Chapter 11

LEVERAGING CYBER-PHYSICAL SYSTEM HONEYPOTS TO ENHANCE THREAT INTELLIGENCE

Michael Haney

Abstract Honeypots and related deception technologies have long been used to capture and study malicious activity in networks. However, clear requirements for developing effective honeypots for active defense of cyber-physical systems have not been discussed in the literature. This chapter proposes a next generation industrial control system honeynet. Enumerated requirements and a reference framework are presented that bring together the best available honeypot technologies and new adaptations of existing tools to produce a honeynet suitable for detecting targeted attacks against cyber-physical systems. The framework supports high-fidelity simulations and high interactions with attackers while delaying the discovery of the deception. Data control, capture, collection and analysis are supported by a novel and effective honeywall system. A hybrid honeynet, using virtualized and real programmable logic controllers that interact with a physical process model, is presented. The benefits provided by the framework along with the challenges to consider during honeynet deployment and operation are also discussed.

Keywords: Cyber-physical systems, honeypots, threat intelligence

1. Introduction

There is growing evidence that attackers, whether individual miscreants, hactivists, terrorists or state-sponsored actors, are increasingly targeting industrial control systems via Internet-connected computers to wreak havoc on critical infrastructure assets [4, 6, 24, 30, 47, 49, 54]. According to recent U.S. government reports [21, 48], attacks and intrusions as well as fingerprinting and scanning activities against U.S. critical infrastructure assets continue to rise. In 2013, more than 250 incidents were reported and analyzed by the U.S. Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) [21],

Published by Springer Nature Switzerland AG 2019
J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XIII, IFIP AICT 570, pp. 209–233, 2019.
https://doi.org/10.1007/978-3-030-34647-8_11

primarily in the enterprise networks of industrial companies. In 2017, the U.S. National Cybersecurity and Communications Integration Center (NCCIC) [48] detected 447 separate incidents and received roughly 106,000 reports of incidents affecting communications, enterprise and process control systems. In July 2017, officials from the U.S. Department of Homeland Security and the Federal Bureau of Investigation briefed a number of nuclear and other power systems operators on ongoing cyber threats [35].

The U.S. Department of Homeland Security reports that many incidents are not detected due to the lack of adequate detection and logging capabilities, and because details of detected incidents are insufficient. Much of the dramatic rise in incident reporting is the result of increased awareness, detection technologies and information sharing efforts across government and the private sector. It is unclear whether the prevailing industrial safety mechanisms are or will be adequate against complex attack vectors and varying attacker motivations. Current diagnostic tools are deficient at identifying compound exposure interactions, creating pathways by which control systems may be attacked or emergency and safety systems may be defeated or co-opted for use against control systems. This environment makes threat intelligence a vital area of cyber security research.

Several proposals have been made to adapt traditional information technology security techniques and systems to the operational technologies in cyber-physical system (CPS) environments. While these proposals may meet the common requirements of information and operational technologies, some are less suited than others to meeting the unique operational technology demands, especially "always on" availability and minimizing the negative impacts of changes. Patching firmware and rebooting on "Patch Tuesday" every month are not viable for many critical infrastructure systems. In these cases, a more passive approach can provide greater threat intelligence while minimizing the operation impact. Honeypots have been used very effectively in information technology environments. These deceptive systems show promise for defending operational technology environments.

Over the last fifteen years, honeypots – systems designed to be attacked in order to learn attacker tactics, techniques and motives – have been proposed, developed, improved and implemented with great success [3, 5, 9, 20, 25, 26, 29, 33, 36, 37, 44, 51], including in detecting and analyzing the celebrated Stuxnet attack [30]. By their nature, activity in honeypot systems is malicious, or at best unintended, except for any background replay traffic or activity. Because all system resources are devoted to intrusion discovery and the background activity should be relatively easy to filter out, higher levels of system logging and other instrumentation can be provided that are not always possible in production environments. Also, false positive and false negative intrusion alerts can be drastically reduced if not eliminated. For these reasons, honeypots are well suited to defending critical infrastructure systems.

However, recent years have seen only incremental advances in honeypot technologies. Many tools are starting to show their age, incompatible with current

versions of operating systems and dependent software. Tried-and-true tools have had little need to change when supporting traditional information technology systems (e.g., virtual honeypots for Windows and Linux operating systems such as `honeyd`), but they are not necessarily suited to industrial control systems, SCADA (supervisory control and data acquisition) environments and large-scale networks of embedded devices that are coupled to physical processes.

The intelligence gathering capabilities of the proposed honeypot framework can dramatically enhance the security of industrial control networks. The framework is lightweight, highly scalable and designed specifically to protect industrial control systems.

Several requirements for next generation honeynet technologies are proposed in this chapter. Many of these requirements are already assumed by current technologies and approaches; however a formal set of requirements is necessary to drive metric-based advancement. A proof-of-concept architecture is presented to meet these requirements. The architecture links process simulation tools, an advanced network simulator and a complete monitoring system to present a complex attack surface for virtualized industrial cyber-physical systems.

The next six sections discuss industrial control systems, honeypots and honeynets, Security Onion, high-interaction honeypot data collection, virtual networks and the Shodan search engine.

2. Industrial Control Systems

Industrial control systems is a general term that describes engineering process support systems in industrial settings. Modern industrial control systems generally comprise physical components (e.g., gauges, pumps, valves and other sensors and actuators) and digital computer components (e.g., embedded systems). Because of their dual nature, these systems are often referred to as cyber-physical systems. Nearly all modern critical infrastructure assets (e.g., electric grids, water treatment facilities, transportation management systems, and oil and gas pipelines) are managed by cyber-physical systems, which are increasingly being interconnected using traditional information technology networks.

A SCADA system is a distributed system where digital devices connected to a process through sensors and actuators communicate over a network to drive a physical process to a desired state or set point. The idea of using computing devices to monitor and control physical systems is not new. Digital computers were used as early as 1959 when computer control was introduced at Texaco's Port Arthur refinery [55]. These early systems were supervisory in nature. Plant loops were controlled by conventional pneumatic or electrical controllers, but monitored and optimized by computers.

Programmable logic controllers, introduced by Modicon (now Schneider Electric) in the 1960s, were originally designed to replace circuits used in sequential control. Modern programmable logic controllers can implement control loops

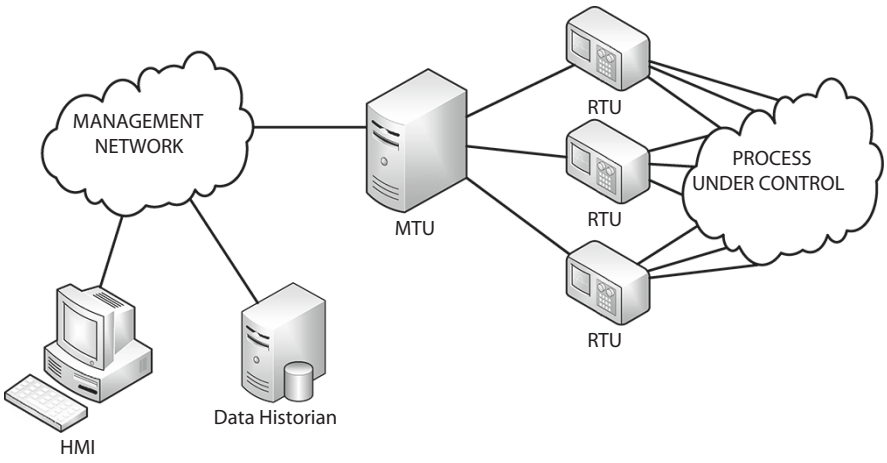

Figure 1. Typical process control network.

such as proportional-integral-derivative (PID) as well as logic functions and sophisticated algorithms.

