NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEVELOPING A DUAL-PURPOSE WEB HONEYPOT FOR CHARACTERIZING ATTACKS ON A SIMULATED POWER GRID

by

Andrew D. Sill

December 2023

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

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** December 2023	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE DEVELOPING A DUAL-PURPOSE WEB HONEYPOT FOR CHARACTERIZING ATTACKS ON A SIMULATED POWER GRID	**5. FUNDING NUMBERS**
6. AUTHOR(S) Andrew D. Sill	
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)

Protection of industrial control systems (ICS) is a critical security task since failure can lead to large-scale damage. Exposing these systems to the Internet makes them more useful but also more vulnerable to costly attacks. This thesis explored using honeypots to help defend Internet-connected ICSs. Honeypots are deceptive systems deployed to identify and gather intelligence on cyberattacks. Our work developed a dual-purpose Web server that functions both as a Web honeypot and as the front end of an ICS honeypot simulating a residential electrical microgrid. Our Web server ran reliably and without any identified compromise on a major cloud server. We observed significant scanning, and some HTTP-based attack attempts, including the Mirai botnet malware. Our results showed that the dual-purpose Web honeypot improved data collection and protection of the Internet-exposed user interface of the ICS honeypot. This could help improve the security of critical systems in industries and the federal government including the Department of Defense.

14. SUBJECT TERMS Web honeypot, deception, microgrid, cybersecurity, iec-60870-5-104, power distribution, electrical devices	**15. NUMBER OF PAGES** 97
	16. PRICE CODE

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

DEVELOPING A DUAL-PURPOSE WEB HONEYPOT FOR CHARACTERIZING ATTACKS ON A SIMULATED POWER GRID

Andrew D. Sill
Captain, United States Army Reserve
BS, Tennessee Technological University, 2010
MS, Missouri University of Science and Technology, 2019

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

NAVAL POSTGRADUATE SCHOOL
December 2023

Approved by: Thuy D. Nguyen
Advisor

Neil C. Rowe
Co-Advisor

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Protection of industrial control systems (ICS) is a critical security task since failure can lead to large-scale damage. Exposing these systems to the Internet makes them more useful but also more vulnerable to costly attacks. This thesis explored using honeypots to help defend Internet-connected ICSs. Honeypots are deceptive systems deployed to identify and gather intelligence on cyberattacks. Our work developed a dual-purpose Web server that functions both as a Web honeypot and as the front end of an ICS honeypot simulating a residential electrical microgrid. Our Web server ran reliably and without any identified compromise on a major cloud server. We observed significant scanning, and some HTTP-based attack attempts, including the Mirai botnet malware. Our results showed that the dual-purpose Web honeypot improved data collection and protection of the Internet-exposed user interface of the ICS honeypot. This could help improve the security of critical systems in industries and the federal government including the Department of Defense.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

APCI	application protocol control information
APDU	application protocol data unit
APT	advanced persistent threat
ASDU	application service data unit
HMI	human-machine interface
ICS	industrial control system
IEC 104	IEC 60870-5-104
IOA	information object address
OT	operational technology
SCADA	supervisory control and data acquisition
SQL	server-query language
VM	virtual machine
VNC	virtual network computing
VPN	virtual private network

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to thank my wife, Katlyn, for her wise advice and willingness to sacrifice for my career. You have been a constant source of inspiration and good ideas throughout our marriage. I would not have applied to this program or changed my career without your enthusiastic support.

Professor Nguyen and Dr. Rowe, thank you both very much. You both helped me enormously and guided me through an unfamiliar and challenging process to get to this point. I heard only good things about your team, and I am proud to be a part of it.

I want to acknowledge the efforts of Harold Fisher who served as our subject matter expert on penetration test. Thank you, your results informed many of our design decisions and my applications would have been weaker without your help. It was an educational experience to witness your skills in action; I hope we can work together again.

Dr. Krause, thank you for being a part of the team and helping me troubleshoot Python and Git. Your contribution to the project through code, IoaManager, is much appreciated and stands out for the quality of work provided.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. MOTIVATION

Much of modern life hinges on systems managed by operational technology. This is especially true for the industrial control systems (ICS) that oversee our electrical power, water, and fuel distribution (Di Pinto et al., 2018; Hemsley & Fisher, 2018). As these services grow more complex, it is desirable to integrate newer technologies to cut costs and improve service quality. Now we are connecting these devices to the Internet and to cloud-service providers. However, this shift to online exposes systems to new attack possibilities. The increasing number of recent ICS cyberattacks demonstrates this.

Previous work at the Naval Postgraduate School has focused on an energy-grid ICS named GridPot to detect these types of cyberattacks. A recent thesis introduced a Windows application interface to GridPot to record the attacks (Meier et al., 2022). A challenge was the reluctance of cyber-attackers to work with the industrial control system itself. Instead, they targeted the system that provides access, i.e., the Web interface. This thesis aimed to overcome that challenge by offering a more secure Web interface to the honeypot.

The provision of Internet accessibility to industrial control systems is desirable for improved efficiency but has only been explored recently. Safeguarding these systems starts from the design and continues to their decommissioning, which usually covers a long time. This thesis explores using a robust dual-purpose Web server that concurrently functions as an HTTP honeypot and as a Web front end for a microgrid honeypot. A honeypot is a device solely intended to be a target for attacks. It provides no real service, thus ensuring that most harms to it are confined to the virtual realm. Using honeypots also enables researchers to quickly alter device attributes and to systematically log attacks, obtaining considerable data. This supports and ultimately speeds the task of distinguishing between genuine threats and benign traffic.

We tested the IEC 60870-5-104 network protocol ("IEC 104" for short) because it is widely used and frequently targeted in cyberattacks (Ramirez et al., 2022; Zafra et al., 2022). IEC 104 is predominantly used in Europe and Africa, regions strategically important

to the United States and its allies. An Advanced Persistent Threat (APT) group Sandworm exploited the IEC 104 protocol in two significant cyberattacks against Ukraine, viz., Black Energy and Industroyer, along with the less effective Industroyer2 attack (Hemsely & Fisher, 2018; Mitre Corporation, 2023a). More recent defensive success suggests that the IEC 104 protocol can still be implemented securely with standard network-security controls (Mitre Corporation, 2023b).

The United States Department of Defense has bases worldwide, including the regions where IEC 104 is prevalent. While specific details remain undisclosed for security reasons, these bases often rely on local power grids using IEC 104 for managing power with limited backup capabilities. Other local services such as water distribution and Internet connectivity often also depend on the local power grid. This means that attacks targeting IEC 104 jeopardize not only nearby communities and the host nation, but also U.S. military bases used for humanitarian and peacekeeping efforts.

The United States wants to ensure political stability in Africa, a continent with a significant IEC 104 use. Unstable electrical power in these regions can undermine the United States' humanitarian and peacekeeping efforts. Considering the remoteness of these operations and the high cost of procuring and transporting equipment, the U.S. must prioritize safeguarding these systems. By preventing power grid attacks by terrorists or rival nations, the U.S. can improve the security and support of peacekeeping and humanitarian missions, improving the overall efficacy of these operations.

B. RESEARCH METHODOLOGY

The Naval Postgraduate School (NPS) has researched ICS security, including the IEC 104 protocol, over the past four years. Researchers have developed an improved version of the open-source honeypot GridPot (Redwood, 2016). GridPot is based on modular open-source software and is resource-efficient. Recent NPS research has added more data points beyond GridPot's 13-node grid model (Institute of Electrical and Electronics Engineers, 2004) and hardened the logging process (Meier et al., 2022).

To improve the Web interface for GridPot, we initially considered using GridPot's built-in Web server, which runs through Conpot. Conpot is a low-interaction open-source

honeypot that supports many protocols including some ICS ones (MushMush Foundation, n.d.); however, Conpot was not user-friendly and required a deep understanding of both the service and command templating. The Web server also had an unusual method of internal communication within the Conpot framework, culminating in an HTTP call to another Web server which provided a user interface for GridLAB-D, an open-source electrical power-grid simulator (Pacific Northwest National Laboratory, 2018). To reduce system complexity, we chose modularization. We decided to adopt an approach centered on IEC 104, which enables running our Web server separately from GridPot. This strategy aimed to improve the honeypot's security and emulate more authentic behavior, as all system-related traffic was ultimately converted into IEC 104 packets.

C. THESIS OUTLINE

Chapter II explores the importance of industrial control systems, our dependence on them, and the motivation for the Department of Defense to defend IEC 104 networks. Chapter III provides an overview of GridPot, detailing its initial design, subsequent improvements, identified weaknesses, and its subsequent improvements.

Chapter IV details our integration of new software with GridPot and describes our plan for testing it. Chapter V presents the test results and offers interpretations based on the data. Chapter VI summarizes insights from the project, discusses new questions that emerged, and suggests possible future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND AND RELATED WORK

This chapter provides background information on industrial control systems and recent attacks on them, honeypots, and electrical power systems.

A. OVERVIEW OF INDUSTRIAL CONTROL SYSTEMS

An industrial control system (ICS) is a cyber system comprised of networked devices that control physical processes. These systems receive inputs from sensors or control devices and send outputs to actuators or other devices. While these systems share many similarities with information technology (IT) systems, they are usually called operational technology (OT). In general, operational technology focuses more on safety and consistent operations with low latency, whereas information technology tends to focus on security, ease of use, and bandwidth.

Industrial control systems often support processes that do not tolerate disruption (Stouffer et al., 2023). If the devices are damaged or lose connectivity to control services, the consequences can be catastrophic. Many of these devices are in locations that humans rarely visit, or they are protected by physical enclosures to deter tampering. The processes they support may run without interruption for years or decades. Hence the devices on operational-technology networks are often older than typical devices on information-technology networks. These older devices often lack many security features and cannot support robust software-level security. The need for continuous operation limits upgrading of operational-technology systems. Consequently, OT networks are more vulnerable to cyberattacks than conventional networks.

Understanding the equipment is also a challenge with operational-technology systems. Their networks use different protocols than information-technology systems, with priorities that rarely emphasize security or "plug and play" capabilities. Since industrial control systems are designed and maintained for long-lived, specialized industrial processes, many of their protocols such as IEC 60870-5-104 make it difficult to add new devices (Matousek, 2017).

Power generation and distribution has long been a concern of industrial security, as many cyberattacks such as Industroyer have targeted these services (Hemsley & Fisher, 2018). Power-distribution networks deliver power to customers over long distances, enabling the effect of an attack to be more widespread than on more localized targets such as water distribution and sewage management. An effective attack on a power grid can also disrupt telecommunications, making it an appealing vector for attackers.

IEC 60870-5-104 (IEC 104) is a widely used protocol for managing power distribution systems despite its minimal security. It was standardized in late 2000 and has been revised since (International Electrotechnical Commission, 2023). While some revisions addressed security, the age and constraints of the equipment involved meant these upgrades were not universally applied. As new ICS networks using IEC 104 connect to the Internet, devices operating this protocol risk damage from malicious traffic. A well-designed system will have safeguards like network segmentation to prevent malicious traffic from interacting with these vulnerable devices. However, as the Stuxnet attack demonstrated, even an isolated network can be compromised (Chen & Nimeh, 2011). A less secure system faces additional risks, and a weakly protected information-technology network connected to an operational-technology network can be an entrance to the latter (Mitre Corporation, 2023a).

IEC 104 is an application-layer protocol that uses IP/TCP as the underlying networking protocol. IEC 104 uses three-packet formats for its traffic: an I-frame for sending and receiving data, a U-frame for managing connections between two IEC 104 devices, and an S-frame for synchronization. A typical interaction between two devices begins with a TCP handshake to establish a TCP connection. U-frames will then be issued to test the connection and establish an IEC 104 data-transfer channel. The two devices use I-frames and S-frames to transfer information and acknowledge receipt of data. U-frames close the IEC 104 channel before ending the TCP connection. The sequence of events in a simple connection between GridPot and the application GridPot Talker (GPTalker) is illustrated in Figure 1.

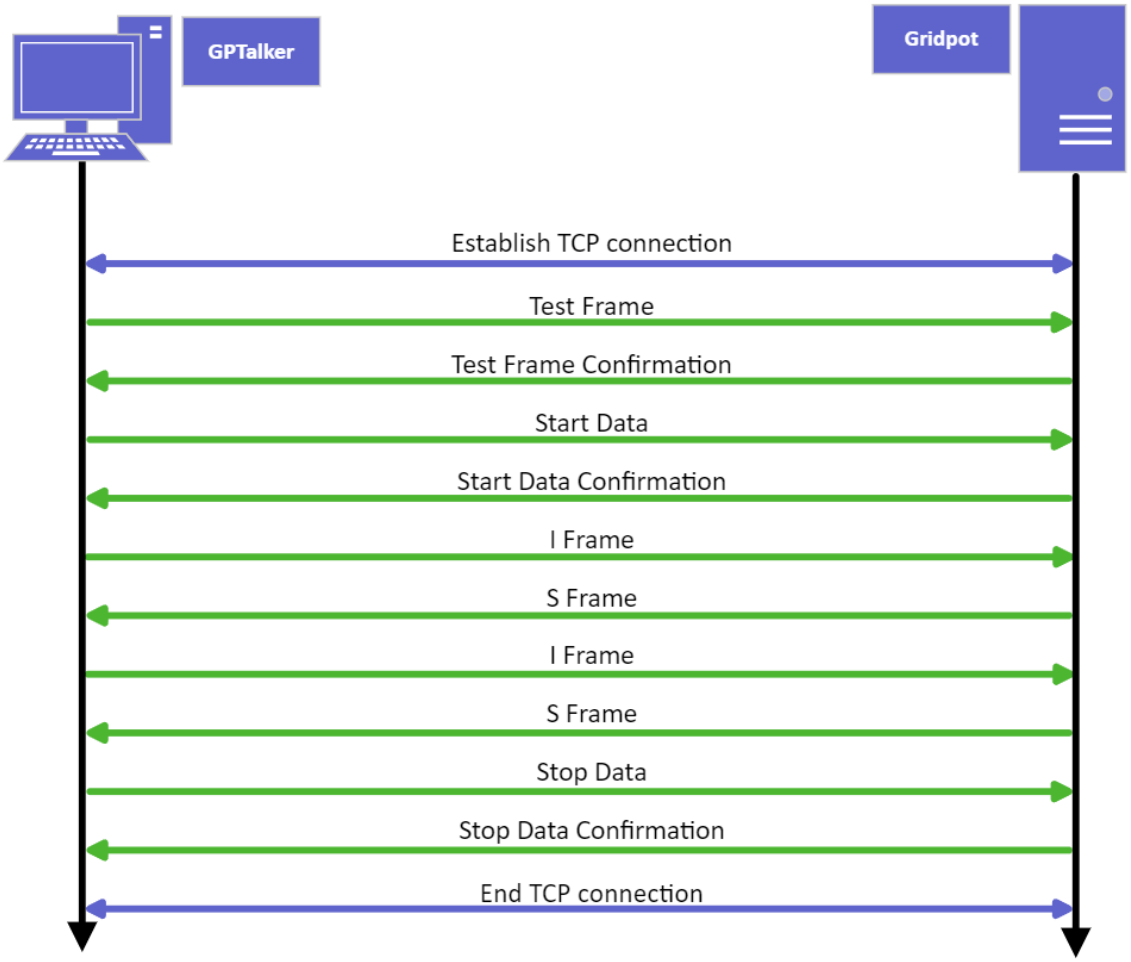

Figure 1. A short IEC 60870-5-104 connection demonstrating the different frames.

B. ATTACKS ON POWER SYSTEMS

Due to its widespread use and security vulnerabilities, cyberattacks on the IEC 104 protocol are often seen at sites using it. Attribution for these attacks is difficult but three major "campaigns" have been reported: Black Energy, Industroyer, and Industroyer2 (Mitre Corporation, 2022; Mitre Corporation, 2023b). Black Energy, the earliest, moved through information-technology networks to find controls of the operational-technology equipment. Black Energy then ran a distributed denial of service (DDoS) attack against vulnerable equipment and networks, blocking operators from controlling their ICSs (Hemsley & Fisher, 2018). It also used network intrusion to control the human-machine interfaces and create outages (Miller, 2021).

Industroyer, also called CRASHOVERRIDE was "the first publicly known malware specifically designed to target and affect operations in the electrical grid" (Mitre Corporation, 2022). It used the IEC 104 protocol, as well as other protocols, to control the devices within the Ukrainian electrical grid and left hundreds of thousands of citizens without power for three hours (Hemsley & Fisher, 2018). Industroyer2, built from successful portions of Industroyer, was actually less successful (Hjelmvik, 2022; Mitre Corporation, 2022). We analyzed public packet captures (PCAPs) and found this attack used properly formed and valid IEC 104 commands to try to damage electrical equipment (Hjelmvik, 2022). This means that the common defensive strategy of looking for malformed packets does not detect these attacks. The understanding required to identify valid components within a power grid and correctly target them leads to the conclusion that Industroyer2 required industry knowledge and detailed intelligence collection. We inferred that the attack failed because of its similarities to Industroyer and that its intended victim was ready for it.

C. ATTACKS ON OTHER ICS SYSTEMS

In 2000 a disgruntled worker was accused of 46 separate attacks on the local sewage system of Queensland, Australia (Abrams & Weiss, 2008). The worker had detailed knowledge of the industrial control systems and the equipment that controls them through a radio connection. He released hundreds of thousands of gallons of raw sewage into the local water systems, causing the local water to "turn black" and killing marine wildlife. This attack was only stopped when the worker was arrested for a traffic violation. This incident demonstrates the impact of insider knowledge and the difficulty in identifying attackers.

Attackers of unknown origin caused physical damage to a steel mill in Germany (Lee et al. 2014). They gained initial access through a phishing (deceptive) email and used that access for reconnaissance on its networks. The attackers demonstrated a good understanding of the industrial process at the mill and could inflict significant damage according to the workers.

Operational-technology networks can be disrupted by attacks on their supporting information-technology networks, as shown in the Colonial Pipeline ransomware attack. Attackers made it into the virtual private network of the Colonial Pipeline company and encrypted key portions of the company's software to hold it for ransom (Beerman et al. 2023). The company shut down its operations, likely for safety and security concerns. Researchers later found no evidence that the attackers accessed the actual industrial control systems or equipment. This demonstrates that a breach into the system through information-technology networks can affect operational-technology networks, even with network segmentation and much security throughout a network.

D. OVERVIEW OF HONEYPOTS

Honeypots are decoy computer systems that are designed to be attacked. For cybersecurity research, they gather information about malware, system vulnerabilities, and attackers. If attackers target a research honeypot, they may reveal a vulnerability or payload before they can use it against a genuine target. Honeypots facilitate data collection because they are generally low-cost and disposable, allowing organizations to deploy them in large numbers over long periods.

For active defense, honeypots can also enhance the security of an ICS by integrating them into both the IT and OT networks. Honeypots can serve as an early warning mechanism, often when deployed on a less-secure machine that is likely to be compromised. Deploying them in the OT and IT networks alongside genuine equipment adds sensing capability for the administrators. Honeypots can detect early signs of compromise such as scanning and lateral movement (Zobal et al. 2019). A honeypot serves no purpose for production, so any connection to it is abnormal and should trigger an alert. Honeypots can be used at many points within the IT or OT network, depending on the needs of the organization (Trend Micro, 2020). Honeypots could be placed in the "demilitarized zone" for a public service, in the testing network, or in the production network to provide detection. System administrators can monitor the honeypots, ensuring that any attack on them immediately alerts security personnel. This approach focuses on the timely detection of a breach rather than its prevention. By identifying threats early,

organizations can respond before the attackers have time to build persistence (footholds) within networks. Some attackers have been known to lurk within systems for years before starting their assault (Hemsley & Fisher, 2018). A commercial version of GridPot known as Q-GridPot provides this functionality to an organization with hardware (Quantalytics, 2020).

Using Web honeypots with HTTP to protect an ICS environment is not new. Researchers in 2013 used a custom Web interface to link a simulated ICS to the Internet (Wilhoit, 2013). They found that attackers used the open HTTP service to enter the IT network, allowing eventual access to the ICS equipment. They observed the behavior of attackers using the HTTP and Modbus protocols and characterized the attacks by the level of effort involved. They found that the attacks followed the traditional stages of reconnaissance, exploration, and persistence.

E. RELATED RESEARCH

Honeypots are widely used for research and security, with likely tens of thousands of honeypots active on the Internet (Morishita et al., 2019). Some of these honeypots belong to research organizations, businesses, and government entities, providing a valuable source of attack information. Including the HTTP protocol in GridPot indicates the value of IT protocol analysis to OT technology (Dougherty, 2020; Mesbah et al., 2023).

One project examined IEC 60870-5-104 security, identifying potential attack vectors (Gyorgy & Holczer, 2020). The protocol has an important security weakness since it cannot validate traffic; this means anyone on its network could send an IEC 104 command to cause a change to an ICS. The protocol is also susceptible to "adversary in the middle" and "replay" attacks. Furthermore, though it is not a flaw of the protocol itself, they found they could create denial of service by sending IEC traffic with out-of-sequence numbers.

