2019

ENERGY-GRID THREAT ANALYSIS USING HONEYPOTS

Kendrick, Marian M.; Rucker, Zaki A.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ENERGY-GRID THREAT ANALYSIS USING HONEYPOTS

by

Marian M. Kendrick and Zaki A. Rucker

June 2019

Thesis Advisor: Neil C. Rowe
Co-Advisor: Thuy D. Nguyen

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE
ENERGY-GRID THREAT ANALYSIS USING HONEYPOTS

5. FUNDING NUMBERS

6. AUTHOR(S) Marian M. Kendrick and Zaki A. Rucker

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

This research addresses cyber threats to smart energy grids. This research identified and characterized threats to electric-power grids by analysis of traffic in a simulated grid. We deployed a high-interaction honeypot that simulates a grid named GridPot. Network-traffic captures and honeypot-activity logs were analyzed to demonstrate the effectiveness of our honeypot at collecting intelligence for threat analysis. This study will contribute to the efforts of the Department of Defense and Department of Homeland Security to protect U.S. critical infrastructure from cyber threats.

14. SUBJECT TERMS
cyber threat, honeypot, critical infrastructure, cybersecurity

15. NUMBER OF PAGES
81

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ENERGY-GRID THREAT ANALYSIS USING HONEYPOTS

Marian M. Kendrick
Lieutenant, United States Navy
BS, Ohio University, 2007

Zaki A. Rucker
Lieutenant, United States Navy
BA, The Citadel, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 2019

Approved by: Neil C. Rowe
Advisor

Thuy D. Nguyen
Co-Advisor

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This research addresses cyber threats to smart energy grids. This research identified and characterized threats to electric-power grids by analysis of traffic in a simulated grid. We deployed a high-interaction honeypot that simulates a grid named GridPot. Network-traffic captures and honeypot-activity logs were analyzed to demonstrate the effectiveness of our honeypot at collecting intelligence for threat analysis. This study will contribute to the efforts of the Department of Defense and Department of Homeland Security to protect U.S. critical infrastructure from cyber threats.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADU	application data unit
CDF	cumulative distribution function
COTP	connection-oriented transport protocol
CPS	cyber-physical system
DOS	denial-of-service
FERC	Federal Energy Regulatory Commission
GOOSE	Generic Object Oriented Substation Event
GLM	GridLAB-D model
GPM	GridPot model
HTTP	Hypertext Transfer Protocol
ICS	industrial-control system
IEC	International Electrotechnical Commission
IED	intelligent electronic device
I/O	input/output
MBAP	Modbus Application Protocol
MIB	management information base
MMS	Manufacturing Message Specification
NAT	Network Address Translation
NCCIC	National Cybersecurity and Communications Integration Center
NERC	North American Electric Reliability Corporation
NMAP	Network Mapper
OS	operating system
PCAP	packet capture
PDU	protocol data unit
PLC	programmable logic controller
SCADA	supervisory control and data acquisition
SNMP	Simple Network Management Protocol
S7comm	S7 communication
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

VM	virtual machine
XML	Extensible Markup Language

ACKNOWLEDGMENTS

We would like to thank you, the reader.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

In support of the National Security Strategy, this work addresses the defenses of cyber-physical systems and critical infrastructure pertaining to bulk electric systems. It examines cyber threats to smart energy grids and aims to identify and characterize threats as learned through data analysis of traffic within a simulated grid. This threat survey aims to build awareness of potential vulnerabilities, thereby reducing the risk to actual critical systems. Identification and characterization of threats applicable to energy resilience contribute to the risk management of critical infrastructure as directed in SECNAVINST 3501.1D [1].

A. MOTIVATION

For most of the twentieth century, security for a critical-infrastructure operation such as the bulk power system meant only physical security. Securing of that infrastructure was achieved with a fence around dangerous or critical equipment such as regulators and transformers. Later in the twentieth century, locks and barbed wire were added to increase security in light of more direct threats of domestic terrorism. Current security postures have expanded to include a cybersecurity architecture for risk management and assessment [2].

In the early days of cyber-physical system development, insufficient resources were spent mitigating risks to industrial systems [3]. Added risk can be seen in the loss of expertise in the actual operation of the power equipment. A switch or transformer that was once operated manually by an experienced operator may now be operated by an individual who has not physically visited the site or has a limited understanding of the safe operation of the power-system equipment. Alternatively, an experienced operator is now interacting with a computer system instead of the physical equipment and must learn how to correctly operate the computer equipment along with the power system. An increasing risk is that the cyber-physical system could be controlled by an unauthorized operator. Before the advent of cyber-physical systems, to control an industrial system an unauthorized operator had to gain physical access to the system. Now any device connected to the Internet risks unauthorized access.

There is an elevated risk to network-connected critical infrastructure, over other information technology systems, due to the potential catastrophic impact if it were to fail. Cyber security of critical infrastructure systems cannot be limited to traditional information-technology defense measures, and defense in depth must be applied. An industrial-control systems honeypot can be deployed as an extra layer for defense in depth, for either all the system or a single high-value component. Hackers could be fooled into thinking a honeypot is the real system. Adversarial targets, objectives, or techniques could be learned by capturing the traffic patterns or noting system changes within the honeypot. Not only would this be valuable intelligence to ensure our limited time and resources are being applied against the most targeted systems, but honeypots waste attacker time.

The National Cybersecurity and Communications Integration Center (NCCIC) reports approximately 1.3 million reports of indications of compromise (IOCs) related to cyber and communications since March 2016 [4]. Industrial Control Systems (ICS) have been targeted by four families of malware; the most recent of which is CRASHOVERRIDE, aka Industroyer [5]. Stuxnet, BlackEnergy 2, and Havex were earlier tailored malware [6]. CRASHOVERRIDE used methods from these three previous malwares, creating a new framework designed specifically to attack electric grids [6]. In December 2016, a transmission-level substation was attacked in the Ukraine using CRASHOVERRIDE [6]. CRASHOVERRIDE exploits network-communications and grid-operations knowledge not specific to any configuration or vendor, and with minimal tailoring can be repurposed to affect grid operations in North America, Europe, and portions of Asia, and the Middle East [6]. Further proof that grid operations can be affected by a cyberattack was demonstrated in a U.S. Department of Energy test at Idaho Labs in 2007 known as the Aurora Experiment, which caused the self-destruction of a replica power plant generator by means of a cyberattack [7].

Sridhar et al. [8] identified cyber threats that target the distribution portion of the bulk power system as load shedding, advanced metering infrastructures, and demand-side management. The United States has seen load-shedding incidents in recent years that have caused cascading power outages affecting tens of thousands if not millions of customers. In 2007, Tempe, Arizona experienced large-scale load shedding which affected 98,700

customers for almost an hour [8]. In 2003, the North America experienced its most severe blackout on record [9].

B. RESEARCH PLAN

Our research constructed a honeypot, a device with only the purpose of collecting information about those that interact with the honeypot [10]. This research implemented a high-interaction honeypot simulating a smart energy grid. We used a high interaction honeypot to draw in more advanced attackers. We used GridPot, an open-source symbolic cyber-physical honeynet framework, to emulate realistic protocols common to industrial-control systems [11]. The primary module we studied was Powerflow created by GridLAB-D, which models an electrical distribution system [12].

C. THESIS OUTLINE

Chapter II provides background information about honeypots and previous work involving honeypots. Chapter III addresses the North American Bulk Power System, industrial-control systems (ICS), GridPot's architecture, five common ICS network protocols, and similar work. Chapter IV outlines our research methodology, with host-environment details, GridPot modifications, data collection phases, and design of parsers for data analysis. Chapter V discusses the results of our collection and data analysis. Chapter VI contains our conclusions and recommendations for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

A. HONEYPOTS

A honeypot is a network node with no purpose other than to detect and analyze unauthorized activity directed at computers and digital devices [13]. Honeypots can provide data and information about an intrusion afterwards to simplify its future detection or provide information about an attacker [10]. Honeypots can be classified based on their reactivity of low, medium, and high. Honeypots are a safe way, with limited risk to a systems infrastructure, to capture, analyze, and characterize threats against an information system.

Honeypots have two general uses as either a tool for research or production [14]. Research honeypots collect data to be analyzed by a researcher instead of notifying an administrator. Production honeypots work with other security measures, like an intrusion detection system (IDS), to provide defense-in-depth of networks. An intrusion-detection system uses known signatures and threat characteristics to alert administrators of unauthorized users or tools on a network. A honeypot could be employed in conjunction with such a system to detect activity for which the system has no signature.

The usefulness of honeypots is related to how much data they collect. Honeypots can use deception to entice attackers into revealing a richer set of information about their attacks [15]. Honeypots that conceal their purpose through deception are more productive because attackers do not want to interact with honeypots [15]. High-interaction honeypots can confuse attackers through program-based or scripted interaction designed to encourage further exploration. The longer an attacker interacts with a honeypot, the more complete the data set will be.

B. PREVIOUS WORK

1. The Honeynet Project

A honeynet is a network of honeypots to simulate production networks without actually producing anything. The Honeynet Project worked to discover using honeynets

the tools and tactics of potential attackers, to improve cyber-security methods [16]. How an organization might integrate a honeynet into an existing production network is depicted in Figure 1. A sensor captures key information from attempted accessing of the honeynet from the Internet. A honeynet deployed in this manner could collect information from attackers outside of the organization as well as insider threats.

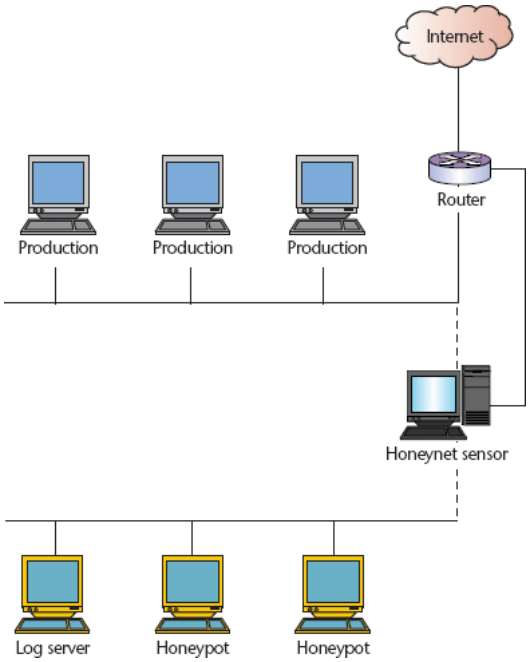


Figure 1. A Honeynet. Source: [16].

To collect data from attackers, one could focus on logging key strokes of attackers [16]. However, keystrokes do not capture the activity of a script that the attacker might deploy. Focusing on any one layer of the TCP/IP stack with focused tools has weaknesses since no single layer tells a complete picture. Instead, several collection points are necessary to understand motives and effects of attackers as with packets, log flows, and log keystrokes.

2. Honeypot-Aware Botnets

A botnet is a collection of compromised computers controlled by a single site (Figure 2). A botnet might be used to send spam, mine cryptocurrencies, or promote accounts on social media.

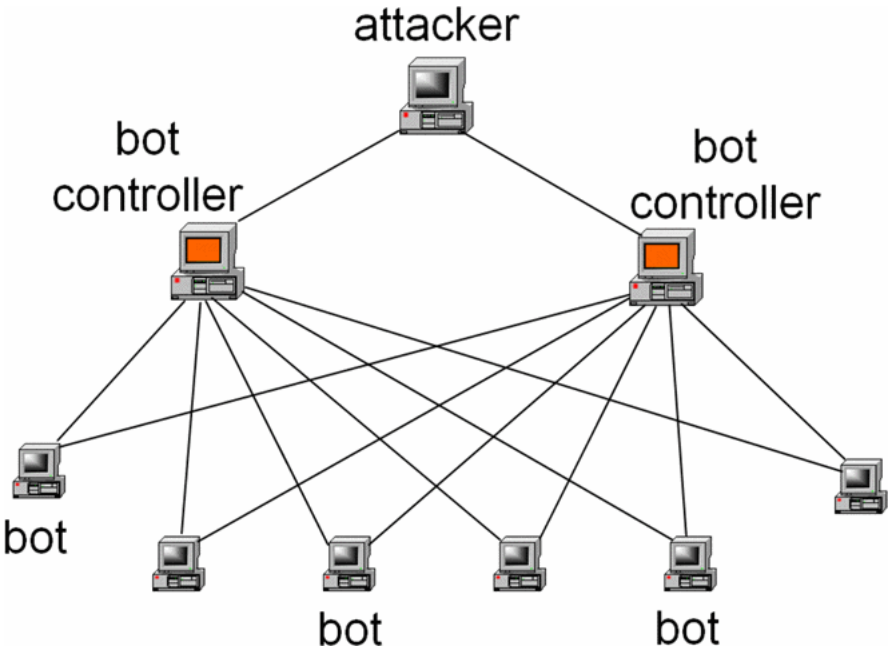


Figure 2. A Botnet Layout. Source: [17].

Botnets have evolved to become aware of honeypots by detecting firewalls and filters on outbound traffic [17]. Honeypots desire to limit their liability in case they are used to launch an attack on a third party. This could be done with an intrusion-detection system that filters for outbound activity. However, a bot controller can detect if a bot is prevented from sending malicious data (Figure 3).

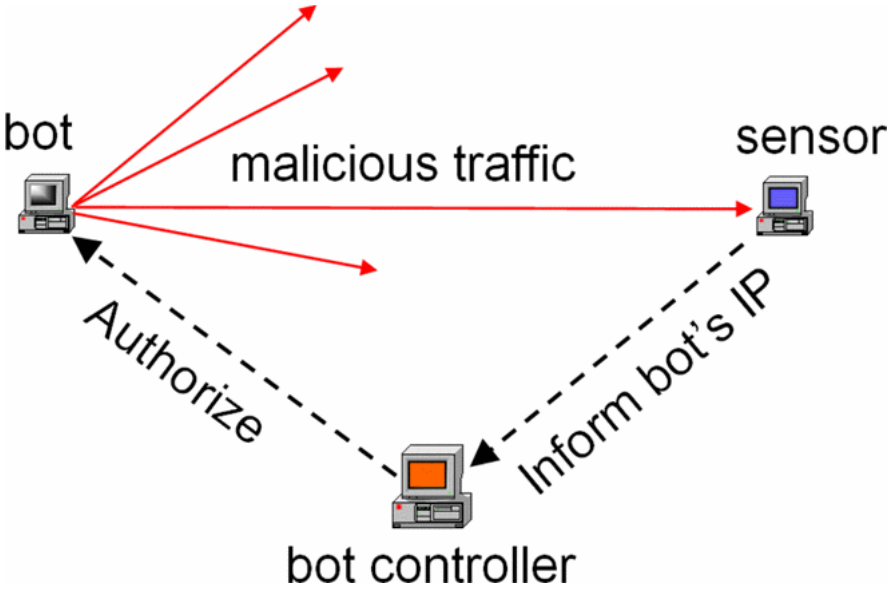


Figure 3. Botnet Detection of a Honeypot. Source: [17].