Communications capabilities were introduced in programmable logic controllers as early as 1979. The Modbus protocol started as a proprietary protocol for Modicon equipment, but is now maintained by the Modbus Organization, a vendor-independent non-profit entity. The Modbus specifications [31] provide a simple communications mechanism for transmitting commands and data between a master terminal unit (MTU) at a control center and remote terminal units (RTUs) at field sites. The original protocol specification defines framing, encoding and error control for transmission over serial lines such as EIA-232 and EIA-485. It also defines standard operations and parameters for MTU-RTU interactions. More recently, the Modbus Organization defined a version of the protocol that uses TCP for transmission [31].

Most modern programmable logic controllers carry control and data messages over TCP/IP using industrial control protocols such as Modbus (widely used in the oil and gas industry), DNP3 (widely used in the electric power industry) and closely-related protocols such as Profibus and IEC 61850. Figure 1 shows a typical process control network and its supporting components.

JAMOD is used for prototyping Modbus-based SCADA systems. JAMOD is a Java Modbus library that supports three transport mechanisms: (i) TCP;(ii) UDP; and (iii) serial (UDP transport is experimental and is not part of the Modbus standard). JAMOD can be used to develop three key components of a SCADA system: (i) a Java class that emulates Modbus-capable programmable logic controllers with the four types of variables that Modbus defines: discrete inputs, discrete outputs, analog inputs and registers; (ii) control logic; and (iii) the process to be controlled. A typical operation cycle has a programmable logic controller read process variables, use the control logic to compute a control

action and then pass it to a programmable logic controller output (e.g., an actuator) to perform an action on the process (e.g., close a valve). Using the JAMOD functionality and library of configuration settings and capabilities, many types of programmable logic controllers can be emulated in software.

3. Honeypots and Honeynets

Spitzner [44] defines a honeypot as a "security resource whose value lies in being probed, attacked or compromised." While Spitzner formally proposed honeypots in 1999 as trap systems for discovering information about attackers, credit is also assigned to Stoll, who published *The Cuckoo's Egg* in 1989 [45] and Cheswick, who authored "An evening with Berferd" in 1992 [8]. These two works describe how clever system administrators lured and studied attackers in order to understand their techniques and ultimately discover their identities. Since the formal creation of the Honeynet Project in 1999 [44], a number of technologies have been introduced to perform this type of intelligence gathering.

A honeypot is the most basic type of attacker trap – a non-production system that is set up to record user and software activities that are assumed to be malicious. Additional terminology has been coined following the "honey" theme to describe various levels of complexity and interactivity. The simplest is a "honeytoken" [43], which can be an unused file, registry key, database entry or email address with some level of additional monitoring to alert when the file is read, the key or database entry is modified or the email address is stolen and added to a spammer's target list.

A "honeyfarm" combines multiple honeypot systems in a cluster or server farm. A more complex setup, which includes honeypots or honeyfarms networked together with out-of-band management and monitoring systems, is referred to as a "honeynet." A honeynet is a complete system of systems with specific goals of attacker control and data collection. In order to fully monitor and control hacker interactions with a number of components in a honeyfarm, a specialized pass-through chokepoint system called a "honeywall" is deployed.

A common way to classify honeypot technologies is according to their levels of interactions with attackers. A honeytoken is generally a low-interaction system that simply waits for someone to come along and access it, upon which it triggers an alarm. Other low-interaction honeypots provide some responses to attacker stimuli. A common low-interaction honeypot tool is `honeyd` [37], which can emulate thousands of networked systems by faking IP addresses and listening on various ports. It provides various banners to attackers who scan the IPs and ports, and it may launch other services and scripts in order to customize responses.

High-interaction honeypots are often deployed as fully functional operating systems within virtual machines or on "bare metal." Virtual machines offer the advantage of being able to scale up more honeypots while limiting the hardware resources that are required. They also offer a means for taking snapshots of system state and rolling back the honeypot systems after compromises. However, they cannot be scaled beyond a handful of systems.

In order to support the management of a honeynet and contain attackers that break into honeypots, a modified approach to a firewall – called a "honeywall" is employed. Although several systems or tools can be used to construct a honeywall, Honeywall, developed by the Honeynet Project, is a specific system build with supporting tools on a single CD-ROM that runs as a live operating system (i.e., read-only). Two major versions of Honeywall have been released – Eeyore and Roo. Honeywall Roo [7] was designed to embody the requirements for GenIII honeynet technologies (discussed later in this chapter). Note that the general term, honeywall, is used to define the proposed data control and collection gateway system.

The first explicit requirement – data control – for a honeynet is to ensure that a compromised honeypot system does not cause harm to other systems in the network. One of the main concerns with deploying a honeypot is the liability and legality of deploying a known-vulnerable system that is intended to be hacked. As stated by Cheswick [8], "[d]ata control always takes priority over data capture." Specific requirements include having both automated and manual control mechanisms and two-deep protection to safeguard against the failure of any one control.

Honeywall Roo is designed to work on a system with three physical network interface cards. Two network cards are configured to bridge two networks, one is the upstream connection to the Internet and the other is the honeynet. The third network card is designated for connections to manage the system and is "out of band" from the other two networks.

Honeywall Roo handles data control using iptables for routing and firewall functions and `snort_inline`, a tool that is no longer actively supported. The `snort_inline` tool is a modified version of Snort[41], an open-source intrusion detection system, that reads packets from iptables and, based on the configured rules, updates the firewall to drop unwanted connections. Note that iptables is configured to provide rate limits for connections outbound from the honeynet to minimize the effects of denial-of-service attacks. As described later, the proposed approach incorporates these design elements, but also makes some significant improvements.

The second explicit requirement for GenIII honeynets is data capture. Capture is the recording of all (or as much as possible) activity in a honeynet. This includes system activity as well all the data entering and leaving the honeynet. An important part of the data capture requirement is that it should be done "without attackers knowing they are being watched" [19].

The third closely related requirement for honeynets is information sharing between disparate honeynets in a standardized way. This supports a community via the disclosure of pertinent threat information. It can lead to shared details about specific malware or targeted attacks from advanced attackers that can benefit other organizations. Honeywall Roo supports data captures and analyses in a number of ways. The primary method is via two Snort processes that listen on the bridge interface, one configured to perform full packet captures and the other configured to match traffic against the enabled rules to generate

alerts. It also provides Argus [39], a network flow processing tool that provides connection summaries and statistics such as bandwidth and packets per second for each observed network flow. This data is tracked in a database managed by Hflow2 so that complete network sessions of flows that match a Snort signature can be reviewed in a web-based tool called Walleye.

Although Honeywall Roo provides excellent support for GenIII honeynets along with a menu-driven system-wide configuration capability [1], there are several problems with the current distribution that render it unusable. The first problem is that Honeywall Roo is based on Fedora Core v3, which reached its end-of-life in January 2006. It also uses Snort v2.6 (which has no available rulesets) and `snort_inline` for data control (which is no longer supported). For these reasons alone, the deployment of Honeywall Roo is ill-advised. Additionally, the Walleye web-based management and analysis interface must be run on the honeywall system, but it only enables data from the single honeywall system to be collected and reviewed. It is not currently possible to separate data control, data capture and data analysis functions on different systems.