Another project proposed a "stateful intrusion detection" method to protect ICS networks (Yang et al., 2014). This defined abnormal behaviors such as an incorrect sequence of events, malformed packets, and an improper flow of traffic. They created a rule-based application that processed a packet capture and identified anomalous packets.

Their testing on a mix of intentionally malicious packets, corrupted packets, and legitimate traffic identified all anomalous entries. However, this method of defense requires significant effort to create and has limited generality. The stateful intrusion detection rules must be custom-built for each system, as in some systems a behavior would be considered anomalous while in others it would be routine traffic. This system also required updating as the configuration of the network changes as a result of device addition or removal.

A later project prototyped a "multivariate" intrusion-detection system that combined access controls and outlier detection based on machine learning (Grammatikis et al., 2020). Their system analyzes the flows of traffic within the network through two main modules. The first compares the actions to a user-defined source-address access control list and identifies anomalies as traffic originating from outside the control list. The second module uses an unsupervised learning model to identify anomalous traffic flows based on their similarity to other flows. This system requires additional software within the network but achieved an accuracy of 98% and F1 score of 87% when identifying malicious traffic.

An industry project developed an interesting ICS honeypot (Hilt et al., 2020). They realistically simulated an industrial control system, provided an interface for it, and allowed unauthorized users to access it through virtual-network computing (VNC). Their experiment ran for eight months, and mostly attracted scanners and malicious actors. Until the final month of their experiment, little interaction with the simulated ICS occurred. They concluded that it was a challenge to find attackers interested in more than ransomware and crypto miners (Hilt et al., 2020).

Recent work on ICS and SCADA attacks used Conpot, a low interaction honeypot that supports the HTTP, SNMP, Modbus, S7Comm, and BACNET protocols (Mesbah et al., 2023). They describe how a honeypot using HTTP can help protect an OT network and experimented with Conpot in particular. They connected the honeypot to the Internet and analyzed the traffic received. They reported that the HTTP protocol was used by the most IP addresses, accounting for over 83% of the sessions. This research suggests that an associated HTTP service could be an attractive target and lead to more interactions for an ICS honeypot.

Several options for running an industrial control system honeypot offer HTTP services (Dalamagkas et al., 2019). Conpot is a commonly used open-source honeypot with community support and continuing development. Including HTTP in Conpot makes it a more likely target for attackers (Dalamagkas et al., 2019).

Our project at NPS built upon a honeypot named GridPot (Redwood, 2016) as mentioned in section I.B. We have several custom variants of GridPot for security research. An early project deployed GridPot on a standalone machine connected to the Internet (Kendrick & Rucker, 2019). It configured ports for the HTTP, MODBUS, and S7Comm protocols and found that all three were routinely scanned. This suggested that an HTTP interface would be fruitful as it was the most frequent protocol. A subsequent project added a user interface to a Windows desktop by the Remote Desktop Protocol and let users interact with GridPot (Dougherty, 2020). This made the ICS more accessible and increased attacks. However, attackers preferred to compromise the Windows operating system and the Remote Desktop Protocol rather than engage the ICS simulation (Ramirez et al. 2022). Another project added security and remote-logging features to "harden" the honeypot system against attacks (Meier et al., 2022).

III. SYSTEM DESIGN

Gathering data on industrial-control-system attacks is a challenge. In real-world incidents, attackers often cover their tracks by deleting software, corrupting firmware, or damaging hardware (Redwood, 2015). A victimized organization may withhold or limit information about the breach due to concerns about security or damage to their public image (Lee et al., 2014).

On the other hand, it is difficult to entice attackers to engage with the industrial control systems (Hilt et al., 2020). Most attackers ignore the ICS features and focus on exploiting the host system instead, a target usually more useful to them. This behavior has been observed in two previous studies at NPS (Dougherty, 2020; Meier et al. 2022) that used a modified GridPot. A weakness of the previous implementation was its Web server, which only displayed a Web page showing some devices and did not provide any user interaction. The Web server running through Conpot did receive a few HTTP messages about simulated data from GridLAB-D but did not simulate a real ICS well nor did it provide an attractive attack surface.

To improve the user interface, we created a custom Web server called the SCADA Web Server (SWS), and a middleware program named GPTalker that is used by SWS to communicate with GridPot over the IEC 104 protocol. SWS and GPTalker are Python programs that can function, in a reduced capacity, independently from each other. This let us upgrade them separately.

Figure 2 shows the work that we added or modified for this project in green and existing services that we used for this project in blue. Red represents the attackers and their access to the system. Orange represents the firewall that only allowed HTTP traffic, over port 80, to the SCADA User Interface Server. The firewall blocked all other requests to that server, as well as external access to the logging server and GridPot server. All servers shown in Figure 2 run on the Linux operating system.

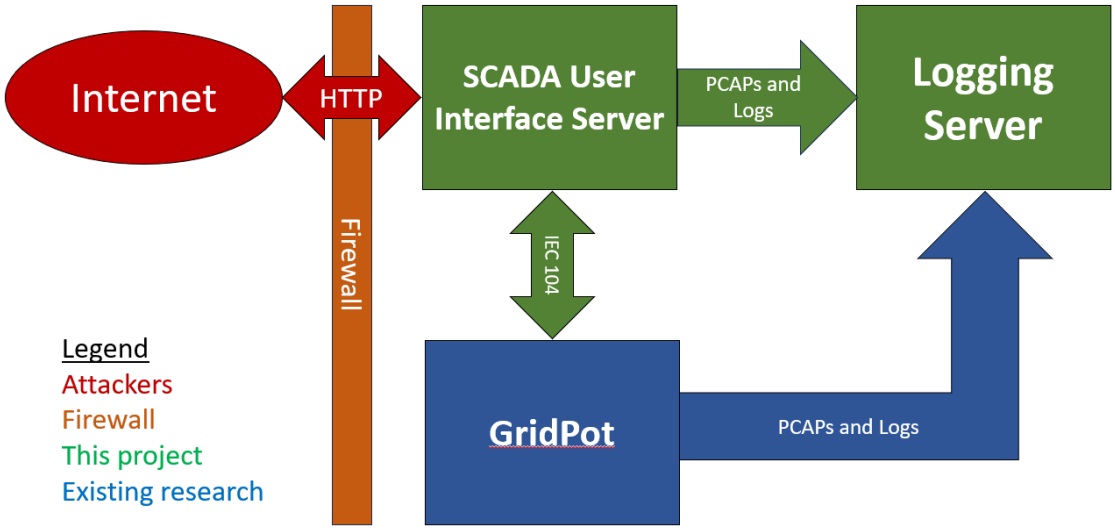

Figure 2. High-level overview of the honeypot's components.

The high-level design of the dual-purpose Web honeypot is shown in Figure 3. The main components of this system are: SCADA Web Server which acts as a honeypot and provides a Web-based human-machine interface to interact with the ICS, GridPot Talker which allows the Web server to communicate with Conpot, Conpot which translates the protocols and communicates with the power-grid simulator, GridLAB-D. The logging server runs a few services through the Linux operating system to retrieve and store logs from the other servers.

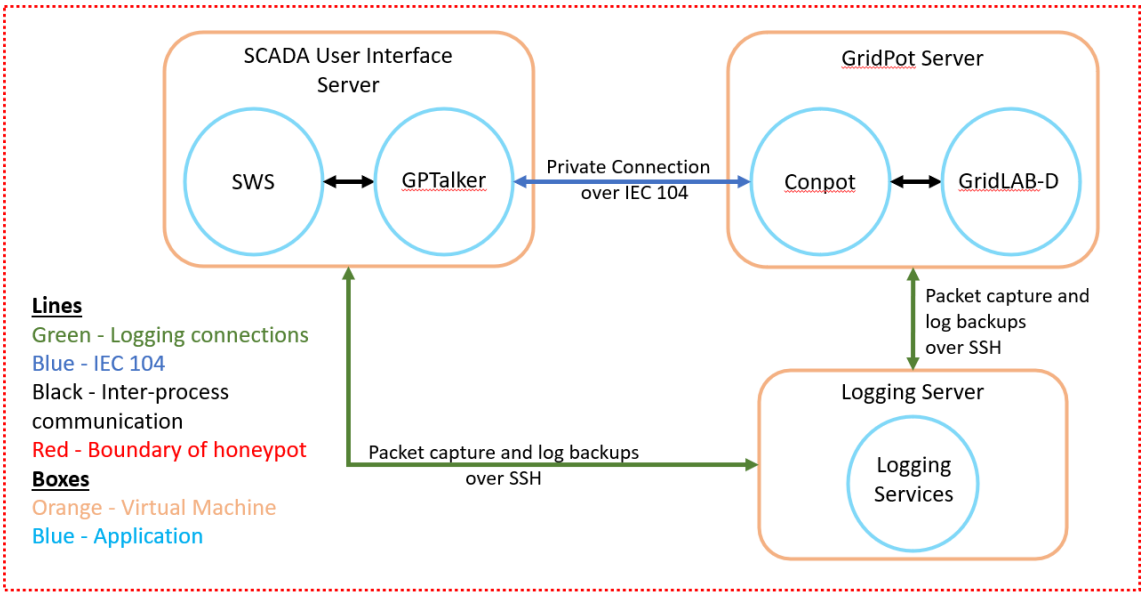

Figure 3. Overview of dual-purpose Web honeypot.

A. IEC 104 PROTOCOL

The GPTalker application communicated with GridPot's IEC 104 server using the IEC 104 protocol. Restricting GPTalker to only IEC 104 ensured that the behavior within our system was consistent with an industrial control system and had plausible operational-technology traffic from human-machine-interface engagement. In IEC 104 terminology, the data structure carried by TCP is called a frame or, more formally, an Application Protocol Data Unit (APDU). A frame starts with a byte of hexadecimal value 68, followed by a byte that specifies the remaining length. The first section of frame contains an Application Protocol Control Information (APCI) structure. IEC 104 uses three kinds of frames: the I format for transmitting information, the S format for synchronizing the flow, and the U format for controlling the connection. The format of an APDU is shown in Figure 4.

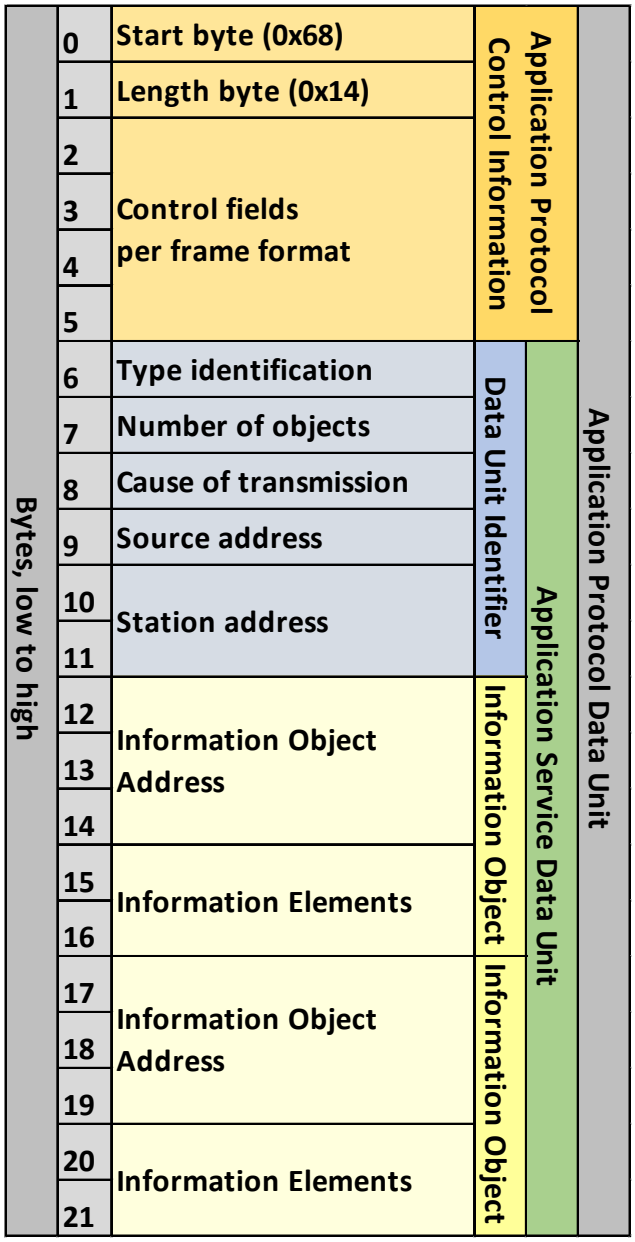

Figure 4. Visualization an I-format APDU.

The three APDU formats are identified by the four bytes after the start byte and the length byte, as shown in Figure 5.

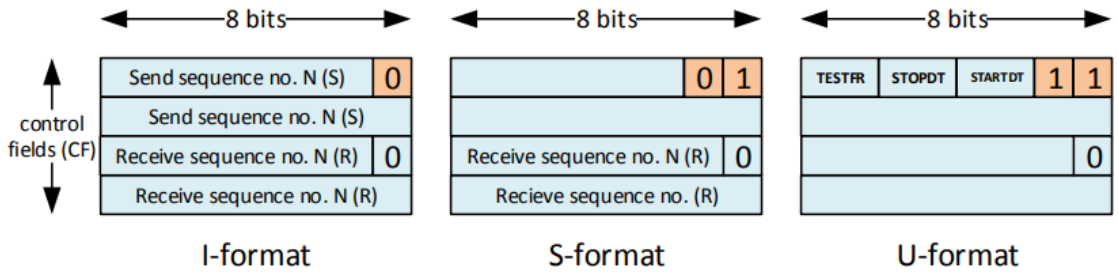

Figure 5. The three formats for IEC 104 APDUs. Source: Matousek (2017).

The I-format frame is the primary method of transmitting and receiving information between devices. It includes both a sending sequence number and a receiving sequence number. IEC 104 devices monitor these sequence numbers to ensure they process traffic in the correct order and that information is not lost. Only the I-format frame contains an Application Service Data Unit (ASDU) structure.

The S format synchronizes the devices, especially when the traffic flow is imbalanced. If one device is sending less traffic, it will periodically send an S-format frame to update the other device with the last received sequence number. Although devices can handle sequence number issues differently, GridPot disconnects a device that is more than 100 sequence numbers behind.

The U format tests and manages the data connection. The "start data transfer" and "stop data transfer" control fields establish data connections between the controlling and controlled device. GridPot requires a U-format frame to start the data transfer, including acknowledgement from the other device, before it will handle frames of I or S format. Once GridPot receives the U-format "stop data transfer" frame, it will only respond to other U-format traffic.

The I-format frames carry additional data in the form of an ASDU. We observed that GridPot will send multiple frames in a single TCP packet such that one frame, beginning with a 0x68 byte, starts immediately after the previous frame's information element ends. An ASDU can hold multiple information objects. The size of an APDU varies as shown in Figure 6.

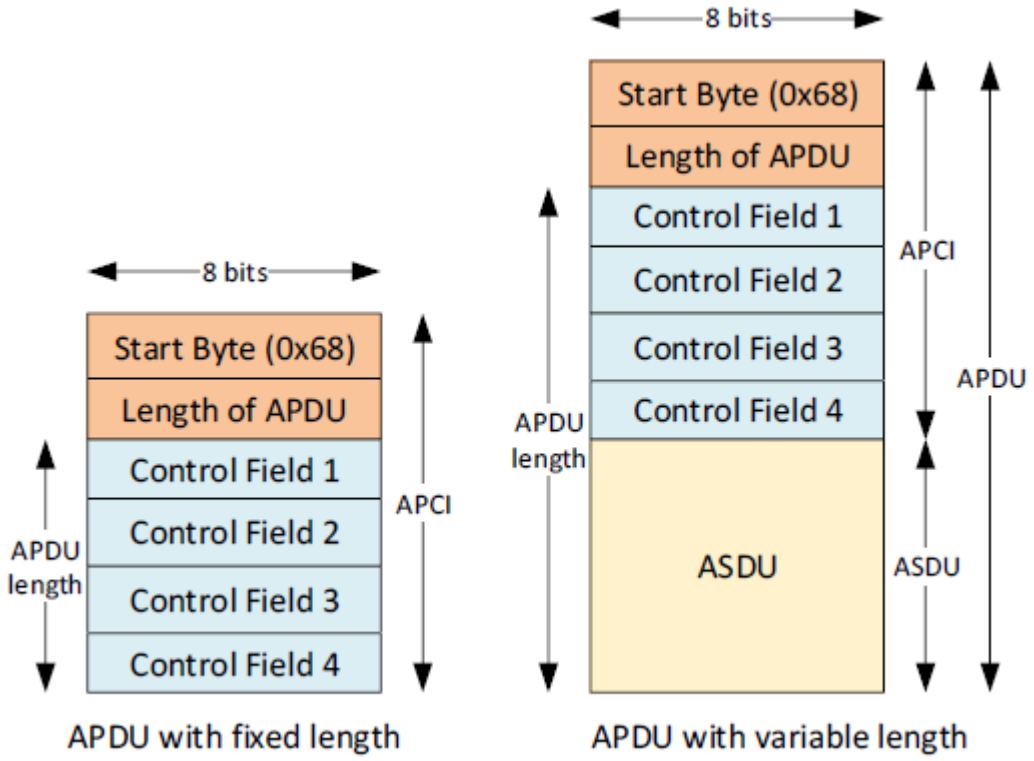

Figure 6. The two formats of Application Protocol Control Information. Source: Matousek (2017).

The ADSUs specify actions to be executed and identify the recipient device. Typically, ten to twelve fields within the ASDU control the behavior of the receiving device. The ASDU format is shown in Figure 7.

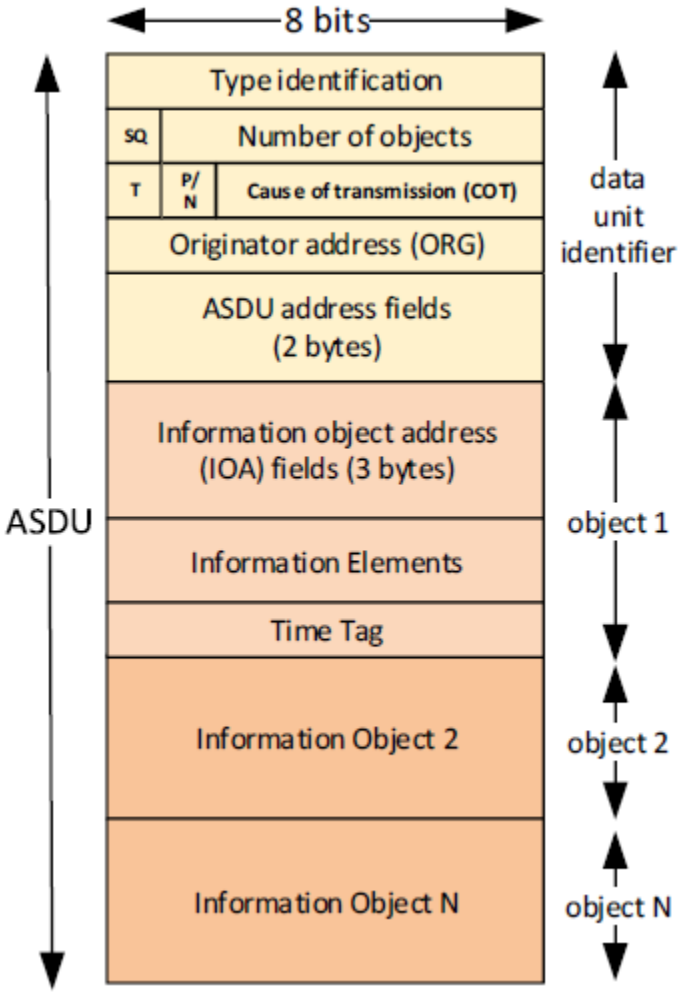

Figure 7. Application Service Data Unit format. Source: Matousek (2017).

The important fields in the ASDU are type identification, address fields, Information Object Address (IOA) fields, and information elements. The type-identification field specifies whether this is a command or request for data. Despite 255 potential type-identification values (excluding zero), GridPot uses less than a dozen in practice. The ASDU address field, also called the "station address," specifies which devices these commands affect. If the sender of the IEC 104 frame wants to address more than one station at a time, they have two options; they can send an ASDU for each address, or set the ASDU address field to 65,535 as a "broadcast" address. The broadcast address can be disabled on devices for security reasons.

The Information Object Address designates the logical address set to either receive a command or provide data. A device can have hundreds of such addresses as each sensor and actuator could have a distinct address. The address is three bytes of data, but the third byte is used to "define unambiguous addresses within a specific system" (Matousek, 2017). A systems designer could use the third byte to extend their address space by a factor of 256. A single ASDU can contain multiple information objects with both Information Object Addresses and information elements, so it can address up to 127 information objects.

An IEC 104 command has three major components. The first is command type that determines what type of interaction will happen such as setting a variable to a new value or requesting updates from a device. The second component is the Information Object Address, which names the affected object or variable, and the third component is the information element which contains the required information to complete the command. Figure 8 shows these components as expressed in a simple pseudo-code statement.

