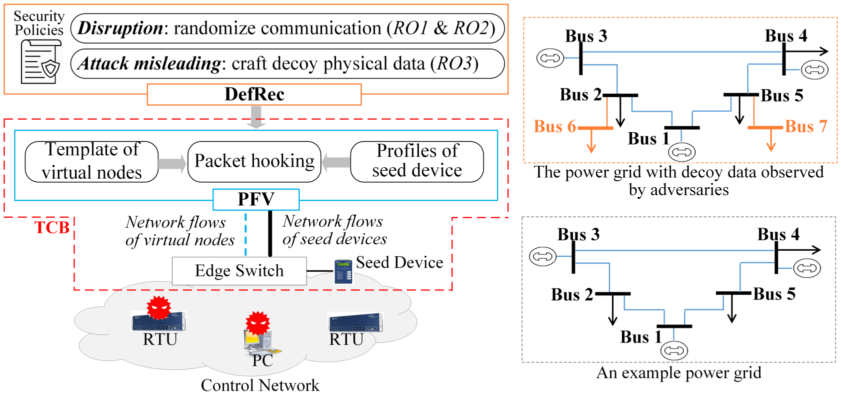

Security Policies
Disruption: randomize communication (*RO1* & *RO2*)
Attack misleading: craft decoy physical data (*RO3*)
DefRec
Template of virtual nodes
Packet hooking
Profiles of seed device
PFV
TCB
Network flows of virtual nodes
Network flows of seed devices
Edge Switch
Seed Device
RTU
PC
RTU
Control Network
Bus 3
Bus 4
Bus 2
Bus 5
Bus 6
Bus 1
Bus 7
The power grid with decoy data observed by adversaries
Bus 3
Bus 4
Bus 2
Bus 5
Bus 1
An example power grid

Fig. 2: **Design overview:** (i) PFV constructs virtual nodes that follow the actual implementations of seed devices and (ii) DefRec specifies security policies based on PFV to randomize communications and to craft decoy data for virtual nodes.

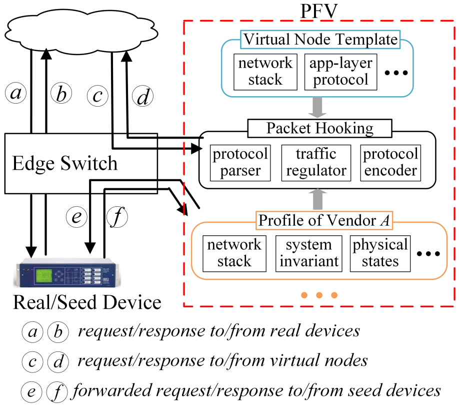


Fig. 3: **Components of PFV:** hook network interactions with real devices based on virtual nodes template and runtime profiling.

in ICSs [35], which are more frequently appearing in real utilities [17], [31] and are becoming a critical and damaging threat for ICSs, including power grids. In previous *targeted attacks*, adversaries have used in-depth knowledge of the target systems (obtained through reconnaissance) to stealthily deliver malicious attacks. We specify our anti-reconnaissance objectives (RO) as follows:

- ***RO1: for passive attacks on a control network***, we aim at significantly lengthening the time required by adversaries to successfully learn the knowledge of the control network.
- ***RO2: for proactive attacks on a control network***, we aim at revealing adversaries' existence with a high probability and isolating the compromised devices from the network.
- ***RO3: for physical knowledge obtained by passive or proactive attacks***, we aim at leveraging intelligently crafted decoy data to mislead adversaries into designing ineffective attacks.

III. Design Overview of DefRec based on PFV

In Figure 2, we present the design overview of DefRec, including the components of PFV and two security policies implemented on top of it. We position PFV as a complementary service to defense mechanisms, such as the design and implementation of the proposed security policies to disrupt and mislead adversaries' reconnaissance. We believe that PFV's functionality is not limited to DefRec, but can be used in other security solutions.

A. Components of PFV

The objective of PFV is to build lightweight virtual nodes that follow the implementation of network stack, system invariants, and physical state variations of real devices. In Figure 3, we present three components of PFV in detail.

Virtual Node Templates. We use these templates to contain basic configurations of the target control networks. For example, the templates include network stack information, such as IP addresses that can be assigned to virtual nodes, and the specification of application-layer protocols used to deliver physical data and control operations. Configurations stored in the templates are not specific to the context of a power grid.

Profile of Seed Devices. We select a few *end devices* as seed devices and profile three aspects of each.

- *The actual implementation of network stack* can be different from the protocol specification. For example, the DNP3 protocol used in power grids specifies 37 function codes [26], but the SEL 751A relay used in our experiment implements only 14 of them [50].
- *System invariants* refer to the characteristics that can be used to identify or fingerprint real devices, such as the latency of executing control commands [19].
- *Physical state variations* usually fall in a deterministic range for a specific power system, such as voltage magnitudes varying within the range of $\pm 5\%$ around a nominal value [21].

The device profile includes a range of variations for a certain property observed at runtime (e.g., command execution time) and the probability distribution over that range. We only need to select one seed device to represent each vendor or model, based on which we profile the runtime behavior. The device profile makes network flows from virtual nodes follow the probabilistic behavior of real devices, rather than replicating the same pattern. In this paper, DefRec focuses on the reconnaissance of power grids' applications. As such, we build virtual nodes following these three aspects of seed devices. When using DefRec for other applications, we can profile other application-specific knowledge.

Packet Hooking. This component uses SDN's programmability to hook network packets from seed devices; it tailors these packets based on the information from device profiles and virtual node templates, and uses the resultant packets as the network flows from virtual nodes.

As shown in Figure 3, when network packets reach an edge switch, a *protocol parser* extracts header information from the packets; a *traffic regulator* redirects them to a seed device if their destinations are virtual nodes (network flow ⓒ and ⓔ in Figure 3). Upon receiving the forwarded packets, the seed device responds (ⓕ). The responding packets serve two purposes: (i) building the device profile and (ii) serving as the input to a *protocol encoder*, which tailors the network packets according to the system invariants and physical state variations profiled before. The tailored packets, which are not deterministic but follow the same probabilistic properties of seed devices, are sent out by virtual nodes as the responses for the original request (ⓓ). The responses from seed devices reflect the actual implementation of the whole network stack, including the TCP/IP implementation. Consequently, virtual nodes are able to respond to lower layer network probing, e.g., ARP requests.

There are two requirements for the packet hooking component so that virtual nodes follow the runtime behavior of seed devices without causing physical damage and revealing the real physical state of a power grid.