A key contribution of this research is a honeywall system design that provides much-improved data analysis capabilities while meeting the data control requirement.

4. Security Onion

Several advanced intrusion detection and network security monitoring techniques and tools [2] can be leveraged to study honeynet data, especially when dealing with unknown attacks. Security Onion is a Linux distribution that combines many tools in a single platform to support out-of-the-box network security monitoring capabilities.

Security Onion provides a network sensor and storage system with utilities for PCAP recording and manipulation, stream identification and flow analysis, passive host identification, payload parsing and signature-based intrusion detection. Two core tools included with Security Onion are Snort and Argus, similar to Honeywall Roo. However, Security Onion offers many advanced management tools and a slew of analysis tools that make the difficult task of network security analysis more manageable.

A significant feature of Security Onion is its three-tier architecture that provides separation and secure communications between sensors, the server and the client. Security Onion Sensor provides applications, libraries and kernel modules for performing full network data captures. Network streams are fed to Bro, Snort or Suricata, Argus and others for processing against policies and signatures that can be updated daily. Security Onion Server provides centralized storage and processing for many sensors. Alerts and connection summary data are written to a MySQL database. Apache hosts multiple web-based applications for presenting data to analysts. Security Onion Client provides the front-end graphical user interface and command-line tools for processing PCAP files and analyzing network flows. In addition to the Sguil monitoring tool, the Security Onion Client includes Wireshark and NetworkMiner for brows-

ing through network traces and reviewing embedded artifacts. Together with the web-based analysis and visualization tools of the ELK stack (Elasticsearch, Logstash and Kibana), tremendous capabilities are provided for network data analysis in an environment that is much more scalable and manageable than Honeywall Roo.

However, Security Onion is not designed to control a honeynet in an in-line deployment – it supports network data collection on a receive-only network interface card connected to a SPAN port or network tap. Another contribution of this research is a design for deploying Security Onion in an in-line manner.

5. High-Interaction Honeypot Data Collection

A full packet capture in a network is invaluable, but it does not offer complete visibility of system activity. Just as network and systems administrators have migrated from cleartext to encrypted protocols (e.g., from Telnet to SSH), so have attackers who often connect to vulnerable `sshd` systems or use stolen or guessed user credentials to gain access. Once in, the attackers install their own, sometimes highly modified, versions of `sshd` or other encrypted remote access tools. This is done deliberately to obfuscate attacks from conventional network monitoring.

The Honeynet Project provides a solution to this problem in a tool called Sebek [18]. Sebek runs on a target honeypot system and collects attacker keystrokes and filesystem access data. Because this is done at the end node, it is possible to view and process data after it has been decrypted by the network software. This provides a critical monitoring capability that is not possible with network-only approaches. Sebek operates stealthily – like a rootkit, it runs in kernel space on a honeypot, hides from users and sends logs covertly to a Sebek server.

However, Sebek is detectable as an unlinked kernel module and because network statistics are increased significantly when keystroke logs are transmitted. Fortunately, many of these shortcomings have been addressed in other tools. For example, Qebek [42] adds hooks to the Qemu system emulator and Ether [13] does the same for Xen. These tools provide the necessary stealth logging in honeynets by addressing data capture in virtual machines.

6. Virtual Networks with IMUNES

Virtual networks of communicating nodes can be generated quickly using the Integrated Multiprotocol Network Emulator and Simulator (IMUNES), a general-purpose network simulation architecture for large-scale real-time experiments [38]. Built on FreeBSD, the simulator offers kernel-level network stack virtualization without the overhead associated with frameworks that make extensive use of virtual machines. This is achieved using virtual nodes in `chroot` jails that have the same capabilities as the underlying kernel. Each IMUNES node can run an independent replica of the FreeBSD network stack as well as unique instances of user-level applications.

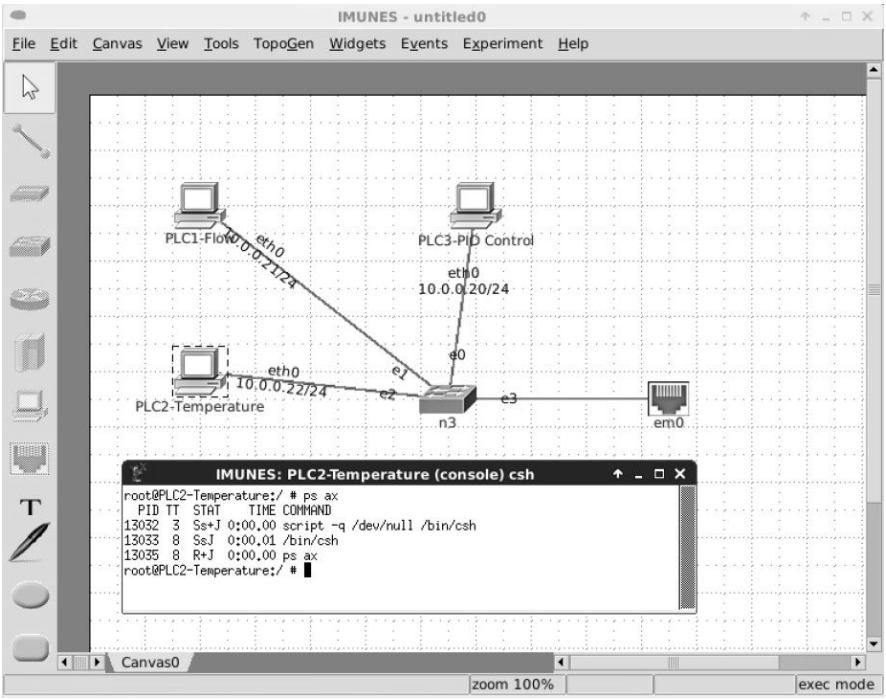

Figure 2. IMUNES graphical user interface.

IMUNES has a user-friendly graphical user interface (Figure 2) that enables arbitrary network topologies to be designed and deployed rapidly. The interface provides seven basic building blocks: (i) workstations; (ii) servers; (iii) hubs; (iv) switches; (v) routers; (vi) physical communications links (i.e., wires); and (vii) connectors for associating physical networks on host systems with virtual networks. Each building block can be reconfigured quickly by right-clicking a node and modifying it in a pop-up window.

Figure 3 shows a more complex model of an advanced metering infrastructure (AMI). IMUNES models can scale to 10,000+ nodes on a moderately powered workstation.