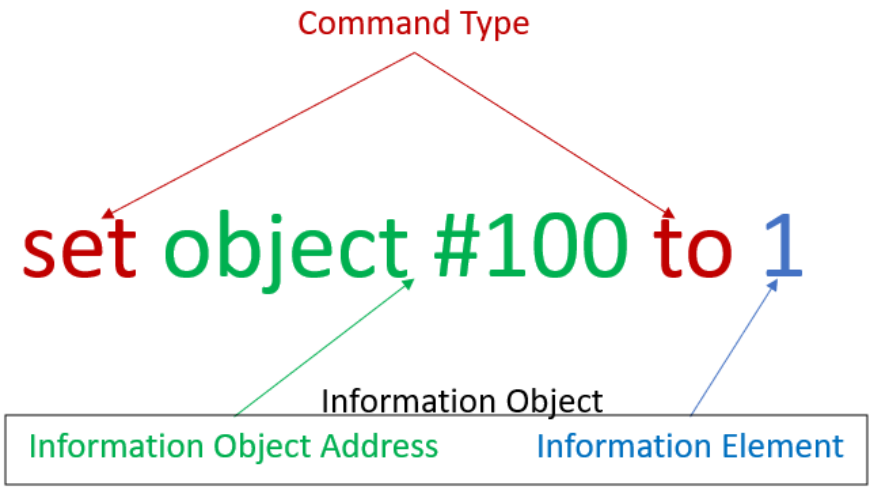

Figure 8. Components of an IEC 60870-5-104 command.

B. SCADA USER INTERFACE SERVER

The design of the SCADA User Interface Server is shown in Figure 9. It handled all Web requests and GPTalker handled all communication with GridPot. The SCADA

Web server (SWS) and the GPTalker application communicated through two shared files, a text-based commands file and a text-based JSON-formatted data file. SWS used the data from the JSON file to fill in information on the webpage for the attacker. Each application read its input file every two seconds, and created a new output file each time it had an update.

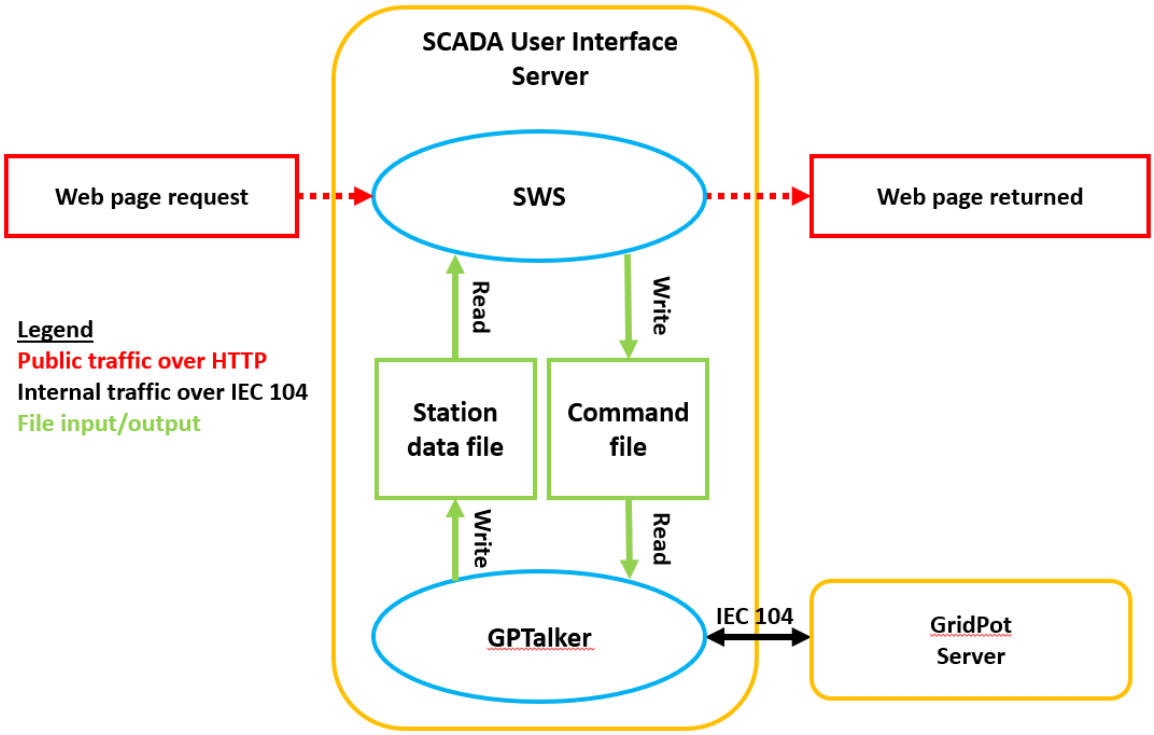

Figure 9. Communications Flow for SWS and GPTalker.

C. GPTALKER APPLICATION

The GPTalker Python program handled the IEC 104 communication between the Web server and GridPot with shared input and output files. GPTalker periodically generated a Javascript Object Notation (JSON) file for each station (labelled as "Station data file" in Figure 9). This file contained information on all IOAs for which the GPTalker application received updates during a particular session. Information stored in the station data file included the most recent value for each IOA along with historic information such

as highest and lowest value recorded, which can be displayed by the SCADA Web server on the main Web page. Every two seconds, GPTalker read a text file produced by the SCADA Web server (labeled as "Command file" in Figure 9) to create IEC 104 frames to send to GridPot. Each line of the text filed contained an Information Object Address and the desired value that the IOA should have. For commands GPTalker translated the values in the command text file into the appropriate IEC 104 commands using an internal key-value mapping between IOAs and object type. This mapping allowed the program to determine the proper data and command formats for an Information Object Address. GPTalker provided connection status output to the user, and the only accepted input at the Linux command-line interface was the command to end the application; all inputs to interact with GridPot were received from the command text file. The GPTalker application did not display device information received from GridPot to the user; that information was saved in the JSON file. The run time output was saved to a text file, "lastgptalkeroutput.txt," to provide debugging information and rewritten on each execution. The GPTalker application was standalone and could communicate with GridPot, without user or SWS input, to update values displayed to the user. It automatically sent "interrogation commands" to GridPot upon establishing the data transfer. GridPot provided updates on devices every few seconds which keep its TCP connection from timing out. GPTalker also sent interrogation commands if it did not receive updates from GridPot within the predetermined threshold of 5 seconds, thus preventing GridPot from ending the connection due to time out.

An example station data file created by the IoaManager class inside GPTalker is shown in Figure 10. The example shows the data for a device at station address 7720 with three associated IOAs: 27653 (transformer voltage out), 27661 (meter 2), and 27396 (switch voltage out). Historic information such as lowest value of individual IOA observed, highest value observed, and total observations is included. The lowest value and highest value are the values reported by GridPot; the running average is the mean of all observed values from the start of the program to the time the station data file was updated.

```
{
    "station_address": "7720",
    "ioa_data": {
        "27653": {
            "current_value": "56561.33",
            "total_count": 1626,
            "historic_low": "56556.66",
            "historic_high": "59788.57",
            "cumulative_sum": "95065737.59000003",
            "running_average": "58466.0132779828"
        },
        "27661": {
            "current_value": "10690.9",
            "total_count": 1319,
            "historic_low": "10688.3",
            "historic_high": "11480.2",
            "cumulative_sum": "14684191.199999992",
            "running_average": "11132.821228203178"
        },
        "27396": {
            "current_value": "358679.91",
            "total_count": 936,
            "historic_low": "283884.7",
            "historic_high": "403784.84",
            "cumulative_sum": "348796463.4399995",
            "running_average": "372645.79427350376"
        },
```


Figure 10. Data in JSON format from testing runs.

D. SCADA WEB SERVER APPLICATION

The Web server SWS processed the information from the station data file (a JSON file) and displayed it in a user-friendly format. It used cascading style sheets (CSS), Java, and HTML to provide a more appealing and realistic user interface. This functionality was built for the GridPot version used by this work. An example of the interface display is shown as in Figure 11.

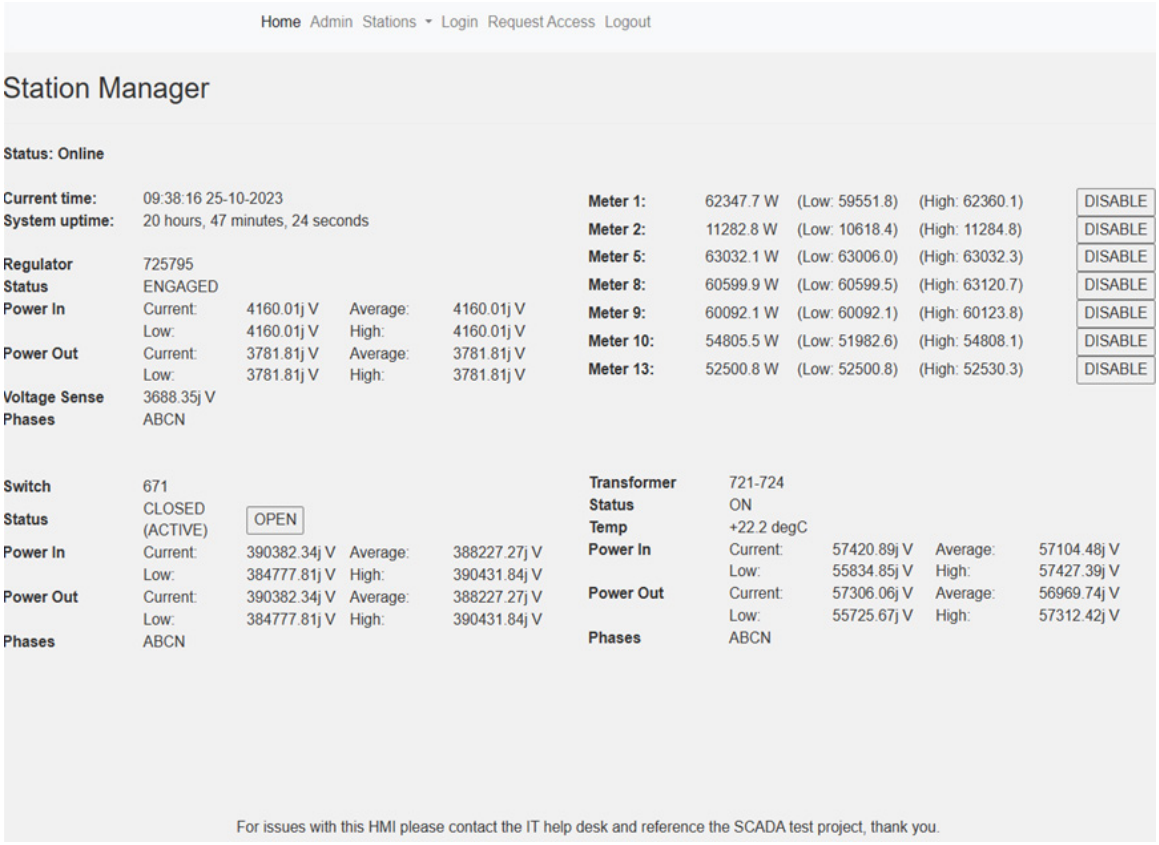

Figure 11. Screenshot of the user interface on the Scada Web Server.

The Web server logged interactions with users, provided debugging opportunities for GPTalker, and acted as a HTTP honeypot. It supported GET and POST HTTP requests, and logged received traffic in several Python-standard logging files. The host system stored all external traffic in packet captures that can be cross-referenced with the logs. The Web interface was intended to encourage users to explore the simulated power grid.

The GridPot version we used supported multiple GridPot meters, a switch, a transformer, and a regulator. We implemented functionality for the Web interface presented by the SCADA Web server to control the switch and meters. The homepage displayed all received information for all devices on GridPot and included additional information such as temperature which was present on the original Conpot-hosted Web page, but not part of the IEC 104 network. The SCADA Web server included other standard Web pages such as a login page, a page for requesting access, and administrative files.

IV. IMPLEMENTATION AND EXPERIMENTATION

A. CONSTRAINTS AND ASSUMPTIONS

The GPTalker and SCADA Web Server (SWS) applications were designed for Linux, and ran as user processes on the base Linux operating system to reduce issues with networking. The SWS application was the Web server for our Web honeypot, and GPTalker was our IEC 104 handler that connected SWS with the simulation software GridPot. We originally tested the SCADA Web server as an application within a VirtualBox virtual machine and found VirtualBox's network address translation interfered with our logging mechanisms. Translation converted all external IP addresses into a single internal IP address for a private network that was exclusive for the host machine and the virtual machine. This meant that packet captures and logs showed all external traffic coming from the same external IP address. Private IP addresses were necessary since each cloud platform could only have one external IP address.

We relied on the cloud-service provider's firewall to restrict network access to our cloud platforms based on a list of their hosted services. We designed GPTalker to communicate with a GridPot instance the same as GridPot's Windows SCADA application does, using only a subset of the available IEC 104 commands. The capabilities of SWS were designed based on the results of previous experiments that showed Web requests did not exceed 10 interactions per minute.

The Web server had no domain name. We chose this to reduce the cost and effort involved in obtaining a domain name and supporting services. The absence of a domain name prevented our research group from using a certificate for the Web server, a necessary feature for the secure HTTP version HTTPS.

B. LOW-LEVEL DESIGN

We used an existing GridPot instance running on the DigitalOcean cloud environment to provide the simulated power grid. Our honeypot instance requires three cloud platforms: a user-interface workstation, a power-grid decoy running GridLAB-D, and a logging server. Each cloud platform is a Linux virtual machine running Ubuntu

20.04.3. GPTalker and SWS ran on the GridPot user-interface workstation, which is called the SCADA User Interface Server for this work (Figure 12).

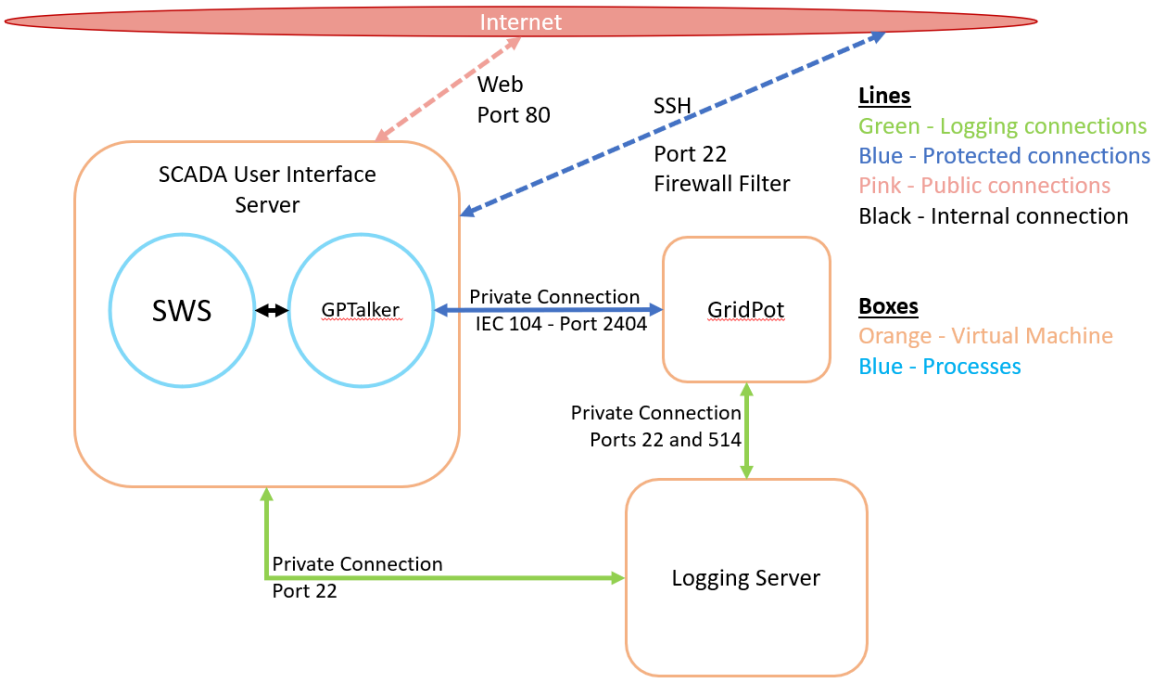

Figure 12. Network configuration; processes in GridPot server and Logging server omitted.

We kept as many logging features from previous research as possible to enable data comparisons. We built the applications to be easy to update and maintain, with scripts for some repetitive tasks. They used public Python libraries and methods to speed development.

SWS and GPTalker ran under a dedicated user account named "SWSuser." This account had no elevated privileges and was not included in the system's "sudoers" list. Were an attacker to compromise the Web server or GPTalker, their ability to execute commands or run code would be restricted by the operating system. Also, the Web server logs and packet captures were backed up every five minutes to preserve data in case of a compromise.

We used the Linux utility "authbind" to allow the Web server to bind to port 80, the default port for hosting HTTP applications. Using authbind let us keep the SWS user account at a lower privilege level while ensuring compatibility with our scripts and the configuration file used by the underlying Gunicorn web-application software (Gunicorn, n.d.).

1. GridPot Talker

GPTalker used a custom Python program "gptalker.py." It had a default hardcoded IP address and TCP port for communicating with GridPot during testing, but otherwise allowed the address and port to be specified with a runtime argument. The program also featured an option to activate a "deception" layer which simulated operational statuses of GridPot's electrical devices. Figure 13 shows the inputs, outputs, and arguments for gptalker.py.

GPTalker	
Execution:	python3 gptalker.py
Option	**Description**
address	IP address of the Gridpot
example:	python3 gptalker.py address=192.0.0.1
port	The port number to connect to the Gridpot
example:	python3 gptalker.py port=2404
deception	Uses deception to simulate the behavior of meters in Gridpot
example:	python3 gptalker.py --deception
help	Displays help for gptalker.py
example:	python3 gptalker.py --help
Inputs	
Stdin (Keyboard)	Ignored except for control-c to exit program
gridpotcommands.txt	Reads commands from file every two seconds as "IOA value" pair
example:	15366 1
Outputs	
Stdout (Display)	Displays information on connection and status of connection
Stderr	Not used
Logging	Not used
gptalker_output/ station_address_7720.json	JSON formatted text file generated by the IOAManager class

Figure 13. GPTalker program runtime arguments, inputs, and outputs.

Besides GridPot, GPTalker could communicate with another IEC 104 device using IEC 104 commands for data transmission and command relays, but additional configurations would be required if the other IEC 104 device used different IEC commands from those used by GridPot. When connected to a GridPot instance, GPTalker received periodic updates on the electrical system's values, transmitted with the associated Information Object Addresses (IOA). If no I-format frames were waiting to be sent to GridPot, it sent an S-format frame to acknowledge receipt. GPTalker used I-format frames to request updates on specific IOAs or to dispatch commands to manage continuous operations. GPTalker maintained communication with GridPot by receiving I-format frames when new data was available or transmitting S-format frames to acknowledge receipt. Information about the connection process and updates on traffic sent or received were displayed on a screen like that in Figure 14.

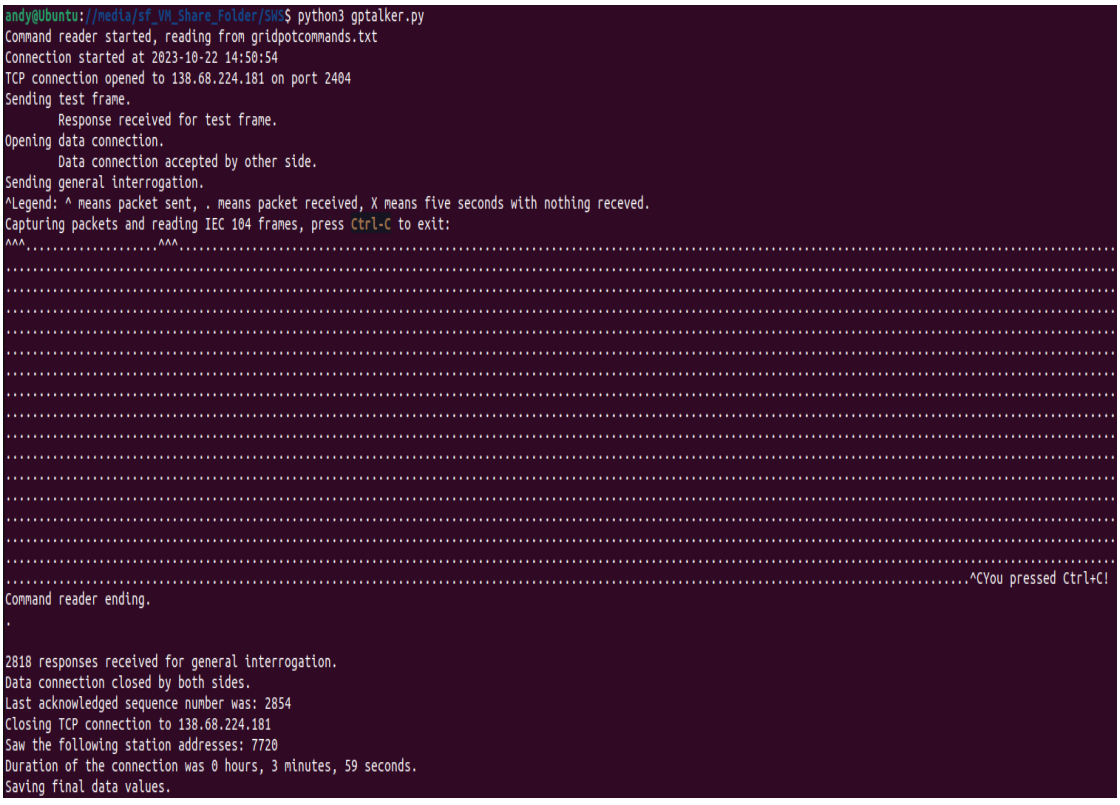

Figure 14. Screen shot of gptalker.py output.