- *Tailor application-layer payloads.* A *protocol encoder* tailors the application-layer payloads of network packets sent out by virtual nodes to (i) avoid leaking real physical data and (ii) introduce entropy (according to the device profile) to the data sent out by virtual nodes.
- *Redirect control operation without physical impact.* If a control operation (e.g., turn off a circuit breaker) reaches a virtual node, we redirect it to a seed device, connecting to a breaker that is already turned off, such that the operation introduces no physical impacts on the power grid.

B. Security Policies based on PFV

Based on PFV, DefRec specifies security policies to achieve anti-reconnaissance objectives defined in Section II-B. Specifically, to achieve RO1 and RO2 related to the reconnaissance of power grid networks, we proposed a *disruption* policy that randomizes network communications. To achieve RO3, we propose an *attack-misleading* policy that crafts decoy data for virtual nodes. Consequently, adversaries collect misleading information about a power grid different from the one under protection (e.g., the power grid with virtual nodes highlighted in orange in Figure 2). Based on such information, adversaries design ineffective attacks.

Disruption Policy. Adversaries can use *passive* attacks to stealthily identify real physical devices, if we issue requests only to real devices (such as ⓐ in Figure 3). To disrupt *passive* attacks in RO1, we randomize network packets by issuing requests to (i) randomly selected virtual nodes and (ii) randomly selected real devices. We refer the set of randomly selected virtual nodes that we sent requests as "accessible virtual nodes" and accessing them will not raise alerts. These randomization activities further balance the number of requests sent to real devices and requests to accessible virtual nodes. They prevent adversaries from identifying real devices based on the biased distribution of the number of network requests sent to the real devices, significantly increasing the time for adversaries to stealthily identify real devices.

Adversaries can perform *proactive attacks* to probe physical devices or virtual nodes and use the responses to identify real devices. To disrupt *proactive* attacks in RO2, we design a *probabilistic dropping protocol*, introducing randomness when responding to probing of virtual nodes, revealing adversaries' existence with a high probability while reducing the information an adversary can learn. Probing *accessible* virtual nodes always results in responses; the proposed protocol handles proactive probing destined to other "inaccessible" virtual nodes, which is suspicious, as the requests are neither from legitimate applications nor DefRec. However, directly isolating the probing machine can immediately reveal the identity of an inaccessible virtual node. The details of the protocol can be found in [36].

Attack-Misleading Policy In addition to disrupting the reconnaissance on network configurations, we further use the "attack-misleading" policy to achieve RO3: disrupting adversaries' reconnaissance on power grids' physical infrastructure by providing misleading knowledge. The core of this policy is an algorithm that crafts decoy data for virtual nodes. Without careful design, data piggybacked by network packets from virtual nodes can still allow adversaries to learn grids' physical knowledge, e.g., physical topology and measurements.

The decoy data construction algorithm takes the following inputs: the states of real physical devices, the number of virtual nodes, and the topology graph representing their connectivity. We put those inputs into a power system's mathematical representation to obtain decoy states or decoy data for virtual nodes, which follow the normal variations observed in real physical devices. For example, in Figure 2, we add four virtual nodes to represent two substations with load units (Bus 6 and Bus 7) and two transmission lines connecting them to the power system; we use the algorithm to obtain decoy data, such as the impedance of the added lines and power consumption in the added buses. We can implement the decoy data construction algorithm in any power grid analysis tools running on general-purpose computers, such as MATPOWER that we use for our evaluations [59].

There are two requirements for constructing decoy data: (i) mislead adversaries into designing ineffective strategies; and (ii) follow the physical model of power grids to avoid suspicions from adversaries. The algorithm to craft decoy data varies according to the attack that we attempt to mislead. For example, in our recent work [36], we focus on crafting decoy data for false data injection attacks (FDIAs), which is one of the most critical attacks in power grids with the aim to downgrade the accuracy of state estimation and the performance of many power grid applications. In [36], we presented the details on how to leverage the well-established theoretical model in FDIAs to formally prove the effectiveness of decoy data.

IV. IMPLEMENTATION

To evaluate security properties and performance overhead of PFV and DefRec, we implemented a cyber-physical testbed. The implementation needed to observe the following requirements [7]: (i) representing real-world operations in power grids; (ii) scalable to a reasonable size; and (iii) cost-effective such that other researchers can efficiently reproduce experiments. As shown in Figure 4, the testbed includes four major parts: implementations of PFV and DefRec, communications networks, simulations of power grids, and physical IEDs used as end devices. The first part is for the research design of DefRec discussed in [36] and in Section III, while the remaining three parts are generic and can be reused in other designs related to ICSs and communications networks. In each of the following subsections, we present not only the implementation used in our published work, but also preliminary explorations leading to the ultimate setups.

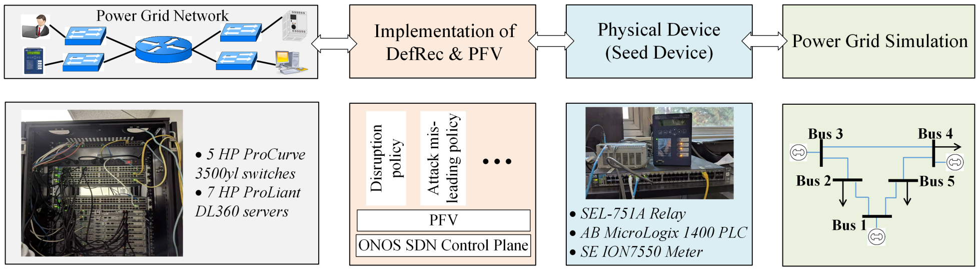


Fig. 4: **Cyber-physical testbed for evaluation.**

TABLE I: **Evaluation cases.** We include the number of nodes for each network in parentheses.

Case	*Power Grid Simulation*	*Network (# of nodes)*
1	IEEE 24-bus	Datax (11)
2	IEEE 30-bus	Abilene (22)
3	RTS96 73-bus	Hurricane (30)
4	IEEE 118-bus	Chinanet (56)
5	Poland 406-bus	Cesnet (78)
6	Poland 1153-bus	Forthnet (124)

A. Implementation of PFV and DefRec

PFV does not require a dedicated virtual environment; we implemented it as an SDN application in ONOS [8], including around 1,500 lines of code (LOC). Based on PFV, we also implemented DefRec's disruption policy in ONOS, using less than 200 LOC. For DefRec's attack-misleading policy, we implemented the decoy data construction algorithm by using MATPOWER, an open-source MATLAB toolbox [59]; the implementation used around 400 LOC. All implementations were carried out on a 64-bit Ubuntu 18.04, deployed in a workstation with four Intel Xeon 2.8 GHz processors and 16 GB RAM.