After a network topology has been defined and configured in IMUNES, the simulation is instantiated and executed. During the execution, it is possible to open a shell on any node and run installed applications. Wireshark and the Links web browser run by default on any node. Router nodes use Quagga to fully emulate routers while server nodes can be configured to run DHCP, an HTTP server and other services. It is then possible, by manipulating configuration scripts in IMUNES, to install additional FreeBSD packages in the root virtual filesystem used by each node. This mechanism has been used in this research to incorporate Java and JAMOD, additional industrial control protocols

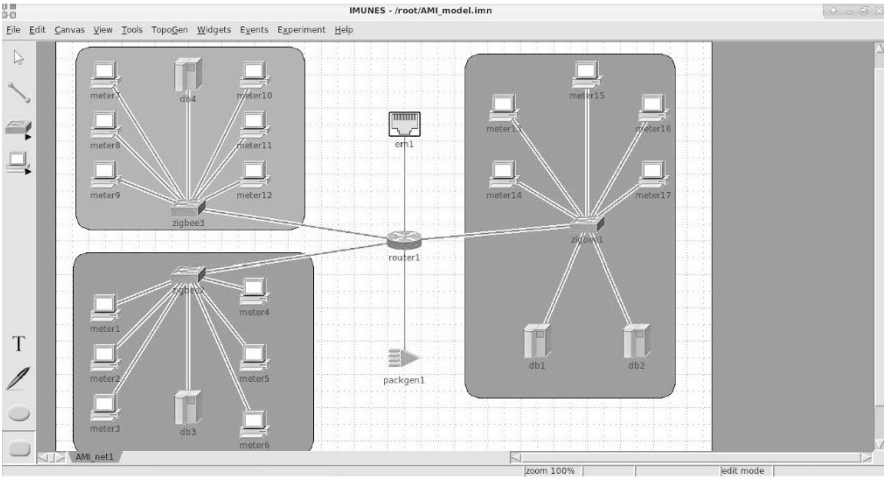

Figure 3. Advanced metering infrastructure model.

and network security tools for system monitoring and logging, including Sebek. Additional details are provided later when discussing the proposed honeynet architecture.

7. Shodan Search Engine

The Shodan search engine (`www.shodanhq.com`) finds Internet-facing systems of certain types (e.g., by manufacturer, operating system, application and configuration) using web-crawling and system fingerprinting techniques. The crawlers scan the Internet for ports 80 (HTTP) and 443 (HTTPS) like most search engines, but also port 22 (SSH), port 53 (DNS), port 161 (SNMP), port 502 (Modbus/TCP) and many others. They query the ports for banners and elicit other responses (e.g., performing an SNMP-walk), and display the results in a searchable format.

For example, if the search term "`Debian port:22`" is entered, results for thousands of systems that are listening on port 22 and run the Debian operating system are presented. Shodan then creates clickable links to the IP addresses and, if available, provides latitude and longitude coordinates from GeoIP data.

When Shodan scans an IP address and finds an open Modbus port 502, it follows up with two Modbus queries to identify the system via its device ID and other string outputs. Often, the results show "`DEVICE FUNCTION FAILURE`." However, a quick search using "`port:502`" can actually discover systems, such as when they return responses like "Schneider Electric BMX P342020." Details from the `whois` database for IP addresses are also provided in the Shodan results, letting a searcher know, for example, that one such device is available on AT&T's U-Verse home fiber optic service. This may suggest that the pro-

grammable logic controller is at a home with a smart meter installed. A click on the GeoIP coordinates takes the searcher to a Google Maps satellite image of the home.

The Shodan search engine is powerful and it actively scours the Internet. As discussed later, these facts should be considered when building cyber-physical system honeynets.

8. Requirements and Prototype Architecture

This section presents the architecture of a next generation hybrid cyber-physical honeynet.

8.1 Next Generation Honeynet Requirements

The formal requirements adopted by the Honeynet Project for GenIII honeynets (third-generation technology) incorporate the essentials of successful honeypotting [19]. These are: (i) data control; (ii) data capture; and (iii) data collection. Additionally, several informal requirements are taken for granted or included as sub-requirements. While these requirements are certainly met by the most successful honeynets, they are not formally specified in the literature.

Therefore, three additional requirements are proposed for next generation honeynets: (iv) realism; (v) scalability; and (vi) detection resistance. These requirements are met by the proof-of-concept design; but they are likely met by other technologies as well. The three additional requirements are essential to updating honeynet deployments to make them viable as an additional network defense layer and as valuable research tools:

- **Realism:** GenIV honeynets should appear to would-be attackers as real production systems that are worth investigating. This requirement is prioritized based on the types of attacks that are desired to be attracted. Some systems on the Internet are very clearly set up as honeypots. While these systems may be effective at catching automated malware and initial reconnaissance scans, they would not fool active attackers who are seeking targets for manual exploitation. An attacker could find such systems by searching for "`honeyd`" in Shodan and avoid their IP addresses altogether.

  Realism in the context of industrial control systems suggests that a honeypot should have basic services such as HTTP, SSH and Telnet, but that the fake programmable logic controllers should be indistinguishable from real ones in network scans. This would mean constructing systems with the same open ports and providing the same responses that real systems provide to Shodan.

  Another aspect of realism is consistency. Specifically, each service must be realistic and the services running together in a honeynet must be consistent. An example honeypot system that is easily discoverable by Shodan would present a Microsoft IIS FTP server on port 21, Debian SSH server listening on port 22 and an open webserver on port 80 that lists pages for

all manner of PHP and database management tools (e.g., different versions of `phpmyadmin` running simultaneously). Potential attackers would avoid this system because it is obviously a honeypot.

CryPLH [5] effectively imitates a Siemens S7 programmable logic controller by providing proper SNMP responses, HTTP and HTTPS pages and an emulation of the SIMATIC Step7 protocol over ISO-TSAP. The techniques used for reverse engineering a real programmable logic controller to create a honeypot are easily replicated for other cyber-physical systems. Another approach involves using Ettercap to record traffic to a real programmable logic controller and then automatically generate a system that provides the same or similar responses [50].

- **Scalability:** GenIV honeynets scale to realistic sizes. Attackers would expect to see dozens or hundreds of programmable logic controllers, not just one or two. However, when deploying full virtual machines it is difficult to meet the realism and consistency requirements – especially if 100 systems in the network are seen to have the same hostname or other identical features. Thus, scalability implies not simply running many virtual systems in the honeynet, but configuring diversity in the deployed systems, possibly by automatically changing configuration files that dictate their "personality" or appearance in the network.

- **Detection Resistance:** GenIV honeynets should not be detectable by attackers. In the official requirements for GenIII honeynets [19], a subrequirement under data control says to "control connections in a manner as difficult as possible to be detected by attackers." Because of the significance of this need and the tools and methods by which it may be met, this item is upgraded to a stand-alone requirement.

  Much research has focused on detecting the presence of a virtual machine versus a physical computer. It has also been shown that many of the GenIII honeynet technologies are identifiable by attackers. For example, `honeyd` and Sebek are technologies used exclusively in honeynets and are key components for meeting the monitoring and data capture requirements. However, research has shown that, with some effort, these systems can be identified by attackers, possibly redirecting them to other production systems [17, 37].

  Thus, a critical requirement for GenIV honeynets is detection resistance. Artifacts are bound to be present in any production system. However, the artifacts should, by design, present themselves in the absolute minimum way. For example, `honeyd` users must replace the initial configuration to hide basic facts that are presented by default, such as known timestamps of filesystem artifacts, protocol unimplemented functions and service banners that present the string "`honeyd`." Detection resistance also stipulates that software used in honeynets must be updated to address new research that publishes how to fingerprint a service such as Sebek, which led to the release of version 3.