GPTalker used Scapy, a public packet-manipulation module that offered traffic customization (Biondi, 2023). We chose Scapy because other open-source IEC 60870-5-104 protocol modules such as Parser IEC104 and 104server could dissect packets but could not create IEC 104 frames. We used Scapy to create TCP connections between GPTalker and GridPot, create frames, and transfer frames between devices. GPTalker used hardcoded U-format frames to test the connection and start the data transfer. The I-format and S-format frames were created based on information obtained from prior research on GridPot and IEC 104 (Matousek, 2017; Haynes et al., 2023). An example of a packet interrogating an IOA created by GPTalker is shown in Figure 15.

		Purpose:	Find value of IOA 5652 at station 7720	
	Byte Number	**Label**	**Integer**	**Hex**
APDU	0	Start byte	104	68
	1	Length (of bytes after)	14	E
	2	Send Seq	1	00
	3	Send Seq		01
	4	Rec Seq	X	X
	5	Rec Seq	X	X
ASDU	6	Type	1	1
	7	Number of objects	1 so 128 (SQ) + 1 (number) = 129	81
	8	Cause of Transmission	5 (interrogation)	5
	9	Originator Address	0	0
	10	Common ASDU address	7720	1E
	11			28
Information Object	12	Object Address	5652	16
	13			14
	14	Unused		00
	15	Information Elements	0	00
	16			
	17		0	00
	18			

Figure 15. Packet to interrogate IOA 5652 at station 7720.

GPTalker ran as a threaded application. The main thread "general interrogation" watched for the "control-c" command from the keyboard to end the application, and monitored the "listener" and "command reader" threads (Figure 14). GPTalker built an IEC 104 command based on input from the text file "gridpotcommands.txt" and cross-referenced it with an internal mapping of IOAs and types. GPTalker used text files to

communicate with the SCADA Web server to simplify the design, though this is inflexible. IOAs used by GridPot for operations were stored in GPTalker's memory and key-value mapped to their information type. The information object address serves as the key and the information object type as the value. Commands that did not correspond to a mapped IOA were dropped. The reader thread translated the requested command value to the corresponding information-object element value, such as an interrogation or single point command, based on the IEC 104 type identification in GPTalker's internal mapping. This approach made the operations for the Web server easier as it did not have to store or convert IOA-specific data types but instead provided a desired value to GPTalker through the command file (Figure 9). The approach also increased security because an injection attack attempting to insert new traffic would be restricted to the mapped IOAs; anything outside those would be ignored by GPTalker. The listener thread within GPTalker processed incoming IEC 104 frames from GridPot, extracting the ASDU and information objects, converted the values, and saved them in the IoaManager class. After extracting relevant information from the frame, GPTalker discarded the frame. The listener thread handled the multiple frames per TCP packet and multiple IOAs per frame that GridPot often sent. Figure 16 depicts the relationship between the threads within the application.

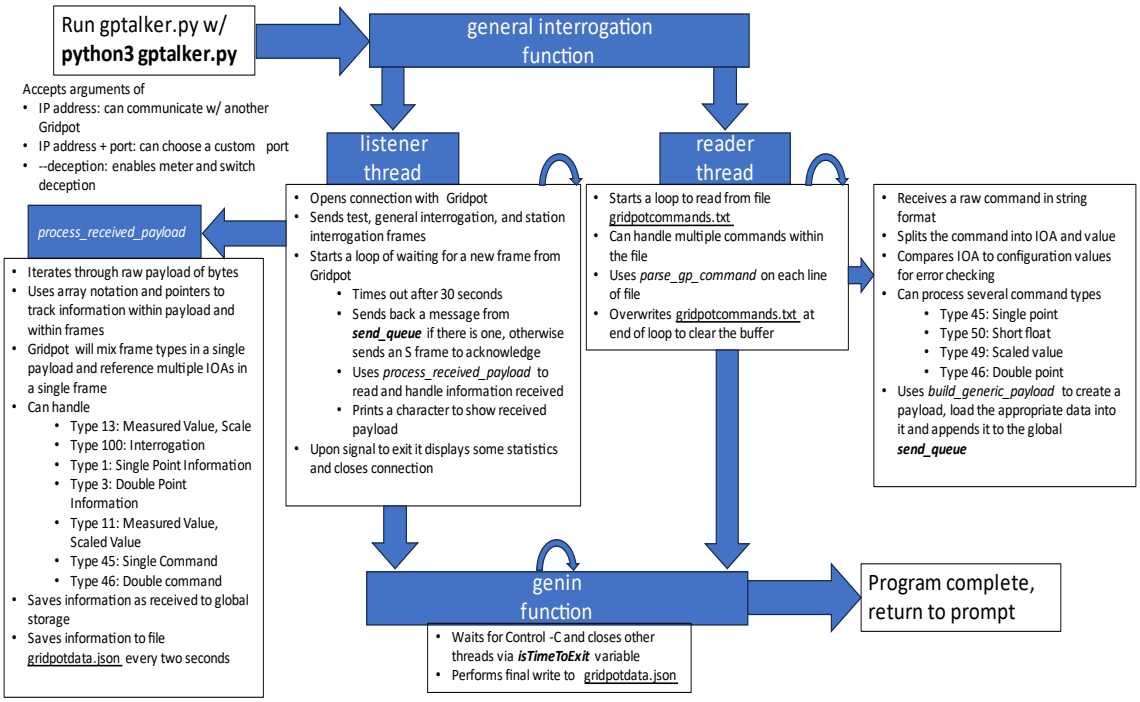

Figure 16. Flow of GPTalker operations.

GPTalker generated frames based on templates that we created from packet inspections and created new frames using the protocol standards. GPTalker monitored the traffic flow from GridPot, and reconnected when no activity on the connection occurred for five seconds. GPTalker saved its internal data including all GridPot information every two seconds, as a file station_address_7720.json in the directory "gptalker_output"; the station address used by GridPot was 7720. A custom Python class, IoaManager, was created to manage the received data from GridPot, perform calculations such as average value observed, and save the converted data in a JSON file (Appendix) which was used by the SCADA Web server to provide device information in webpages.

If the deception layer was enabled, GPTalker simulated the actions of the electrical switch and meters available from GridPot, and maintained the deception throughout its execution. When GPTalker received the information from an IEC 104 frame, it checked whether deception was enabled. If so, it replaced the value received from GridPot with a deceptive value. The substitution was made before the information was loaded into the IoaManager object, so that historical extremes and average values were reflected in the

substituted value. For inactive meters (set to "off"), GPTalker adjusted the value received to zero; for the switch, it verified the status of the corresponding meters and linearly adjusted the switch's output voltage to match. Deception was enabled for all live experiments.

2. SCADA Web Server

The SCADA Web Server (SWS) was designed for ease of use and reliability. It used Gunicorn, free and open-source Web-application software built on Flask. Flask is a public Python module for hosting a simple Web server. The Web server could be launched in debug mode or production mode with a configuration file to simplify operations. Figure 17 shows the interaction between SWS and GPTalker.

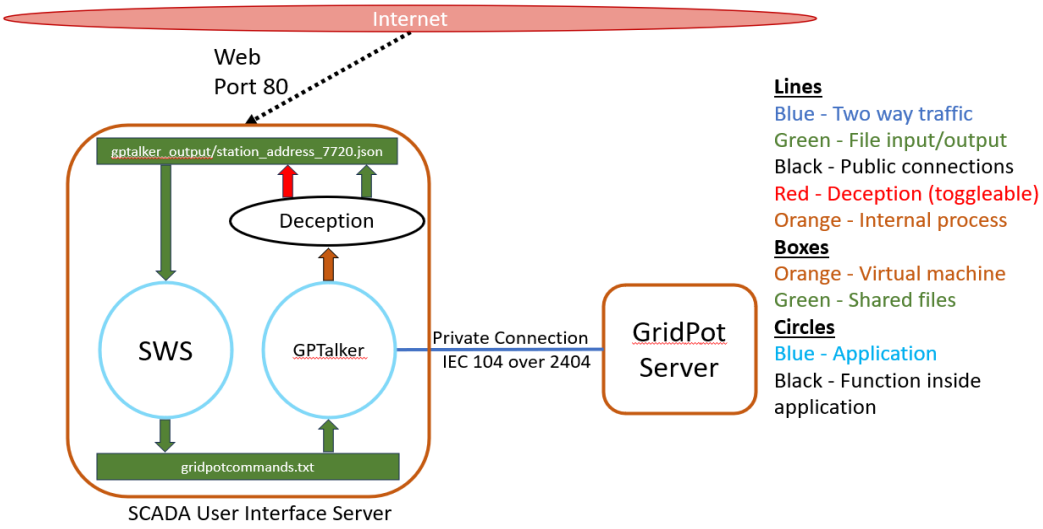

Figure 17. Interactions between SWS and GPTalker.

In Figure 17, SWS and GPTalker are shown running simultaneously on the SCADA User Interface Server and each accessing two shared files. GPTalker also maintained a connection to the GridPot machine and handled deception if enabled. Figure 18 shows the different execution and configuration options for the SCADA Web server.

SWS	
Execution Options	
Debug:	Includes detailed error logs displayed to screen and web page.
example:	python3 sws.py
Production:	Uses configuration file, errors are hidden from users.
example:	gunicorn -c gunicorn_config.py sws:app
Authbind + Production	Uses configuration file, uses port 80 without privileges.
example:	authbind --deep gunicorn -c gunicorn_config.py sws:app
Configuration	**Description**
bind	Controls address restrictions and port
example:	"0.0.0.0":80
threads	Launches multiple threads to serve client requests
example:	multiprocessing.cpu_count()*2 + 1
accesslog	Specify file where to save access (page requests) log
example:	./logs/ + date_string + "_gunicorn_access.log"
errorlog	Specify file where to save error log
example:	./logs/ + date_string + "_gunicorn_error.log"
Inputs	
Stdin (Keyboard)	Ignored except for control-c to exit program
gptalker_output/ station_address_7720.json	JSON formatted text file generated by the IOA_manager class
Outputs	
Stdout (Display)	Displays information on web server status and requested pages
Stderr	Saves a copy of errors to the SWS general log
Logging	Saves copies of page requests and errors to a log in /logs/ directory
Requests	Logs all HTML requests associated with a POST HTTP transaction
Passwords	Logs all username/password attempts
gridpotcommands.txt	Reads commands from file every two seconds as "IOA value" pair
example:	15366 1

Figure 18. SWS configuration options, inputs, and outputs.

We chose Gunicorn due to its support for multiple threads or processes for a Flask application, allowing the Web server to handle several clients simultaneously. This multi-client capability was essential for collecting more data and providing a realistic interface. Flask allowed Python programs to build pages based on Jinja2 templates (Flask, n.d.). We used this feature to develop functions for managing logins, requesting an account, displaying system data, and interfacing with a back-end application such as GPTalker. Flask was designed to be simple and secure, which let us focus on adding only the features we needed. We relied on Flask to securely handle user input and block unauthorized

resource access. Flask met our needs with a straightforward interface and its many available tutorials.

To improve the honeypot's realism, we introduced a login page as a prerequisite for privileged actions. Instead of using Web cookies to track a user's activities, we implemented a "whitelist" strategy to avoid storing data on the user's computer; the "whitelist" database contains authorized IP addresses that SWS uses to track Web clients. Gunicorn allows threads to share resources such as global variables to implement the whitelist and other data structures holding information from GridPot, as shown in Figure 19.

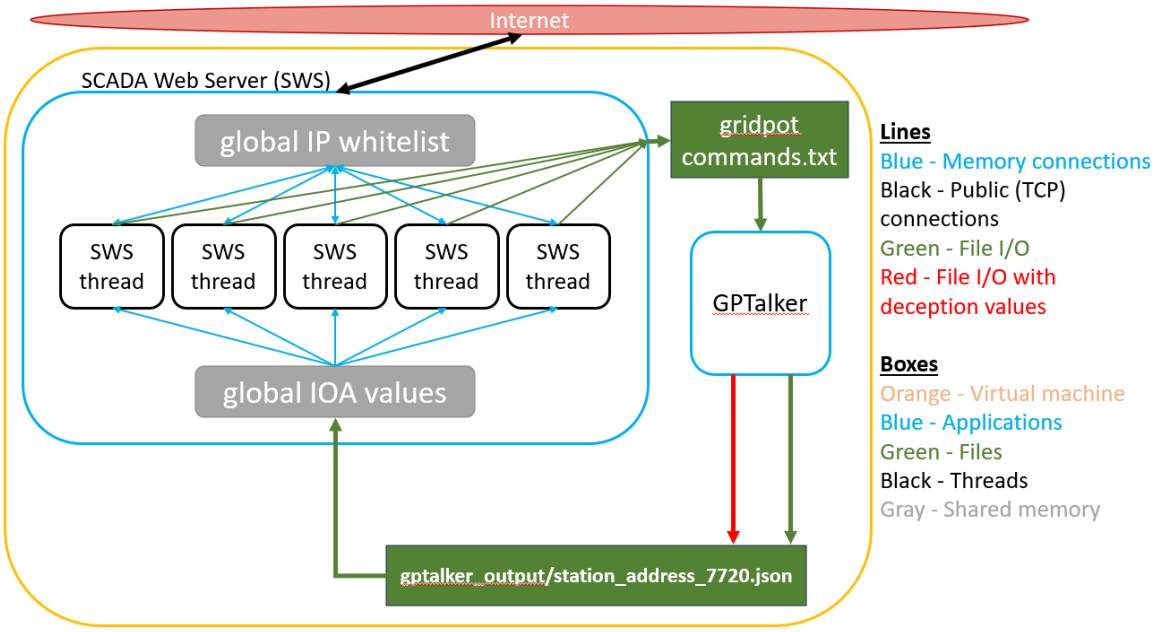

Figure 19. Overview of SWS execution and shared resources.

The Web interface presented by SWS was plain and straightforward. The GridPot software version we used only allowed limited user interaction with a single switch. This switch was associated with safety controls that seemed nonfunctional. GridPot consistently responded to requests to modify the switch's value at either the "measurement" or "command" IOAs with a negative acknowledgment. Conventionally, this response indicates the IOA was valid, but the system would not execute the command, possibly due to safety issues.

To address our research question of whether we could persuade attackers to attack the simulated electrical devices instead of launching common HTTP attacks, we implemented additional safeguards. First, SWS "sanitized" all data from request forms to minimize the risk of injection or memory-based attacks. The sanitization was done by removing characters found in the user-submitted data, but not in a list of safe characters. SWS maintained an internal list of IOAs that accepted commands, and compared the requested IOAs to the list to prevent unauthorized changes. Also, GPTalker kept a map between IOAs and command types to prevent the Web client from relaying unpermitted commands to GPTalker.

3. Bash Scripts

We used Linux Bash scripts to improve the usability of the system and the applications. The logging script rsynclogging.sh backed up captured network data and system logs from the SCADA Web server machine to the logging virtual machine every five minutes. The startup script startup.sh" launched SWS and GPTalker with the parameters specified in the script, Gunicorn configuration file, and Python application defaults as appropriate. Figure 20 shows output of its startup script. The script gploop.sh relaunched GPTalker when the current sessions ended due to an error condition rather than a user-initiated exit. This maintained continuous communications between GridPot and GPTalker.

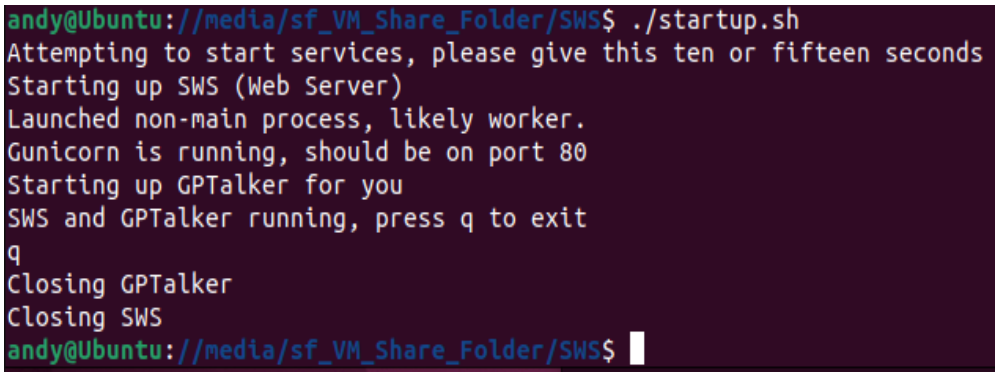

Figure 20. Screenshot of startup script output.

C. TESTING AND EXPERIMENTS

We ran three experiments with three separate GridPot instances running in the DigitalOcean cloud environment. The SCADA User Interface Server used Ubuntu 20.04.3. The Web server used Gunicorn Version 20.1.0 and Flask 2.3.2. GPTalker used Scapy 2.5.0 and IoaManager (included with GPTalker). The logging server ran on Ubuntu 20.04.3 and used the included services systemctl and rsync to transfer logs and packet captures.

Before the live experiments, an external subject-matter expert (SME) in penetration testing probed our system to check that the logging and security features worked as intended. The configuration of the system they tested was the same setup used in live experiments. The SWS and GPTalker applications were modified because of their testing.

For each experiment we started fresh instances of the Tshark network analyzer, the SCADA Web server, and GPTalker. Upon startup, GPTalker connected to the GridPot server. The connections from GPTalker to GridPot, and from the logging server to the Web server's machine, were made over our honeypot's virtual private network. This kept the traffic logically separate and prevented Tshark from logging those connections. We used DigitalOcean's firewall interface to open the HTTP port on the SCADA User Interface Server machine to the Internet. We protected the logging and GridPot servers with firewall rules that only allowed incoming connections from authorized machines.

1. Experiment 1

Experiment 1 ran from 30 September 2023 to 8 October 2023. This experiment used the same website layout and GPTalker as the development testing. DigitalOcean's firewall was set to allow any IP address to connect to the SCADA Web server on port 80.

The initial experiment plan is shown in Figure 21. It set simple metrics for experiment to meet by a particular timeline; in the event the metric was not met, we planned to adjust the experimental setup.

Time	Trigger	Response
Day 0	Start experiment	Allow any IP address to access honeypot on port 80.
Day 2	Less than 10 unique IP addresses seen	Initiate Shodan Scan, initiate Censys Scan.
Day 3	More than 5 unique IP addresses seen	Save copies of passwords.log, requests.log, and gunicorn_access.log for analysis.
Day 5	Less than 25 unique IP addresses seen	Re-examine methods of getting IP address visible; possibly post it on forums or allow it to be scrapped.
Day 6	Successful admin/user access	Pull requests.log, passwords.log, and gunicorn_access.log for analysis
Day 12	Successful admin/user access beyond day 6	Pull requests.log and passwords.log, compare results to Day 6 to validate method.
Day 18	Successful admin/user access beyond day 12	Rerun scripts and processes from day 12, note relevant information for analysis
Day 21	Results of data analysis	Start analysis of requests.log, passwords.log, and guinicorn_access.log; PCAPs can be used for additional information.
Day 28	Experiment complete	Return server to original status, make backup of server image

Figure 21. Experimental plan for Experiment 1.

We planned to start Shodan and Censys scans for the IP address of the SCADA Web server if fewer than ten IP addresses connected to the server in the first 48 hours. However, it only took six hours to receive the first ten. In this traffic, we saw scans from Bingbot, Censys, and zgrab; Bingbot and Censys are bots that scan the Internet for servers and map the information discovered on them. Shodan and Palo Expanse scans showed up on the second day. In the nine days that the experiment ran, we did not observe any login attempts by people outside the research group.

2. Experiment 2

Experiment 2 ran from 8 October 2023 to 13 October 2023. We introduced new Web pages and changed the home page to display the statuses of the simulated electrical grid with no login required. However, the Web server still required a user to log in before they could change the devices. An attempt to change a device by an unauthenticated user gave an error message prompting them to log in.