There are many network manipulation platforms based on SDN, such as POX [45], RYU [48], Floodlight [18], and ONOS [8]. The former two are written in Python while the latter two controllers are written in Java. All these controllers follow a similar implementation structure; they run a core service, which can load applications written by developers following their provided APIs. We chose to use ONOS as it provides rich functionalities; a unique one is that ONOS allows building a cloud infrastructure, in which multiple SDN controller instances are distributed in different locations and exchange information through an extension of Atomix framework.

Repeatability. Repeating PFV's implementation requires to load an ONOS application. Because the aforementioned SDN controllers follow similar APIs, other researchers can implement the same network manipulation logic in a different controller.

B. communications networks

We classify current implementation of communications networks found in existing research work into three categories, which are based on emulations, cloud environments, and hardware network switches. Each category has its own pros and cons, which can be leveraged at different stages of research design.

In the last two decades, the emulation-based implementation evolves from simulators such as NS2 [40] to real time emulators such as Mininet [37]. Using Mininet as an example, we can easily leverage the emulator to build a network of arbitrary topology in a single desktop. Communications in the emulated network can be established based on applications installed in the desktop at which the Mininet is running. These advantages allow us to quickly instantiate a network and obtain a first-hand evaluation of research design, identifying any design flaws at an early stage. However, major disadvantages stem from the fact that all communications in the emulated network go through a loopback interface of the desktop. As one emulates heavy communications or a large-scale network, the limited bandwidth can introduce severe traffic congestion. Meanwhile, all emulated network nodes follow the same clock, which can make the evaluation of research work related to synchronization in distributed systems less convincing [23].

Another option to implement communications networks is to use a cloud computing environment. Generally, we can find two types of cloud environments. The first type is the cloud environment specifically deployed to run network experiments, especially the one involving advanced network technology such as SDN or network function virtualization (NFV). The representative examples include NSF Geni and Fabric platforms [6], [9]; these platforms leverage a technique called "network slicing" to provide spare network resources in commercial networks to research projects [2], [51]. The second type is the cloud environment for computing tasks, such as Amazon AWS and Google Cloud. In this environment, we can leverage open-source virtual switches such as Open vSwitch [43] to change a computing node into a network processing node; the computing node needs to be equipped with additional Ethernet interfaces to allow this change. The major advantage of using cloud environments is that they allow users to allocate network and computing resources at different physical locations, constructing a realistic wide-area network. However, using cloud environments can also suffer from scalability. Based on our experiences, NSF testbed is available for most US research and education institutions; the network and computing resources are shared by different projects, making it challenging to reserve sufficient resources to build a large-scale network. The commercial cloud environment can be cost-prohibitive, especially owning to the need of a large number of Ethernet interfaces.

The last option of implementing communications networks based on hardware switches can meet the trade-off between using emulations and using cloud environments. Specifically, in a lab environment, we implemented a large-scale network (up to 124 nodes) by building logical switches on top of five HP ProCurve 3500yl switches. Each physical switch has 48 ports; we grouped certain switch ports into different VLANs (virtual local area networks), according to given network topology. Each VLAN behaves like a logical switch. To avoid the default broadcast communication within each VLAN (or a logical switch), we need to set up forwarding/routing rules for the ports grouped in each VLAN, which could be a tedious task requiring us to manually work with proprietary configuration interfaces. In our experiments, instead of using proprietary interfaces, we use the API supported by SDN controllers, e.g., *ofp_flow_mod* API, to automatically configure forwarding rules in each VLAN. In addition, the communication between two VLANs corresponds to the logical link connecting two logical switches, constructed by connecting physical ports belonging to two VLANs via Ethernet cables. In addition to building logical switches in physical switches, we used lightweight Docker containers to build computing nodes as end hosts of the constructed network in seven HP ProLiant DL3600 servers. We extended each server by deploying five PCI 4-port Ethernet adapters. To manipulate network flows for each individual end host, we associated each end host implemented within a Docker container with a separate Ethernet interface. Under this setup, we could include up to 140 end hosts ($7 \times 4 \times 5$). Then, we manually connected end hosts with the corresponding logical switches through Ethernet cables.

Compared to automatic "network slicing," we need to form VLANs and connect them manually. However, owning physical switches allows us to dedicate network resources to our experiments and to perform many custom configurations, the capability that is challenging to obtain in a cloud environment. Also, purchasing switches is acceptable for many research groups. For example, we purchased refurbished HP ProCurve 3500yl switches with the unit price under 200 dollars in Amazon. Due to this easy deployment, we find many research works related to SDN adopt this implementation option [58].

Regardless of implementation options, we always need a dataset to implement networks of different topology that can represent the real-world communications networks. There are two common datasets used by network communities, i.e., TopologyZoo [30] and Rocketfuel [53]. These two projects relied on network scanning to estimate the topology of real networks managed by different Internet Service Providers (ISPs). However, there are no communications networks used by ICSs included in the dataset, as the topology of those networks may help adversaries reduce the reconnaissance effort significantly.

Repeatability. All three implementation options of communications networks can be repeated. Other researchers can build an emulated network or a network in a cloud environment by parsing the file used to store a network topology, e.g., "gml" file used by the TopologyZoo dataset. Then following appropriate APIs to connect network nodes and links results in the corresponding network. Repeating the implementation on physical switches requires slightly more effort. Building VLANs to group physical ports in a switch is proprietary and requires manual effort. However, we automated the setup of forwarding rules in each VLAN in an SDN application, which can be loaded and run in ONOS.

C. Physical Devices

The best way to resemble end devices in a real utility environment is to directly use the appropriate IEDs suitable for lab environments. In our experiments, we have used IEDs from three different vendors as end devices: a Schweitzer Engineering Laboratories (SEL) 751A feeder protection relay [50], a Allen Bradley (AB) MicroLogix 1400 PLC [1], and a Schneider Electric (SE) ION7550 power meter [49]. Since we were using these devices for communications based on the DNP3 protocol, we have selected them by going through certification equipment listed in DNP3's official website [26]. In the process of identifying the appropriate model of IEDs for our experiments, we have encountered two challenges: (i) some equipment models are not for sale in the U.S. and (ii) some vendors can only perform business with utilities companies, not with academic institutions. To overcome those challenges and save expenses, we purchased refurbished equipment from eBay, which, in some cases, can make us purchase proprietary companion software of the devices separately.

Because IEDs usually include proprietary design and implementations, their unit price can be high, making it challenging to purchase and use a large number of different models. To resolve these issues, we have found current research work in different areas, such as robotic vehicles [11], water treatment plants [5], and power grids [19], leveraging the simulations of physical processes as end devices. This is a feasible alternative approach, because IEDs usually perform deterministic functionality that can be accurately simulated in general-purpose computing devices.