3. Privacy Concerns

Despite the benefits in deploying a honeypot to gather information about a potential vulnerability or threat to a network, there are some risks as well. Some researchers would argue a honeypot that collects IP addresses, timestamps, and protocol data could be considered a personal-data collection system [18]. This article states there is a privacy concern when collecting network data, since IP addresses have been used to link criminal behavior to individuals. The article mentions the existence of potential privacy concerns in the deployment of honeypots and honeynets and in their ability to share threat data with peers and partners. Finally, this article mentions publishing information collected from a honeypot could provide attackers with information to launch an improved attack.

4. Cloud-Based Industrial Honeypot

A cloud-based industrial honeypot project deployed a large-scale low-interaction honeypot system for 28 days using Amazon's EC2 cloud environment [19]. This experiment monitored the protocols DNP3, ICCP, IEC-104, MODBUS, SNMP, TFTP, and XMPP. This experiment also categorized interaction as either related to Shodan (or Shodan's subdomain) or not. The researchers concluded that reconnaissance activities occurred more often than actual attacks and this reconnaissance targeted individual

industrial protocols rather than combinations of different protocols. Counts by protocol are shown in Figure 4. Additionally, the researchers identified a positive correlation between MODBUS reconnaissance activity by non-Shodan sources following Shodan discovery of MODBUS-enabled devices.

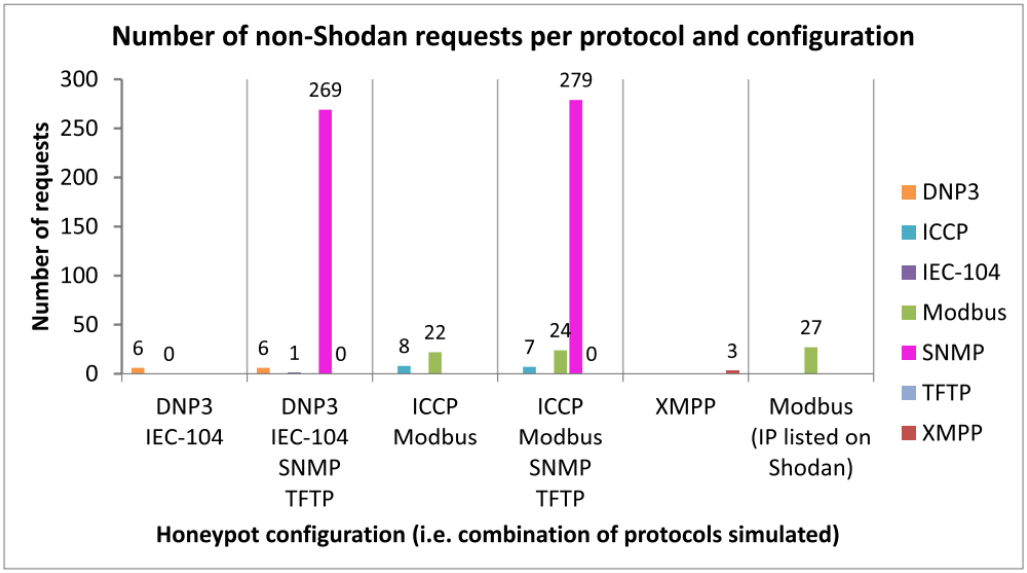


Figure 4. Results from Cloud-Based Industrial Honeypot Deployment. Source: [19].

5. HoneyPhy

HoneyPhy, a cyber-physical system honeypot framework, was developed to better understand attacks against cyber-physical systems [20]. It addressed the problem that existing frameworks could be unrealistic in modeling device physics and device-actuation times and therefore could be easily identifiable as a honeypot. The authors designed two proof-of-concept systems aimed at presenting convincing honeypots. The initial version provided general structure-modeling processes and devices implementing a simple heating-ventilation system. The extended version provided deployment and log-collection specifications for a simple water-treatment system. The authors claimed both versions appeared to work well [20].

THIS PAGE INTENTIONALLY LEFT BLANK.

III. GRIDPOT

We used GridPot as the base for our experiments. It is the only open-source high-interaction industrial-control system honeypot available at the time of this writing.

A. BULK POWER SYSTEM

Within the utilities industry, the term 'electric grid' refers to the high-voltage transmission lines which crisscross the area of transmission of a bulk power system, connecting generation sources to local distribution sub-stations [21]. The bulk power system of the United States is owned and operated by dozens of individual corporations, municipalities, and cooperatives. The U.S. Department of Energy is responsible for policies concerning power generation and transmission in the United States, among other things. The Federal Energy Regulatory Commission (FERC), an independent regulating agency, is responsible for the safety, security, and availability of the whole of the U.S. bulk power system. FERC has appointed the North American Electric Reliability Corporation (NERC) as its agent to develop and monitor standards for reliability and safety. NERC provides guidance to the regional authorities responsible for geographic zones depicted in Figure 5.

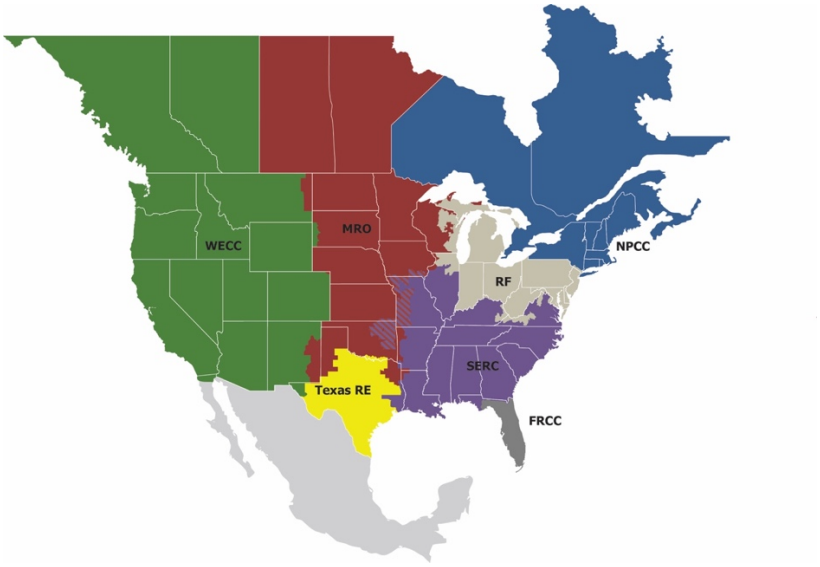


Figure 5. North American Bulk Power System Interconnections. Source: [22].

A bulk power system is divided into four sectors: generation, transmission, distribution, and end-use. The generation of electricity occurs in coal-fired plants, natural-gas plants, solar farms, wind turbines, and hydroelectric plants. Generated electricity is passed through transformers to step up voltage to a very high level where it is then transferred to transmission lines [21]. Transmission lines deliver electricity to substations. Substations use a transformer to step down the voltage from high to low voltage before it is distributed to end users. The process of power flow from generation to customer is depicted in Figure 6.

Key components of substations include transformers and switches [23]. Transformers change voltage when passing current from one circuit to the next. A switch is a device used to direct the flow of current by opening and closing a circuit. A substation employs each of these devices to safely control the transfer of current from transmission to distribution. We also used regulators in our model. Regulators ensure a constant and safe voltage level is maintained throughout the bulk power system.

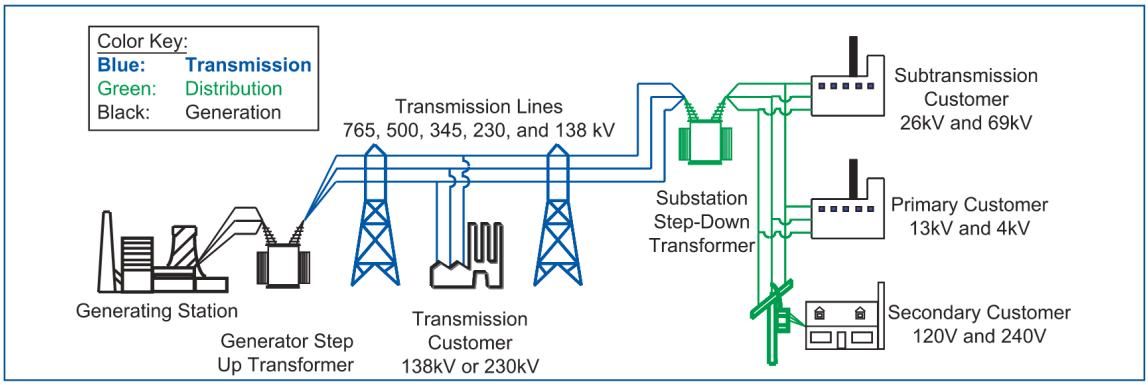


Figure 6. Basic Structure of the Electric System. Source: [9].

In the second half of the twentieth century, technological advances were made that allowed grid operators to monitor and control portions of the electric grid without being at each physical location. Industrial control systems (ICS), Supervisory Control and Data Acquisition (SCADA) systems, cyber-physical systems (CPS), and Intelligent Electronic Devices (IEDs) were part of a larger movement of industrial automation. Industrial-control

systems provide the ability to control multiple physical devices through connected communications. SCADA devices, a subset of industrial-control systems, allow operators to monitor many devices over a wide area. Cyber-physical systems integrate physics and logic to allow interaction between digital, analog, physical, and human components [24]. An IED, like a controller or a digital relay, is a device capable of sending or receiving data or control to or from an external source or a combination thereof [3].

U.S. Navy ships and shore facilities depend on critical infrastructure that includes industrial-control systems. Shipboard cyber-physical systems include mechanical and electrical-control systems used to generate power for driving the ship. Critical power-supply systems and components necessary for operations are found on many military bases, including naval facilities. All are vulnerable to the same threats as a shore-based industrial-control system.

Internet-connected devices including industrial-control systems, can be found using Shodan, an Internet-connected device-search engine. Released in 2009, Shodan provides a suite of services used by researchers and technical professionals [54]. Tools offered on Shodan's subscription site include location mapping of connected devices and searching using protocol headers to fingerprint devices. Shodan's tool "Honeypot Or Not" returns a report indicating if a device at a particular IP address is a honeypot or another type of connected device [26].

B. GRIDPOT ARCHITECTURE

1. Conpot

Conpot is a low-interaction industrial-control-system honeypot associated with the Honeynet Project, an international organization dedicated to security research [27], [28]. Conpot was developed in 2014 to provide researchers with attack data on supervisory control and data acquisition by simulating a Siemens SIMATIC S7-200 programmable logic controller (PLC) with common industrial-control-system protocols [29]. Conpot was written in Python 2.7 and later upgraded to Python 3. Conpot is connected to the Internet and listens on common ports used by industrial-control system devices; it records data from other computers and devices that attempt to connect to those ports over the Internet. The

data is recoded in a log for a researcher to analyze. The latest version of Conpot, version 0.6.0, runs nine protocol-related servers with the default template [30]. These protocol-related servers are MODBUS, S7Comm, HTTP, SNMP, BACNET, IPMI, ENIP, FTP, and TFTP. Our honeypot uses Conpot version 0.4.0, enabling MODBUS, S7Comm, HTTP, and SNMP servers on startup.

2. GridLAB-D

GridLAB-D is a simulation and analysis tool for power-distribution systems [31]. It was developed by the Department of Energy and Pacific Northwest National Laboratory to offer users algorithms to model and test distribution systems at a low cost. We used the Powerflow module to simulate voltage and current values across an IEEE 13 node grid model with 15 houses [32]. The Powerflow module is written using GridLAB-D model and Extensible Markup Language (XML) syntax and contains Powerflow objects and schedules [33]. GridLAB-D model syntax is similar to C++ though it is not a procedural language [34]; objects are described in terms of properties and parameters. Schedules use local time to change values in a predefined manner [35].

GridLAB-D model objects that were important to our honeypot were node, link, switch, transformer, and regulator. Object node properties include ‘phases’, which is used to represent a three-phase connection in terms of ‘A’, ‘B’, and ‘C’, as well as, ‘N’ for neutral phase. Object link properties include the following: ‘from’ and ‘to’ for referencing node object connections; ‘status’ used in terms of open or closed; and ‘power-in’ and ‘power_out’ to express power flow in volt-amperes. The switch, transformer, and regulator objects all include properties inherited from the node and link objects. Inherited object switch properties include ‘status’, ‘phases’, ‘from’, ‘to’, ‘power_in’, and ‘power_out’. Inherited object transformer properties include ‘status’, ‘phases’, ‘from’, and ‘to’, and contain three additional properties: ‘ambient_temperature’ to express the temperature around the transformer, ‘winding_hot_spot_temparture’ to express the temperature of the transformer windings, and ‘configuration’ to describe the specific transformer implementation. Inherited object regulator properties include ‘status’, ‘phases’, ‘from’, and ‘to’, with four additional properties: ‘tap_A’, ‘tap_B’, and ‘tap_C’ to express the position

of the tap, and 'configuration' to describe the specific regulator implementation. GridLAB-D model objects that load or transform values can include schedules as parameters to change values over time. GridLAB-D model schedules are defined in <minutes hours days months weekdays value> form.

3. GridPot

GridPot is an open-source honeypot framework for modeling electric grids [36]. GridPot uses a honeypot layer and a modeling layer to add electrical components and integration between GridLAB-D and Conpot, including IEC 61850 communication [36]. GridPot's honeypot layer is derived from Conpot, adding an XML-formatted GridPot template that specifies to which GridLAB-D model (GLM) to link. Additional Python-coded GridPot files are included in the honeypot layer to retrieve parameter values from the running model in real-time using TCP port 6267. GridPot's primary modeling layer uses GridLAB-D's Powerflow module. GridPot model (GPM) configuration files were added to the modeling layer. A GPM configuration file names objects specific to a GLM and is referenced by the GridPot template, thus linking the two layers together. A visual of the linkages between GridPot's honeypot and modeling layers is shown in Figure 7. We obtained the GridPot source code at https://github.com/sk4ld/gridpot [36]. Our GridPot configuration will be discussed more in Chapter IV.

GridPot source code includes additional modeling features for intelligent electronic devices (IEDs) under an electric-components subdirectory [36]. This subdirectory contains code written in C and XML syntax to simulate a GE Brick Merging Unit and a generic input/output (I/O) switch control device. GridPot demo commands state to "start configured IEDs," which implies to start the GE Brick Merging Unit and the switch control device [36]. Our honeypot does not utilize the GE Brick device and we left this additional modeling piece for future work.