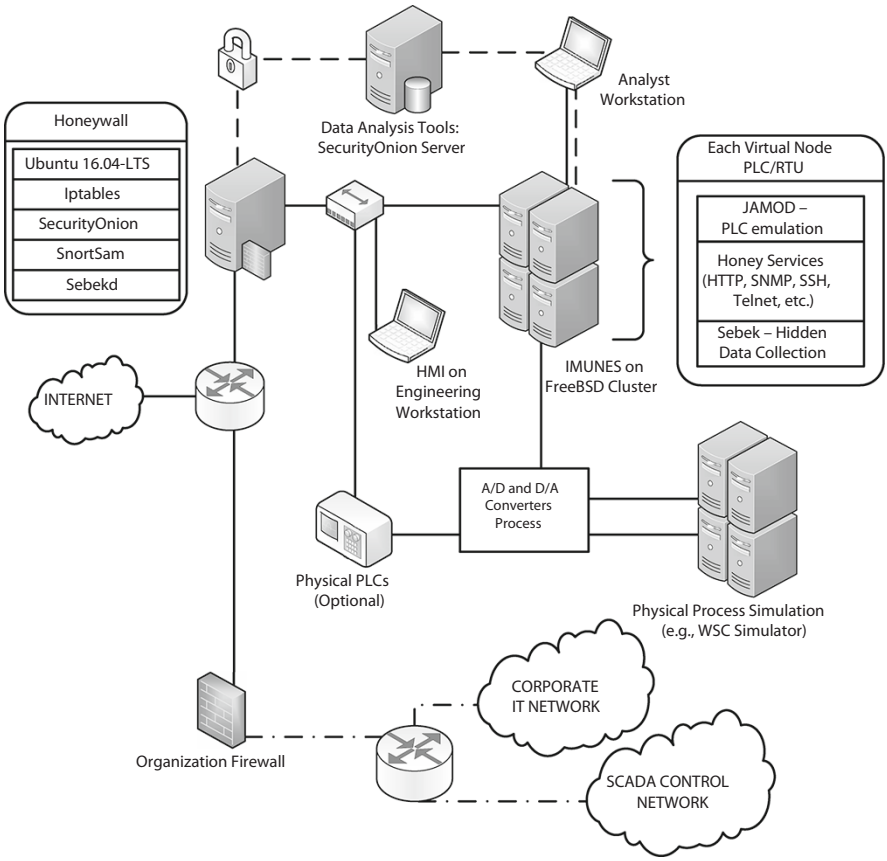

Figure 4. Next generation industrial control system honeynet architecture.

8.2 Proposed Honeynet Design

The proposed honeynet architecture has five components: (i) honeyfarm, comprising a number of virtual honeypots generated by the IMUNES network emulator; (ii) physical honeypot components (e.g., programmable logic controllers); (iii) simulated physical process, which is kept separate from the cyber components of the network; (iv) modernized honeywall running updated data collection tools and providing control of the environment; and (v) management network segment providing separated data analysis tools and access for honeynet operators to connect in an out-of-band manner to the various components. Figure 4 shows the proposed next generation industrial control system honeynet architecture.

8.3 Honeypots

The prototype GenIV honeynet design incorporates a scalable number of individual honeypot systems. The majority of these systems correspond to virtual IMUNES nodes. Some additional hosts may be provided by Qemu, Virtual Box and/or VMware. The remainder are physical devices, such as single programmable logic controller units, that are plugged into a hub or switch downstream from the honeywall.

- **IMUNES for Virtual Nodes:** The prototype honeynet centers on an IMUNES network emulator running on a FreeBSD workstation with two physical network interface cards. The first network interface card is used for connections from the honeypot management network segment (e.g., operator workstations) to manage the base FreeBSD system, which is invisible to the honeynet. The second network interface card is bound to an Ethernet adapter node in IMUNES and given an Internet-routable address range.

  The IMUNES script `prepare_vroot_10.sh` was modified to include additional packages for installation in `vroot` of the virtual network nodes. Basic packages were included, such as `bash` shell, Quagga for router emulation and Wireshark for troubleshooting. Additionally, `netcat`, `honeyd`, `ftelnetd`, `kojoney` and other tools were incorporated. OpenJDK was installed as the Java runtime environment. The script prepared the virtual node filesystems in `chroot` jails. IMUNES was used to create default network configurations for the nodes. After adjusting the first node's network with a publicly-available IP address, additional nodes connected to the same subnet were automatically configured with incremental IP addresses and a matching subnet mask and default route. After the network topology was configured in IMUNES, the honeywall was adjusted to enable packets to be forwarded to this range of IP addresses.

  IMUNES meets the scalability requirement by enabling additional tools to be quickly added to each `vhost` node in the environment. As a honeypot operator decides to expand or modify the services offered, additional packages can be installed on all `vhost` nodes or individual files may be added instantly while the system is operating. These files are created by IMUNES in each virtual node filesystem; changes in each node are isolated to the node, including file deletions, additions and modifications. Additional nodes may be added by simply dragging and dropping nodes into the network topology. This makes it possible to rapidly modify the state of the honeynet and maintain customizations unique to each node or groups of nodes. This is far more manageable than running fully separate virtual machines; indeed, it is a key benefit of the proposed approach.

- **IMUNES Honey Tools and Realism:** The goal of the proof-of-concept honeynet is to obtain information about threats to critical infrastructure systems. While providing general-purpose honeypots with

a variety of services to entice attackers, the main focus is on attracting attackers interested in launching Modbus attacks. In order to meet the realism requirement, supplemental services typically found in cyber-physical systems must be incorporated. Full services such as `telnetd` and `sshd` can run on each `vhost`, providing high-interaction honeypot functionality.

The initial proof-of-concept experiments sought to emulate only a realistic network fingerprint. Therefore, it was decided to implement `ftelnetd` (fake Telnet) and `kojoney` (fake SSH) [10]. The two services present the usual banners to attackers upon initial connection and proceed to log the usernames and passwords; the login attempts always fail. Web-based management tools were scraped from real programmable logic controllers and hosted in the IMUNES nodes using `lighttpd`.

The first step in this effort is to mimic the TCP/IP stack options used by the real devices, which is accomplished using FreeBSD `pf`. Also, by scraping any web-based management interfaces, these tools can be recreated and deployed with `lighttp` in running IMUNES nodes.

Another aspect of realism and enticement is the context in which a honeypot is deployed. A honeypot must blend into its surroundings. Thus, the prototype honeynet was deployed on an Internet-reachable subnet of the university network. If the goal is to masquerade as an oil refinery or water treatment facility, the honeynet should certainly not be located in the network address block of a university's computer science department. A realistic honeynet should be deployed in the IP address space of an actual industrial facility to maintain realism and consistency. This is accomplished using a layer-2 virtual private network to forward traffic bound to an unused IP address range at an asset owner's site to the honeynet laboratory environment, which would be invisible to the attacker.

- **JAMOD for Modbus Slave Devices:** JAMOD is a critical component of the honeynet design because it can emulate a production cyber-physical system. The library enables customized ladder logic to run on multiple IMUNES virtual nodes, simulating master and slave devices in a working implementation of the Modbus/TCP protocol. A master program on one node is configured to connect to slave programs on other nodes, querying and setting various data registers. Slave devices can be configured to be "vulnerable" to remote queries and set points to be writeable by any remote master process. Attackers may then interact with any of the slave devices in the network by sending Modbus/TCP commands to the open ports of the nodes. It is also important to write JAMOD code that implements Modbus functions such as Slave ID, which are used in Shodan queries and by other scanning methods. Other tools such as Conpot [9], OpenPLC [40] and `opendnp3` [11] can also be deployed at IMUNES nodes to emulate other programmable logic controller platforms.