We added a Web-standard file "robots.txt" to the Web server that listed webpages and directories that scanning traffic and search engines should avoid. We added some of these resources to the Web server and any attempt to access them or a nonexistent file was logged by the server. The entries in the document were selected to appeal to an attacker

such as the paths "/SCADA" and "/technician-logon." The pages for these paths returned a message denying access and directing the attacker to log in to the honeypot organization's virtual private network to gain access. This network did not exist, and only was mentioned to make deception more realistic. We added several common login page URLs such as "login.php" and "login.html" that a typical Web server would have. Requests for those pages were redirected to the standard login page.

We monitored the logs of the SCADA Web server for attacks and traffic rates. The scans from Censys, Shodan, Bingbot, and others took time to update with the new Web pages. After several days, traffic was still either scanning or running automated attempts to compromise the Web server.

We ended Experiment 2 after six days (on 13 October 2023) because there were no attempts to use the Web SCADA user interface, no known compromises of the system, and no attempts to log into the Web server. Although we recorded a large variety of attempted compromises, we saw no indication of conscious testing of the system for vulnerabilities, as it appeared all attacks were automated.

3. Experiment 3

Experiment 3 began on 14 October and ended on 27 October 2023. We shut down the Web server used in the first two experiments and created a new one with a new public IP address. We expected traffic patterns like those of the first two experiments, with possibly new traffic once the Web server's existence was well established by scanning. We saw the first scanning traffic, by Bingbot, on the new IP address within six minutes. This experiment yielded several notable attack attempts on the Web server, including new types of attacks not observed in the first two experiments.

We stopped Experiment 3 on 27 October 2023 with no known compromises to the Web server or interactions with the user interface. We had a gap of six and a half hours in the logs when Gunicorn encountered an error. After further investigation, we do not believe this error was caused by an attack.

D. DATA ANALYSIS

The log files created by the SCADA Web server were our primary data source. They contained important information from the Web connection such as remote IP address, HTTP method, and URL for the method. We created a Python program to parse the logs and create data structures for further analysis.

We identified "HTTP patterns" in the logs to classify the types of traffic with the Web server. The patterns are defined by the order of HTTP methods consecutively sent to the server from the same IP address. Once a request from a different IP address was received, the HTTP pattern ended. These patterns identified when traffic used the same techniques.

We examined the Web server logs after each experiment to identify attacks and traffic patterns. We manually classified the traffic into three categories:

- Malicious: If the traffic requested protected information, such as a .env file, tried to access an administrative dashboard, or tried to exploit a server vulnerability.

- Scanning: If the traffic connected only to pages that were accessible, and only viewed common Web server documents like robots.txt or security.txt.

- Legitimate: Any traffic that was not categorized as malicious or scanning. This traffic generally loaded all the Java script and cascading style sheet documents necessary to render the Web page for human use. This traffic also normally requested Web pages that were linked to the pages displayed, and it did not try to exploit vulnerabilities.

For some portions of the data analysis, we further separated the scanning traffic from the rest of the legitimate traffic and labeled it as such. This separation improved legibility of the graphs as scanning was a small part of the total traffic.

To identify if an IP address as malicious, as opposed to legitimate traffic, we compared its history with the HTTP patterns. Any IP address that ever used a malicious

HTTP pattern was labeled as a malicious address, regardless of whether that interaction would be considered malicious. Scanning traffic was treated similarly; any IP address that demonstrated scanning behaviors was considered a scanning address. Possibly multiple devices could access the Web server from the same IP address, but we did not track connections any further than the IP address level. It is also possible that a device could use multiple IP addresses to access the Web server, creating several malicious IP addresses within the dataset. Due to the short timeline of individual experiments, we tracked the HTTP patterns over the entire period of all experiments. This method could cause overclassification of some traffic because an IP address may have belonged to one device in one experiment and to another device in another experiment. We considered the value of the classification compared to these possible cases of overclassification, and chose to use the traffic data with the classification labels.

The amount of HTTP traffic (by total connections) created by legitimate sources was much more than the amount of HTTP traffic created by malicious sources. Due to this disparity, we chose to reduce most subsets of data to unique IP addresses. This reduction was necessary because, in general, a browser would refresh a page every five seconds, resulting in hundreds or thousands of HTTP connections. Using unique IP addresses let us examine the relationship between behavior and IP addresses without the most active connections overshadowing the rest.

We used MaxMind's IP2 geolocation service to estimate physical locations associated with IP addresses (MaxMind, 2023). Since IP addresses were reported from the connection to the Web server and not the starting location of the request, we did not attribute any behavior to the IP addresses identified in the dataset, the geographical location, or the organizations involved. We used geolocation data to identify trends within the traffic but do not draw any conclusions about the characteristics or nature of the organizations and locations themselves. All IP addresses used in our dataset were gathered from the SCADA Web server logs which indicated that the device connecting to the Web server could receive and respond to traffic at the IP address. Given the difficulty in spoofing an IP address through a full TCP connection, and the likely low value of the target, we did not consider the IP addresses to be spoofed.

The accuracy of the geolocation service is advertised at 99.8% for identifying the country of origin; the dataset contained over 900 unique IP addresses which suggests it includes a few misattributed IP addresses (MaxMind, 2023). The same service also provides information on the organization for the IP address, such as an Internet service provider or owning business, but we could find no information regarding the accuracy of this service.

We used the public Python module "pyplot" to graph the results of our analysis. We organized the data and set cutoffs on the rankings to demonstrate consistencies between phases and to highlight anomalies.

THIS PAGE INTENTIONALLY LEFT BLANK

V. RESULTS AND DISCUSSION

We ran our experiments for 27 days, collecting information from the SCADA Web server logs to identify traffic patterns and assign categories to IP addresses. We analyzed the logs and found results consistent with previous research, in that the operational-technology interface was mostly ignored, as attackers focused on the operating system (Hilt et al., 2020).

We recorded 55407 HTTP connections from 933 unique IP addresses, representing 216 organizations and 61 countries (Table 1). We categorized the traffic as malicious, scanning, or legitimate based on their use of manually coded traffic patterns as discussed in Section IV.D. We noted that the totals of the individual experiments did not exactly match the totals in the third column as some IP addresses occurred in more than one experiment. The duration in hours had a small difference when comparing all phases to the individual phases, due to the time taken to switch systems between experiments.

Table 1. Statistics of all experiments.

Phase	Duration (hours)	Unique IPs	Malicious HTTP patterns	Legitimate HTTP patterns	Scanning HTTP patterns
1	192	299	108	171	21
2	124	222	82	123	17
3	338	549	209	290	50
TOTAL	656	933	344	523	67

The length of the HTTP patterns, the number of different Web pages and the order they were requested, was used to correlate with the type of traffic we received. We identified malicious HTTP patterns as having a spike in traffic at length of one, often an

attempt to read a ".env" file, or lasting longer than 15 requests. Legitimate traffic was generally five to seven requests long, enough to load the Web page itself and supporting documents. Scanning traffic was often three to five queries, typically for the index Web page, the robots.txt file, the Web page icon, and sometimes the security.txt or sitemap.xml files. Figure 22 depicts the relationship between HTTP pattern length along the horizontal axis and the number of patterns that match that length along the vertical axis. The lines are colored to differentiate the types of traffic.

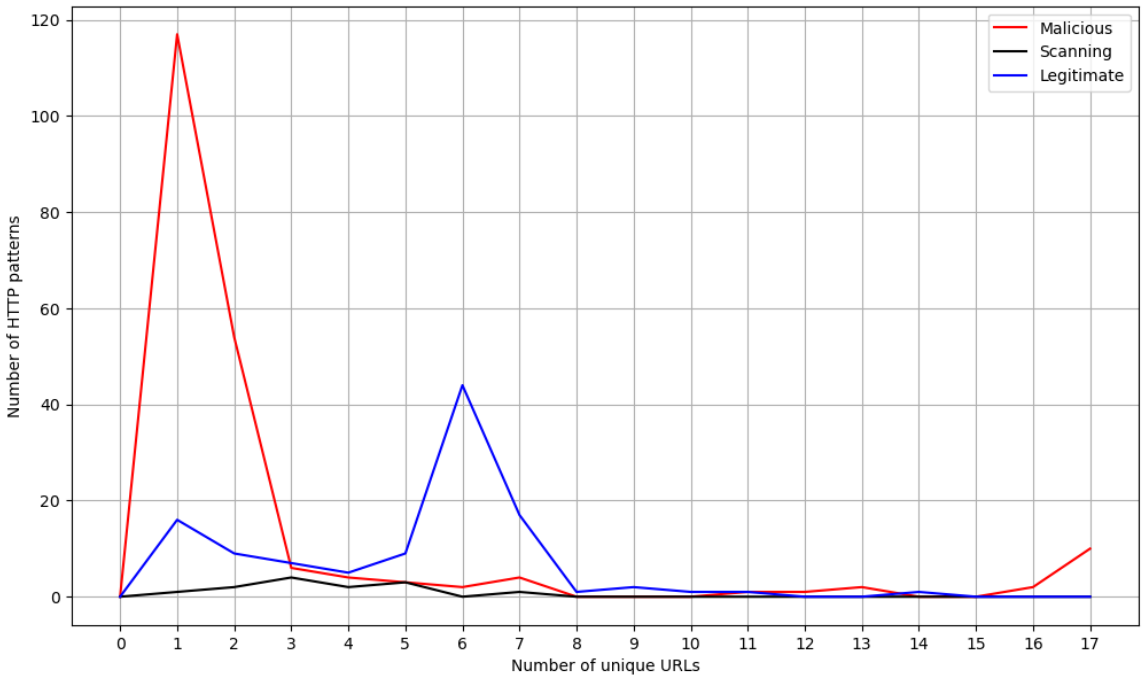

Figure 22. Number of HTTP patterns by length and traffic type over all experiments.

We used a geolocation service Google My Maps to visualize the approximate location of each IP address, obtained from the MaxMind geolocation service, as shown in Figure 23 (Google, n.d.). Red pins indicate IP addresses that sent at least some malicious traffic, and blue pins correspond to IP addresses that sent only legitimate traffic.

Figure 23. Unique IP addresses overlaid world map. Source: Google (n.d.).

We used cosine similarity to compare the results of the three experiments to prior research. Cosine similarity measures similarity between two sets of vectors of equal length. We observed significant similarities between our experiments and two previous projects (Dougherty, 2020; Bieker & Pilkington, 2020). We reduced the vector size of the previous research to match the data collected during our experiments. The Dougherty experiments used an implementation of GridPot hosted on an on-premises system using an IP address associated with the Naval Postgraduate School (Dougherty, 2020). The Bieker & Pilkington experiments ran the GridPot version used by Dougherty on the same cloud-service provider that we used (Bieker & Pilkington, 2020). Table 2 shows basic information about HTTP traffic for each research project.

Table 2. HTTP methods for three GridPot projects (weekly average).

Research Project	HTTP Calls (Weekly)	HTTP GET (Weekly)	HTTP POST (Weekly)	HTTP Other (Weekly)
1 – On-premises GridPot (Dougherty)	1198.7	528.2	612.7	57.8
2 – Cloud-based GridPot (Bieker & Pilkington)	12150.4	10546.7	1346.0	257.7
3 – Cloud-based GridPot with dual-purpose Web server (this work)	14157.8	13930.7	192.9	34.2

We used the information from Table 2 to calculate cosine similarity between the three projects. Projects 1 and 2 ran in 2020 while Project 3 ran in 2023. Despite using the same Conpot-based Web server, Project 2 received roughly ten times as much weekly traffic than Project 1. Running on the same cloud-service provider, Project 2 and Project 3 showed roughly equal amounts of weekly traffic. Table 3 shows that Project 2 and Project 3 have a strong similarity in traffic received. Their similarity score, 0.9934 is higher than that of Project 1 and Project 2, which ran within a few months of each other. This could indicate the IP address received from the Internet service provider or cloud-service provider influences the amount and type of traffic a honeypot receives.

Table 3. Cosine similarity scores for three GridPot projects.

Datasets	Cosine Similarity
Project 1 versus Project 2	0.7432
Project 2 versus Project 3	0.9934
Project 1 versus Project 3	0.6619

A. INTERNAL TESTING

We tested the SWS and GPTalker applications to ensure our system worked as expected before exposure to the Internet. In these tests, communications between GPTalker and GridPot were observed to be reliable for days and restored communications automatically. The subject-matter expert examined the SCADA Web server before the experiments, and confirmed that the firewall provided by the cloud-service provider was effective, and that no additional firewall features would be useful.

During testing, the default Gunicorn logging and Tshark packet captures were found insufficient for periodic monitoring. To fix this, we added two new custom Python logging methods in the Web server to monitor it more easily. The password logging method saved all attempted usernames and passwords in a text document. The request logging method saved all HTML requests to the Web server such as form responses or login prompts.

The subject-matter expert gathered information about the SCADA Web server to identify its applications and its version of the operating system. They checked the public Critical Vulnerabilities and Exposures (CVEs) database to find issues relating to those applications that would affect the security of the server. They found no relevant critical CVEs or known weaknesses of Gunicorn or the other services on the server, and the latter were also protected by a firewall.

This testing did reveal some vulnerabilities in the implementation of the Web server. They noted that the lack of the secure protocol HTTPS and using an IP address instead of a domain name for the server left it open to an "adversary-in-the-middle" attack. The tester successfully cloned the website's login page, Java scripts, and style sheets to create an imposter page. While this could be an effective tactic against a real website, it did not prevent the honeypot from gathering data.

They noted that the SCADA Web server did not use the industry-standard method of storing the login status with cookies, opting instead for a server-based list of IP addresses recording the IP address of approved connections. An attacker who impersonated the IP address of a logged-in user, or connected from the same IP address as another authenticated

user, would be automatically logged in. This was a known limitation of the implementation and did not affect honeypot operations.

The Web server used a typical username and password mechanism for user authentication. The system itself was resistant to injection attacks but did not limit login attempts to prevent "brute-force" attacks. Despite this, the login system did prevent the tester from accessing the rest of the website. To allow future attackers to access the system, a capability was added to the login process to identify common SQL (Server Query Language) injection attacks and grant access to the attacker, though this bypass would be logged by the server.

The expert did identify the Web server as the front end of a SCADA system, and suspected it used the IEC 60870-5-104 protocol. They discovered that the cloud-service provider permitted thousands of failed login attempts through HTTP on port 80, but they were quickly blocked from Secure Shell (SSH) access on port 23. This meant their interactions with the Web server along with all other servers hosted by the cloud-service provider worldwide were blocked for several days. Notably, the research team was not told by the cloud-service provider of the attempted intrusion.

B. EXPERIMENT 1 RESULTS

Experiment 1 ran 192 hours. The observed attacks on the SCADA Web server ranged from old techniques such as directory traversal to find the password file to more modern attacks exploiting Log4j and Laravel vulnerabilities (Lacework Labs, 2022). These attacks were unsuccessful and did not find any vulnerabilities in the Web server. Several were periodically repeated despite their previous failures.

In Experiment 1, we recorded 299 unique IP addresses interacting with the Web server. Based on their traffic patterns, 108 addresses were malicious, 21 were scanning, and 171 were legitimate as defined in section III.D (Figure 24).

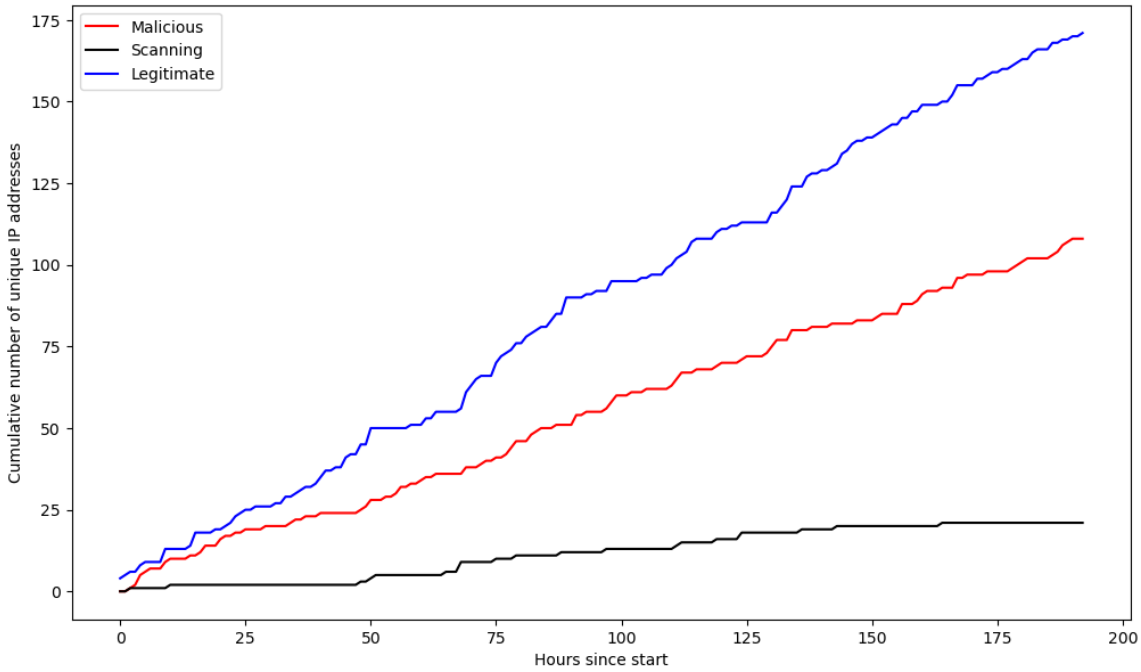

Figure 24. Experiment 1: Cumulative unique IP addresses by traffic type.

Each type of traffic increased over time, with legitimate traffic increasing the most quickly, followed by malicious traffic. The fewer unique IP addresses associated with scanning suggests that scanning devices use consistent source IP addresses and run continuously. In this experiment some scanning services visited the Web server each day.

Geolocation reported 40 different countries with the United States having the most unique IP addresses, 152 of the 299 total. The unique IP addresses are graphed in two categories, malicious and legitimate plus scanning. Figure 25 shows 15 countries with the most unique IP address and the other 25 countries are combined into the "other" category.

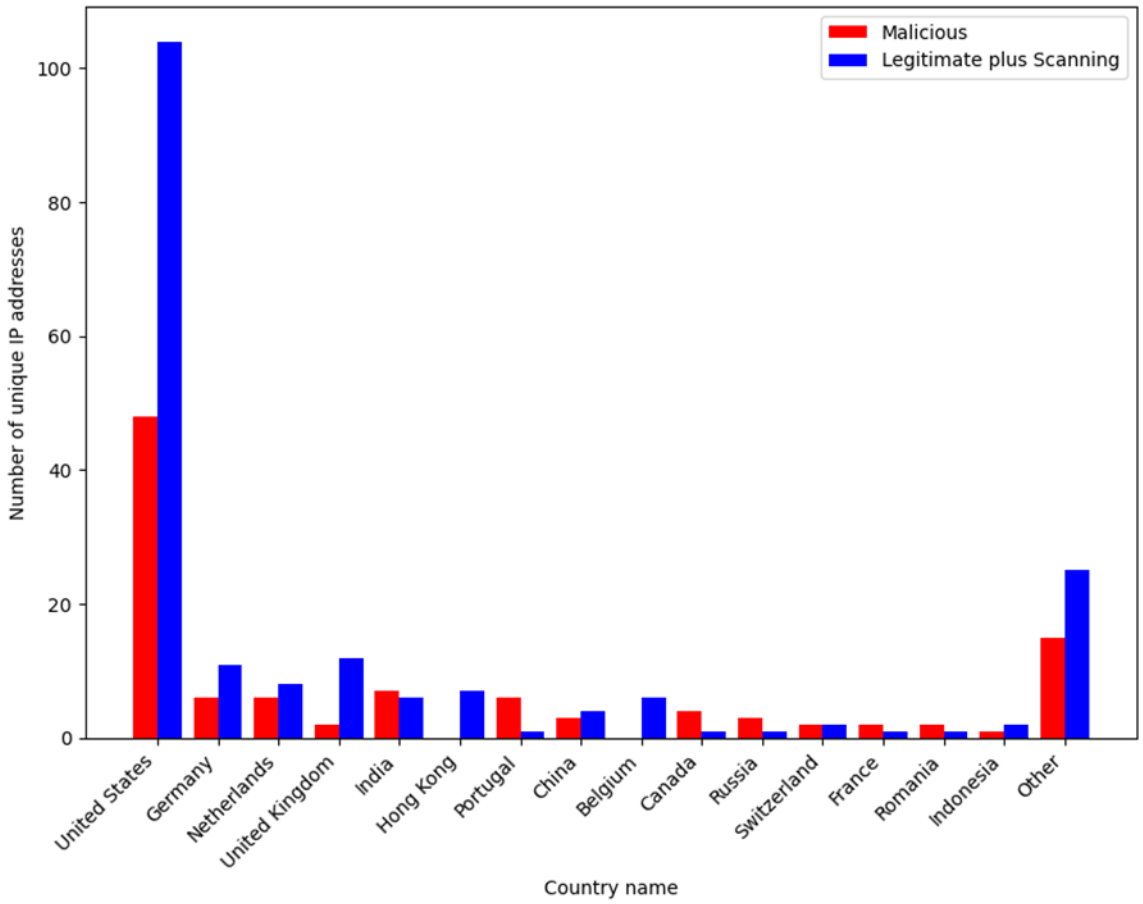

Figure 25. Experiment 1: Unique IP address by traffic type by country.