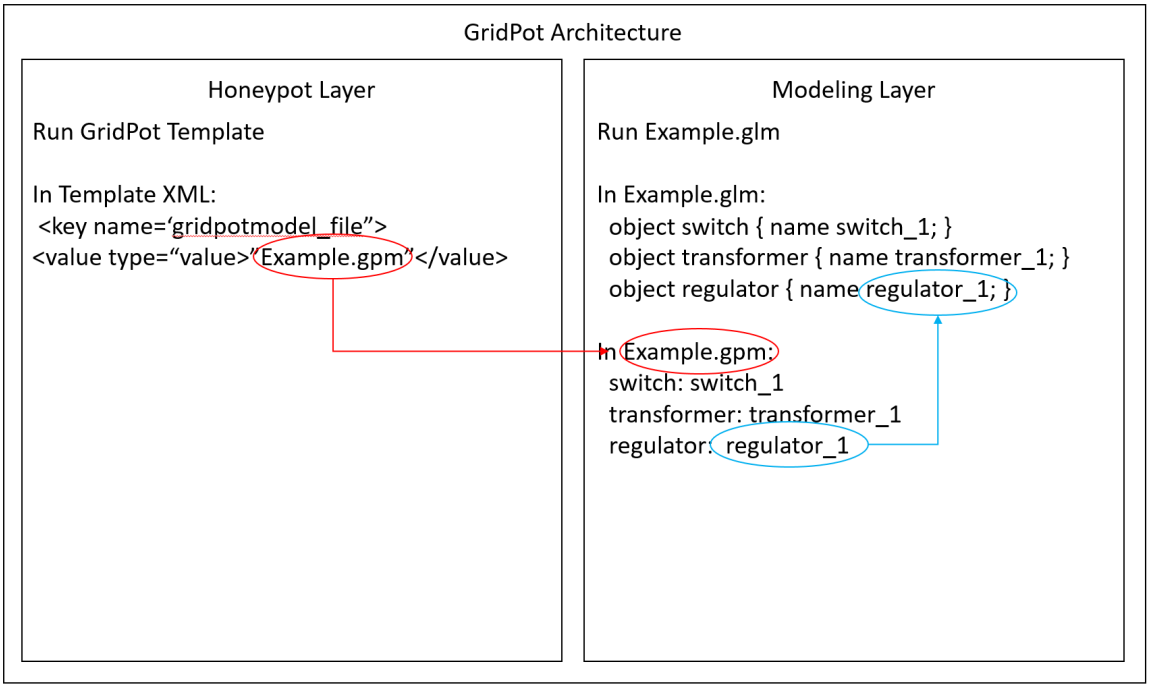


Figure 7. GridPot Architecture

C. NETWORK PROTOCOLS

1. HTTP

The Hypertext Transfer Protocol (HTTP) is an application-layer protocol used to exchange messages between two programs, a client and a server. The World-Wide Web uses HTTP to send request and response messages across the Internet, most often using transmission control protocol (TCP) port 80 [37]. Clients can make requests of the server via pre-defined methods, such as GET, POST, OPTIONS, HEAD, and TRACE. The GET method retrieves the requested specified information. The POST method requests the identified resource accepts the enclosed entity and is designed to add, append, or make comments to resources. The OPTIONS method requests information (i.e., server capabilities) about a specific resource. The HEAD method requests a return of meta-information (header fields) of the requested entity and is identical to GET except without the message-body. The TRACE method is used by a client to gain diagnostic information by requesting a reflection of what the server receives in the response body of the message. Status codes are sent by the server in response to client method requests.

2. MODBUS

MODBUS is an industrial-control-system application-layer protocol for client/server communication shown in Figure 8. MODBUS was introduced in 1979 and is the most common industrial-control-system protocol currently [38]. It is used to remotely start operations by devices including programmable logic controllers (PLC), control panels, and input/output (I/O) devices. [39].

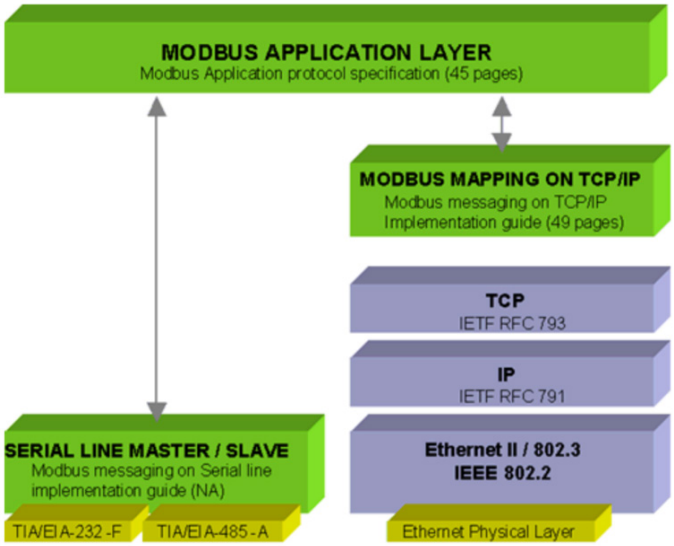


Figure 8. MODBUS Communication of Serial Line (left) and Ethernet Using TCP/IP (right). Source: [38].

Four MODBUS message types are sent between client and servers on an Ethernet TCP/IP network: MODBUS Request, MODBUS Confirmation, MODBUS Indication, and MODBUS Response [40]. Information exchange is used to read and write data access and for diagnostics purposes. MODBUS messaging consists of a MODBUS Application Protocol header (MBAP header) and a protocol data unit (PDU) consisting of a function code and data. All of these are encapsulated by the MODBUS TCP/IP application data unit (ADU).

3. S7COMM

S7comm is a Siemens proprietary protocol used for data exchange between PLCs and PLC programming. It is used to enable supervisory control and data acquisition (SCADA) systems to access PLC data [41]. It uses the Connection-Oriented Transport Protocol (COTP), the International Standards Organization ISO-on-TCP transport services, and TCP port 102. Message types include 0x01-Job Request, 0x02-Ack, 0x03-Ack-Data, and 0x07-Userdata [42]. A basic model for S7comm is shown in Table 1.

Table 1. S7comm OSI Layer Model. Adapted from [41].

	OSI layer	Protocol
7	Application Layer	S7 communication
6	Presentation Layer	S7 communication
5	Session Layer	S7 communication
4	Transport Layer	ISO-on-TCP (COTP)
3	Network Layer	IP
2	Data Link Layer	Ethernet
1	Physical Layer	Ethernet

4. SNMP

The Simple Network Management Protocol (SNMP) uses the User Datagram Protocol (UDP) on port 161 and allows network administrators to remotely manage, monitor, and configure network devices [43]. SNMP uses an SNMP Manager on the controller, an SNMP Agent on the device, and a 'management information base' [44]. Central management consoles for industrial control systems use SNMP to manage and maintain a network of PLCs [3].

5. IEC 61850

Manufacturing Message Specification (MMS) and Generic Object Oriented Substation Event (GOOSE) protocols are nested under IEC 61850. MMS is similar to the proprietary S7Comm protocol in that it uses TCP port 102 for SCADA monitoring and

supports client/server communications [45]. GOOSE is used for sending command requests and status updates between IEDs and controllers using Ethernet-based multicast communications [45].

D. PREVIOUS GRIDPOT AND CONPOT WORK

A 2015 Florida State University study about vulnerabilities of cyber-physical systems created GridPot as a proof of concept for physics-based intrusion detection, threat intelligence collection, and as an additional capability for CPS real-time situational awareness [46]. This experiment integrated a modified GridLAB-D with simulated substations, a SCADA operator interface, and IEDs using IEC 61850 protocols. A switching attack against IEC 61850 protocols was replicated during this experiment to illustrate the real-time physics analysis performed by GridPot. Our research differs in that it is focused more on network-based threat characterization rather than physics-based impact provided by a modified GridPot.

The effectiveness of Conpot was analyzed in another project [47]. This experiment ran multiple virtualized Conpot instances using both the default template and a template simulating a gas-tank level, which were deployed globally using Amazon Web Services for an 18-day period. Conpot instances that were dependent on obsolete repositories had to be manually added. Analysis was conducted on scans using multiple Nmap flag settings in addition to results of Shodan scans. They concluded that Conpot accurately simulated SCADA ports but could be identified as a honeypot based on other open ports that appear when the system is scanned.

Work at NPS conducted a study to generate cyberattack data specific to industrial control systems using Conpot version 0.5.1 [48]. The objective was to better understand indications of compromise on industrial control systems. This honeypot was deployed outside of the Naval Postgraduate School's firewall from October 2017 to February 2018. It used Conpot's default template to monitor the protocols HTTP, EtherNet/IP, MODBUS, S7Comm, SNMP, BACnet, and IPMI. This honeypot averaged 10.51 attacks per day from 54 countries with a decrease in traffic over time, as shown in Figure 9. This research showed that Conpot is a viable platform for honeypot research for industrial control

systems, and suggested that the logs provided by Conpot need to be supplemented with a packet capture tool like Wireshark to extract more information about attacks.

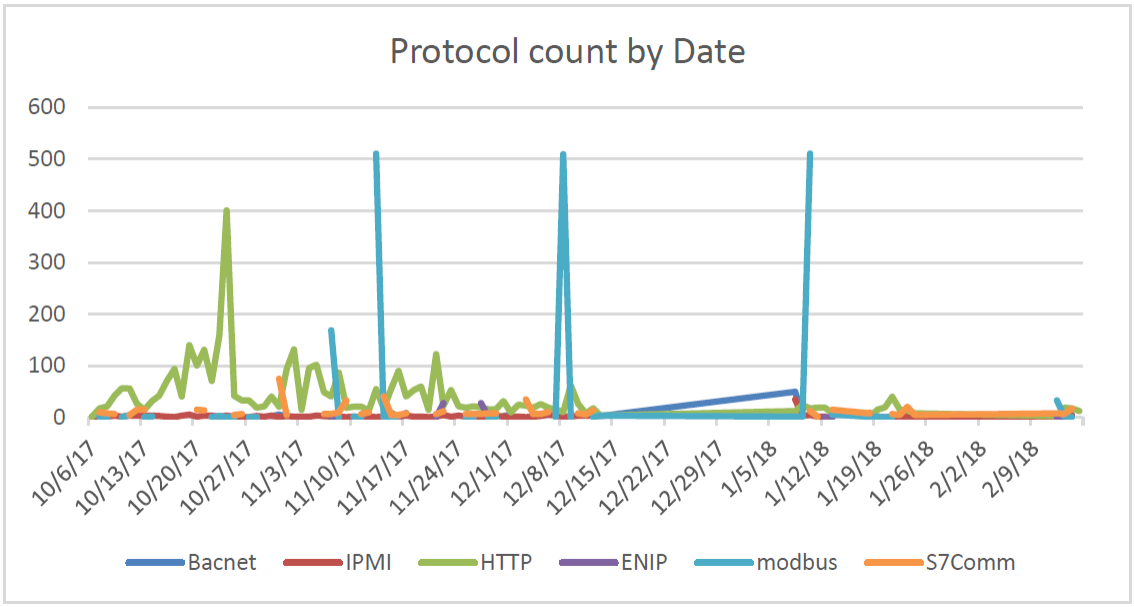


Figure 9. NPS Conpot Data. Source: [48].

IV. METHODOLOGY

A. HOST ENVIRONMENT

Our experiment used two environments, a test environment and a live environment. The test environment was used for the initial code review and baseline data collection. It also enabled internal testing and host-to-virtual-machine testing to establish baseline data for analysis and comparison. The live environment enabled external user access and threat data collection from outside of the campus firewall.

Our live environment was a Dell XPS 8910 desktop computer running a Windows 10 Home OS with 16 GM RAM and a 925 GM hard disk. We used Oracle VM Virtualbox 5.2.22 to import and open our GridPot VM from the test environment. Our test environment was a Dell Precision M6800 laptop computer running a Linux x86-64 Ubuntu 18.04.01 LTS operating system with 16 GB RAM and a 750 GB hard disk. We used Oracle VM Virtualbox 5.2.22 to install a virtual machine in which GridPot was installed. It ran the same operating system as the host and was configured with 10.7 GB of RAM and a 300 GB virtualized hard disk. We used Network Address Translation (NAT), Host-Only-Adapter, and Bridged-Adapter network settings in our live and test environments. The design and layout of our live setup is depicted in Figure 10.

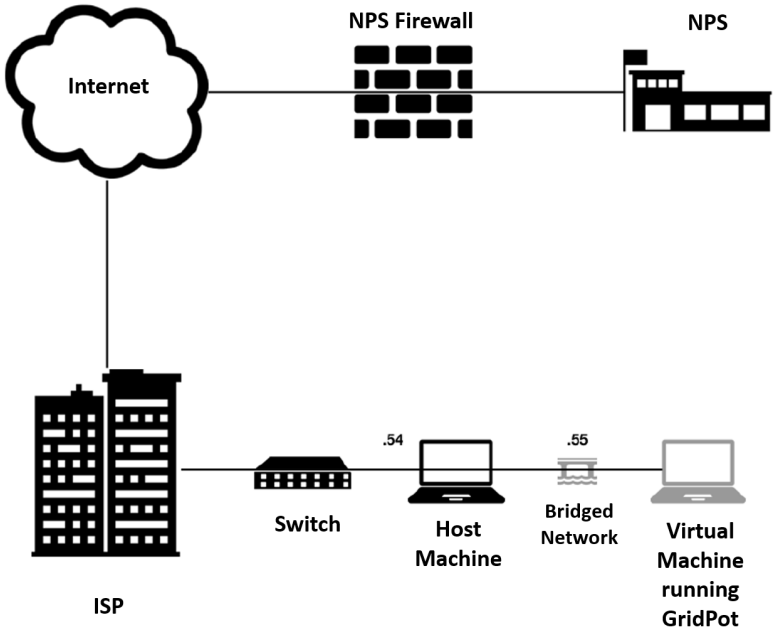


Figure 10. Live Environment Setup. Adapted from [48].

B. GRIDPOT

Our GridPot had a honeypot layer and a modeling layer as discussed in Chapter III. The honeypot layer initialized Conpot using the GridPot template and the modeling layer initialized a GridLAB-D model named IEEE_13_Node_With_Houses. Our GridPot installation steps are provided in Appendix A.