- **HMI on a Windows XP Virtual Machine:** Documented attacks on industrial control systems, including Stuxnet [14] and others, often attempt to compromise the human-machine interfaces (HMIs) used by operators to manage cyber-physical components. A human-machine interface running on a full Windows XP virtual machine was employed to meet the realism and detection resistance requirements; this is not much different from what is used by other honeypots. The proof-of-concept honeynet has a custom interface built in LabVIEW [32]. The virtual machine executes on the same FreeBSD cluster that hosts IMUNES, but it is managed by the VirtualBox hypervisor.

- **Physical Switch/PLCs and Scalability:** Another important design element incorporated in the prototype honeynet is a switch or hub downstream from the honeywall so that multiple systems can be added to the honeynet quickly and easily. This also provides a rapid manual data control mechanism – a system can be physically separated from the network if needed. The design element contributes to scalability because additional devices can be added to the honeynet without modifying the honeywall significantly; only the iptables rules need to be updated in order to forward traffic to the new honeypots.

  The initial experiments employed a Direct Logic programmable logic controller. This device provides an Ethernet connection and includes a web server on port 80 and Modbus listener on port 502. The web server provides access to a configuration interface that requires no authentication so that values can be set via an HTTP form using POST commands. The Modbus listener provides responses to valid queries. All network traffic to and from the programmable logic controller is automatically recorded by the honeywall system without any extra configuration effort.

8.4 Simulated Physical Process

While simply positioning a Windows XP virtual machine in the cloud or connecting a typical programmable logic controller to the Internet would certainly draw attacks, they would not attract advanced threat actors who specifically target industrial processes. The honeynet architecture leverages a research simulation to construct a physical process that can be controlled via digital automation. This method for representing a real physical process is a significant contribution to the honeynet literature. The deployment of simulation software is shielded in a unique way from attacks – by converting signals from digital to analog and back again, which emulates real-world sensors and actuators.

Established methodologies and tools are available for modeling and simulation: (i) continuous process control systems, which typically involve differential equations and have real-time constraints; (ii) discrete systems, which keep a process under control using feedback from sensors to send commands to actuators; and (iii) communications networks, which use TCP/IP or direct-link serial lines. While the distributed nature and communications of a large-scale cyber-

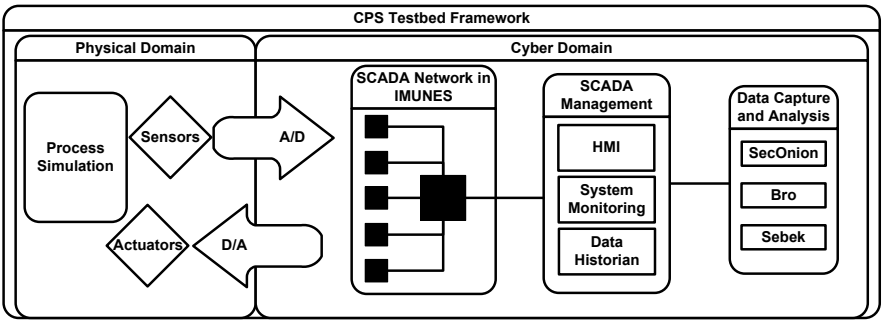

Figure 5. Cyber-physical system testbed for simulation and analysis.

physical system are handled by IMUNES, the honeynet implements a process simulator with an API for programmed sensors and actuators associated with real-world process components.

Models of power system buses and breakers (e.g., IEEE test buses [15]) have been created using software packages such as MATLAB and Simulink. Human-machine interface and power system simulators have been developed in LabVIEW [32] or acquired from entities such as Western Services Corporation [53]. Modelica, an object-oriented, equation-based language has been used to model complex physical systems [12, 16]. However, modeling and simulation tools do not always scale well and do not prioritize the goal of mimicking a physical process in a manner that deceives attackers.

Several testbeds have been developed for modeling and simulating physical systems such as electricity generation and transmission facilities, as well as cyber-physical systems that provide virtual representations of physical components [27, 28]. Simulators require modifications or new interfaces that share control and sensor feedback information with the virtual network. Simulations developed using MATLAB with Simulink [23, 46] and other platforms can be adapted for this purpose. Simulink modules provide the physical equations to create virtual systems that masquerade as a real physical process to attackers. Programmed transformations implementing digital-to-analog (D/A) and analog-to-digital (A/D) conversions provide an interface and convert data from the physical simulator to the cyber (relay) domain, enabling the modeled physical process to remain separate and unexposed, except via instrumentation. Figure 5 shows the cyber-physical system testbed for simulation and analysis.

8.5 Honeywall Design

Three novel aspects of the honeywall design are: (i) honeynet control; (ii) honeynet data capture; and (iii) honeynet data analysis:

- **Honeynet Control:** A GenIII honeynet should not be compromised by attackers to wreak more havoc on other unsuspecting users. How-

ever, the honeynet should also capture the full range of attacker techniques, which often involve making outbound connections to command-and-control servers that contain tools for download. Outbound connections from the honeypots to the Internet must be permitted, but automatic and manual intervention mechanisms must cut off attackers if they go too far.

Honeywall Roo was designed to be used with a layer-2 bridge interface to provide data control and data capture, sniffing traffic as it is forwarded from the Internet to the honeynet internals. Support for in-line monitoring and control using a network bridge interface is not supported by Security Onion. Nevertheless, it is possible to install the Security Onion Sensor package and tools such as network bridge support and SnortSam [22] to provide data capture and data control. The honeywall provides this control using Linux `netfilter` capabilities via iptables and SnortSam, a tool that dynamically adds firewall rules based on Snort alerts. The design incorporates an Ubuntu Server v16.04-LTS on a hardware configuration that mirrors that of Honeywall Roo and requires three network interface cards. The first network interface card provides the management connections; the other two cards are then bridged together using `bridge-utils` at layer 2 as in the case of Honeywall Roo.

- **Honeynet Data Capture:** The Security Onion Sensor package was installed on the Ubuntu system in order to perform data capture and collection. After manually modifying the network configuration for bridging two interfaces, the `br0` interface was selected to monitor inbound and outbound traffic on the bridge. This enables the installed tools to operate in an in-line manner, although this configuration is officially unsupported by Security Onion. The `sebekd` tool was then installed from the Honeywall Roo distribution with the necessary modifications to run on the Ubuntu platform. A Perl script was written to parse the `sebekd` data received by the honeywall without modification. The Sebek client was ported to FreeBSD and compiled into the honeypot kernel. It was then made available on every `vhost` node running in IMUNES.

- **Honeynet Data Analysis:** Data analysis employs a second system configured and installed with the Security Onion Server and Client packages. The system provides the database and web-based applications, and receives data from a sensor over an encrypted SSH tunnel. This enables data collection, offline storage and detailed analysis to be performed without impacting the honeywall. Additionally, multiple honeywall sensors may be deployed across a network and data can be securely aggregated for analysis at a primary system. This design supports the scalability requirement while maintaining good data control.