The same geolocation service tried to identify the organization that owned the IP address. We saw that most traffic came from large corporations such as cloud-service providers. Figure 26 shows the distribution of organizations across addresses. Relationships between location and organization are visible such as “Hong Kong” and “Hong Kong Zhengxing Technology Co., Ltd.” The difference in the traffic types in the two largest organizations could indicate different business practices. The “other” category contains all the organizations with fewer unique IP addresses seen than Gemnet LLC.

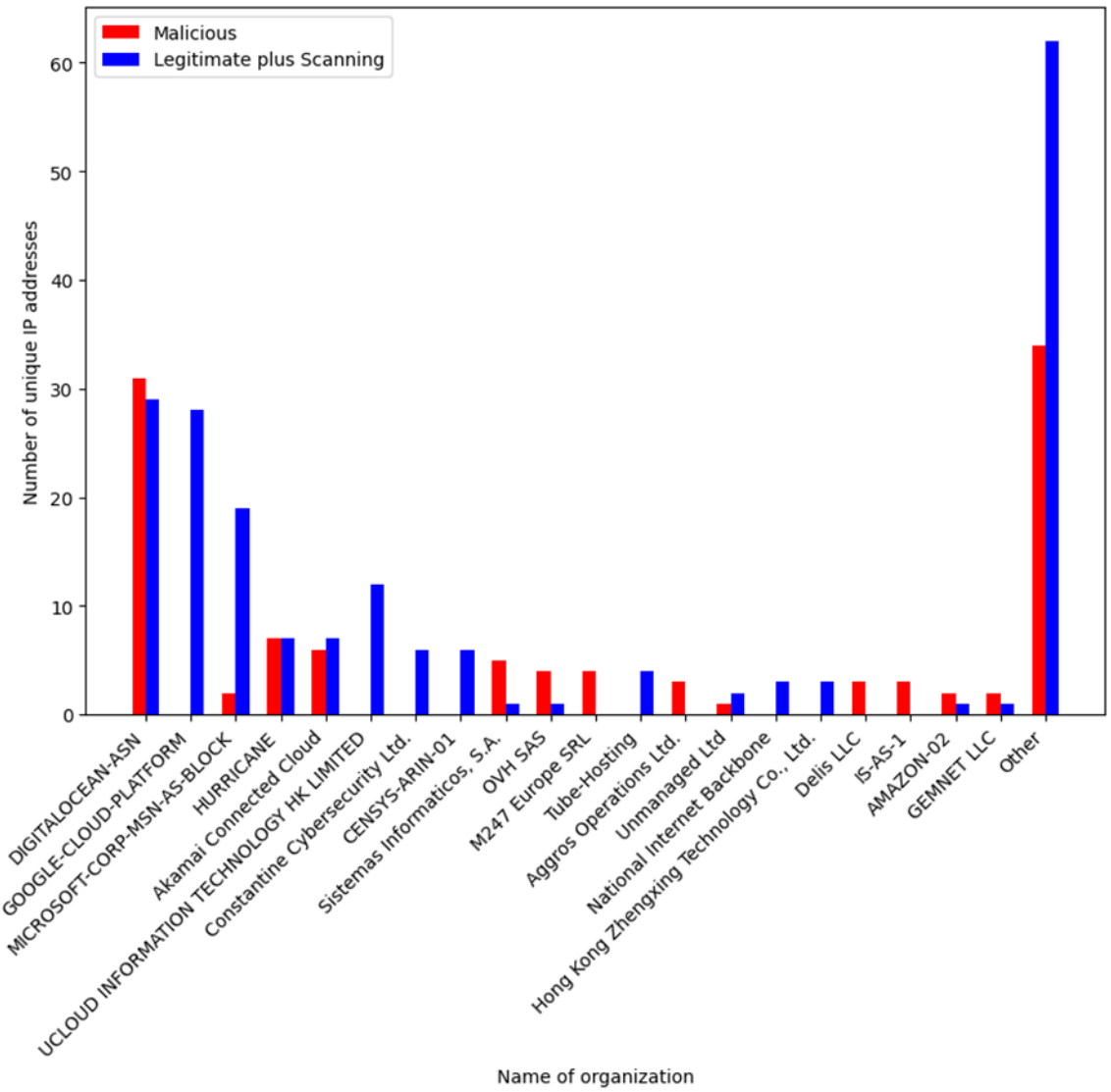

Figure 26. Experiment 1: Unique IP addresses by traffic type by organization.

We did not detect any compromises of the Web server or other services on the target machine. We saw no denial-of-service attempts. We ended this experiment due to the lack of interaction with the Web server's login page.

C. EXPERIMENT 2 RESULTS

Attackers tried to exploit several vulnerabilities in attacking our hardened Web server. One attack made 126 attempts to access administrative resources in 32 seconds.

One recurrent attack focused on Laravel vulnerabilities that were over a year old, well-known, and only valid against specific instances of that software.

Attackers tried to use the resources of the SCADA Web server by "tunneling" through the HTTP CONNECT method. That attack tries to redirect the attacker's traffic to another website using the Web server as a relay. The attackers sent requests to redirect to www.google.com, but likely would have changed it to something more malicious later had the redirection worked.

Attackers also tried to use the HTTP POST method to access files on the Web server. Two attackers focused on the "/bin/sh" application which is critical for Linux operating systems as it interprets commands and hosts scripts. Such an attack could have obtained privileged access and compromised the entire server. Another attack made 85 attempts to access ".env" files in different directories. Such files may hold private keys used by an organization to manage login information, encryption, and authentication. Attackers likely wanted to duplicate the keys and access protected resources. They could also replace the private keys with their own to gain access for themselves and deny it to the administrators.

Traffic patterns were similar to those found in Experiment 1. Legitimate traffic was the most common and scanning traffic was the least. All three categories of traffic increased over time (Figure 27) indicating that the honeypot would continue to identify new IP addresses and new traffic patterns if it ran longer.

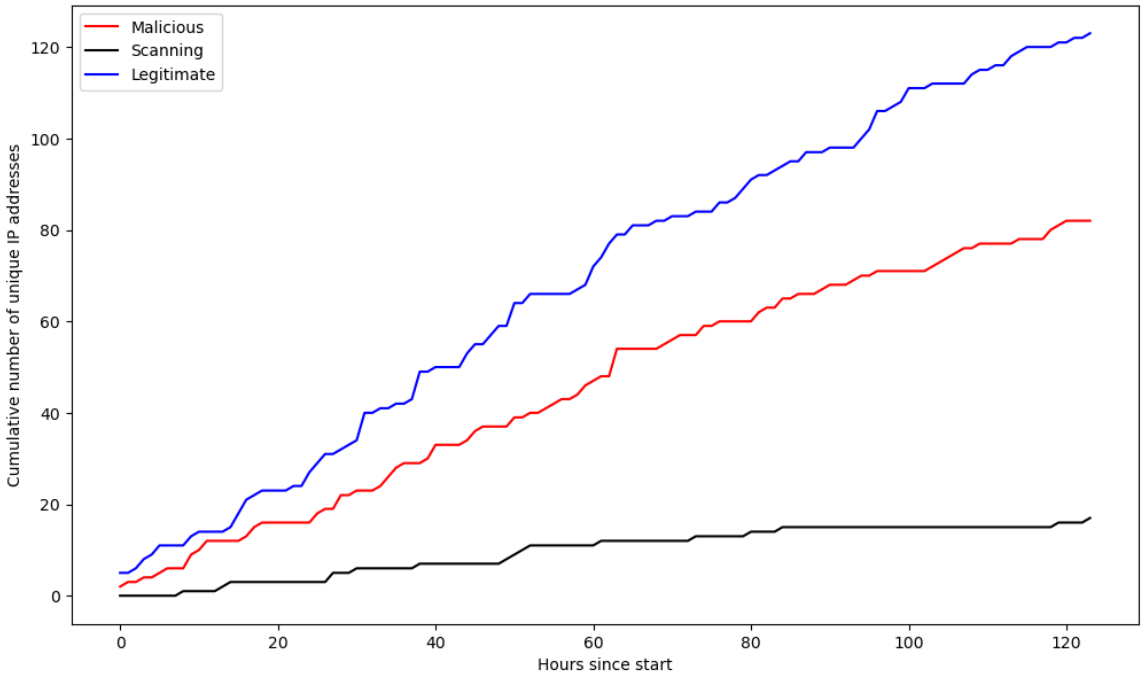

Figure 27. Experiment 2: Cumulative unique IP addresses by traffic type.

The ratios of malicious to legitimate IP addresses according to country differed between Experiment 1 and Experiment 2. Malicious traffic originated at the same rate across countries. However, the United States showed proportionally more legitimate traffic than before (Figure 28).

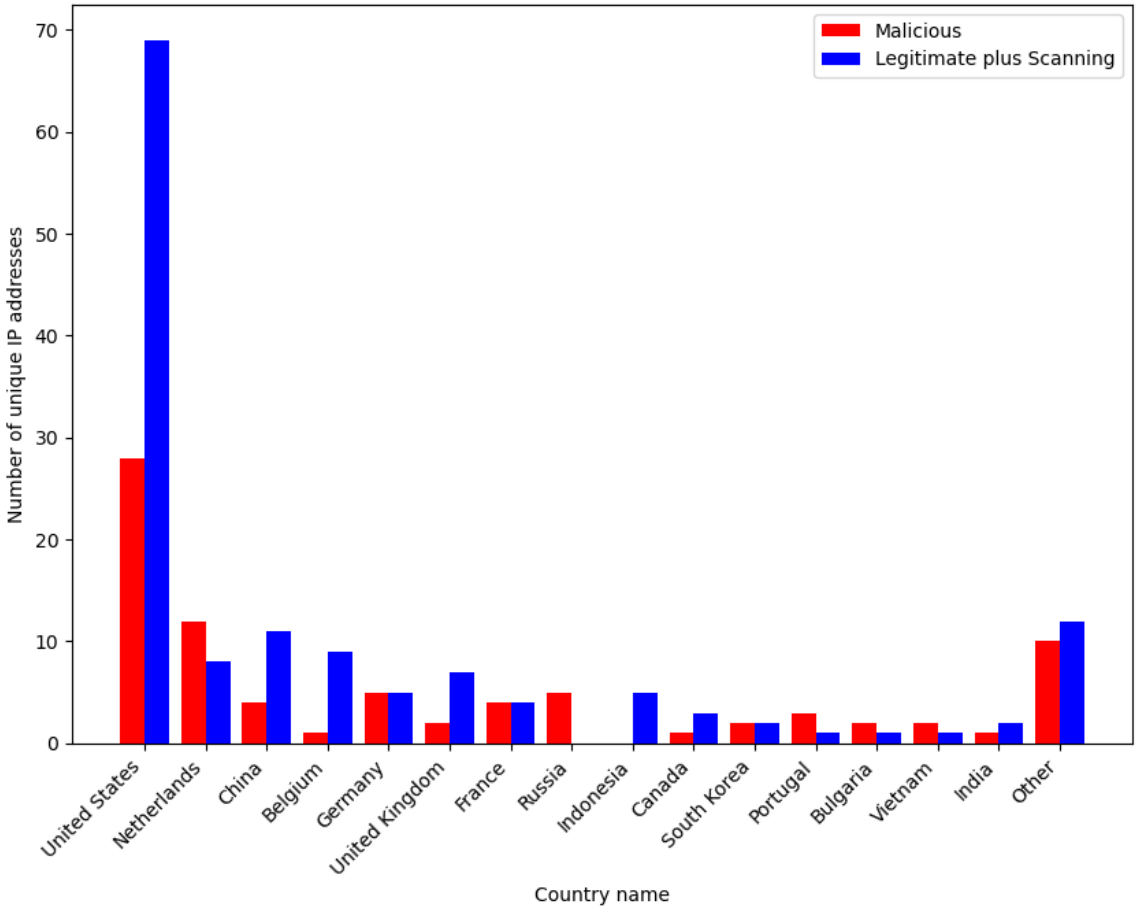

Figure 28. Experiment 2: Unique IP addresses by traffic type by country.

The distribution of malicious and legitimate IP addresses resembled that of Experiment 1. The three largest organizations remained the same, but no malicious traffic came from a Hong-Kong-based organization according to CENSYS and Constantine Cybersecurity. Figure 29 shows the 20 organizations with the most IP addresses and their traffic types; 62 others are in the "other" category.

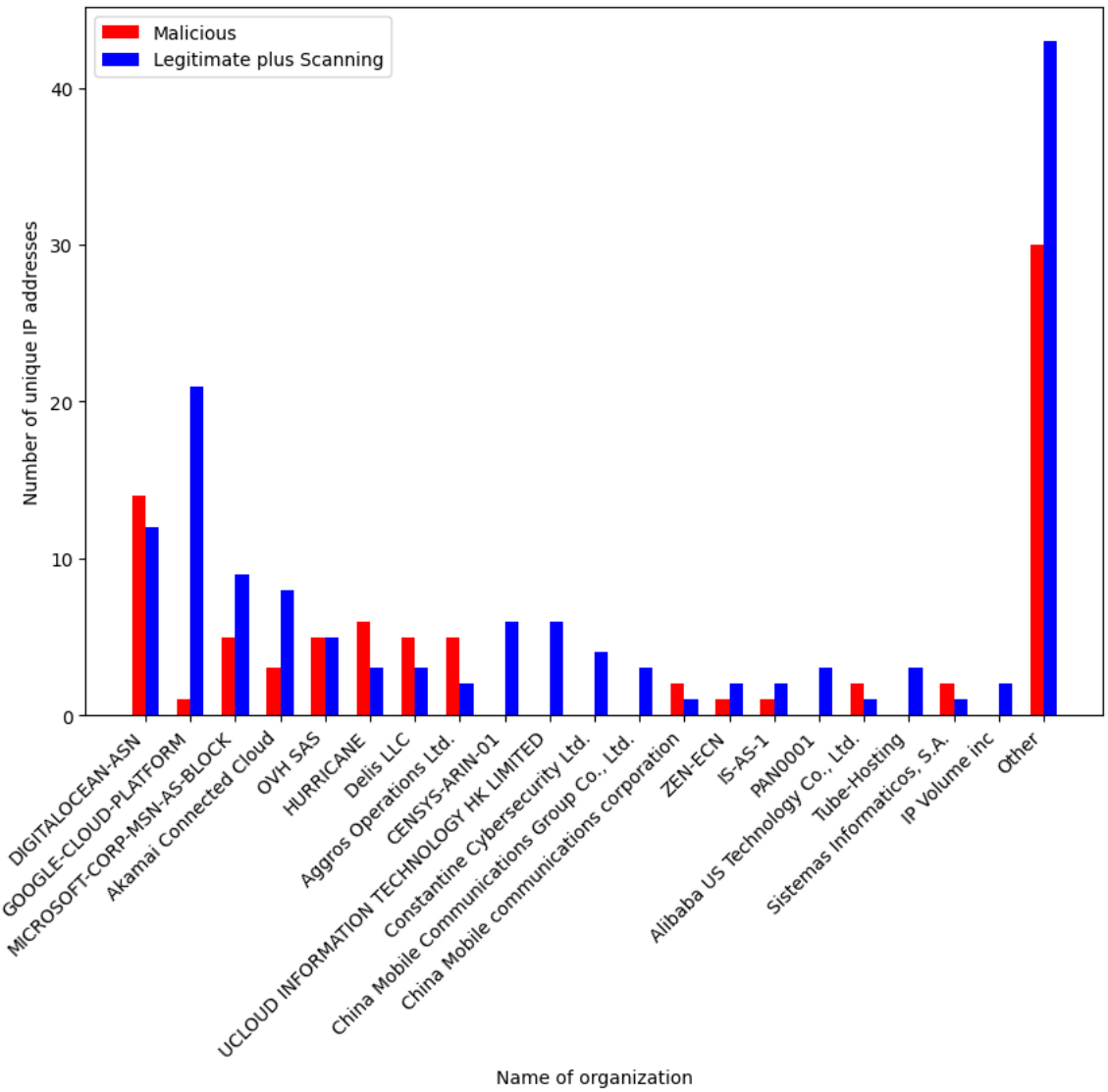

Figure 29. Experiment 2: Unique IP addresses by traffic type by organization.

In Experiment 2, we saw no indication that the Web server was compromised or that an individual outside the research team interacted with the Web human-machine interface. The cosine of similarity between Experiment 1 and Experiment 2 was 0.9917 indicating that the HTTP traffic did not change much despite the changes to the main Web page.

D. EXPERIMENT 3 RESULTS

Experiment 3 used a new cloud-based virtual machine with a different IP address. Within an hour and a half, we received a novel attack using the Log4J vulnerability to try to cause our Web server to execute an encoded payload. The malicious code tried to trick the logging portion of the server into executing several shell commands within the Linux operating system. Since our cloud-service provider recycles IP addresses, possibly our IP address had been assigned to a vulnerable site previously. The first step of the attack has the Web server connect to another server and use an application to decode the Base64 encoded portion of malicious payload. The decoded code instructs the target machine to connect to a second server to download a file "paraiso.x86." The attack then would change the permissions on the file to allow execution, execute the file, and then delete it.

We decoded the attack and manually retrieved the malware from the server, putting it on a low-value virtual machine. We examined the malware with the "strings" command and determined that it was likely encoded by the application "UPX." We submitted a hash of Paraiso to a well-known malware-tracking website but it could not be identified. We submitted the executable itself to the website VirusTotal which gave us additional information. Paraiso is a Linux executable that was identified as malicious by 36 out of the 63 security vendors that scanned it (VirusTotal, n.d.). The consensus was that it was "malware/Mirai," which was also supported by the sandbox evaluation from VirusTotal. We examined the artifacts found within the VirusTotal reports, and surmised that this malware attempted to make the executing machine a bot in the Mirai botnet.

Experiment 3 lasted 338 hours and recorded 549 unique IP addresses. Despite the new Web server, IP address, and extended run time, Experiment 3 demonstrated similar traffic patterns to the previous two experiments. Legitimate IP addresses were most of the traffic, and new IP addresses were continuously recorded, indicating that the honeypot would likely continue receiving new traffic. Figure 30 shows the relationship between unique IP address and traffic type.

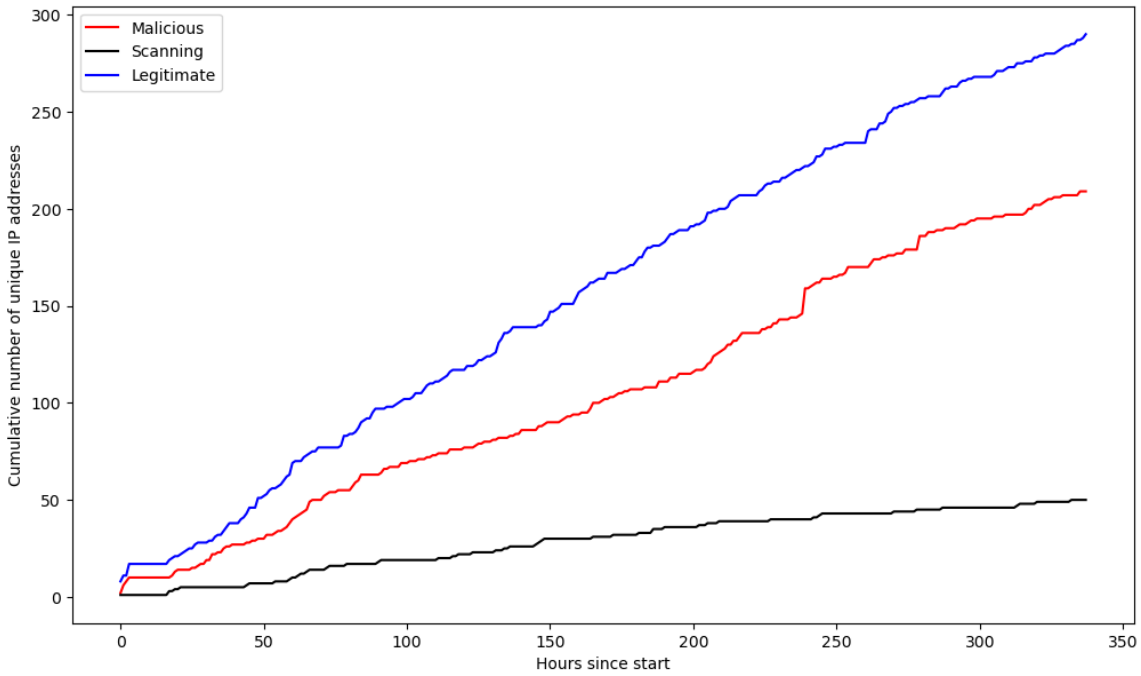

Figure 30. Experiment 3: Cumulative unique IP addresses by traffic type.

The relationship of country to traffic type is shown in Figure 31; the top fifteen countries are listed. The other countries are in the "other" category.

