A Resilient Dual-Purpose Web Honeypot for Analyzing Attacks on Industrial Control Systems

Andrew D. Sill
Naval Postgraduate School
andracen@gmail.com

Thuy D. Nguyen
Naval Postgraduate School
tdnguyen@nps.edu

Neil C. Rowe
Naval Postgraduate School
ncrowe@nps.edu

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 manageable but also more vulnerable to costly attacks. Honeypots are deceptive systems deployed to gather intelligence on cyberattacks and can help defend Internet-connected ICSs. We developed a resilient server that functions both as a Web honeypot and as the front end for an ICS honeypot simulating a residential electrical microgrid. Our server underwent third-party penetration testing and ran without any identified compromise on a commercial cloud machine. 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 paper appeared in Proc. Hawaii Intl. Conf. on System Sciences, January 2025.

Keywords: honeypots, World Wide Web, microgrids, cybersecurity, industrial control systems

1. Introduction

Much of modern life depends 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 online shift exposes systems to new attack possibilities. The increasing number of recent ICS cyberattacks demonstrates this.

A honeypot is a device solely intended to be a target for attacks. It provides no real service, thus most harms to it are virtual. Using honeypots also enables defenders to quickly alter device attributes and to systematically log attacks, obtaining considerable data. This supports and ultimately speeds up the task of distinguishing between genuine threats and benign traffic.

Our previous honeypot work focused on detecting cyberattacks on electrical-energy grids. We developed an improved version of the open-source honeypot GridPot (Redwood, 2016), which simulates a small power grid that uses the IEC 60870-5-104 network protocol ("IEC 104" for short). It was standardized in late 2000 and has been revised since (International Electrotechnical Commission, 2023). IEC 104 is predominantly used in Europe and Africa, and is frequently targeted in cyberattacks. The recent defensive success of Ukrainian energy companies defending against Industroyer2 suggests that the IEC 104 protocol can be implemented securely with standard network-security controls (Mitre Corporation, 2023b) (Hemsley & Fisher, 2018).

A challenge to our research was the reluctance of cyber-attackers to interact with the simulated ICS itself; instead, they targeted the Web user-interface system that provides access. That suggests turning the Web interface into a second honeypot.

To improve the Web interface for GridPot, we initially considered using GridPot's built-in Web server, which uses Conpot. Conpot is a low-interaction open-source honeypot that supports many protocols including some ICS ones (MushMush Foundation, 2024). However, Conpot was not user-friendly and required a deep understanding of both the service and command templating. It also had unusual internal communication within the GridPot 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 to pursue an approach centered on IEC 104 which enabled running a Web server separate from GridPot. This approach also improved the honeypot's security and emulated more authentic behavior, as all system-related traffic was eventually converted into IEC 104 packets. Ultimately, we created a robust dual-purpose Web server that concurrently functions as an HTTP honeypot and as a Web front end for Gridpot.

2. Background and Related Work

Industrial control systems often support processes that do not tolerate disruption and lack many security features (Stouffer et al., 2023). The need for continuous operation also limits regular product upgrades. Therefore, these systems are more vulnerable to cyberattacks than IT systems.

2.1. Attacks on industrial control systems

Many cyberattacks such as Industroyer have targeted power generation and distribution services (Hemsley & Fisher, 2018). Although IEC 104 is widely used for managing power distribution, it has minimal security. As new ICS networks connect to the Internet, devices using IEC 104 risk damage from malicious traffic. Attribution for these attacks is difficult but three major "campaigns" have been reported: Black Energy, Industroyer, and Industroyer2. Black Energy moved through information-technology networks to find controls of the operational-technology equipment and then ran a distributed denial-of-service 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, in part, the IEC 104 protocol to control devices within the Ukrainian electrical grid and left many citizens without power for three hours. Industroyer2, built from successful portions of Industroyer, was actually less successful (Hjelmvik, 2022; Mitre Corporation, 2022) because of its similarities to Industroyer and its intended victim was ready for it.