  Because the new honeywall analysis system includes Bro and ELSA [34], analyses can leverage many more queries than are supported by Walleye, which relies entirely on Snort to generate intrusion alerts. However, to

further support analyses, several customized Snort signatures were incorporated to alert to connections to open ports (e.g., 22, 23, 80 and 502), as well as connection attempts (e.g., SYN packets without the full connection-establishment handshakes indicative of scans) with messages such as "HONEYNET Connection established to port 80." This enables quick investigations of all attacker connections whether or not the connections match intrusion signatures. Analysis can be performed on all signatures matching "HONEYNET" and the sessions can be extracted quickly from the stored PCAP files via CapMe for offline examination using tools such as Wireshark or NetworkMiner.

9. Results and Analysis

A prototype honeynet conforming to the proposed architecture was constructed and deployed on the university development network from whence it was exposed directly to the Internet. Attackers began probing the honeynet almost immediately – the first scan was detected just 19 minutes after the Internet connection was made. The results presented in this section do not provide new insights into attacker tactics or techniques; rather, they demonstrate the utility of the prototype.

9.1 Modbus Scanning via Shodan

Within two days of deploying the honeynet on an IP address range that was not used on the Internet for several years, the Shodan web crawler engines found the honeynet, scanned it for the primary list of ports – including Modbus port 502 – and performed fingerprinting techniques (e.g., banner grabbing) to make data about the honeynet available in its search results.

Over the course of two weeks, the honeynet received connections from most of the systems `census1.shodan.io` through `census12.shodan.io`. These systems are located in the United States based on GeoIP and `whois` database records.

The fact that Modbus was scanned so quickly suggests that attackers can reliably search for industrial control system victims before performing any direct system reconnaissance or fingerprinting them. This should be taken into account when planning honeypot deployments as well as operational network defenses. No other systems connected to port 502 during the two-week period of the experiments.

9.2 Brute Force Login Attacks

The `ftelnetd` tool logged the usernames and passwords presented by attackers during Telnet connection attempts. More than 22,300 Telnet-only logins were attempted from 333 unique IP addresses in 55 countries. Because the connections may have come from Tor network exit nodes or addresses from

Table 1. Top ten brute force Telnet attacks by country.

Source Country	IP Addresses	Login Attempts
China	356	6,456
India	59	1,927
Viet Nam	48	1,245
Mexico	35	1,414
United States	31	1,321
Columbia	24	723
Malaysia	20	1,425
Turkey	18	506
Korea	16	67
Russia (tied)	8	1,394
Taiwan (tied)	8	8

other proxies and privacy networks, correctly attributing the countries of origin and unique attackers would be difficult to impossible.

Table 1 provides a breakdown of the Telnet connection attempts during the two weeks of experiments. The most login attempts (744) came from a single IP address in Belarus. It is notable that only 419 distinct username-password combinations were recorded; many of them were attempted hundreds of times. Since automated attack tools tend to draw from the massive stolen username-password lists posted on the Internet, the small number of distinct username-password combinations suggests that a specialized list of login credentials has been tailored for industrial control systems that may be reachable via their Telnet ports while also listening on Modbus port 502.

Table 2. Most common usernames and passwords.

Username	Count	Password	Count
admin	2,229	password	709
root	2,138	<blank>	612
cusadmin	343	admin	572
MGR	253	1234	455
support	168	12345	393
FIELD	96	123456	385
Administrator	91	Root	248
MANAGER	77	smcadmin	217
guest	60	dreambox	195
OPERATOR	55	highspeed	178

Table 2 lists the most common usernames and passwords used in the Telnet connection attempts. Note that limited results are reported here because

future research with the honeynet would be impacted by the publication of detailed information. Published information about the honeynet would likely lead potential attackers to deliberately change their tactics.

10. Conclusions

Honeypots and other deception technologies have long been used to study malicious activity in networks, but clear requirements for honeypots that implement active defenses of industrial control systems have not been discussed. Existing honeynet requirements as well as new requirements for next generation honeynets specified in this chapter have been employed to develop a realistic, scalable and detection-resistant GenIV honeynet with an advanced honeywall and three-tier data collection subnet. The architecture incorporates a physical process simulation that is lacking in existing honeynets for cyber-physical-systems. It also supports the inclusion of additional physical systems to present a robust, hybrid cyber-physical target for attackers. Indeed, deployments of this honeynet architecture could provide valuable advanced threat intelligence for securing critical infrastructure assets.

References

[1] F. Abbasi and R. Harris, Experiences with a Generation III virtual honeynet, *Proceedings of the Australasian Telecommunications Networks and Applications Conference*, 2009.

[2] R. Bejtlich, *The Tao of Network Security Monitoring: Beyond Intrusion Detection*, Addison-Wesley, Boston, Massachusetts, 2004.

[3] J. Briffaut, J. Lalande and C. Toinard, Security and results of a large-scale high-interaction honeypot, *Journal of Computers*, vol. 4(5), pp. 395–404, 2009.

[4] C. Bronk and E. Tikk-Ringas, The cyber attack on Saudi Aramco, *Survival*, vol. 55(2), pp. 81–96, 2013.

[5] D. Buza, F. Juhasz, G. Miru, M. Felegyhazi and T. Holczer, CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot, *Proceedings of the Second International Workshop on Smart Grid Security*, pp. 181–192, 2014.

[6] E. Byres, The air gap: SCADA's enduring security myth, *Communications of the ACM*, vol 56(8), pp. 29–31, 2013.

[7] G. Chamales, The Honeywall CD-ROM, *IEEE Security and Privacy*, vol. 2(2), pp. 77–79, 2004.

[8] B. Cheswick, An evening with Berferd in which a cracker is lured, endured and studied, *Proceedings of the Winter USENIX Conference*, pp. 163–174, 1992.

[9] Conpot Development Team, Conpot ICS/SCADA Honeypot (`conpot.org`), 2019.

[10] J. Coret, Kojoney – A Honeypot for the SSH Service (kojoney.sourceforge.net), 2006.

[11] I. Darwish, O. Igbe and T. Saadawi, Experimental and theoretical modeling of DNP3 attacks on smart grids, *Proceedings of the Thirty-Sixth IEEE Sarnoff Symposium*, pp. 155–160, 2015.

[12] P. Derler, E. Lee and A. Vincentelli, Modeling cyber-physical systems, *Proceedings of the IEEE*, vol. 100(1), pp. 13–28, 2012.

[13] A. Dinaburg, P. Royal, M. Sharif and W. Lee, Ether: Malware analysis via hardware virtualization extensions, *Proceedings of the Fifteenth ACM Conference on Computer and Communications Security*, pp. 51–62, 2008.

[14] N. Falliere, L. O'Murchu and E. Chien, W32.Stuxnet Dossier, Version 1.4, Symantec, Mountain View, California, 2011.

[15] C. Grigg, P. Wong, P. Albrecht, R. Allan, M. Bhavaraju, R. Billinton, Q. Chen, C. Fong, S. Haddad, S. Kuruganty, W. Li, R. Mukerji, D. Patton, N. Rau, D. Reppen, A. Schneider, M. Shahidehpour and C. Singh, The IEEE reliability test system-1996, A report prepared by the reliability test system task force of the application of probability methods subcommittee, *IEEE Transactions on Power Systems*, vol. 14(3), pp. 1010–1020, 1999.

[16] D. Henriksson and H. Elmqvist, Cyber-physical systems modeling and simulation with Modelica, *Proceedings of the Eighth Modelica Conference*, pp. 502–509, 2011.