IEDs often play the role of "slave" machines, which passively perform operations based on requests from a master machine. To implement a DNP3 master to communicate with those IEDs, we used the well-maintained openDNP3 library [41]. However, we found several function codes used by the aforementioned IEDs were not implemented in the OpenDNP3 library. After contacting the developers, we learned that those function codes are deprecated and no longer supported by most upcoming products. By going through some other protocols used in power grids, such as Modbus [27], IEC 61850 [24], Ethernet/IP [16], and DLMS [14], we find that most of them have the corresponding open-source implementations [29], [34], [42].

Repeatability. Even though physical IEDs cannot be easily shared, we can share the configurations that define the application layer format of the DNP3 protocol used to deliver measurement data.

D. Power Grid Simulations

For many CPSs, we can monitor and analyze their physical processes in a single device. In a power grid, however, the physical process is represented by correlating operations of a large number of IEDs in distributed locations, making it challenging to build a real power grid for research experiments. Even though there are a few state-of-the-art laboratories building a small-scale real power system, the scalable and cost-effective experiment methods are to use simulations on the case built based on real power grids.

Instead of using real power grids, start-of-the-art simulations are common techniques to study power systems' steady states. For example, we have used MATPOWER, an open-source MATLAB toolbox, to simulate six different power grids [59], shown in Table I. MATPOWER includes popular power grid cases as benchmarks, such as IEEE test cases as well as cases representing national transmission networks in Poland and France. Even though these test cases follow the topology and configuration of real power grids, they still show two shortcomings, reducing the repeatability of experiments. First, some cases are outdated, and there are some missing parameters. For example, some IEEE test cases, including 57-bus, 114-bus, and 300-bus cases, do not include power flow limit on transmission lines (the maximum amount of power that a transmission line can deliver). Second, these cases only include a snapshot of operational conditions at a certain timestamp; they do not directly include variations of operational conditions.

To simulate normal variations of operations in the simulated power grids, we developed a benchmark profile based on data from real utilities [15], [47]. We extracted one month of real data on power generation and calculated the ratio between actual data value at each timestamp to the peak value of that month. For each simulated system, we randomly selected power generators and load units, adjusting measurements for each unit according to the benchmark. In addition, in recent years, some newly developed evaluation cases, which directly include normal variations of operational conditions, become popular and are widely used in power engineering communities [54].

In recent years, an approach called hardware-in-the-loop (HIL) emerges as a popular option, achieving a trade-off between the construction of real power grids and software simulation. HIL seamlessly integrates physical hardware (e.g., the real IEDs used in our experiment) and software simulations, accurately representing a real operational environment and evaluating the performance and reliability of power systems in great fidelity. At this stage, there are two companies, i.e., OPAL-RT technologies and RTDS technologies, achieving commercial success on providing such high-fidelity simulations.

Repeatability. The power grid cases used in our experiments serve as benchmarks in power engineering community. Their parameters are free to download from most power flow analysis tools.

V. Evaluations

In [36], we performed security and performance evaluations for both PFV and security policies in DefRec. In this section, we provide more details on experiments for those evaluations and present results that are regarded as "primitive," leading to the ultimate results. The objective is to present experiences that we learn through those primitive evaluations. As shown in Section IV, there usually exist multiple options of implementations. In some aspects of the experiments, we have tried several options, whose results are presented. Due to resource limitations, for implementations used in other work, we direct readers to the corresponding cited literature.

A. Experiment Setup

Hypothesis. We used the DNP3 protocol as the communication protocol used by networks in ICSs. This protocol can represent common functionalities found in other ICS protocols, e.g., Modbus [27], IEC 61850 [24], Ethernet/IP [16], and DLMS [14]. Also, the security and performance evaluations focused on the lower layer characteristics of networks; we believe that the evaluation results can be representative of current ICS networks.

Methodology. For each evaluation case listed in Table I, we constructed the corresponding network. In this network, we selected a computing node as a DNP3 master to issue requests, including data acquisition retrieving analog data and control operations opening/closing breakers, to DNP3 slaves. The responded data and the control operations are obtained from the simulation of the corresponding power grid case.

We built DNP3 slaves differently. To evaluate the effective-ness of PFV, we used the three IEDs as the DNP3 slaves. To evaluate the performance of PFV, we used Docker containers to simulate a large number of end hosts in the network, each implementing a DNP3 slave based on the OpenDNP3 library.

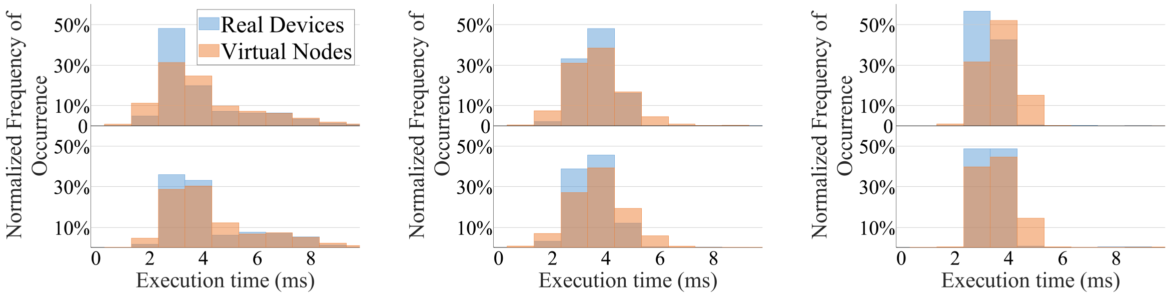
Data Collection and Analysis. For each request supported by the OpenDNP3 library (see Table II for details), we repeated the request-response communication for 500 times with and without PFV/DefRec enabled. We run *Tcpdump* on the DNP3 master to store all network communication locally, e.g., in pcap files, which were processed by the Zeek network analyzer [44]. All experiments lasted a week, and we collected two sets of traces for each network with and without DefRec enabled: (i) the trace of more than 400,000 DNP3 packets that represent the data acquisitions and control operations, and (ii) the trace of more than 19,000 OpenFlow packets that represent the outbound traffic of virtual nodes.

Repeatability. Repeating those experiments involves developing a DNP3 master and DNP3 slaves. We developed them by modifying the example master and slave modules included in the OpenDNP3 library, without any changes in its core libraries or APIs. The major modification included the setup of periodically data acquisition and the format to issue various control operations supported by the physical IEDs used in our experiments. Our analyses focused on common network performance metrics, e.g., throughput and latency. We used Zeek to perform such analyses, which can be performed by most network analysis tools, e.g., Wireshark or Tcpdump.

B. *Effectiveness of PFV*

In this section, we present additional details of evaluation on the effectiveness of PFV, demonstrating the challenges due to the proprietary implementation in different IED models. Specifically, we applied fingerprinting methods proposed for ICSs on both real physical devices and virtual nodes. As shown in [19], the time to execute commands in ICS devices is an effective system invariant to identify device types and models. Based on the methods presented in [19], we measured the difference between the timestamp in the response carrying measurement data and the timestamp in the corresponding TCP acknowledgment, to accurately reveal execution times in real devices or virtual nodes.