In 2000 a disgruntled worker was accused of 46 separate attacks on the local sewage system of Queensland, Australia (Abrams & Weiss, 2008). Hundreds of thousands of gallons of raw sewage were released into the local water systems, causing the local water to "turn black" and killing marine wildlife. This incident demonstrates the potential of insider threats on ICSs and the difficulty in identifying attackers.

Attackers also caused physical damage to a steel mill in Germany (Lee et al. 2014). They gained initial access through a phishing (deceptive) email and performed reconnaissance on its networks. The attackers demonstrated a good understanding of the industrial processes at the mill and could have inflicted significant damage according to the workers.

In the Colonial Pipeline ransomware attack, attackers got 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). Researchers later found no evidence that the attackers accessed the actual industrial control systems or equipment.

2.2. Honeypots

Honeypots are decoy computer systems that are designed to be attacked. A honeypot serves no other purpose, so any connection to it is abnormal and should trigger an alert. Honeypots can be used at many places within an information-technology or operational-technology network (Hilt et al., 2020). Honeypots can detect early signs of compromise, allowing organizations to respond before the attackers have time to build persistence within networks. Some attackers have lurked within systems for years before starting an assault (Hemsley & Fisher, 2018).

Using Web honeypots with HTTP to study attacks on an ICS environment is not new. In 2013 (Wilhoit, 2013) used a custom Web interface to link a simulated ICS to the Internet. They found that attackers used the open HTTP service to enter the information-technology network, eventually accessing the ICS equipment. They observed the behavior of attackers in the HTTP and Modbus protocols and characterized the attacks by the level of effort involved.

2.3. Related work

(Gyorgy & Holczer, 2020) examined IEC 104 security, identifying potential attack vectors. Since the protocol cannot validate traffic, anyone on an IEC 104 network can send an IEC 104 command to change an ICS. The protocol is also susceptible to adversary-in-the-middle and replay attacks. They also found they could create denial of service by sending IEC traffic with out-of-sequence numbers.

(Yang et al., 2014) proposed a "stateful intrusion detection" method to protect ICSs. They created a rule-based application that processed captured network traffic and identified anomalous packets. However, this requires much effort and has limited generality. The stateful intrusion detection rules must be customized for each system, as only in some systems would a particular behavior be considered anomalous. This system also requires updating when the configuration of the network changes with device addition or removal.

(Grammatikis et al., 2020) prototyped a "multivariate" intrusion-detection system that combined access controls and outlier detection based on machine learning. Their system analyzed the flows of traffic within the network with two modules. One compared the actions to a user-defined source-address access-control list and identified anomalies as traffic not in the list. Another module used unsupervised learning to identify anomalous traffic flows based on their similarity to other flows. This system required 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 ICS, provided an interface for it, and allowed unauthorized users to access it through virtual-network computing. Their experiment ran for eight months, and mostly attracted scanners and suspected bots. Until the final month of their experiment, little interaction with the simulated ICS occurred. They concluded that it was difficult to attract attackers interested in more than ransomware and "crypto jacking" (Hilt et al., 2020).

Recent work on ICS and SCADA attacks used the Conpot honeypot described in Section 1 (Mesbah et al., 2023). They described how an HTTP honeypot can help protect an operational-technology network. They connected the honeypot to the Internet and analyzed the traffic received. They reported that the HTTP protocol was used for over 83% of the sessions.

Several other options for running an ICS honeypot offer HTTP services (Dalamagkas et al., 2019). However, Conpot is a well-known open-source honeypot with community support and continuing development so including HTTP in Conpot makes it a more likely attack target.

3. Honeypot design

Our own previous research has also confirmed that most attackers ignore the ICS features of a honeypot and focus on exploiting the host system instead (Meier et al., 2023). Previously we used a modified GridPot that included both a Windows-based SCADA user interface and a Web-based user interface. A weakness of the previous implementation was that its Web server displayed a page showing a subset of devices and provided no user interaction.