1. Honeypot Layer

Our Conpot was launched using a modified GridPot template. An example output of a successful launch is shown in Figure 11. We updated the 'gridpotmodel_file' field value to link with our modified GPM file named IEEE_13_Node_With_Houses shown in Appendix C. Four protocol servers were started upon launch as written in the original source code and are depicted in Table 2. MODBUS is used on TCP/IP port 502 connecting to one MODBUS client and two MODBUS servers. Details of how we started Conpot using the GridPot template can be found in Appendix B.

```
2019-04-19 09:33:57,993 Starting Conpot using template: /usr/local/lib/python2.7/dist-packages/Conpot-
0.4.0-py2.7.egg/conpot/templates/gridpot
2019-04-19 09:33:57,993 Starting Conpot using configuration found in: /usr/local/lib/python2.7/dist-
packages/Conpot-0.4.0-py2.7.egg/conpot/conpot.cfg
2019-04-19 09:33:57,994 Resetting databus.
2019-04-19 09:33:57,994 Initializing databus using /usr/local/lib/python2.7/dist-packages/Conpot-0.4.0-
py2.7.egg/conpot/templates/gridpot/template.xml.
2019-04-19 09:33:57,994 Initializing power_simulator with
conpot.protocols.gridpot.gridpot_simulator.GridPotSimulator as a function.
2019-04-19 09:33:57,996 DataBus: Storing key: [power_simulator] value:
[<conpot.protocols.gridpot.gridpot_simulator.GridPotSimulator object at 0x7f0621839e90>]
2019-04-19 09:33:57,996 Initializing gridpot_obj with "" as a value.
2019-04-19 09:33:57,997 DataBus: Storing key: [gridpot_obj] value: []
2019-04-19 09:33:57,997 Initializing gridlabd_ip with "localhost" as a value.
2019-04-19 09:33:57,997 DataBus: Storing key: [gridlabd_ip] value: [localhost]
2019-04-19 09:33:57,997 Initializing gridlabd_port with "6267" as a value.
2019-04-19 09:33:57,997 DataBus: Storing key: [gridlabd_port] value: [6267]
2019-04-19 09:33:57,997 Initializing gridpotmodel_file with "IEEE_13_Node_With_Houses.gpm" as a value.
2019-04-19 09:33:57,997 DataBus: Storing key: [gridpotmodel_file] value: [IEEE_13_Node_With_Houses.gpm]
```

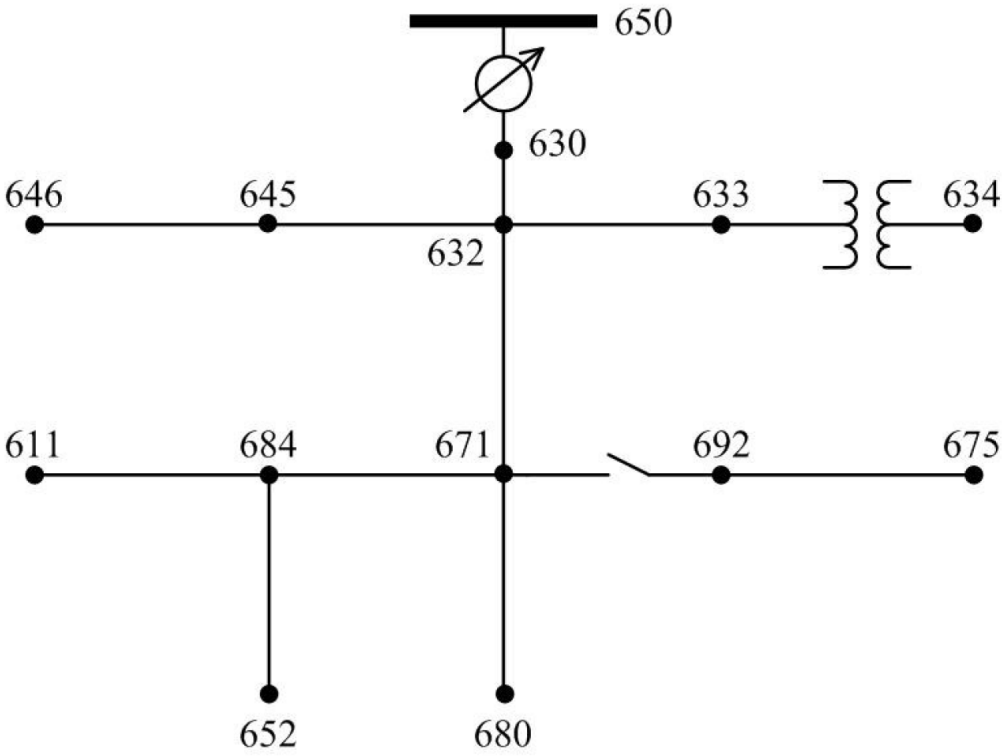
Figure 11. Launching Conpot Using the GridPot Template

Table 2. Protocol Servers Used by Conpot's GridPot Template

Protocol	Port Number
MODBUS	502
S7Comm	102
HTTP	80
SNMP	161

2. Modeling Layer

We used the IEEE_13_Node_With_Houses model, which simulates an IEEE 8500 distribution model with 13 nodes and 15 houses. A partial diagram of this distribution model is shown in Figure 12. We chose this model since it contained switch, transformer, and regulator objects used for the Conpot integration and required minimal code modifications. A schedule was used to alter power flow readings across the switch. Real-time power-in and power-out simulated switch parameter values, which were displayed on our web-based interface. We created a GPM file to link with the switch, transformer, and regulator objects specified in the IEEE_13_Node_With_Houses GLM file by modifying an existing GPM file to ensure proper formatting. These modifications are shown in Appendix C. Chapter III explained how the honeypot layer and modeling layer were linked using the GPM and GLM files.



The switch is located between nodes 671 and 692. The transformer is located between nodes 633 and 634. The regulator is located between nodes 630 and 650.

Figure 12. Partial Layout of our IEEE 13 Node Grid Model. Source: [46].

C. DATA COLLECTION

We collected data in three phases; an internal testing phase from within the virtual machine, a host-to-virtual-machine phase, and a live data collection phase. The internal phase tested GridPot's source code and how Conpot and GridLAB-D interacted. The host-to-virtual-machine phase established a baseline data repository to compare with live data results. The live data collection phase was the period in which the honeypot was connected to the Internet.

We used the tools Wireshark, Nmap, Nessus, Metasploit, and Netstat. Wireshark, a network-protocol analyzer, captured packets on the network [49]. Nmap probed a target system to determine port status, active services, operating system, and MODBUS identification and device information [50]. Nessus, a vulnerability-assessment tool,

scanned for operational-technology devices and SCADA protocols active on the network [51]. Metasploit, a penetration-testing framework, detected MODBUS, found MODBUS units, and read MODBUS client data [52]. Netstat, a command line tool, printed networking subsystem information and showed listening and non-listening network sockets [53].

1. Internal Testing

We conducted internal tests of our GridPot installation using a variety of tools. We altered the Conpot code to use its localhost IP address instead of retrieving the host environment's external IP address upon start. This enabled our honeypot to keep all network traffic internal to our machine. First, we tested if our web-based interface display was accurate to the running model by pointing a web-browser to GridPot's HTTP server using localhost IP address and TCP port 80 and comparing the results to the GridLAB-D model instance that listened on port 6267. We then used the Netstat tool to determine which ports were opened by GridPot. We tested this by executing 'netstat –ano' from the command line before and after starting GridPot. We then ran scans using Nmap, Nessus, and Metasploit against our honeypot. We focused these scans on open ports and probed for operational-technology devices using the MODBUS protocol by running MODBUS detection, MODBUS discovery, and MODBUS interaction scans. We focused on MODBUS since it is the most commonly known industrial-control-system protocol. Details of our internal testing are in Appendix D.

2. Host-To-Virtual-Machine Baseline

Host-to-virtual-machine baseline testing also used Nmap, Nessus, and Metasploit. Before starting this phase, we altered our network connectivity from disabled-network status to host-only status. This allowed for connectivity between our host machine and our GridPot virtual machine without allowing the GridPot virtual machine to connect to the Internet. To prevent GridPot from being viewable from outside our host machine, we left the previous change to the Conpot startup using its localhost address.

We tested the host-only network status using *ping* commands between our host and our GridPot virtual machine and confirmed receipt of a 'network is unreachable' error

when trying to ping an arbitrary IP address. The same scans using Nmap, Nessus, and Metasploit were then performed with the target IP address set to the IP address of our GridPot virtual machine instead of the localhost IP address. To increase our log data for comparison against potential live denial-of-service (DOS) attacks, we conducted an additional scan using an auxiliary DOS Metasploit module during this baseline testing phase. Details are found in Appendix E.

3. Live Data Collection

We modified portions of the Conpot configuration that are common across Conpot instances to attract more interaction from would-be attackers. It was our desire that removing the simple indicators may prolong the attacker's identification of our system as a honeypot. The longer we can keep an attacker interacting with GridPot, the more data we have to determine what their intentions may be. Live data collection altered the static default configurations that could be used to fingerprint an instance of Conpot. These changes were accomplished by modifying the GridPot template XML file found in the Conpot subdirectory. We assumed by altering these default values we would increase our honeypot's deception to both attackers and web crawlers. We used Shodan to determine the success of our deception efforts to suggest our honeypot was an actual energy distribution system [54]. To do this, we surveyed our IP address using Shodan's Honeyscore website [26].

Our honeypot collected data over 19-days from April 11-30, 2019. GridPot ran continuously during this time period except when we brought down the honeypot layer on April 12 due to a broken link between our honeypot and our modeling layer. We fixed this issue and re-launched Conpot using the GridPot template, using the same procedure shown in Appendix B. We used Wireshark to complement the Conpot log to obtain more details about external source interactions with our honeypot. Wireshark packet capture (Pcap) files were saved every three hours. The Pcap files and honeypot logs were backed up to the host machine and stored on a NPS web-based shared platform twice a day on weekdays and once a day on weekends. GridPot was not affected by this back-up procedure and Pcap files older than two days were deleted from the live environment to free disk space.

D. PARSER DESIGN

Wireshark has many built-in capabilities to analyze and report information from a packet capture (Pcap) file. However, these capabilities are limited to single Pcap file. To analyze sets of files, we wrote a Python program that uses the DPKT Python package [55] to examine the packets to determine the source IP address of traffic, the unique ports with which each address is attempting to connect, and the time period in which each IP address is actively sending packets to the honeypot. This data indicates actors that persisted and were searching for vulnerabilities, as well as actors conducting port scans. Results of our analysis tool were compared to the Conpot logs which provide source information as well as the payload of each packet. However, these logs do not provide enough information without an additional layer of analysis to monitor trends over time. We left this additional capability as future work. The Conpot log is sufficient, however, to provide an indication that an actor is active on a network without authorization.

We used Microsoft Excel to filter the Conpot log by protocol. We then were able to count specific protocol instances to include the number of HTTP, MODBUS, and S7Comm sessions, HTTP requests and responses, HTTP versions, HTTP methods, MODBUS connections, MODBUS function codes, S7Comm connections, COTP connection requests, and S7 packets and PDU Types. We also used comparison functions to count repeat attacks, and graphing functions to show HTTP requests by source IP address, MODBUS connections and traffic by source IP address, S7Comm connections by source IP address, HTTP request method distribution, country interaction counts using HTTP, MODBUS, and S7Comm, and cumulative distribution functions (CDFs) of obvious scanning attacks using HTTP.

E. UNEXPECTED COMPLICATIONS

The GridPot source code pulled from GitHub was last updated in March 2015. There were many updates to Conpot and GridLAB-D since then, including GRIDLAB-D upgrading to Python 3.6. This caused many broken connections between the honeypot and the modeling layers which resulted in additional dependencies needing installation before we could successfully run our GridPot. We used the latest release of Ubuntu which

probably contributed to some of the additional older dependencies needing installation. A list of the additional dependencies we installed are listed in Appendix A.

The GridPot source code also did not include all of the code needed to run the IEDs found in the electric-components subdirectory or reference what open-source tool was used to integrate a functioning SCADA operator interface. In-depth code review leads us to believe IEC 61850 communications would occur by launching the IEDs. We assume there is additional software needed to implement both the IEDs and SCADA operator interface and suggest they be implemented in future work.

V. DATA AND RESULTS

Results of our internal and host-to-virtual-machine testing eventually proved successful. It confirmed our web-based interface accurately displayed values of the running GridLAB-D model. Netstat results confirmed that the four protocol ports shown in Table 2 were open. Results of the Nmap, Nessus and Metasploit scans also saw these ports as open and saw that MODBUS-enabled devices were running on our honeypot. We confirmed that results received from internal testing were seen again during host-to-virtual-machine testing.

The honeypot first went online on April 11, 2019. The data collected by Wireshark as of April 29 totaled 9,240,989 packets. The bulk of this data was the result of network broadcast messages, address resolution protocol messages, and standard administrative traffic from the host machine, such as software updates. Filtering this data for network traffic to and from our honeypot reduced the data to 1,525,059 packets and 165 MBs. The traffic flow averaged 1.1 packets per second, and the average packet size was 108 bytes for 113 bytes per second. The greatest number of packets to the honeypot came from an IP address to a California-based cloud-hosting corporation which accounted for 1,013,726 packets and 63 MB of data. The second-highest source of packets was an IP address registered to an LLC in St. Petersburg, Russia, which was responsible for 56,280 packets and 3,221KB. Censys.io traced this address to a Debian-based SSH server in Amsterdam. This address sent 45,657 packets to GridPot over 12 days of which 38,754 were SYN packets sent to GridPot; also RST packets were sent to ports 5160, 5164, 5110, 5093, 5094, 5112, 5134, and 5156.

A. PROTOCOL DATA AND RESULTS

1. Overall Statistics

Our Conpot log recorded 9,641 HTTP requests, 621 MODBUS connections, 606 MODBUS traffic instances, and 102 S7Comm connections from April 11-30, 2019. HTTP was the most commonly used protocol during this 19-day period with heavy scanning almost every day. Heavy scanning using MODBUS was seen twice. Thirty-nine unique

source addresses sent packets to our honeypot multiple times, with return visits either repeating the initial interaction or showing more HTTP requests which could be attributed to further reconnaissance efforts using learned information from the initial contact. Overall honeypot activity by protocol from April 11-30, 2019, is shown in Figure 13.