Our original plan was to measure the execution time of different control commands for each IEDs. Even though we can communicate each IED by using the DNP3 protocol, it turns out that each IED supported different function codes of DNP3 requests. Consequently, we grouped those function codes into two categories, i.e., data acquisition and control, as shown in Table II. All three IEDs supported periodic data acquisition by sending reading requests. However, the data format in the responses has small differences. In the response from SEL 751A, measurement data are stored as 32-bit integers, while the responses from the other two IEDs attach a property flag with each 32-bit integer to store accessory information, e.g., the time when the data is sampled [26]. In addition to the data acquisition, it is worthwhile to note that



(a) SEL 751A

(b) AB MicroLogix 1400

(c) SE ION 7550

Fig. 5: **PDF** (y-axis) **of execution time** (x-axis) of data acquisition (at top) and control operations (at bottom) for three IEDs.

different *control* operations interact with IEDs differently. For example, the OPERATE function code can trigger the change of physical process, but WARM RESTART function code only affects the software stack of an IED. In future work, we will further investigate how those different interactions with IEDs can affect the effectiveness of device fingerprinting.

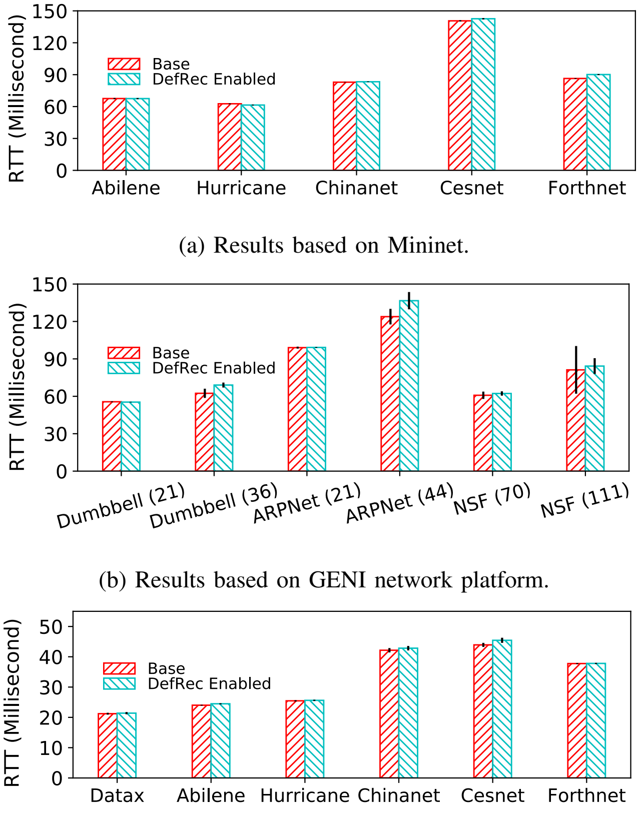
TABLE II: **Group function codes supported by different IEDs into two types: data acquisition and control.**

Supported Function Codes	***Devices***		
	SEL 751A	AB 1400	ION 7550
Data Acquisition			
READ	32-bit integer without flags	32-bit integer with flags	32-bit integer with flags
Control			
WRITE	✓		
SELECT	✓	✓	✓
OPERATE	✓	✓	✓
DIRECT OPERATE	✓	✓	✓
COLD RESTART	✓		✓
WARM RESTART	✓	✓	✓
ENABLE UNSOLICITED	✓		
DISABLE UNSOLICITED	✓		

Based on the coarse-grained classification, we evaluated the effectiveness of PFV, whose results are presented in Figure 5. In this figure, we show the probability density functions (PDFs) of execution time measured for both data acquisition and control operations. We can see that PDF patterns vary in different operation types and devices. Consequently, data acquisitions, which mainly access the storage space of an IED, can present very different characteristics compared to control operations, which can initiate different physical operations in an IED. Based on these experiments, we can demonstrate that virtual nodes can follow the communication patterns of real devices.

C. Performance Overhead in Different Network Implementations

The disruption policy specified in DefRec introduces additional network traffic from virtual nodes. We conducted experiments to understand the impact of the injected packets on the performance of existing networks. Specifically, we measured and compared round-trip time (RTT) of all data acquisitions and control operations with and without DefRec enabled. In Figure 6, we specify in the x-axis different network topology. In the y-axis, we present the measured RTT in milliseconds (ms) with 95% confidence interval. In these figures, we specify the name of networks from the TopologyZoo dataset [30]. When we performed experiments in the NSF Geni testbed, we have adjusted the number of hosts of networks (included in the parenthesis).



(a) Results based on Mininet.

(b) Results based on GENI network platform.

(b) Results based on our lab environment.

Fig. 6: **Compare RTTs measured in different network implementations.**

In Figure 6, we present selected primitive results based on three options to implement communications networks, which are based on Mininet emulation, a cloud environment (e.g., the NSF GENI network platform), and hardware network switches. In all three environments, we can observe that the overhead due to DefRec is small. However, those results reveal different characteristics of the corresponding network implementation. While using a network emulation, the bottleneck of the size of emulated networks depended on the bandwidth of the physical Ethernet interface of the desktop at which we run the Mininet emulation. As we increased the size of communications networks, more network packets triggered re-transmissions. Re-transmission can introduce challenges to calculate RTT, because it is difficult to determine the specific request, among all re-transmitted requests, that reaches end devices and triggers the response. Due to this reason, we needed to run experiments for an extended period to obtain a sufficient number of communications that include no re-transmissions. Even though building emulated networks took a short time, running experiments took a much longer time than experiments based on other implementations. This experience has demonstrated that using Mininet can benefit small scale experiments to identify early-stage design flaws; it is not suitable for large-scale experiments that can represent real-world networks.

Using cloud environments, such as NSF Geni testbed, introduced bigger variations in RTT, especially when we used a large-scale network for evaluations. The reason is that we were sharing computing resources in the GENI testbed with other projects, as we were using hardware switches, instead of virtual switches. As the scale of the networks increased, we could not allocate sufficient computing resources for network switches and end hosts. We have observed dropped packets due to software errors and hardware errors. Because many network resources required more complicated maintenance, it was usually difficult to reserve sufficient resources to simulate a network with more than 100 hardware switches.