To improve the Web user interface, we modified our honeypot design to include a new component we created, the SCADA User Interface Server. This runs a custom Web server called the SCADA Web Server (SWS) and a middleware program, GPTalker, we also created. SWS acts as a Web honeypot and provides a Web-based human-machine interface (HMI) to interact with GridPot. GPTalker allows SWS to communicate with Conpot's IEC 104 server, which translates the protocols and communicates with the power-grid simulator, GridLAB-D. SWS and GPTalker are Python programs that can function independently from each other.

A benefit of this approach is that the data collected by SWS while waiting for potential attackers to interact with the HMI can be used to compare to the data gathered from the HMI to look for insights. In particular one could look to see if IP addresses are shared between the data sets, and if so, the delay between initial traffic and later interaction with the HMI.

The high-level design of our dual-purpose honeypot, minus the virtual machine hosting it, is shown in Figure 1. Although the Logging Server is hardened and not exposed to the Internet, it could potentially be compromised if the firewall managed by the cloud-service provider, i.e., DigitalOcean, failed to block unauthorized traffic.

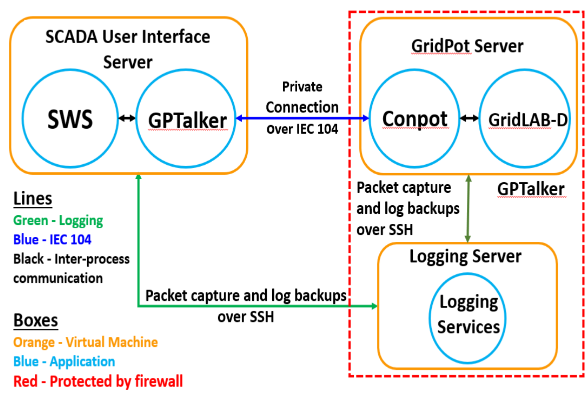

Figure 1. Overview of dual-purpose honeypot.

The design of the SCADA User Interface Server is shown in Figure 2.

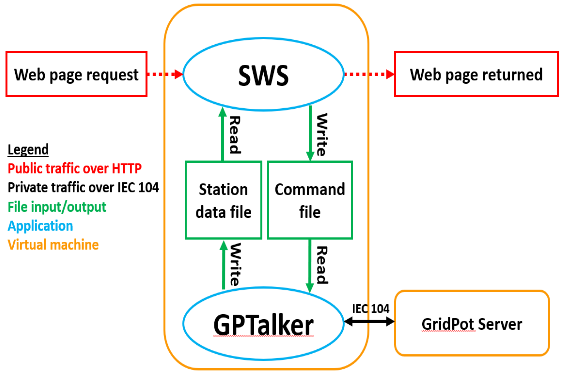

Figure 2. SWS and GPTalker interactions.

SWS handles all Web requests and GPTalker handles all communication with GridPot. SWS and GPTalker communicate through two shared files, a text-based command file and a text-based JavaScript Object Notation (JSON)-format data file. SWS uses the data file to fill in information on the webpage for the attacker. Each application should read its input file every two seconds and create a new output file each time it has an update.

3.1. The GPTalker application

The GPTalker program handles IEC 104 communication between the Web server and GridPot with shared input and output files. GPTalker periodically generates a JSON file for each station (labelled as "Station data file" in Figure 2). This file describes all IEC 104 logical addresses (Information Object Addresses or IOAs) in a system for which GPTalker receives updates during a particular session. The station data file holds the most recent value for each IOA, along with historic information such as highest and lowest value recorded, which can be displayed by SWS on the main Web page. Every two seconds, GPTalker reads a text file produced by SWS (labeled as "Command file" in Figure 2) to create IEC 104 frames to send to GridPot. Each line of the text file contains an IOA and its desired value. For commands, GPTalker translates the values in the command text file into the appropriate IEC 104 commands using an internal key-value mapping between IOAs and object type. GPTalker provides connection status output to the user; all inputs to interact with GridPot are taken from the command text file.