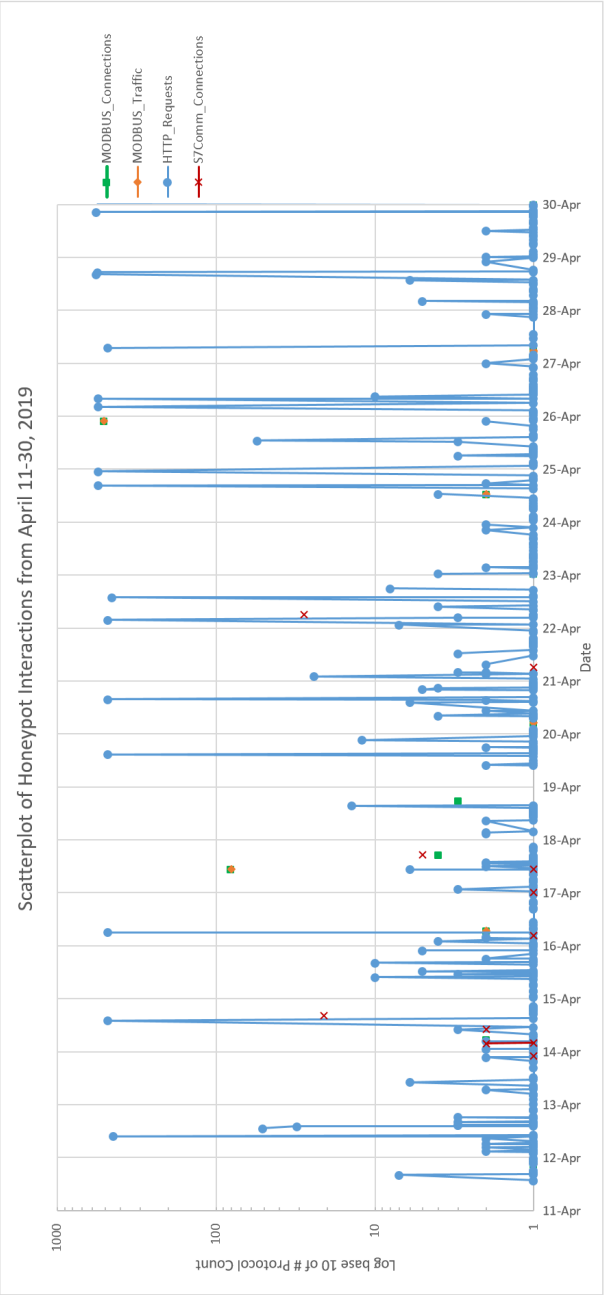


Figure 13. Protocol Count by Source IP Address from April 11-30, 2019

The source addresses observed are shown in Figure 14. This was determined using the GeoLite2 plugin for Wireshark, developed by MaxMind [56]. The countries with more than 100 unique source addresses that probed our GridPot are depicted in Figure 15.

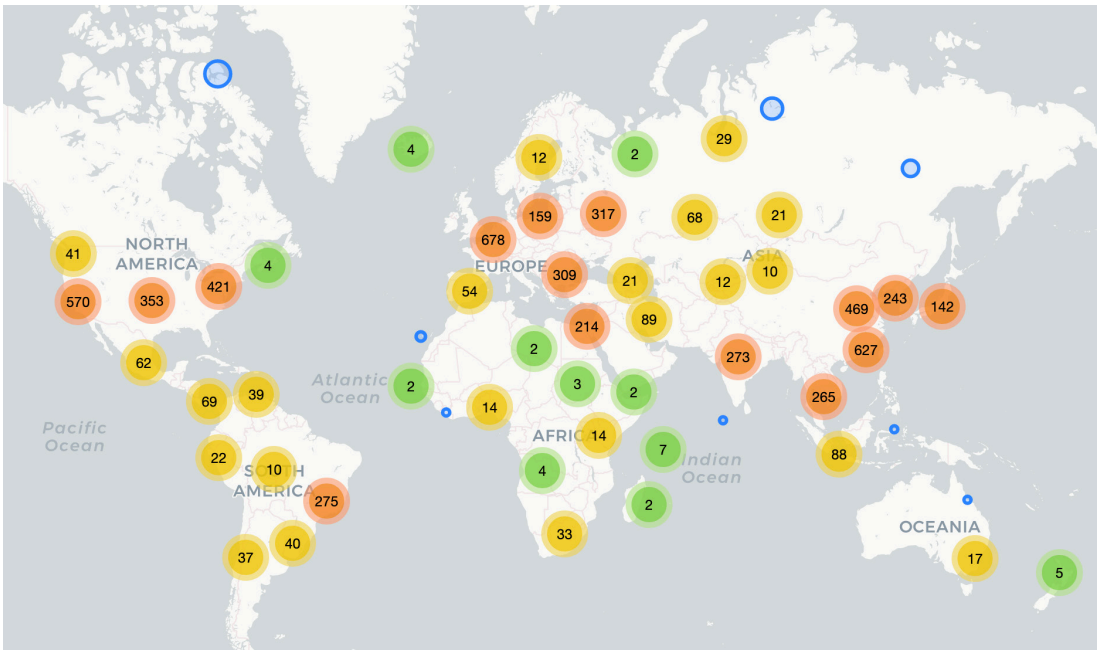


Figure 14. Source IP Address Endpoints

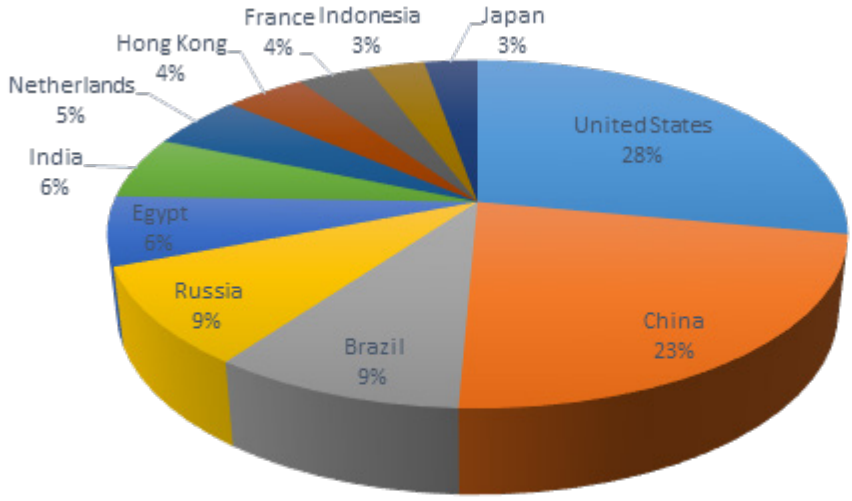
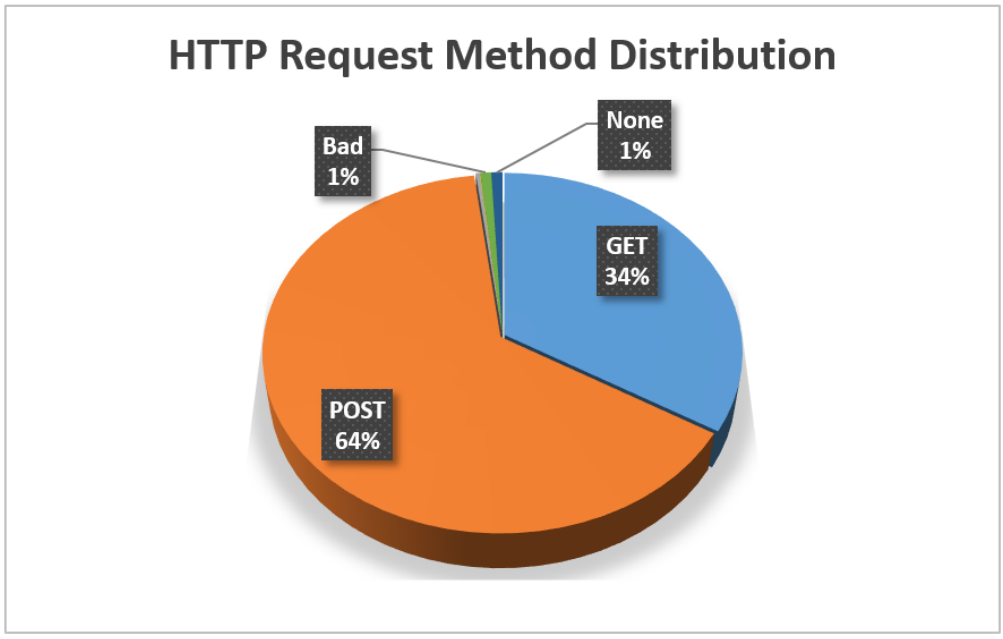


Figure 15. Countries Using More Than 100 Unique IP Source Addresses

The greatest number of packets came from an address registered to Fastly, a content delivery network provider. GridPot exchanged 84,588 packets with just one Fastly address which MaxMind attributes to a location in Seattle. Traffic from this address covered the entire duration of our collection. The traffic can be distinguished by its volume and persistence on our honeypot, but it contained over 26,000 retransmissions of nearly identical ACK messages.

2. HTTP

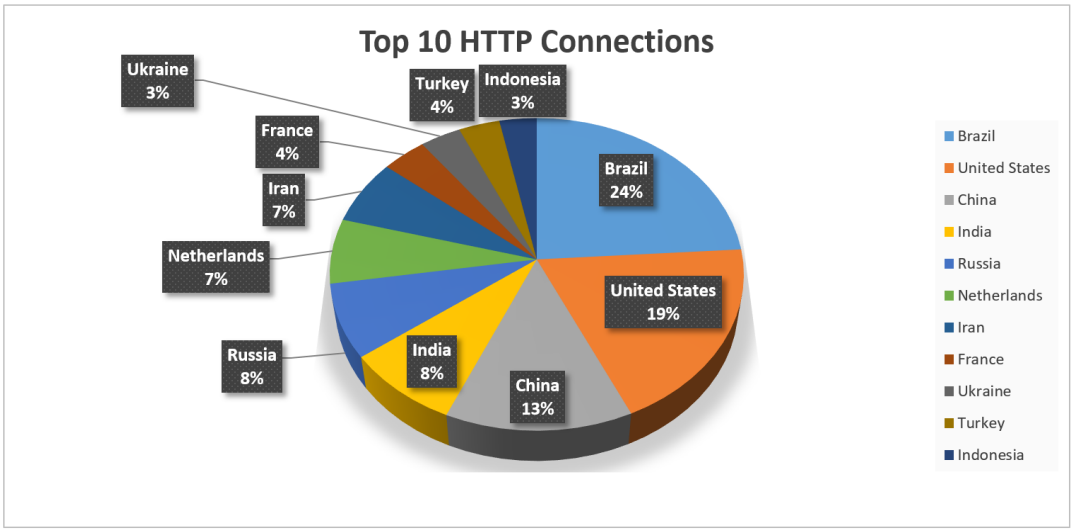
Our Conpot log recorded 453 new HTTP sessions, 9,641 HTTP requests, and 9,591 HTTP responses in April 11-30, 2019. Of the 19,232 combined HTTP requests and responses, 18,886 were HTTP v1.1, 160 were HTTP v0.9, 108 were HTTP v1.0, and 78 were either bad requests or had no HTTP version listed. Seven different HTTP methods were seen in the 9,641 HTTP requests (Figure 16), including 79 requests that did not use any methods and 78 invalid requests, shown as 'None' and 'Bad', respectively.



	GET	POST	HEAD	OPTIONS	PROPFIND	Bad	None
Count	3243	6209	27	4	1	78	79

Figure 16. HTTP Request Method Distribution

Sixty-seven different countries interacted with our honeypot using HTTP during April 11-30, 2019. The top three were Brazil, the United States, and China. Eleven countries interacted with our honeypot at least 10 times, and 42 of the countries only interacted with our honeypot three times or less (Figure 17).



	Brazil	United States	China	India	Russia	Netherlands	Iran	France	Ukraine	Turkey	Indonesia
Count	75	60	42	26	24	21	21	12	11	11	10

Figure 17. HTTP Country Count from April 11-30, 2019

Significant spikes in the number of HTTP requests were seen almost daily as shown in Figure 18. Each peak contained roughly the same number of GET and POST request methods in the same order with varying speeds. This led us to believe these attackers used the same HTTP scanning tool to conduct the attacks. Fourteen attacks were using IP addresses associated with China, two with Hong Kong, and one with Mexico.

We compared this apparent scanning to our host-to-virtual-machine logs and determined that the real scans did not match any of our test scans. We assess these scans are likely all using the same scanning tool. The real scans presented a strong correlation in the quantity and order of GET and POST request methods sent. This trend was not seen in any of our test scan results. To show the real scan correlation, we plotted CDFs of these attacks using distributions of HTTP GET shown in Figure 19, HTTP Post shown in Figure

20, and the combination of HTTP GET and HTTP POST shown in Figure 21. The x-axis for each of these figures is calculated using the start and end time of the attacks; the y-axis shows the distributions of the protocol method.