In [36], we used the results from the experiments performed based on hardware switches in our lab environment. Compared to the implementations on Mininet and cloud environments, the results are more consistent and stable, with a tolerable overhead of manual configurations. By comparing Figure 6(a) and Figure 6(c), we can see that the same network experiences very different RTTs in the lab environment. TopologyZoo dataset specifies the latency of communication links in each network. While Mininet and the GENI testbed provide API for us to directly specify the latency of a communication link, the HP switches used in our lab lack such APIs. In those HP switches, we could only reduce the bandwidth of physical ports to increase the latency of traffic going through those ports. Since there is a minimum bandwidth required for each port, we could not accurately configure the latency as indicated in the dataset but with some adjustments.

D. Performance Overhead in SDN Controllers

To evaluate PFV's capability to manipulate network packets, we mainly focused on the southbound goodput of ONOS SDN controller in [36]. The *southbound* communications refer to the interactions between an SDN controller and network switches to which this controller connects.

ONOS provides an interface to deploy multiple SDN controller instances in different locations and to share information among them. The inter-controller communication is often referred to as *northbound* communication. In DefRec, inter-controller communications do not play a critical role, which were left out in [36]; we present the results here for the completeness of experiments.

To achieve inter-controller communications, we built an ONOS cluster including SDN controller instances deployed in the NSF Geni network platform. The ONOS cluster uses an extension of Atomix framework for the included instances to exchange information. The physical machines hosting the SDN controller instances (deployed in virtual machines) were at three different physical locations. We allocated publicly reroutable IP addresses to all SDN controllers, so they communicated with each other by public Internet, not an air-gaped network.

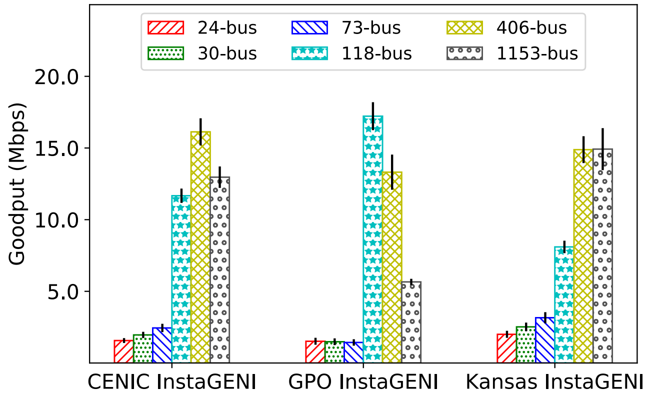


Fig. 7: **The goodput of inter-controller communication** at three instaGENI sites in six evaluation cases (error bars show 95% confidence interval).

In Figure 7, we present the average goodput (in megabits per second or Mbps) of the communications among distributed controllers at different locations. On the *x*-axis, we specify the three different instaGENI sites [20]. The number of data exchanged among different SDN controllers, which is specified in Table II of [36], varied with the size of simulated power grids. Consequently, we evaluate the goodput for the six power grids, shown in different bar patterns. We use the *y*-axis to include the average goodput and the error bars to show 95% confidence interval.

In our experiment, there are three major factors affecting the goodput of inter-controller communications: the allocated network bandwidth, the configuration of Atomix protocol, and the amount of data. From the figure, we can see that the goodput observed in the first three evaluation cases are smaller, around 2 Mbps, compared to the goodput observed in the latter three cases. Actually, we observed similar round trip times of the inter-controller communications in all six cases. Consequently, with the small amount of exchanged data, the communication bandwidth is not fully exploited in the first three evaluation cases. After exchanging more data, we can see the steady increase of the goodput in all three instaGENI sites, varying from 5 Mbps to 15 Mbps. In GENI testbed, we could not control the bandwidth for public network communications, which were allocated by the local IT infrastructure.

Focusing on the evaluation of each individual power grid, we find that the goodput of the controllers installed in different instaGENI sites appear strange at first sight, as the goodput varies in different sites even though the same number of data is exchanged. In addition to the limited network bandwidth and physical location of different sites, we find that the detailed implementation of the Atomix protocol used by the ONOS cluster also contributes to the result. By observing the network trace, we found that the controllers at GPO InstaGENI site often played the role of data subscribers (or consumers) and thus did not send a large number of packets to controllers at other sites. On the contrary, the controllers deployed at the other two sites played the role of both data subscribers and producers, and we observed more network packets.

VI. Related Work

Instead of repeating the comparisons between DefRec and previous related research, we focus on the implementation and evaluations of the related work in this section.

Network Function Virtualization (NFV). NFV is an emerging technology to virtualize network nodes according to specific functionality, such as load balancing and access control [39]. NFV is not necessarily dependent on SDN, but SDN's network programmability and visibility can significantly benefit its design. Recent work has applied NFV to improve performance and flexibility of security designs. Li et al. propose to use NFV to virtualize detection logic of network IDSs, allowing efficient and flexible state sharing and resource migration [33]. Deng et al. leverage SDN and NFV to overcome resource limitations of hardware-based firewall applications, enabling elastic and scalable access control for virtual computing environments [13]. In [13], [33], since the authors focused on the evaluations of NFV, which were implemented with the help of SDN or network intrusion detection systems, they used custom virtual machine environments, e.g., Zen, to implement the evaluation environment, without complicated implementations on real communications networks.

Moving Target Defense (MTD) in ICSs. Traditional MTD approaches disrupt adversaries by randomly changing system and network configurations, e.g., IP addresses and port numbers [4], [28]. Some recent work leverages similar designs to disrupt control operations in ICSs. Rahman et al. randomly change the set of physical data used for power system analysis, attempting to remove some compromised data and to reduce the effectiveness of FDIAs [46]. Another group of MTD approaches intentionally disrupt physical processes in an ICS and use deviations from expected consequences to detect attacks [3], [38]. Because those approaches used physical perturbations, the evaluations on their impact were integrated into the simulation of power grids or control systems, without any implementation related to the cyber infrastructure used by power grids.

Honeypots for ICSs. Honeypots or honeynets interact with adversaries with simulated network packets. Several honeypot projects aim at building separate computing or network environments to trace adversaries' activities on ICS devices, e.g., PLCs [10], [12], [56]. Han et al. further propose to use SDN to automate interactions with adversaries [22]. Those ICS honeypots can mimic a cyberinfrastructure of an ICS. However, in their constructed networks, the honeypots lack supports for constructing meaningful application-layer payloads, e.g., measurements exchanged between ICS devices.

Instead of mimicking and simulating network packets, we design PFV, a completely different technique, by virtualizing physical devices. PFV is not a honeypot for ICSs: it does not require interactions with adversaries to disrupt their reconnaissance. Adversaries that passively monitor network packets can be significantly delayed; they can end up using decoy data to design damage-free attack strategies.

Increasing SDN's resilience. Rich capabilities provided by SDN also make SDN controllers a popular target of attacks [52], [55]; therefore, SDN-based approaches require more complex protections. To evaluate the work related to attacks that aim to compromise SDN's control plane, many researchers implemented a realistic SDN controller as well as communications networks. However, because those research works usually dealt with representative attack cases, they usually used a few network switches to construct a small-scale network; there was no necessity to implement a large-scale wide-area communications network in their evaluations.