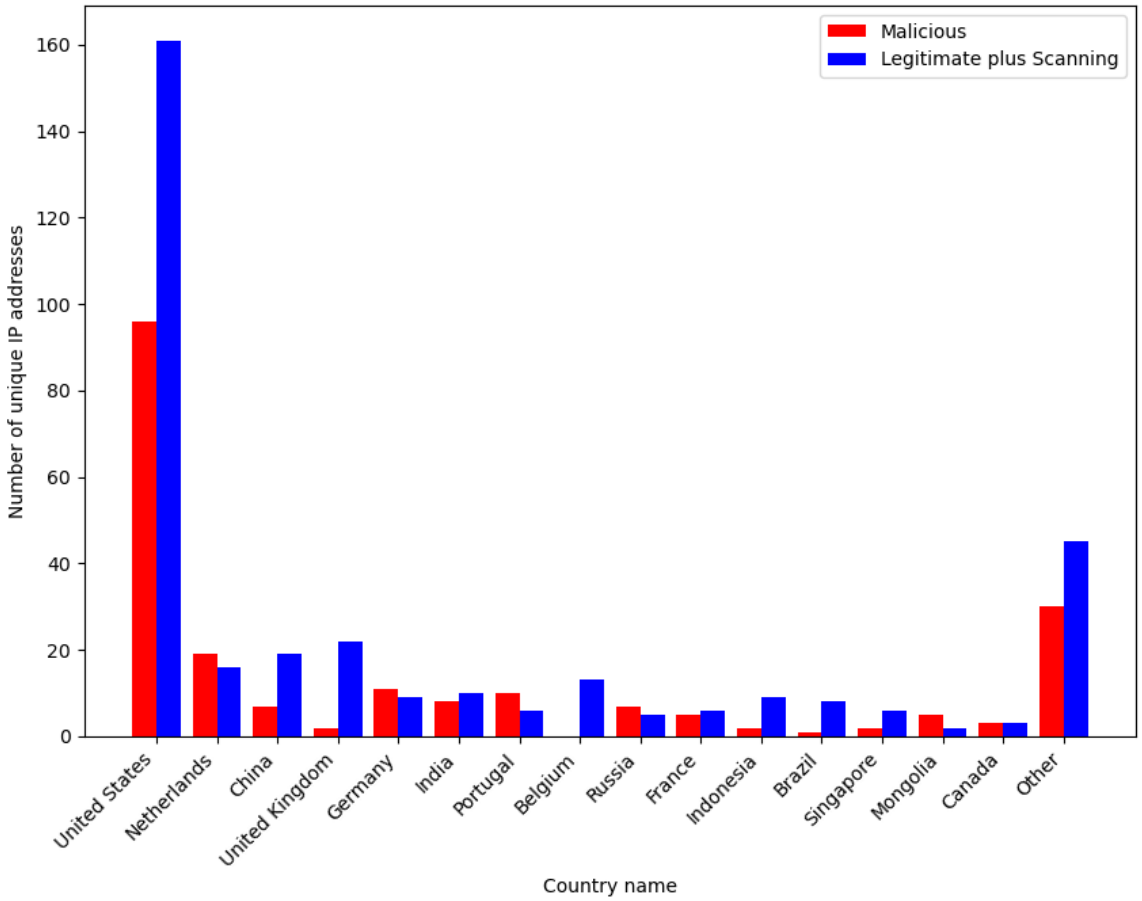

Figure 31. Experiment 3: Types of unique IP address by traffic type country.

The distribution of traffic over organization also matches the previous two phases, with the same top three organizations. The top twenty organizations are shown individually, with all others combined, in Figure 32.

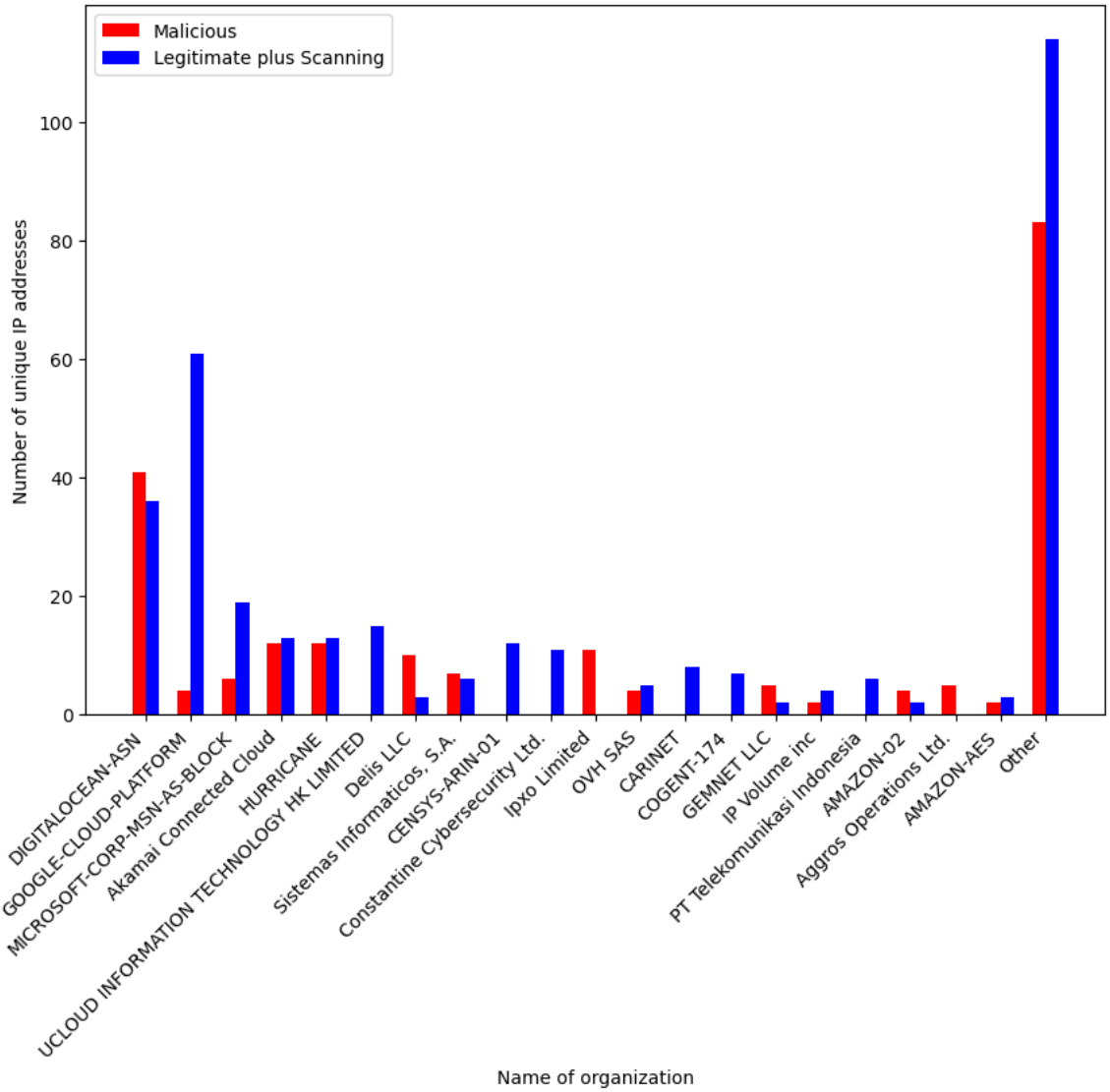

Figure 32. Experiment 3: Unique IP addresses by traffic type by organization.

We noticed several attempts to compromise the Web server including the Mirai botnet attack, and one IP address that sent over 400 HTTP messages to find information about the Web server or attempt exploitation of a potential vulnerability. We saw no indication that any attacks in this experiment were targeted to Gunicorn, the Web server software we used since Gunicorn has protection against unauthorized file access. The cosine similarity between Experiment 2 and Experiment 3 was 0.9991, suggesting a strong similarity in the HTTP traffic received despite using a new IP address.

E. DISCUSSION

We combined the logs from the three experiments. In total, we had 933 unique IP addresses over 656 hours. Figure 33 shows the results of the classification; the vertical lines represent the transitions between experiments. The graph shows that the amount of unique IP addresses seen over time was nearly constant even with a new server IP address.

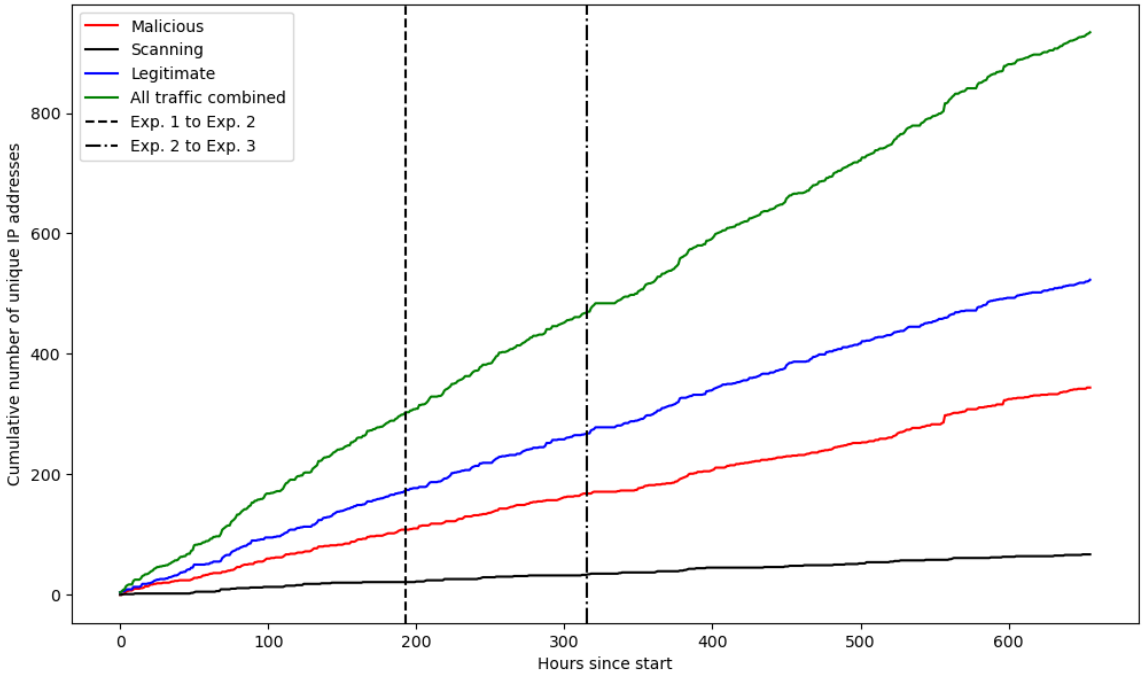

Figure 33. Cumulative unique IP addresses by traffic type over time across all three experiments.

We note that the slope of the graphs in Figure 33 is nearly linear which did not match our expectations. The rate of new unique IP addresses per hour stays almost constant from when the experiment started to the end of the experiment. No substantial change was observed when the IP addresses were scanned by Shodan, when the servers changed the layout of the index page, or when the server moved to a new IP address.

Figure 34 shows the number of new unique IP addresses observed each day of the three experiments. A substantial variation occurred in each day but the overall trend reflects

the results of Figure 33. The portion of the graph that represents the 27th day covers a partial day, 10 hours, and has proportionally less traffic as a result.

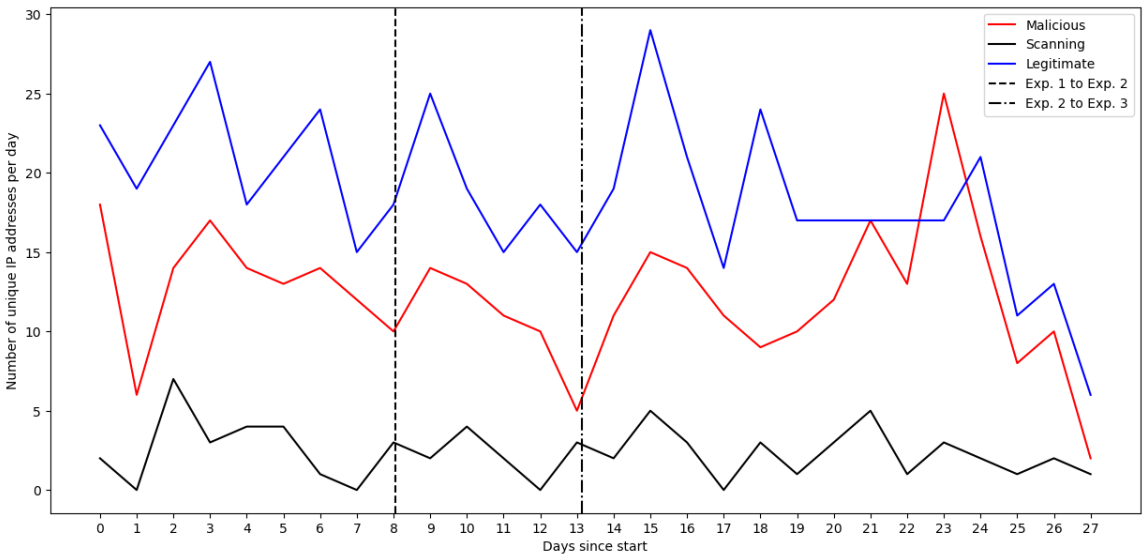

Figure 34. Unique IP addresses by traffic type over time across all three experiments (non-cumulative).

We used the daily traffic patterns of three experiments combined to separate all unique IP addresses into 24 subsets, one for each hour of the day. We divided each of the 24 subsets into "malicious" and "legitimate plus scanning" categories. The results are shown in Figure 35.

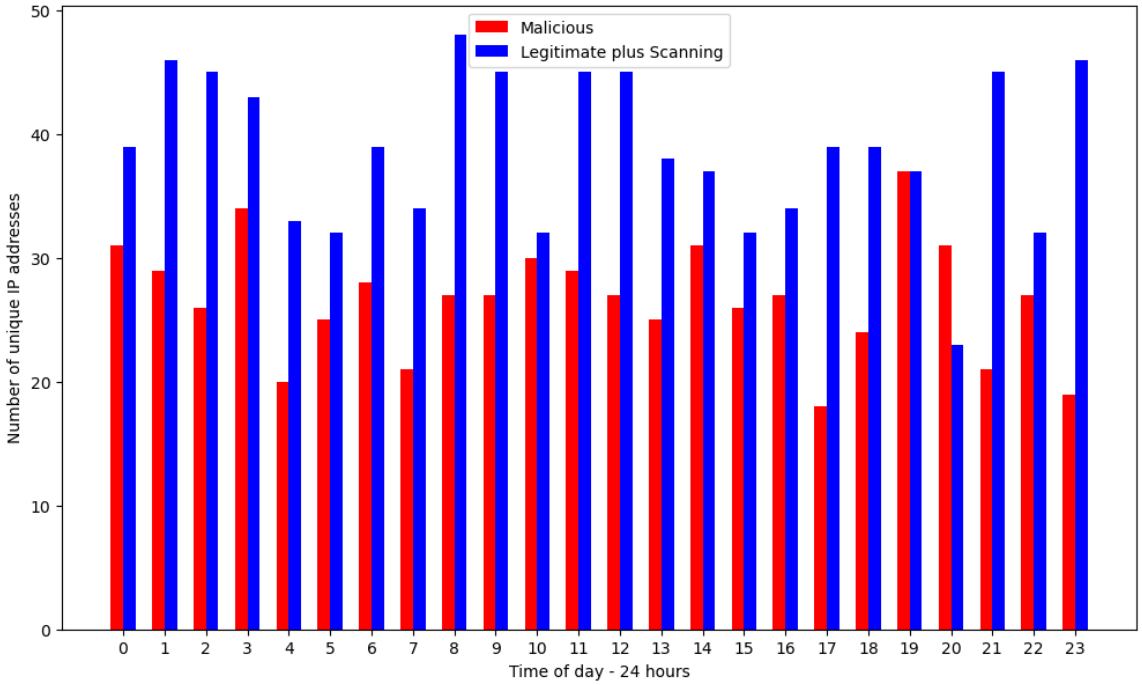

Figure 35. Unique IP addresses by traffic type by hour of day for all three experiments.

No obvious patterns are apparent. Almost half of the IP addresses were linked to the United States, 455 of the 933 observed. Figure 36 shows the distribution by countries.

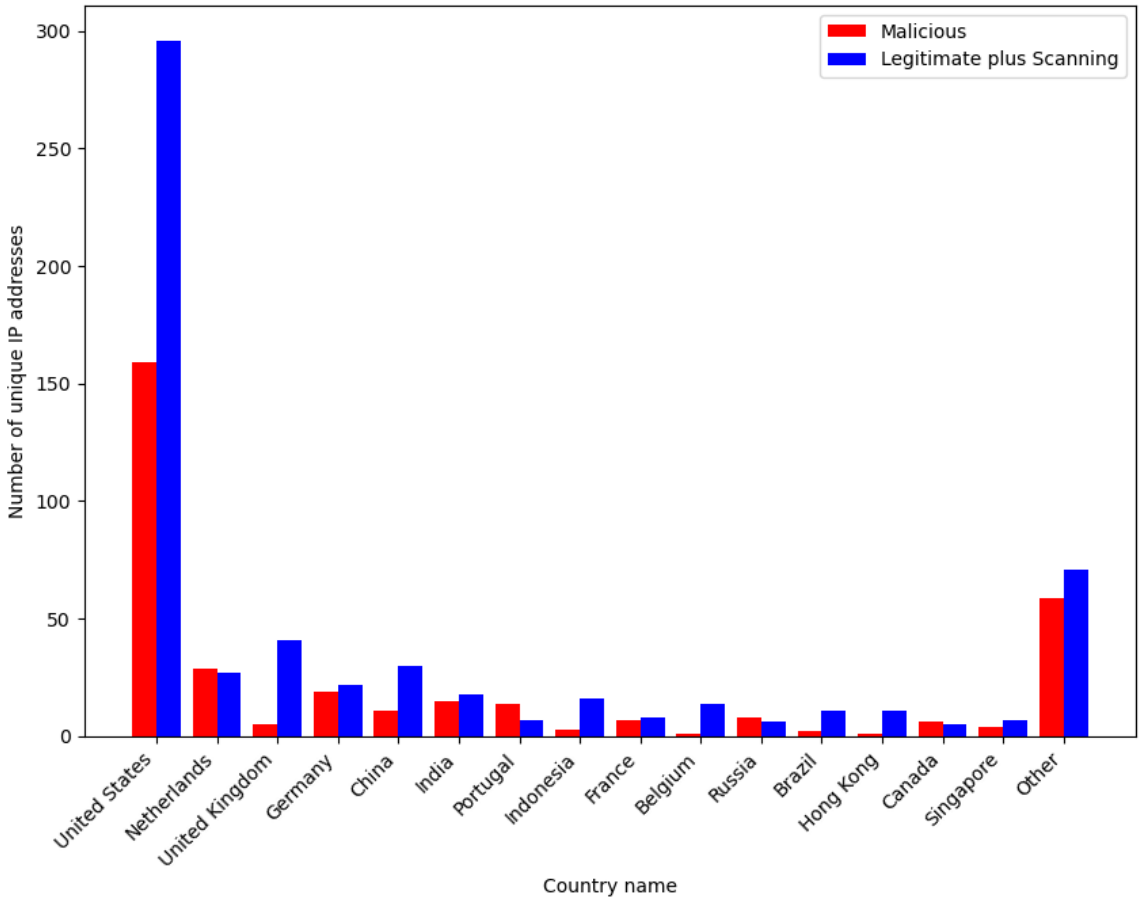

Figure 36. Types of unique IP address by traffic type by country across all three experiments.

Figure 37 shows unique IP addresses by organizations across all experiments. The "other" category contains nearly 200 organizations and more than a third of the addresses.