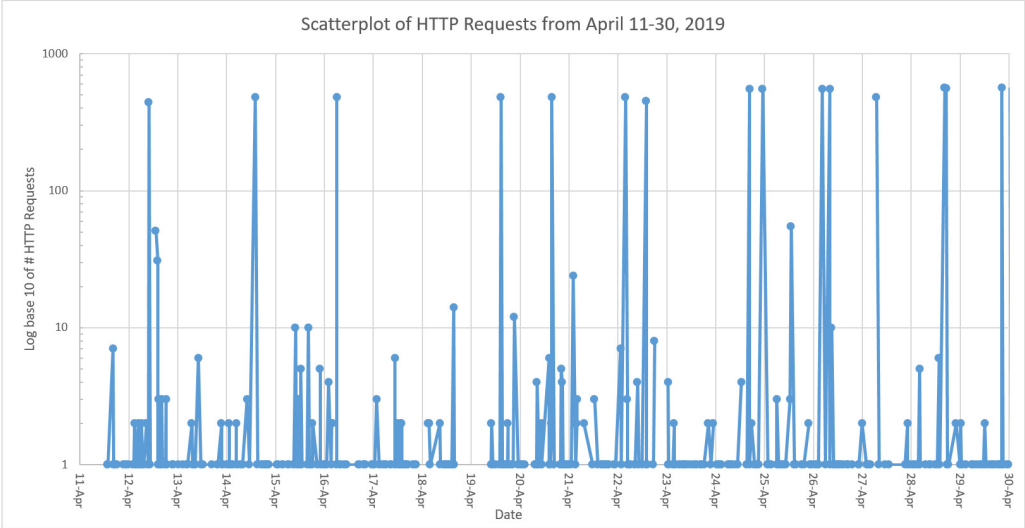


Figure 18. HTTP Requests from April 11-30, 2019

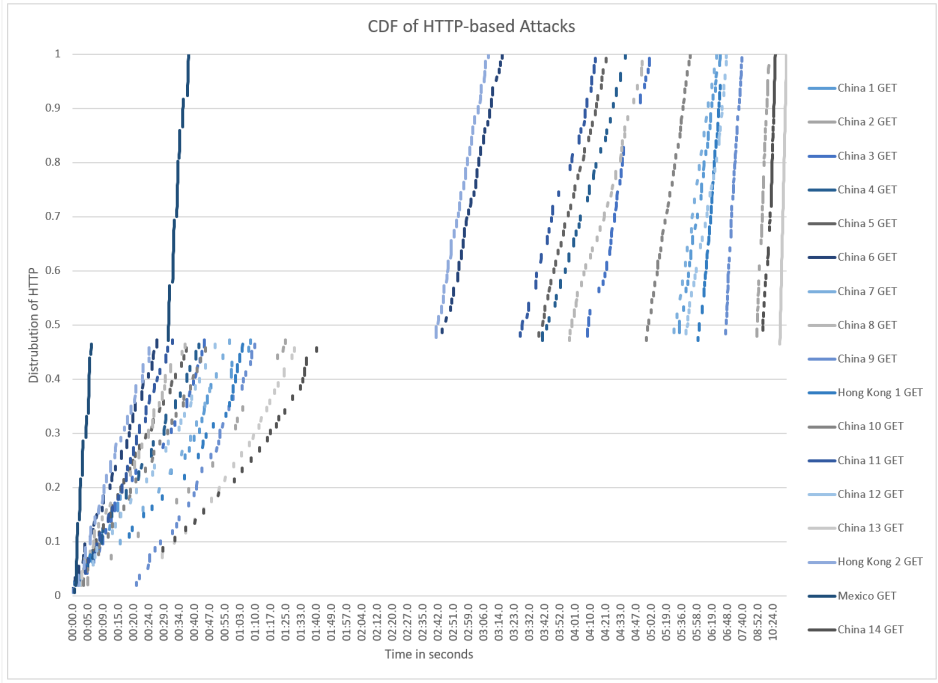


Figure 19. CDF of Attacks Using HTTP GET

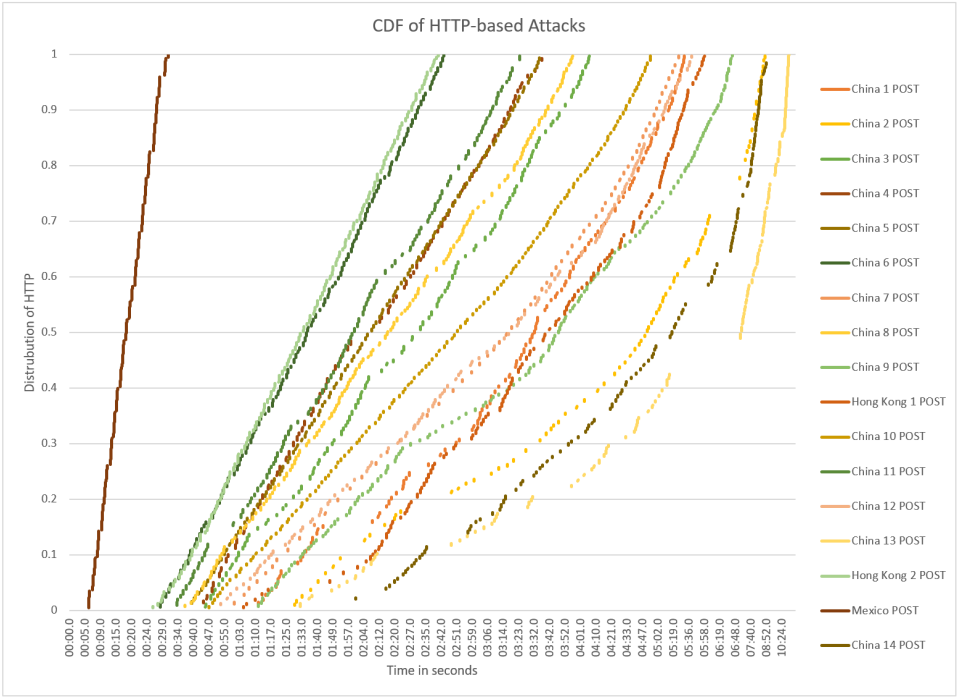


Figure 20. CDF of Attacks Using HTTP POST

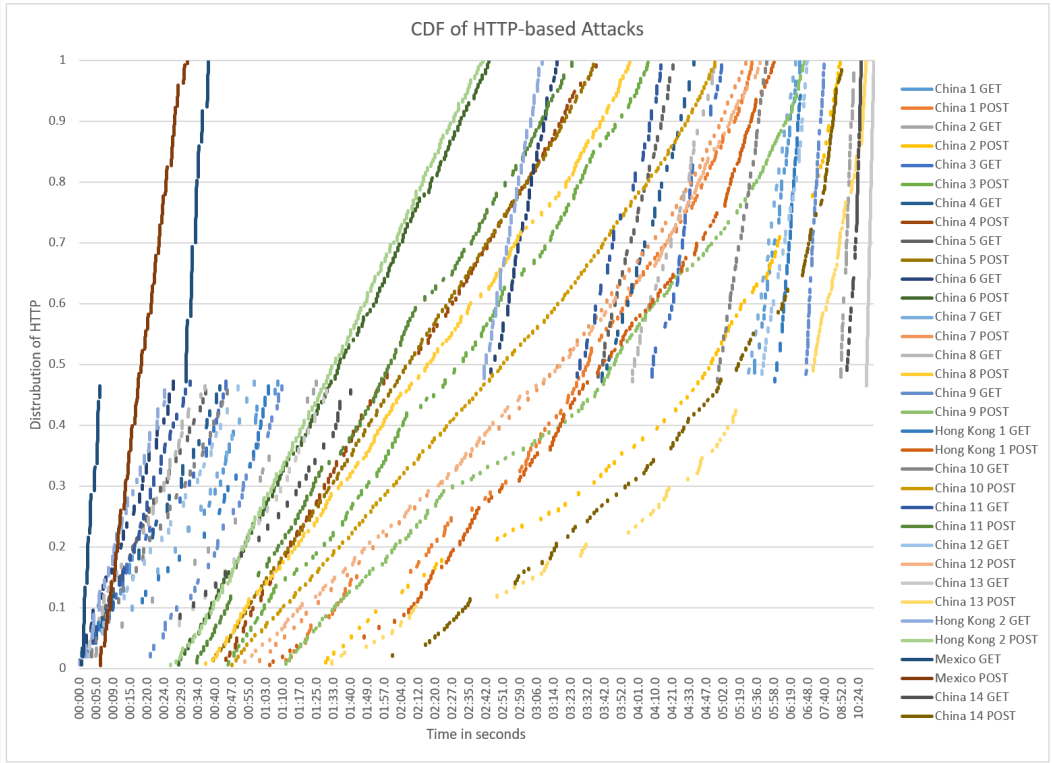


Figure 21. CDF of Attacks Using HTTP

3. MODBUS

Our Conpot log recorded 621 MODBUS connections and 606 MODBUS log entries during April 11-30, 2019. The majority of the MODBUS traffic, 597 entries, did not use a function code and the remainder of the traffic count was split between function codes 17 and 43, totaling 5 and 2, respectively. MODBUS connections and traffic over time are shown in Figure 22, highlighting outliers we investigated further.

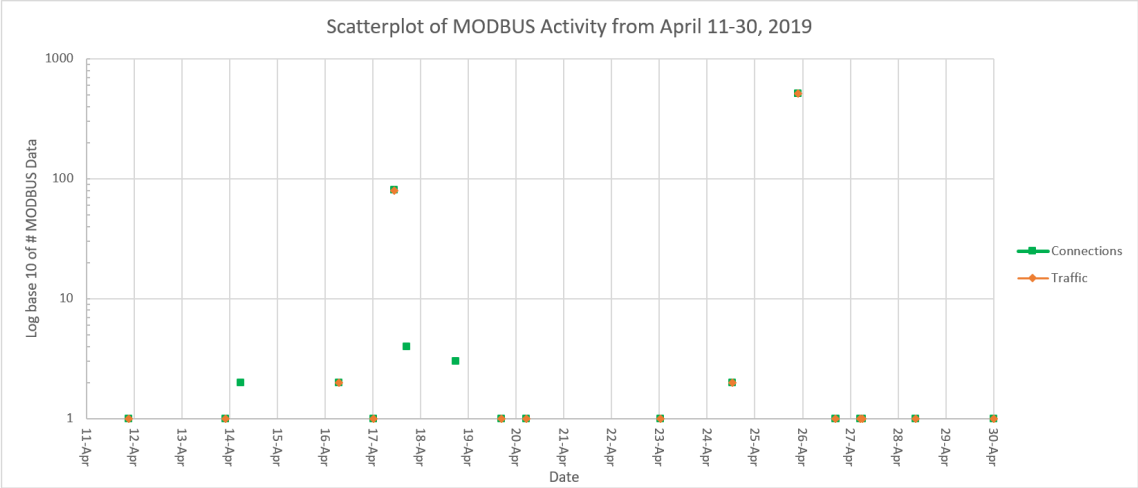


Figure 22. MODBUS Connections and MODBUS Traffic from April 11-30, 2019

Protocol scanning using MODBUS was apparent in a manual review of the Conpot log, and was indicated by incrementing slave ID numbers with each new request seen with both April 17 and April 25. We compared the observed scanning to our host-to-virtual-machine logs and inferred that Nmap and the 'modbus-discover.nse' script were used in both cases. The similarity in the sequencing of function code, slave ID, request values, and response values can be seen in the log of our Nmap scan shown in Figure 23 compared to the logs of attack scanning shown in Figures 24 and 25.

21Mar_conpot_nmap_modbus

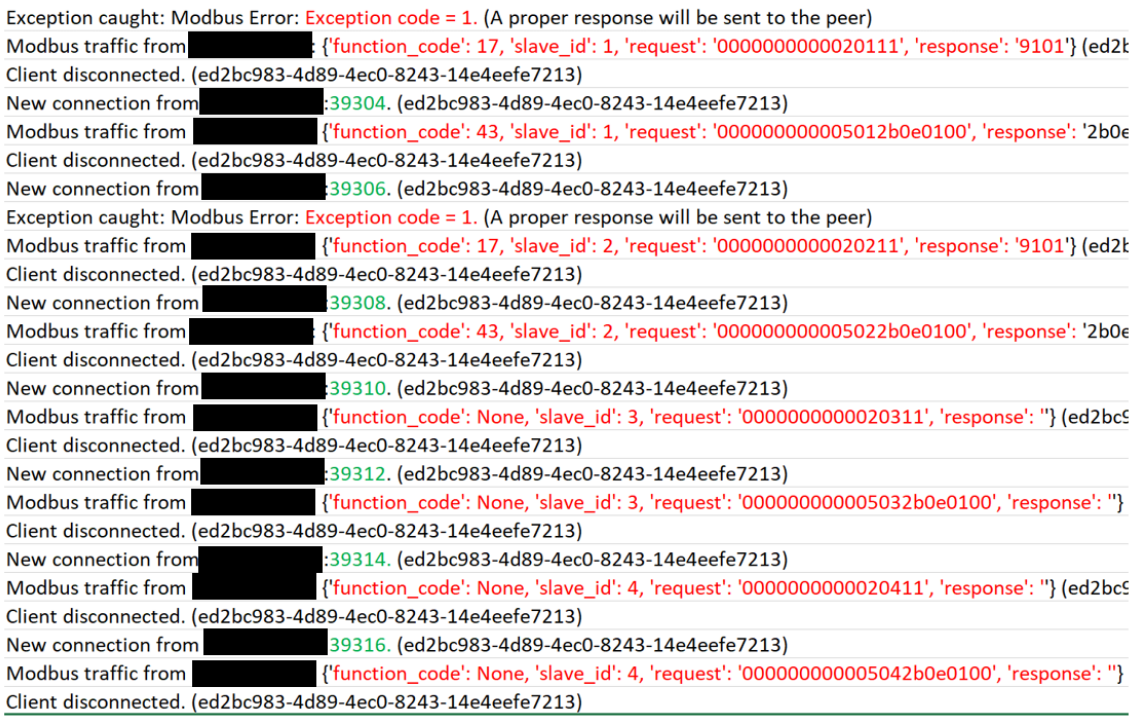


Figure 23. Conpot Log from Host-to-Virtual-Machine Nmap Scan Using MODBUS Script

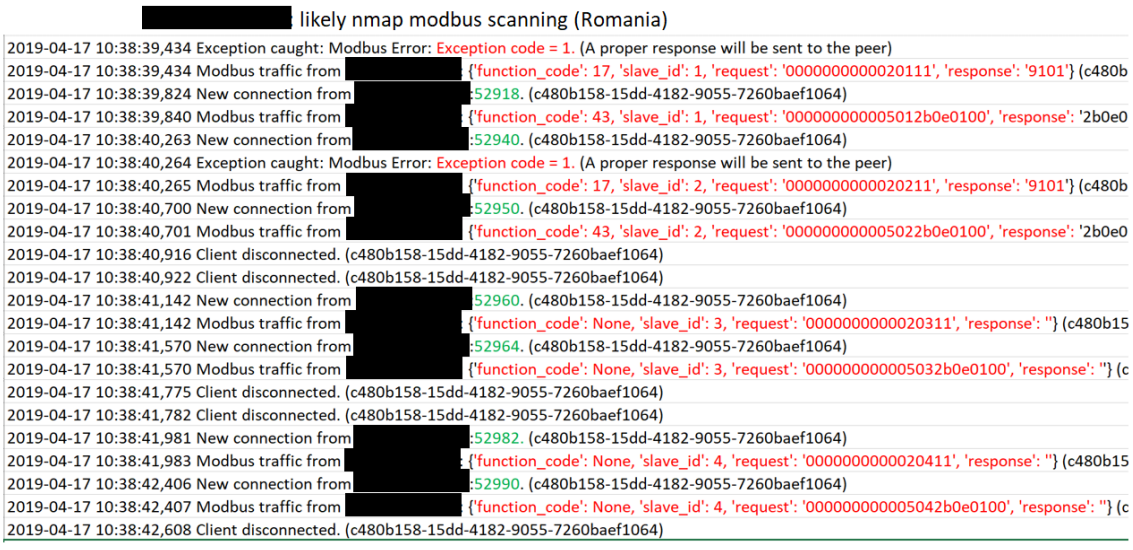


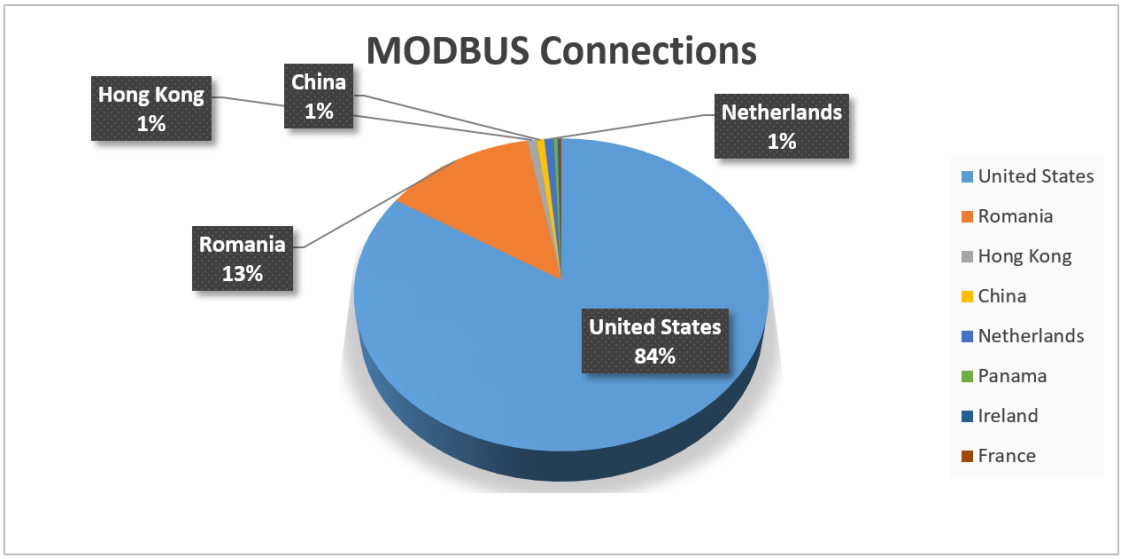
Figure 24. First Conpot Log of Likely Nmap Scan Using MODBUS Script