VII. Lessons Learned

In this section, we briefly summarize what we learned to perform experiments:

- ***Cyber infrastructure.*** We have tried three different implementation options of building communications networks to evaluate research design at different stages. Using Mininet emulation is easy to set up and can be used to identify early-stage design flaws. Using cloud environments can resemble a realistic wide-area network, with the limitation of available network resources. Using hardware switches to construct VLANs allows great flexibility, requiring some manual effort.
- ***Physical infrastructure.*** We have used benchmark cases widely accepted by power engineering community to perform simulation close to real environments. The conventional cases, which have been used for decades, require some manual configuration effort, including random adjust-ment of the normal operational conditions.
- ***Network engineering.*** To implement research design, we have integrated network engineering into an application loadable to common SDN controllers. Using network manipulation to design and implement moving target defense mechanisms is a new research approach, without instrumentation on both server and client ends of communications.

For future researchers, we propose the following suggestions to improve evaluation quality:

- Even though the DNP3 protocol is widely used at this stage, using newly designed protocols such as Ethernet/IP allows researchers to purchase new IED models for experiments, which can be more easily obtained at reduced unit prices.
- Some new network switches allow configurations on the latency and throughput of an individual physical port. With this feature, researchers will be able to configure the latency within a VLAN if they choose to construct logical switches in a hardware switch. Consequently, the air-gaped networks built on physical switches can better resemble realistic communications networks.
- The size of traditional power grid cases is usually limited to a few hundreds of substations. With more large-scale power grid cases (e.g., up to 10,000 substations) becoming

available for academics, we recommend using them in simulations to better evaluate the scalability of the proposed research idea.

VIII. Conclusion

In our recent work, we proposed the concept of PFV, which hooks network interactions with real devices to build virtual nodes, following the actual implementations of network stacks, system invariants, and physical state variations of the real devices. Based on PFV, DefRec specifies two security policies, randomizing communications and crafting decoy data for virtual nodes, to disrupt adversaries' reconnaissance of power grids' cyber-physical infrastructures. In this work, we present additional details on various options related to the implementation of a cyber-physical testbed, evaluating the effectiveness and efficiency of DefRec. We aim to present the pros and cons of different implementation options with "primitive" results, helping other researchers in similar fields.

In future work, we will focus on including the close coupling of implementation of cyber and physical components in the testbed, providing increasing fidelity to simulate the runtime behavior of ICSs.

Acknowledgment

This material is based upon work partially supported by the National Science Foundation under Award No. CNS-1850377 and the National Science Foundation under Award No. CNS-1717313. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

[1] "Allen Bradley MicroLogix 1400 programmable logic controller systems," [Online] Available at: https://ab.rockwellautomation.com/Programmable-Controllers/MicroLogix-1400.

[2] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck, "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions," *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2429–2453, 2018.

[3] M. Q. Ali and E. Al-Shaer, "Randomization-based intrusion detection system for advanced metering infrastructure," *ACM Transactions on Information and System Security*, vol. 18, no. 2, pp. 1–30, 2015.

[4] S. Antonatos, P. Akritidis, E. Markatos, and K. Anagnostakis, "Defending against hitlist worms using network address space randomization," *Computer Networks*, vol. 51, no. 12, pp. 3471–3490, 2007.

[5] W. Aoudi, M. Iturbe, and M. Almgren, "Truth will out: Departure-based process-level detection of stealthy attacks on control systems," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 817–831. [Online]. Available: https://doi.org/10.1145/3243734.3243781

[6] I. Baldin, A. Nikolich, J. Griffioen, I. I. S. Monga, K. Wang, T. Lehman, and P. Ruth, "Fabric: A national-scale programmable experimental network infrastructure," *IEEE Internet Computing*, vol. 23, no. 6, pp. 38–47, 2019.

[7] D. Balenson, L. Tinnel, and T. Benzel, "Cybersecurity experimentation of the future (cef): Catalyzing a new generation of experimental cybersecurity research," National Science Foundation (NSF), Tech. Rep., July 2015.

[8] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and others, "ONOS: towards an open, distributed SDN OS," in *Proceedings of the 3rd workshop on Hot topics in software defined networking*. ACM, 2014, pp. 1–6.

[9] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "Geni: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5 – 23, 2014, special issue on Future Internet Testbeds – Part I.

[10] D. I. Buza, F. Juhász, G. Miru, M. Félegyházi, and T. Holczer, "CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot," in *Smart Grid Security*, J. Cuellar, Ed. Springer International Publishing, 2014, pp. 181–192.

[11] J. Choi, H. Jeoung, J. Kim, Y. Ko, W. Jung, H. Kim, and J. Kim, "Detecting and identifying faulty iot devices in smart home with context extraction," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2018, pp. 610–621.

[12] "Conpot: low interactive server side industrial control systems honeypot," [Online] Available at: http://conpot.org/.

[13] J. Deng, H. Li, H. Hu, K.-C. Wang, G.-J. Ahn, Z. Zhao, and W. Han, "On the safety and efficiency of virtual firewall elasticity control," in *Proceedings of the 2017 Annual Network and Distributed System Security Symposium (NDSS 2017)*, 2 2017.

[14] "DLMS: device language message specification," [Online] Available at: https://www.dlms.com/.

[15] "U.S. energy information administration - EIA - independent statistics and analysis," [Online] Available at: https://www.eia.gov/opendata/.

[16] "Ethernet/IP Overview," [Online] Available at: https://www.odva.org/Technology-Standards/EtherNet-IP/Overview.

[17] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[18] "Floodlight SDN Openflow Controller," [Online] Available at: https://github.com/floodlight/floodlight.

[19] 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 2016 Network and Distributed System Security Symposium*. Internet Society, 2016.

[20] "GeniAggregate - GENI: geni," 2019, [Online] Available at: https://groups.geni.net/geni/wiki/GeniAggregate.

[21] J. D. Glover, M. S. Sarma, and T. Overbye, *Power System Analysis and Design, SI Version*. Cengage Learning, 2012.

[22] W. Han, Z. Zhao, A. Doupé, and G.-J. Ahn, "Honeymix: Toward sdn-based intelligent honeynet," in *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization*, ser. SDN-NFV Security '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1–6. [Online]. Available: https://doi.org/10.1145/2876019.2876022

[23] N. Handigol, B. Heller, V. Jeyakumar, B. Lantz, and N. McKeown, "Mininet performance fidelity benchmarks," Stanford University, Tech. Rep., October 2012.