GPTalker can communicate with GridPot without user or SWS input to receive updated values. It automatically sends IEC 104 interrogation commands to GridPot to accomplish the data transfer; GridPot updates devices every few seconds. GPTalker also sends interrogation commands when it does not receive updates from GridPot within 5 seconds, thus preventing GridPot from timing out. GPTalker uses Scapy to create TCP connections between GPTalker and GridPot, create frames, and transfer frames between devices; Scapy is an open-source packet-manipulation module (Biondi, 2023) which allows us to dissect and create IEC 104 frames.

GPTalker can activate a "deception" layer which simulates operational statuses of GridPot's electrical devices. This layer simulates the actions of the electrical switch and meters available from GridPot without actually changing GridPot. When deception is enabled, GPTalker replaces the value received from GridPot with a deceptive value. For inactive meters (set to "off"), GPTalker adjusts the value received to zero; for the switch, it retrieves the status of the corresponding meters and linearly adjusts the switch's output voltage to match. Deception was enabled for all live experiments.

3.2. The SWS application

The SWS program processes the information from the station data file (i.e., GPTalker's output file), and displays it in a user-friendly format. It uses cascading style sheets (CSS), Java, and HTML to provide a realistic user interface. SWS logs interactions with users, provides debugging data for GPTalker, and acts as a HTTP honeypot. It supports GET and POST HTTP requests, and logs received traffic. The host system stores all external traffic in packet captures that can be cross-referenced with the logs. The Web interface includes meters, a switch, a transformer, and a regulator. Other Web pages include a login page, a page for requesting access, and administrative files.

SWS uses the open-source Gunicorn Web-application software (Gunicorn, 2023) built on Flask (Flask, 2023), an open-source Python module for hosting a simple Web server. We chose Gunicorn due to its support for multiple threads for a Flask application, allowing the server to handle several clients simultaneously. This multi-client capability enables collecting more data and providing a realistic interface. Flask allows Python programs to build pages based on Jinja2 templates 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; we relied on Flask to securely handle user input and block unauthorized access to resources.

To improve the honeypot's realism, we posted a login page which was a prerequisite for privileged actions. Instead of using cookies to track a user, we implemented a "whitelist" database containing authorized IP addresses that SWS uses to track Web clients. To persuade attackers to attack the simulated electrical devices instead of the Web server, we implemented additional safeguards. First, SWS "sanitizes" all data from users by examining it and removing all characters that are not part of a small "safe list", namely letters and numbers, to minimize the risk of injection or memory-based attacks. Additionally, SWS also maintains an internal list of IOAs that accepted commands, and blocks the requested IOAs not in the list. Furthermore, GPTalker keeps a map between IOAs and command types to prevent the Web client from relaying unpermitted commands to GPTalker.

4. Implementation and experiments

Our honeypot requires three systems: a user-interface workstation, a power-grid decoy running GridLAB-D, and a logging server. Each system was a Linux virtual machine running Ubuntu 20.04.3. GPTalker and SWS ran on the user-interface workstation. Except for the public IP address of the Web server, every instance of our honeypot uses the same IT and OT network configurations.

4.1. Constraints and assumptions

GPTalker and SWS ran as user processes on the base Linux operating system to reduce networking issues. We relied on DigitalOcean's firewall to restrict network access to the hosted services on our cloud platforms. We designed GPTalker to communicate with a GridPot instance the same way as GridPot's Windows SCADA application does, using a subset of the available IEC 104 commands. SWS was designed based on the results of previous experiments that showed Web requests did not exceed 10 per minute.

The Web server had no domain name. This prevented us from using a certificate for the Web server, a necessary feature for the secure HTTP version, HTTPS.

4.2. Security features

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 would be restricted by the operating system. Also, the Web server's 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. This 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.

Before the live experiments, an external subject-matter expert (SME) in penetration testing probed our system to check that the logging and security features were resilient against common exploits. The configuration they tested was the same setup used in live experiments.

4.3. Experiments

We ran three experiments with three separate GridPot instances running in the DigitalOcean cloud environment, hosted on the West Coast of the United States. 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.