[redacted] likely nmap modbus scanning (USA)

```
2019-04-25 21:48:15,264 New connection from [redacted]:50038. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,284 Modbus traffic from[redacted]: {'function_code': 17, 'slave_id': 1, 'request': '0000000000020111', 'response': '9101'} (996f731
2019-04-25 21:48:15,339 New connection from[redacted]:50122. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,340 Modbus traffic from [redacted] {'function_code': 43, 'slave_id': 1, 'request': '000000000005012b0e0100', 'response': '2b0e01(
2019-04-25 21:48:15,363 Client disconnected. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,384 New connection from[redacted]:50143. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,385 Modbus traffic from [redacted]: {'function_code': 17, 'slave_id': 2, 'request': '0000000000020211', 'response': '9101'} (996f731
2019-04-25 21:48:15,427 New connection from [redacted]:50171. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,430 Modbus traffic from [redacted] {'function_code': 43, 'slave_id': 2, 'request': '000000000005022b0e0100', 'response': '2b0e01(
2019-04-25 21:48:15,476 New connection from [redacted]:50193. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,477 Modbus traffic from [redacted]: {'function_code': None, 'slave_id': 3, 'request': '0000000000020311', 'response': ''} (996f731e-
2019-04-25 21:48:15,521 New connection from [redacted]:50216. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,522 Modbus traffic from[redacted]: {'function_code': None, 'slave_id': 3, 'request': '000000000005032b0e0100', 'response': ''} (99
2019-04-25 21:48:15,546 Client disconnected. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,546 Client disconnected. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,566 New connection from [redacted]:50261. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,574 Modbus traffic from [redacted] {'function_code': None, 'slave_id': 4, 'request': '0000000000020411', 'response': ''} (996f731e-
2019-04-25 21:48:15,624 New connection from [redacted]:50315. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
2019-04-25 21:48:15,626 Modbus traffic from [redacted] {'function_code': None, 'slave_id': 4, 'request': '000000000005042b0e0100', 'response': ''} (99
2019-04-25 21:48:15,648 Client disconnected. (996f731e-1652-4a16-96a3-e56fa7e6cf7f)
```

Figure 25. Second Conpot Log of Likely Nmap Scan Using MODBUS Script

MODBUS traffic originated from eight different countries, with the United States and Romania being the top two. The MODBUS traffic breakdown by country is shown in Figure 26.



	United States	Romania	Hong Kong	China	Netherlands	Panama	Ireland	France
Count	522	81	5	4	5	2	1	1

Figure 26. MODBUS Traffic Distribution by Country

4. S7Comm

During April 11-30, our Conpot log showed 20 new S7Comm sessions, 102 S7Comm connections, 13 COTP connection requests, and 19 S7 packets. Messages of type-1 and type-7 were the only ones seen, of counts 6 and 13, respectively. S7Comm connections over time are shown in Figure 27.

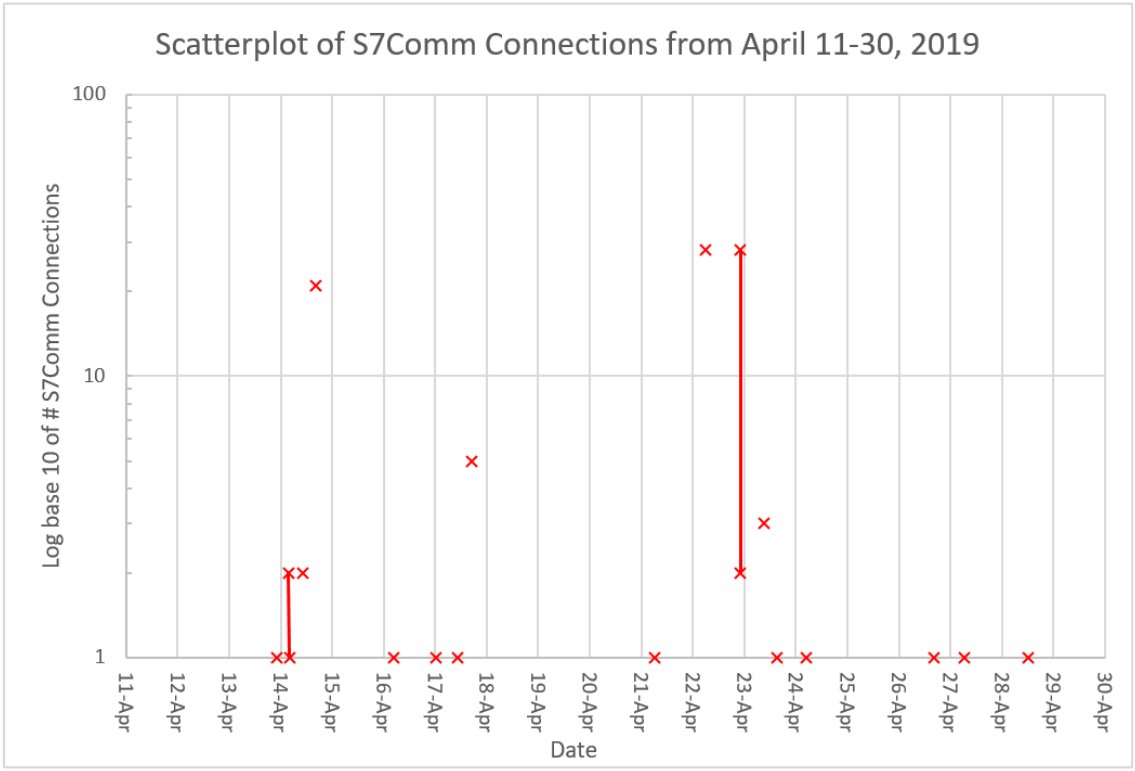
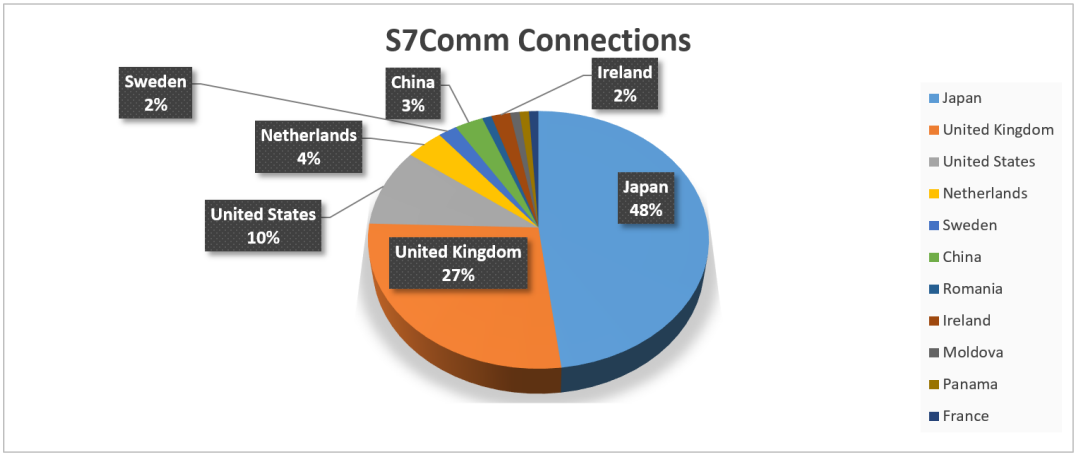


Figure 27. S7Comm Connections from April 11-30, 2019

Eleven different countries connected to our honeypot using S7Comm; Japan connecting the most totaling 49 times, followed by the United Kingdom and the United States with 28 and 10, respectively. S7Comm connections by country is shown in Figure 28.



	Japan	United Kingdom	United States	Netherlands	Sweden	China	Romania	Ireland	Moldova	Panama	France
Count	49	28	10	4	2	3	1	2	1	1	1

Figure 28. S7Comm Connections by Country

B. DECEPTION RESULTS

We used Shodan's 'Honeypot Or Not' tool to determine the convincingness of our GridPot implementation. This tool calculates the probability (ranging from 0.0 to 1.0) that an IP address is a honeypot [26]. Our IP address returned a score of 1.0, indicating with a strong probability that it is a honeypot. Even though our GridPot was not deceptive enough to fool Shodan, this did not prevent attackers from interacting with our honeypot. Either would be attackers were not aware that Shodan had classified our IP address as being attached to a honeypot or they chose to interact with it anyway.

C. BENEFITS OF MULTI-TOOL LOGGING

We used both Conpot log data and Wireshark Pcap files to analyze the data captured by our GridPot. The Conpot log captured high-level interaction data including timestamps, source IP addresses and ports, protocols used, and basic protocol information, and it tagged each interaction with a unique descriptor for source-flow analysis. Wireshark measured traffic patterns and also provided more detailed information for anomalies highlighted with the Conpot log. Wireshark tools provided several useful scripts for conducting analysis. The Conpot log was specific only to our monitored protocols.

Additional knowledge of other attack vectors, like a high number of packets sent to destination port 443, could prove useful for future work. Abnormally large traffic periods were easily found by saving Wireshark Pcaps every three hours. Pcap file sizes 10 times greater than the others stood out for immediate analysis. Hard drive and virtual-memory size constraints were not an issue during our research, but could become a problem the longer the honeypot remained active due to multi-tool logging using Wireshark.

THIS PAGE INTENTIONALLY LEFT BLANK

VI. CONCLUSION AND FUTURE WORK

A. CONCLUSIONS

Much time was spent reconfiguring GridPot to ensure our honeypot and modeling layers were working and integrating properly. This left only a 19-day period for live data collection. Our GridPot implementation proved successful at collecting intelligence for threat analysis during this period. It is apparent that attackers will exploit any web-based vulnerabilities in a SCADA external interface to gain access to the system, as seen in the large amount of HTTP traffic captured by our honeypot. This highlights the need to have SCADA administrators and operators trained on information security to ensure unauthorized web-based access attempts are stopped. Our GridPot also captured MODBUS scanning, indicating our simulated grid was realistic enough to encourage specialized-protocol reconnaissance even though it was fingerprinted as a honeypot by Shodan.

B. FUTURE WORK

There are many different paths for follow-on work using GridPot. As mentioned, GridPot runs on comparatively old versions of Conpot and GridLAB-D. We recommend updating GridPot using the newest versions of Conpot and GridLAB-D. Rebuilding efforts would need to include the GridPot layer links shown in Figure 7 as well as the additional GridPot Python scripts in the original source code under the Conpot protocol subdirectory.

Adding a SCADA operator interface is desirable since most attacks were using HTTP. This would add an additional layer of deception, which could increase the length of time attacks would interact with the honeypot. Presenting a SCADA operator interface would also make a more attractive target as it would be more believable than the read-only web page which GridPot currently uses.

We recommend integrating IED device simulations to learn more about attacks using IEC 61850 communications. This could be simulated similar to the original GridPot source code under electric-components subdirectory, or by connecting a real physical device which could provide real-time visual feedback.

Future work could also increase deception within GridPot or increase interaction through use of honeynets. Deception could be added in the Internet registry or DNS information as a better way to hide the honeypot's location. Attackers can easily fingerprint our honeypot currently by searching a WhoIs registry to determine ownership and note the tie to NPS. Using honeynets, multiple connected honeypots, could increase the number of attacks by presenting more interfaces with which to interact, although it would be harder to set up and maintain a honeynet.

A final suggestion is to use a low-maintenance and easily deployable honeypot like T-Pot, an open-source multi-honeypot platform that can be configured to run dockerized versions of well-established honeypots including Conpot [57].

APPENDIX A. GRIDPOT SETUP

This appendix summarizes the steps required to install GridPot. We changed our virtual-machine network setting to use NAT during this setup.

1. Installing dependencies (from the command line)

```
$ sudo apt-get update

$ sudo apt-get upgrade

$ sudo apt-get install autoconf (this includes automake)

$ sudo apt-get install libtool subversion python-dev mysql-server

$ sudo apt-get install libmysqlclient-dev libmysqld-dev

$ sudo apt-get install libxerces-c-dev python-mysqldb python-pip

$ sudo apt-get install libcurl3 libcurl4-openssl-dev libcurl4-gnutls-dev

$ pip install –U sphinx

$ sudo apt-get install python3-sphinx libxml2-dev libxslt1-dev

$ sudo pip install lxml gevent python-dateutil mixbox

$ sudo pip install pyasn1 pycryptodomex pysmi

$ sudo apt-get install doxygen

$ sudo apt-get install libcppunit-dev libcppunit-doc

$ sudo apt-get install libncurses5-dev libncursesw5-dev
```

2. Pulling GridPot from GitHub

```
$ sudo apt install git

$ git clone [36]
```

3. Setup Conpot

```
$ cd gridpot/conpot/

$ conpot/ sudo make clean (if rebuilding)

$ conpot/ sudo python setup.py install
```

4. Setup GridLAB-D

```
$ cd ../gridlabd/3.1

$ autoreconf -isf

$ make

$ sudo make install
```

5. Setup libiec61850

```
$ cd ../libiec61850

$ make

$ sudo make INSTALL_PREFIX=/usr/local install
```

APPENDIX B. RUNNING GRIDPOT

This appendix summarizes the steps required to clear and enable logging from both the honeypot and modeling layers, enable packet captures, and run GridPot.