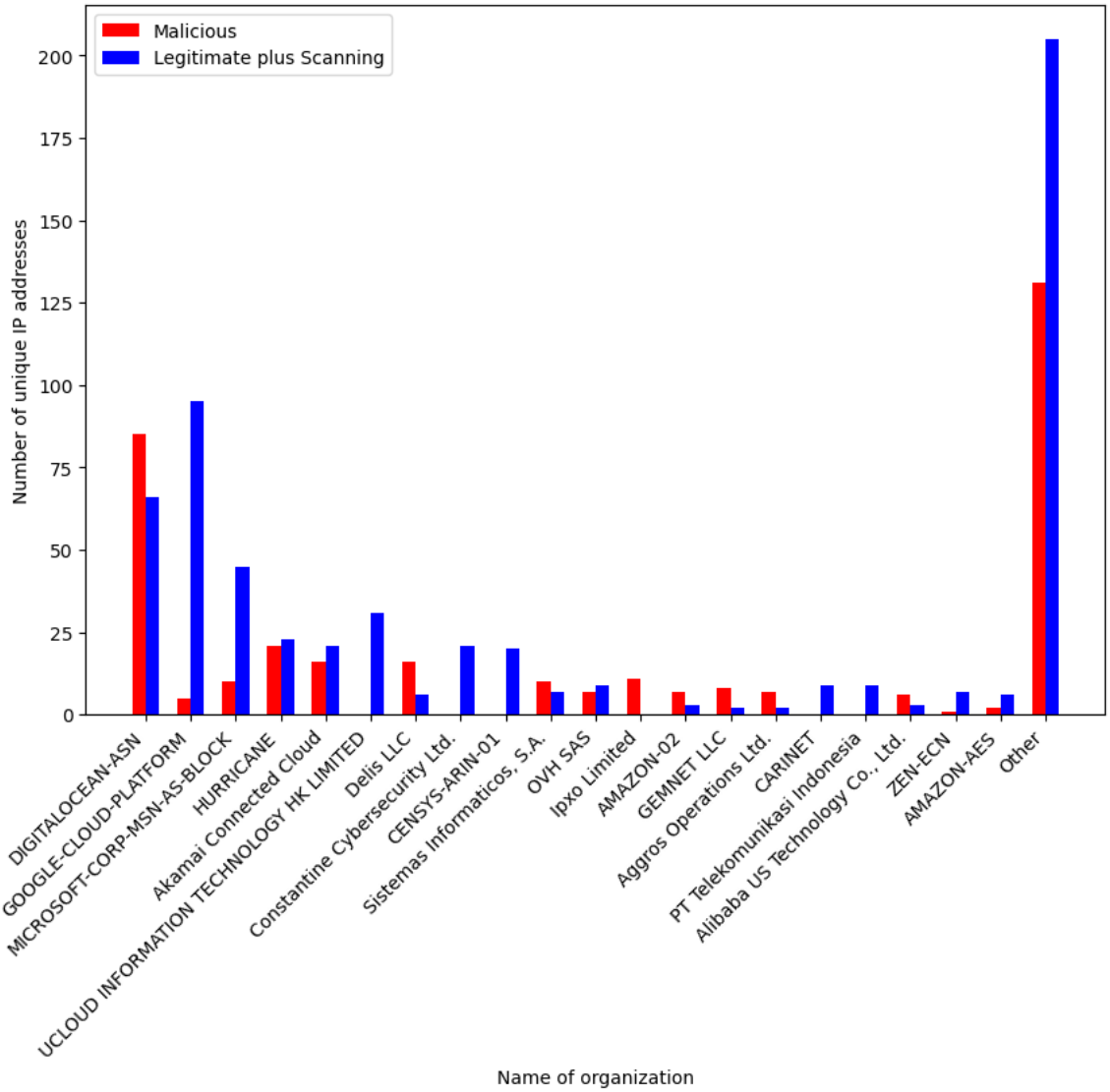

Figure 37. Unique IP addresses by traffic type by organization across all three experiments

VI. CONCLUSIONS AND FUTURE WORK

We successfully built a robust low-resource Web server that provided a straightforward human-machine interface for GridPot, a power-grid honeypot. It acted as an HTTP honeypot to collect attack data.

A. SUMMARY OF FINDINGS

We conducted three experiments, and our honeypot was not identified as a honeypot by well-known scanners during these experiments. All three experiments received new probes with unique IP addresses at roughly the same rate throughout, indicating that the honeypot should continue to gather relevant data indefinitely.

During testing by a cybersecurity subject-matter expert, we found that the cloud-service provider did not regulate traffic to the Web server, leaving it vulnerable to brute-force password attacks. We identified several unsuccessful attempts to exploit the Web server, and we captured a payload from an attack associated with the Mirai botnet malware.

We observed trends in scanning, malicious and legitimate types of traffic based on the browser's alleged Internet service provider and location. We compared characteristics of the Web traffic received during our experiments to previous GridPot projects and measured the similarity.

B. FUTURE WORK

Prior research has shown that an industrial-control-system honeypot may take months to receive ICS-specific attacks (Hilt et al., 2020). Although we stopped our experiments after 27 days, our system continued to receive new traffic. Since our Web server has so far has avoided detection by honeypot scanners, leaving it running should collect additional information on non-ICS attacks while waiting for the IEC 104 honeypot to be discovered by attackers interested in exploiting power-distribution systems.

Creating a new cloud platform with our honeypot virtual machines is relatively easy. Hence, additional honeypots could be quickly deployed elsewhere in the world to compare traffic patterns. The Web pages can be easily adapted to customize the interface.

An identified potential weakness of our Web server implementation was a lack of support for a domain name and HTTPS connections. If the Web server had a domain name it would appear more legitimate and could use digital certificates with the HTTPS protocol. It takes more effort, and potentially money, to host an HTTPS server so it may create the perception that this server is more important. Due to the perceived value of HTTPS, differences may occur in the traffic between an HTTP and HTTPS honeypot.

The SCADA Web server offers a simple working interface with limited information and feedback about the simulated power grid. An improved interface could appeal more to attackers. The ability to control more devices or to control the devices more granularly could create a more realistic or engaging power-grid user interface. Improvements to the interface that revealed more information about the fictional organization or suggested a physical location for the simulated power grid may also increase interest.

The attacks observed in our experiments did not target Gunicorn. The attacks were against other Web server applications or tried to compromise the operating system through the HTTP methods. This may indicate that attackers chose not to attack Gunicorn because there were no published vulnerabilities on Gunicorn. A Web server could impersonate another application, such as older versions of Apache containing the Log4j vulnerability, and attract new or additional adversarial traffic.

Recent work has added capabilities to our GridPot software but it is not fully tested (B. McGuire, personal communication, 17 August 2023). The SWS and GPTalker applications could add support for these capabilities to provide better user interactions with the power-grid simulator. This would keep attackers more interested and could induce them to stay longer.

Data collected from the three experiments was limited due to the short test periods. With a larger dataset, machine-learning tools could identify scanning and malicious traffic patterns more efficiently. The values in “user agent” field of the HTTP requests are a feature rich data set, and our traffic classification can be used with it to train supervised learning models. Large amounts of traffic data could also be classified with an unsupervised learning model to support detection and classification efforts.

APPENDIX. STATION DATA FILE FORMAT

The GPTalker application creates a "station data file" in the Java Script Object Notation (JSON) format to provide information to the SCADA Web server every few seconds. The format of this file is shown below.


```
{
   "station_address": "7720",
   "ioa_data": {
       "27665": {
           "current_value": "56375.7",
           "total_count": 26,
           "historic_low": "56367.6",
           "historic_high": "56414.5",
           "cumulative_sum": "1466257.9",
           "running_average": "56394.53461538461"
       },
       "27653": {
           "current_value": "58592.71",
           "total_count": 26,
           "historic_low": "58592.71",
           "historic_high": "69360.81",
           "cumulative_sum": "1652592.1199999999",
           "running_average": "63561.23538461538"
       },
```

```
"27664": {
    "current_value": "57088.8",
    "total_count": 26,
    "historic_low": "57079.3",
    "historic_high": "57129.1",
    "cumulative_sum": "1484807.7000000002",
    "running_average": "57107.988461538465"
},
"27666": {
    "current_value": "64984.9",
    "total_count": 27,
    "historic_low": "62175.5",
    "historic_high": "65025.5",
    "cumulative_sum": "1707267.0",
    "running_average": "63232.11111111111"
},
"27661": {
    "current_value": "11244.4",
    "total_count": 26,
    "historic_low": "11240.6",
    "historic_high": "11248.6",
    "cumulative_sum": "292350.50000000006",
    "running_average": "11244.250000000002"
},
"27663": {
```

```
    “current_value”: “60585.2”,
    “total_count”: 26,
    “historic_low”: “60569.2”,
    “historic_high”: “60604.8”,
    “cumulative_sum”: “1575347.9999999998”,
    “running_average”: “60590.30769230768”
},
“27662”: {
    “current_value”: “70825.3”,
    “total_count”: 26,
    “historic_low”: “70788.0”,
    “historic_high”: “70840.3”,
    “cumulative_sum”: “1841344.3000000003”,
    “running_average”: “70820.93461538463”
},
“27395”: {
    “current_value”: “405295.22”,
    “total_count”: 26,
    “historic_low”: “396419.62”,
    “historic_high”: “409764.12”,
    “cumulative_sum”: “10387631.93”,
    “running_average”: “399524.305”
},
“3348”: {
    “current_value”: “1.0”,
```

```
    "total_count": 1,
    "historic_low": "1.0",
    "historic_high": "1.0",
    "cumulative_sum": "1.0",
    "running_average": "1.0"
},
"3349": {
    "current_value": "1.0",
    "total_count": 1,
    "historic_low": "1.0",
    "historic_high": "1.0",
    "cumulative_sum": "1.0",
    "running_average": "1.0"
},
"8454": {
    "current_value": "2.0",
    "total_count": 1,
    "historic_low": "2.0",
    "historic_high": "2.0",
    "cumulative_sum": "2.0",
    "running_average": "2.0"
},
"27651": {
    "current_value": "3686.36",
    "total_count": 26,
```

```
    “historic_low”: “3685.84”,
    “historic_high”: “3688.91”,
    “cumulative_sum”: “95878.06999999999”,
    “running_average”: “3687.618076923077”
},
“27652”: {
    “current_value”: “58744.45”,
    “total_count”: 26,
    “historic_low”: “58744.45”,
    “historic_high”: “69524.01”,
    “cumulative_sum”: “1656671.41”,
    “running_average”: “63718.13115384615”
},
“27654”: {
    “current_value”: “0.0”,
    “total_count”: 1,
    “historic_low”: “0.0”,
    “historic_high”: “0.0”,
    “cumulative_sum”: “0.0”,
    “running_average”: “0.0”
},
“27655”: {
    “current_value”: “0.0”,
    “total_count”: 1,
    “historic_low”: “0.0”,
```

```
        “historic_high”: “0.0”,
        “cumulative_sum”: “0.0”,
        “running_average”: “0.0”
    },
    “27656”: {
        “current_value”: “0.0”,
        “total_count”: 1,
        “historic_low”: “0.0”,
        “historic_high”: “0.0”,
        “cumulative_sum”: “0.0”,
        “running_average”: “0.0”
    },
    “27657”: {
        “current_value”: “0.0”,
        “total_count”: 1,
        “historic_low”: “0.0”,
        “historic_high”: “0.0”,
        “cumulative_sum”: “0.0”,
        “running_average”: “0.0”
    },
    “5651”: {
        “current_value”: “1.0”,
        “total_count”: 1,
        “historic_low”: “1.0”,
        “historic_high”: “1.0”,
```

```
            "cumulative_sum": "1.0",
            "running_average": "1.0"
        },
        "5652": {
            "current_value": "1.0",
            "total_count": 1,
            "historic_low": "1.0",
            "historic_high": "1.0",
            "cumulative_sum": "1.0",
            "running_average": "1.0"
        },
        "15366": {
            "current_value": "2.0",
            "total_count": 1,
            "historic_low": "2.0",
            "historic_high": "2.0",
            "cumulative_sum": "2.0",
            "running_average": "2.0"
        }
    }
}
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Beerman, J., Berent, D., Falter, Z., Bhuania, S. (2023). A Review of Colonial Pipeline Ransomware Attack. *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing Workshops*. DOI: 10.1109/ CCGridW59191.2023.00017

Bieker, M. C., & Pilkington, D. (2020). Deploying an ICS honeypot in a cloud computing environment and comparatively analyzing results against physical network deployment [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. https://calhoun.nps.edu/handle/10945/66586

Biondi, P. (2023). *About Scapy*. Read The Docs. Retrieved 10 November 2023 from https://scapy.readthedocs.io/en/latest/introduction.html

Chen, T. M., Abu-Nimeh, S. Lessons from Stuxnet. (2011, April). *Computer*, 44(4), 91–93. DOI: 10.1109/MC.2011.115

Cyber Security and Infrastructure Security Agency (2021, July 20). *Significant historical cyber-intrusion campaigns targeting ICS*. https://us-cert.cisa.gov/ncas/current-activity/2021/07/20/significant-historical-cyber-intrusion-campaigns-targeting-ics

Dalamagkas, C., Sarigiannidis, P., Ioannidis, D., Iturbe, E., Nikolis, O., Ramos, F., Rios, E., Sarigiannidis, A., Tzovaras, D. (2019) A survey of honeypots, honeynets, and their applications on smart grid. *IEEE Netsoft 2019 – 1st Workshop on Cyber-security threats, trust, and privacy management in software-defined and virtualized infrastructures(SecSoft). 93–100. DOI: 10.1109/NETSOFT.2019.8806693*

Di Pinto, A., Dragoni, Y., Carcano, A. (2018). *TRITON: The First ICS Cyber Attack on Safety Instrument Systems*. Black Hat USA 2018. https://www.nozominetworks.com/downloads/US/Nozomi-Networks-TRITON-The-First-SIS-Cyberattack.pdf

Dougherty, J. T. (2020). *Evasion of honeypot detection mechanisms through improved interactivity of ICS-based systems* [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. https://calhoun.nps.edu/handle/10945/66065

Flask. (n.d.). *Welcome to Flask – Flask documentation (2.3.x)*. Retrieved 26 November 2023 from https://flask-docs.readthedocs.io/en/latest/

Google. (n.d.). *My Maps – About – Google maps*. Retrieved 10 November 2023 from https://www.google.com/maps/about/mymaps/

Grammatikis, P., Sarigiannidis, P., Sarigiannidis, A., Margounakis, D., Tsiakalos, A., & Efstathopoulos, G. (2020). An Anomaly Detection Mechanism for IEC 60870-5-104. *2020 9th International Conference on Modern Circuits and Systems Technologies*. DOI: 10.1109/MOCAST49295.2020.9200285

Gunicorn. (n.d.). *Configuration Overview*. Retrieved 11 November 2023 from https://docs.gunicorn.org/en/stable/configure.html

Gyorgy, P. & Holczer, T. (2020). Attacking IEC 60870-5-104 Protocol. *1st Conference on Information Technology and Data Science, 140–150*. https://ceur-ws.org/Vol-2874/paper13.pdf

Haynes, N. J., Nguyen, T. D., Rowe, N. C., "Creating Synthetic Attacks with Evolutionary Algorithms for Proactive Defense of Industrial Control Systems," *Proceedings of the 56th Hawaii International Conference on System Sciences (2023), 1684–1693.* https://scholarspace.manoa.hawaii.edu/items/555dc8c0-68ff-4ac1-84d7-3fbeeb85d371

Hemsely, K. E., Fisher, R. E. (2018). *History of Industrial Control System Cyber Incidents*. https://www.osti.gov/servlets/purl/1505628

Hilt, S., Maggi, F., Perine, C., Rosler, M., Vosseler, M. (2020) *Caught in the Act: Running a Realistic Factory Honeypot to Capture Real Threats*. Trend Micro Research. https://documents.trendmicro.com/assets/white_papers/wp-caught-in-the-act-running-a-realistic-factory-honeypot-to-capture-real-threats.pdf

Hjelmvik, E. (2022, April 25). Industroyer2 IEC-104 analysis [Blog]. *Netresec*. https://www.netresec.com/?page=Blog&month=2022-04&post=Industroyer2-IEC-104-Analysis

Institute of Electrical and Electronics Engineers. (2004) *IEEE 13 Node Test Feeder*. https://cmte.ieee.org/pes-testfeeders/wp-content/uploads/sites/167/2017/08/feeder13.zip

International Electrotechnical Commission. (2023). *International Standard IEC 60870-5-104*. International Electrotechnical Commission. https://webstore.iec.ch/publication/25035

Kendrick, M. & Rucker, Z. (2019). *Energy-grid threat analysis using honeypots* [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. https://calhoun.nps.edu/handle/10945/62843

Lacework Labs. (2022, December 6). AndroxGh0st – the python malware exploiting your AWS keys. *Lacework*. https://www.lacework.com/blog/androxghost-the-python-malware-exploiting-your-aws-keys/

Lee, R., Assante, M., Conway, T. (2014) *German Steel Mill Cyber Attack*. SANS ICS. https://assets.contentstack.io/v3/assets/blt36c2e63521272fdc/bltc79a41dbf7d1441e/607f235775873e466bcc539c/ICS-CPPE-case-Study-2-German-Steelworks_Facility.pdf

Marshall, A. & Weiss, J. (2008). *Malicious Control System Cyber Security Attack Case Study-Maroochy Water Services, Australia*. The Mitre Corporation. https://apps.dtic.mil/sti/citations/AD1107275

Matoušek, P. (2017). *Description and analysis of IEC 104 protocol* (Technical Report No. FIT-TR-2017-12). Brno University of Technology. https://www.fit.vut.cz/research/publication-file/11570/TR-IEC104.pdf

MaxMind. (2023, April 20). *Geolocation accuracy*. https://support.maxmind.com/hc/en-us/articles/4407630607131-Geolocation-Accuracy

Meier, J., Nguyen, T. D., Rowe, N. C., "Hardening Honeypots for Industrial Control Systems" *Proceedings of the 56th Hawaii International Conference on System Sciences (2023), 1684–1693.* https://scholarspace.manoa.hawaii.edu/items/c5a1440d-c7b2-4b80-9082-7f5e585d5d84

Mesbah, M., Elsayed, M. S., Jurcut, A. D., & Azer, M. (2023). Analysis of ICS and SCADA Systems Attacks Using Honeypots. *Future Internet, 15(7),* 241. MDPI http://dx.doi.org/10.3390/fi15070241

Miller, C. (2011, November 11). *Throwback Attack: BlackEnergy attacks the Ukrainian power grid*. Industrial Cybersecurity Pulse. https://www.industrialcybersecuritypulse.com/threats-vulnerabilities/throwback-attack-blackenergy-attacks-the-ukrainian-power-grid/

Mitre Corporation. (2022, October 2). *Industroyer*. https://attack.mitre.org/software/S0604/

Mitre Corporation. (2023a, April 10). *2016 Ukraine Electric Power Attack*. https://attack.mitre.org/campaigns/C0025/

Mitre Corporation. (2023b, April 6). *Industroyer2*. https://attack.mitre.org/software/S1072/

Morishita, S., Hoizumi, T., Ueno, W., Tanabe, R., Ganan, C., Eeten, M., Yoshioka, K., Matsumoto, T. (2019). Detect Me if You… Oh Wait. An Internet-Wide View of Self-Revealing Honeypots. *2019 IFIP/IEEE Symposium on Integrated Network and Service Management*. https://ieeexplore.ieee.org/abstract/document/8717918/authors#authors

Pacific Northwest National Laboratory. (February 2018). *GridLAB-D A Unique Tool to Design the Smart Grid*. https://www.gridlabd.org/brochures/20180212_gridlabd_brochure.pdf

Quantalytics. (2020). *Q-GridPot*. https://www.quantalytics.com/wp-content/uploads/2020/07/Q-GridPot_ENG2.pdf

Ramirez, R. P., Nguyen, T. D., Rowe, N. C., Meier, J. T., "Classifying RDP Remote Attacks on User Interfaces to Industrial Control Systems," *Proceedings of the International Conference on Computational Science and Computational Intelligence, Research Track on Cyber Warfare, Cyber Defense, and Cyber Security*

Redwood, W. (2016). *Cyber Physical System Vulnerability Research* [Doctoral dissertation, Florida State University]. DigiNole. https://diginole.lib.fsu.edu/islandora/object/fsu:360429/datastream/PDF/download/citation.pdf

Rowe, N.C. & Rrushi, J. (2016). *Introduction to Cyberdeception*. DOI 10.1007/978-3-319-41187-3

Stouffer, K., Pease, M., Tang, C., Zimmerman, T., Lightman, S., Hahn, A., Saravia, S., Sherule, A., & Thompson, M. (2023). *Guide to Operational Technology (OT) Security* (NIST SP 800-82r3) https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-82r3.pdf

VirusTotal. (n.d.). *File 7c587a47eba7baa875790791ed49614a8ef49fb0c798dce570e93384b75e5851* Retrieved November 7, 2023, from https://www.virustotal.com/gui/file/7c587a47eba7baa875790791ed49614a8ef49fb0c798dce570e93384b75e5851/summary

Washofsky, A. D., Rowe, N. C., Nguyen, T. D., "Using Cloud Honeypot Platforms for Gathering Industrial-Control-System Attack Intelligence," *Industrial Control System Security (ICSS), ACSAC,* December 2021. https://www.acsac.org/2021/workshops/icss/2021-icss-rowe.pdf

Wilhoit, K. (2013). The SCADA That Didn't Cry Wolf. Trend Micro Forward-Looking Threat Research Team. https://documents.trendmicro.com/assets/white_papers/wp-the-scada-that-didnt-cry-wolf.pdf

Yang, Y., McLaughlin, K., Sezer, S., Yuan, Y., Huang, W. (2014) Stateful Intrusion Detection for IEC 60870-5-104 SCADA Security. *2014 IEEE PES General Meeting | Conference & Exposition*. https://ieeexplore.ieee.org/document/6939218

Zobal, L., Kolar, D., Fujdiak, R. (2019) Current State of Honeypots and Deception Strategies in Cybersecurity. *11th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops.* https://ieeexplore.ieee.org/abstract/document/8970921

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Fort Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California