[24] C. Hoga and G. Wong, "IEC 61850: open communication in practice in substations," in *IEEE PES Power Systems Conference and Exposition*, vol. 2, Oct 2004, pp. 618–623.

[25] A. Houmansadr, C. Brubaker, and V. Shmatikov, "The parrot is dead: Observing unobservable network communications," in *2013 IEEE Symposium on Security and Privacy*, May 2013, pp. 65–79.

[26] "IEEE standard for electric power systems communications-distributed network protocol (dnp3)," pp. 1–821, Oct 2012, [Online] Available at https://standards.ieee.org/standard/1815-2012.html.

[27] "Modbus messaging on tcp/ip implementation guide v1b," Oct 2006, [Online] Available at: http://www.modbus.org/docs/Modbus_Messaging_Implementation_Guide_V1_0b.pdf.

[28] J. H. Jafarian, E. Al-Shaer, and Q. Duan, "An effective address mutation approach for disrupting reconnaissance attacks," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 12, pp. 2562–2577, Dec 2015.

[29] "jDLMS: a Java implementation of the DLMS/COSEM protocol," [Online] Available at: https://www.openmuc.org/dlms-cosem/files/jdlms-doc.pdf.

[30] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.

[31] D. Lee, "Ukraine power cut was cyber-attack," Jan 2017, [Online] Available at: https://www.bbc.com/news/technology-38573074.

[32] R. M. Lee, M. J. Assante, and T. Conway, "Analysis of the cyber attack on the ukrainian power grid," SANS and E-ISAC, Tech. Rep., March 2016.

[33] H. Li, H. Hu, G. Gu, G.-J. Ahn, and F. Zhang, "vnids: Towards elastic security with safe and efficient virtualization of network intrusion detection systems," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '18. New York, NY, USA: ACM, 2018, pp. 17–34. [Online]. Available: http://doi.acm.org/10.1145/3243734.3243862

[34] "libmodbus: A Modbus library for Linux, Mac OS X, FreeBSD, QNX and Win32," [Online] Available at: https://libmodbus.org/.

[35] H. Lin, H. Alemzadeh, D. Chen, Z. Kalbarczyk, and R. K. Iyer, "Safety-critical cyber-physical attacks: Analysis, detection, and mitigation," in *Proceedings of the Symposium and Bootcamp on the Science of Security (HotSoS)*. ACM, 2016, pp. 82–89.

[36] H. Lin, J. Zhuang, Y.-C. Hu, and H. Zhou, "Defrec: Establishing physical function virtualization to disrupt reconnaissance of power grids' cyber-physical infrastructures," in *Proceedings of the 2020 Annual Network and Distributed System Security Symposium (NDSS 2020)*, 2 2020.

[37] "Mininet: An instant virtual network on your laptop (or other PC) - mininet," [Online] Available at: http://mininet.org/.

[38] K. L. Morrow, E. Heine, K. M. Rogers, R. B. Bobba, and T. J. Overbye, "Topology perturbation for detecting malicious data injection," in *2012 45th Hawaii International Conference on System Sciences*, 2012, pp. 2104–2113.

[39] "Network functions virtualisation: An introduction, benefits, enablers, challenges, and call for action," Oct 2012, [Online] Available at: https://portal.etsi.org/nfv/nfv_white_paper.pdf.

[40] "The Network Simulator - ns-2," [Online] Available at: https://www.isi.edu/nsnam/ns/.

[41] "Opendnp3: the de facto reference implementation of ieee-1815 (dnp3)," [Online] Available at: https://www.automatak.com/opendnp3/.

[42] "OpENer: an EtherNet/IP stack for I/O adapter devices," [Online] Available at: https://github.com/EIPStackGroup/OpENer.

[43] "Open vSwitch: Production Quality, Multilayer Open Virtual Switch," [Online] Available at: https://www.openvswitch.org/.

[44] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Comput. Netw.*, vol. 31, no. 23–24, p. 2435–2463, Dec. 1999. [Online]. Available: https://doi.org/10.1016/S1389-1286(99)00112-7

[45] "The POX network software platform," [Online] Available at: https://github.com/noxrepo/pox.

[46] M. A. Rahman, E. Al-Shaer, and R. B. Bobba, "Moving target defense for hardening the security of the power system state estimation," in *Proceedings of the First ACM Workshop on Moving Target Defense*, ser. MTD '14. New York, NY, USA: Association for Computing Machinery, 2014, p. 59–68. [Online]. Available: https://doi.org/10.1145/2663474.2663482

[47] "PJM: regional transmission organization (rto) that coordinates the movement of wholesale electricity in all or parts of 13 states and the district of columbia," [Online] Available at: https://www.pjm.com/.

[48] "Ryu SDN Framework," [Online] Available at: https://osrg.github.io/ryu/.

[49] "Schneider Electric PowerLogic ion7550/ion7650 series," [Online] Available at: https://www.schneider-electric.us/en/product-range/1460-powerlogic-ion7550-ion7650-series/.

[50] "SEL-751a feeder protection relay," [Online] Available at: https://selinc.com/products/751A/.

[51] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?" in *Proceedings of the 9th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI'10. USA: USENIX Association, 2010, p. 365–378.

[52] R. Skowyra, L. Xu, G. Gu, V. Dedhia, T. Hobson, H. Okhravi, and J. Landry, "Effective topology tampering attacks and defenses in software-defined networks," in *2018 48th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2018-06, pp. 374–385.

[53] N. Spring, R. Mahajan, and D. Wetherall, "Measuring ISP topologies with rocketfuel," in *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '02. New York, NY, USA: ACM, 2002, pp. 133–145.

[54] "Electrical Grid Test Case Repository," [Online] Available at: https://electricgrids.engr.tamu.edu/.

[55] B. E. Ujcich, U. Thakore, and W. H. Sanders, "Attain: An attack injection framework for software-defined networking," in *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, June 2017, pp. 567–578.

[56] K. Wilhoit and S. Hilt, "The GasPot experiment: Unexamined perils in using gas tank monitoring systems," A TrendLabs Research Paper, Trend Micro Inc. [Online]. Available: https://www.trendmicro.de/media/wp/the-gaspot-experiment-wp-en.pdf

[57] M. Zalewski, *Silence on the wire: a field guide to passive reconnaissance and indirect attacks*. No Starch Press, 2005.

[58] W. Zhou, D. Jin, J. Croft, M. Caesar, and P. B. Godfrey, "Enforcing customizable consistency properties in software-defined networks," in *12th USENIX Symposium on Networked Systems Design and Implementation (NSDI 15)*. Oakland, CA: USENIX Association, May 2015, pp. 73–85.

[59] R. Zimmerman, C. E. Murillo-Sanchez, and R. Thomas, "MATPOWER: Steady-state operations, planning, and analysis tools for power systems research and education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011.