[17] T. Holz and F. Raynal, Detecting honeypots and other suspicious environments, *Proceedings of the Sixth Annual IEEE SMC Information Assurance Workshop*, pp. 29–36, 2005.

[18] Honeynet Project, Know Your Enemy: Sebek – A Kernel Based Data Capture Tool (old.honeynet.org/papers/sebek.pdf), 2003.

[19] Honeynet Project, Honeynet Definitions, Requirements and Standards (old.honeynet.org/alliance/requirements.html), 2004.

[20] P. Huang, C. Yang and T. Ahn, Design and implementation of a distributed early warning system combined with intrusion detection system and honeypot, *Proceedings of the International Conference on Hybrid Information Technology*, pp. 232–238, 2009.

[21] Industrial Control Systems Cyber Emergency Response Team (ICS-CERT), Trends in Incident Response in 2013, Idaho Falls, Idaho, 2013.

[22] F. Knobbe, SnortSam – A firewall blocking agent for Snort (www.snortsam.net), 2001.

[23] V. Koganti, Cyber-Attack Simulation in MATLAB/Simulink, M.S. Thesis, Department of Computer Science, University of Idaho, Moscow, Idaho, 2017.

[24] B. Krebs, Cyber incident blamed for nuclear power plant shutdown, *The Washington Post*, June 5, 2008.

[25] S. Kuman, S. Gros and M. Mikuc, An experiment in using IMUNES and Conpot to emulate honeypot control networks, *Proceedings of the Fortieth International Convention on Information and Communications Technology, Electronics and Microelectronics*, pp. 1262–1268, 2017.

[26] T. Lengyel, J. Neumann, S. Maresca, B. Payne and A. Kiayias, Virtual machine introspection in a hybrid honeypot architecture, *Proceedings of the Fifth USENIX Workshop on Cyber Security Experimentation and Test*, 2012.

[27] J. Mahseredjian, V. Dinavahi, and J. Martinez, An overview of simulation tools for electromagnetic transients in power systems, *Proceedings of the IEEE Power Engineering Society General Meeting*, 2007.

[28] J. Mahseredjian, V. Dinavahi and J. Martinez, Simulation tools for electromagnetic transients in power systems: Overview and challenges, *IEEE Transactions on Power Delivery*, vol. 24(3), pp. 1657–1669, 2009.

[29] A. Mairh, D. Barik, K. Verma and D. Jena, Honeypot in network security: A survey, *Proceedings of the International Conference on Communications, Computing and Security*, pp. 600–605, 2011.

[30] S. McLaughlin, C. Konstantinou, X. Wang, L. Davi, A. Sadeghi, M. Maniatakos and R. Karri, The cybersecurity landscape in industrial control systems, *Proceedings of the IEEE*, vol 104(5), pp. 1039–1057, 2016.

[31] Modbus Organization, Modbus Application Protocol Specification, V1.1b3, Hopkinton, Massachusetts (`www.modbus.org/specs.php`), 2012.

[32] National Instruments, LabVIEW, Austin, Texas (`www.ni.com/en-us/shop/labview.html`), 2019.

[33] S. Nunes, Web Attack Risk Awareness with Lessons Learned from High Interaction Honeypots, M.S. Thesis, Information Networking Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, 2009.

[34] V. Paxson, Bro: A system for detecting network intruders in real-time, *Computer Networks*, vol. 31(23-24), pp. 2435–2463, 1999.

[35] N. Perlroth, Hackers are targeting nuclear facilities, Homeland Security Dept. and FBI say, *The New York Times*, July 6, 2017.

[36] V. Pothamsetty and M. Franz, SCADA HoneyNet Project: Building Honeypots for Industrial Networks (`scadahoneynet.sourceforge.net`), 2008.

[37] N. Provos, A virtual honeypot framework, *Proceedings of the Thirteenth Annual USENIX Security Symposium*, 2004.

[38] Z. Puljiz and M. Mikuc, IMUNES based distributed network emulator, *Proceedings of the International Conference on Software in Telecommunications and Computer Networks*, pp. 198–203, 2006.

[39] QoSient, Argus: Network Audit Record Generation and Utilization System, New York (`qosient.com/argus`), 2014.

[40] T. Rodrigues Alves, M. Buratto, F. de Souza and T. Rodrigues, OpenPLC: An open source alternative to automation, *Proceedings of the IEEE Global Humanitarian Technology Conference*, pp. 585–589, 2014.

[41] M. Roesch, Snort – Lightweight intrusion detection for networks, *Proceedings of the Thirteenth USENIX Conference on System Administration*, pp. 229–238, 1999.

[42] C. Song, B. Hay and J. Zhuge, Know Your Tools: Qebek – Conceal the monitoring, The Honeynet Project (`www.honeynet.org/sites/default/files/files/KYT-Qebek-final_v1.pdf`), 2010.

[43] L. Spitzner, Honeytokens: The other honeypot, *Symantec Connect* (`www.symantec.com/connect/articles/honeytokens-other-honeypot`), July 16, 2003.

[44] L. Spitzner, The Honeynet Project: Trapping the hackers, *IEEE Security and Privacy*, vol. 1(2), pp. 15–23, 2003.

[45] C. Stoll, *The Cuckoo's Egg: Tracking a Spy Through the Maze of Computer Espionage*, Doubleday New York, 1989.

[46] H. Tsai, C. Tu and Y. Su, Development of a generalized photovoltaic model using MATLAB/Simulink, *Proceedings of the World Congress on Engineering and Computer Science*, 2008.

[47] U.S. Department of Homeland Security, Common Cybersecurity Vulnerabilities in Industrial Control Systems, Washington, DC, 2011.

[48] U.S. Department of Homeland Security, NCCIC Year in Review 2017: Operation Cyber Guardian, Washington, DC (`www.us-cert.gov/sites/default/files/publications/NCCIC_Year_in_Review_2017_Final.pdf`), 2018.

[49] C. Valli and A. Woodward, SCADA security – Slowly circling a disaster area, *Proceedings of the International Conference on Security and Management*, pp. 613–617, 2009.

[50] T. Vollmer and M. Manic, Cyber-physical system security with deceptive virtual hosts for industrial control networks, *IEEE Transactions on Industrial Informatics*, vol. 10(2), pp. 1337–1347, 2014.

[51] S. Wade, SCADA Honeynets: The Attractiveness of Honeypots as Critical Infrastructure Security Tools for the Detection and Analysis of Advanced Threats, M.S. Thesis, Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa, 2011.

[52] D. Watson and J. Riden, The Honeynet Project: Data collection tools, infrastructure, archives and analysis, *Proceedings of the WOMBAT Workshop on Information Security Threats Data Collection and Sharing*, pp. 24–30, 2008.

[53] Western Services Corporation, Power Plant Simulation Overview, Frederick, Maryland (`www.ws-corp.com/default.asp?PageID=1&PageNavigation=Simulation-Overview`), 2019.

[54] K. Wilhoit, Who's really attacking your ICS equipment, *Trend Micro Security Intelligence Blog* (`blog.trendmicro.com/trendlabs-security-intelligence/whos-really-attacking-your-ics-devices`), March 15, 2013.

[55] T. Williams, Computer control technology – Past, present and probable future, *Transactions of the Institute of Measurement and Control*, vol 5(1), pp. 7–19, 1983.