0. Save & clear Conpot log

```
$ cat conpot.log >> conpot_original.log

$ vim conpot.log

:0 (to go to beginning of file)

dG (to erase file)

shift+zz (to save & exit)
```

1. Start Wireshark

```
$ sudo wireshark
```

2. Start GridLAB-D model with output to screen and file

```
$ gridlabd -D run_realtime=1 --server --debug --verbose IEEE_13_Node_With_Houses.glm 2>&1 | tee HousesOutput.txt
```

3. Start Conpot using GridPot template

```
$ sudo conpot -t gridpot –v
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. MODIFIED GPM FILE

This appendix summarizes the modifications made to the GPM file. GPM format is a configuration file written using Python's ConfigParser module. This module contains sections beginning with [section] headers, trailed by 'name: value' entries, and ignores comment lines beginning with a '#' symbol [58].

IEEE_13_Node_With_Houses.gpm

```
#GRIDPOT model for conpot instances
[conpot1]
switch: 671-692
transformer: 633-634
regulator: 650630
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. INTERNAL TESTING

This appendix summarizes the steps we completed for internal testing using our interface, the NETSTAT command, and running Nmap, Nessus, and Metasploit scans of our network. We disabled our virtual-machine network settings during this internal testing phase.

1. Web-based Interface

```
Run GridPot (see Appendix B.)

Open web-browser to localhost:80

Open web-browser to localhost:6267/650630/status

Open web-browser to localhost:6267/tp_line_conductor/resistance

Stop GridPot (ctrl+c)

Pull Logs (Wireshark, Conpot, HousesOutput)
```

2. Netstat

```
$ netstat -ano >gridpot/conpot/Print/netstat_beforeStart.txt

Run GridPot (see Appendix B.)

$ netstat -ano >gridpot/conpot/Print/netstat_afterStart.txt

Pull Logs (Wireshark, Conpot, HousesOutput, Netstat output files)
```

3. NMAP Scans

```
Run GridPot (see Appendix B.)

$ sudo nmap -p- -oN nmap_port.txt localhost

$ sudo nmap -v -sV -O -oN nmap_v-sV-O.txt localhost

$ sudo nmap --script modbus-discover.nse --script-args='modbus-
discover.aggressive=true' -p 502 -oN nmap_modbus.txt localhost

Stop GridPot (ctrl+c)

Pull Logs (Wireshark, Conpot, HousesOutput, nmap output files)
```

4. Nessus Scans

```
$ /etc/init.d/nessusd start

Open web-browser to https://localhost:8834

Run GridPot (see Appendix B.)

Run Nessus Advanced Dynamic Scan: set 127.0.0.1 as the target, select 'scan operational technology devices' (found under Host Discovery) and match plugin description contains SCADA

Run Nessus Advanced Scan: set 127.0.0.1 as the target, select 'scan operational technology devices' (found under Host Discovery) and enable all plugins including SCADA family

$ /etc/init.d/nessusd stop

Stop GridPot (ctrl+c)

Pull Logs (Wireshark, Conpot, HousesOutput, Nessus reports)
```

5. Metasploit Scans

```
Run GridPot (see Appendix B.)

$ msfconsole

$ use auxiliary/scanner/scada/modbus_findunitid

$ show options

$ set RHOST 127.0.0.1

$ run

$ back

$ use auxiliary/scanner/scada/modbusdetect

$ show options

$ set RHOST 127.0.0.1

$ run

$ back

$ use auxiliary/scanner/scada/modbusclient

$ show actions
```

```
$ set ACTION READ_COILS (READ_REGISTERS)

$ show options

$ set RHOST 127.0.0.1, RPORT 502, UNIT_NUMBER 1, NUMBER 1 (10, 100)

$ set DATA_ADDRESS 1 (10001, 30001, 40001)

$ run

$ back

$ exit

Stop GridPot (ctrl+c)

Pull Logs (Wireshark, Conpot, HousesOutput, msf output files)
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX E. HOST-TO-VIRTUAL MACHINE TESTING

This appendix summarizes the steps we completed for host-to-virtual-machine scanning using Nmap, Nessus, and Metasploit. We changed our virtual-machine network setting to use Host-Only Adapter during this testing phase. We testing our connectivity using the ping command to/from our host (X.X.X.1) and client (X.X.X.101) and confirmed receipt of a 'network is unreachable' error when trying to ping an arbitrary IP address.

1. NMAP scans

```
See Appendix D.3., change localhost to X.X.X.101
```

2. Nessus Scans

```
See Appendix D.4., change target from 127.0.0.1 to X.X.X.101
```

3. Metasploit Scans

```
See Appendix D.5., change RHOST from 127.0.0.1 to X.X.X.101 and run
all scans, including additional scan and dos attack listed below before
exiting.

$ use auxiliary/scanner/scada/profinet_siemens

$ show actions

$ show options

$ set INTERFACE enp0s3

$ run

$ back

$ use auxiliary/dos/scada/siemens_siprotec4

$ show options

$ set RHOST X.X.X.101

$ run
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

[1] *Department of the Navy Critical Infrastructure Protection Program*, SECNAV Instruction 3501.1D, Secretary of the Navy, Washington, DC, USA, 2018.

[2] NIST, *NIST roadmap for improving critical infrastructure cybersecurity*, 2014. [Online]. Available: https://www.nist.gov/sites/default/files/roadmap-021214.pdf

[3] K. Stouffer, V. Pillitteri, S. Lightman, M. Abrams, and A. Hahn, "Guide to Industrial Control Systems (ICS) security," NIST, Gaithersburg, MD, USA, NIST SP 800-82r2, 2015.

[4] J. Manfra, "NCCIC year in review 2017: Operation Cyber Guardian," DHS, NCCIC, 2017.

[5] NCCIC, "CRASHOVERRIDE malware," Washington, DC, USA, Alert ICS-ALERT-17-206-01, 2017.

[6] R. Lee, "CRASHOVERRIDE: analysis of the threat to electric grid operations," DRAGOS Inc, Hanover, MD, USA, version 2.20170613, 2017.

[7] C. A. Theohary, "Cyber operations in DoD policy and plans: issues for congress," CRS Report No. R43848, 2015.

[8] S. Sridhar, A. Hahn, and M. Govindarasu, "Cyber–physical system security for the electric power grid," *Proceedings of the IEEE*, vol. 100, no. 1, pp. 210–224, Jan. 2012.

[9] U.S.-Canada Power System Outage Task Force, "Final report on the August 14, 2003 blackout in the United States and Canada causes and recommendations," U.S. Dept. of Energy, Washington, DC, USA, 2004.

[10] M. Nawrocki, M. Wählisch, T. C. Schmidt, C. Keil, and J. Schönfelder, "A survey on honeypot software and data analysis," *Cornell University*, vol. arXiv:1608.06249v1, Aug. 2016.

[11] GridPot, "GridPot: symbolic cyber-physical honeynet framework," Accessed November 3, 2018. [Online]. Available: gridpot.org

[12] Pacific Northwest National Laboratory, "GridLAB-D main page," Accessed February 22, 2019. [Online]. Available: http://gridlab-d.shoutwiki.com/wiki/Main_Page

[13] R. A. Grimes, *Hacking the Hacker*. Indianapolis, IN, USA: John Wiley & Sons, Inc., 2017.

[14] F. Zhang, S. Zhou, Z. Qin, and J. Liu, "Honeypot: a supplemented active defense system for network security," in *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2003, pp. 231–235.

[15] N. C. Rowe, Honeypot deception tactics. Chapter 3 in E. Al-Shaer, J. Wei, K. Hamlen, and C. Wang (Eds.), *Autonomous Cyber Deception: Reasoning, Adaptive Planning, and Evaluation of HoneyThings*, Springer, Chaum, Switzerland, 2018, pp. 35-45.

[16] L. Spitzner, "The Honeynet Project: trapping the hackers," *IEEE Security Privacy*, vol. 1, no. 2, pp. 15–23, Mar. 2003.

[17] C. C. Zou and R. Cunningham, "Honeypot-aware advanced botnet construction and maintenance," in *International Conference on Dependable Systems and Networks (DSN'06)*, 2006, pp. 199–208.

[18] P. Sokol, J. Míšek, and M. Husák, "Honeypots and honeynets: issues of privacy," *EURASIP J. on Info. Security*, vol. 2017, no. 1, pp. 1–9, 2017.

[19] A. V. Serbanescu, S. Obermeier, and Y. Der-Yeuan, "Threat analysis using a large-scale honeynet," in *Proc. of the 3rd Int. Sym. for ICS & SCADA Cyber Sec. Res*., 2015. [Online]. doi: http://dx.doi.org/10.14236/ewic/ICS2015.3

[20] S. Litchfield, "Honeyphy: a physics-aware CPS honeypot framework," M.S. thesis, Dept. of Elec. and Comp. Eng., Georgia Inst. of Tech., Atlanta, GA, USA, 2017.

[21] Department of Energy, "Infographic: understanding the grid," *Energy.gov*, November 17, 2014. [Online]. Available: https://www.energy.gov/articles/infographic-understanding-grid

[22] North American Electric Reliability Corporation, "Key players," Accessed February 28, 2019. [Online]. Available: https://www.nerc.com/AboutNERC/keyplayers/Pages/default.aspx

[23] "Substations," in *Electric Power System Basics for the Nonelectrical Professional*, John Wiley & Sons, Ltd, 2016, pp. 53–89.

[24] K. D. Thompson, "Cyber-physical systems," *NIST*, June 20, 2014. [Online]. Available: https://www.nist.gov/el/cyber-physical-systems

[25] T. Kiravuo, S. Tiilikainen, M. Särelä, and J. Manner, “Peeking under the skirts of a nation: finding ICS vulnerabilities in the critical digital infrastructure,” in *European Conference on Cyber Warfare and Security; Reading*, Reading, United Kingdom, Reading, 2015, pp. 137–144.

[26] Jmath, “Honeypot or not?” Accessed March 26, 2019. [Online]. Available: https://honeyscore.shodan.io/

[27] L. Rist, “About,” *Conpot ICS/SCADA Honeypot*. Accessed February 23, 2019. [Online]. Available: http://conpot.org/

[28] F. Leder, “About the honeynet project,” *The Honeynet Project*. Accessed March 19, 2019. [Online]. Available: https://www.honeynet.org/about

[29] C. Scott, “Designing and implementing a honeypot for a SCADA network,” *SANS Institute Reading Room*, p. 39, Jun. 2014.

[30] L. Rist, “GitHub - mushorg/conpot: ICS/SCADA honeypot,” February 04, 2019. [Online]. Available: https://github.com/mushorg/conpot

[31] Department of Energy, “GridLAB-D a unique tool to design the smart grid,” August 2018. [Online]. Available: https://www.gridlabd.org/

[32] Pacific Northwest National Laboratory, “Power flow user guide,” *GridLAB-D ShoutWiki*, October 20, 2017. [Online]. Available: http://gridlab-d.shoutwiki.com/wiki/Power_Flow_User_Guide

[33] Pacific Northwest National Laboratory, “GridLAB-D basics,” October 13, 2012. [Online]. Available: http://gridlab-d.shoutwiki.com/wiki/GridLAB-D_basics

[34] Pacific Northwest National Laboratory, “GLM syntax,” *GridLAB-D Wiki*, September 25, 2011. [Online]. Available: http://gridlab-d.shoutwiki.com/wiki/GLM_syntax

[35] Pacific Northwest National Laboratory, “Schedule,” October 05, 2014. [Online]. Available: http://gridlab-d.shoutwiki.com/wiki/Schedule

[36] Sk4ld, *Gridpot*. GitHub, 2015.

[37] R. Fielding *et al.*, “Hypertext transfer protocol -- HTTP/1.1.” The Internet Society, 1999.

[38] P. Ackerman, *Industrial Cybersecurity*. Birmingham, AL, USA: Packt Publishing, 2017.

[39] “Modbus application protocol specification v1.1b3.” Modbus, 2012.

[40] “Modbus messaging on TCP/IP implementation guide v1.0b.” Modbus-IDA, 2006.

[41] T. Wiens, “S7 communication (s7comm),” *Wireshark*, May 13, 2016. [Online]. Available: https://wiki.wireshark.org/S7comm

[42] G. Miru, “The Siemens S7 communication - part 1 general structure,” *GyM’s Personal Blog*, April 08, 2018. [Online]. Available: http://gmiru.com/article/s7comm/

[43] J. Case, M. Fedor, M. Schoffstall, and J. Davin, “A simple network management protocol (SNMP).” Network Working Group, 1990.

[44] R. Latchmepersad and G. Singh, *CompTIA Network+ Certification Guide*. Birmingham, AL, USA: Packt Publishing, 2018.

[45] P. Matousek, “Description of IEC 61850 communication,” Brno University of Technology, Brno, Czech Republic, Technical FIT-TR-2018-01, 2018.

[46] W. O. Redwood, “Cyber physical system vulnerability research,” Ph.D. dissertation, Dept. of Comp. Sci., Florida State Univ., Tallahassee, FL, USA, 2016.

[47] A. Jicha, M. Patton, and H. Chen, “SCADA honeypots: An in-depth analysis of Conpot,” in *Proc. of the 2016 IEEE Conf. on Intel. and Sec. Inf. (ISI)*, Tucson, AZ, USA, 2016, pp. 196–198.

[48] D. Hyun, “Collecting cyberattack data for industrial control systems using honeypots,” M.S. thesis, Dept. of Comp. Sci., NPS, Monterey, CA, USA, 2018.

[49] G. Combs, “About Wireshark.” Accessed March 26, 2019. [Online]. Available: https://www.wireshark.org/

[50] G. Lyon, “Nmap: the network mapper - free security scanner.” Accessed March 26, 2019. [Online]. Available: https://nmap.org/

[51] “Nessus Professional,” *Tenable®*, February 04, 2015. [Online]. Available: https://www.tenable.com/products/nessus/nessus-professional

[52] “Metasploit | penetration testing software, pen testing security,” *Rapid7*. Accessed March 26, 2019. [Online]. Available: https://www.metasploit.com/

[53] F. Baumgarten, “Netstat(8) - Linux man page.” Accessed March 26, 2019. [Online]. Available: https://linux.die.net/man/8/netstat

[54] “Shodan.” Accessed March 26, 2019. [Online]. Available: https://www.shodan.io/

[55] “API Reference — dpkt 1.9.2 documentation.” Accessed May 13, 2019. [Online]. Available: https://dpkt.readthedocs.io/en/latest/api/index.html

[56] “GeoLite2 Free Downloadable Databases « MaxMind Developer Site.” Accessed May 14, 2019. [Online]. Available: https://dev.maxmind.com/geoip/geoip2/geolite2/

[57] Deutsche Telekom AG Honeypot Project, “T-pot: a multi-honeypot platform.” Accessed May 06, 2019. [Online]. Available: https://dtag-dev-sec.github.io/mediator/feature/2015/03/17/concept.html

[58] The Python Software Foundation, “13.2. ConfigParser — configuration file parser,” March 18, 2019. [Online]. Available: https://docs.python.org/2.7/library/configparser.html?highlight=configuration

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California