For each experiment, we started fresh instances of the Tshark network analyzer, SWS, and GPTalker. Upon startup, GPTalker connected to the GridPot server and reconnected as needed. 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.

Experiment 1 ran from 30 September 2023 to 8 October 2023. It used the same website layout and GPTalker as the development testing. DigitalOcean's firewall was set to allow any IP address to connect to SWS on port 80. It only took six hours to receive the first ten external addresses. In this traffic, we saw scans from Bingbot, Censys, and zgrab; these scanners 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 unauthorized login attempts.

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 put some of these resources on the Web server and any attempt to access them or a nonexistent file was logged by the server. These resources 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 directed the attacker to log in to a nonexistent virtual private network to gain access. We added several common Web 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 SWS logs 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 successful 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 systematic testing of the system for vulnerabilities, and it appeared all attacks were automated.

Experiment 3 began on 14 October and ended on 27 October 2023. For this experiment, 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 seen 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 concluded that this error was not caused by an attack.

5. Results

We ran our experiments for 27 days, collecting information from the SWS logs to identify traffic patterns and assign categories to IP addresses. We recorded 55407 HTTP connections from 933 unique IP addresses, representing 216 organizations and 61 countries. We categorized the traffic as malicious, scanning, or legitimate based on their use of particular traffic patterns. We defined legitimate traffic as traffic that loaded all associated resources for the Web page such as the icon and JavaScript without attempting to load protected resources or exploit vulnerabilities. This is in contrast with scanning traffic that would load a Web page without loading all of the resources or malicious traffic which might also target ".env" files or search for unlinked pages. We classified the interaction with the Web server by "HTTP patterns", which was created by tracking the resources requested, such as a Web page or image file, in their particular order per transaction. Given the stateless nature of HTTP, we considered a transaction to be all of the traffic from a particular IP address within a few seconds. We then consolidated the patterns into ten categories based on the number of resources requested, from 1 to 9 and then 10 or more. We noted that malicious HTTP patterns were most common when the resource-request count was one, often an attempt to read a ".env" file, or more than 15. Legitimate traffic was generally five to seven resource requests long, enough to load the Web page itself and supporting documents. Scanning traffic was often three to five resource requests, typically for the index Web page, the robots.txt file, the Web page icon, and sometimes the security.txt or sitemap.xml files. Figure 3 depicts the relationship between HTTP pattern length along the horizontal axis and the number of patterns that match that length along the vertical axis over all three phases. The lines are colored to differentiate the types of traffic.

We used Google My Maps to visualize the approximate location of each IP address, obtained from the MaxMind geolocation service (MaxMind 2023), as shown in Figure 4 (Google, 2023). Red pins indicate IP addresses that sent at least some malicious traffic, and blue pins correspond to IP addresses that sent only legitimate traffic. We surmised that the geographic location of the DigitalOcean data center used in the experiments might have influenced the originating IP addresses observed.

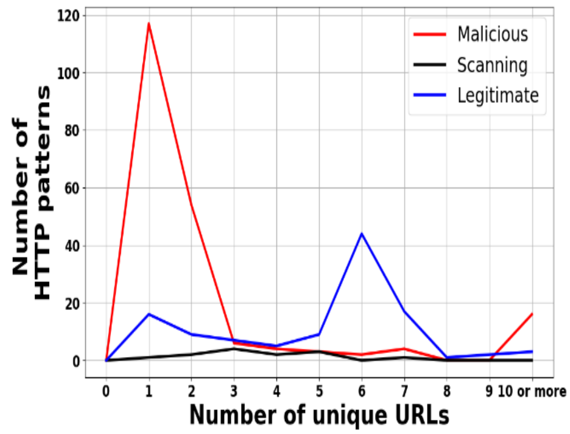

Figure 3. Number of HTTP patterns by request count and traffic type over all experiments.

Figure 4. Unique IP addresses overlaid world map. Source: Google (2023).

5.1. Experiment 1 results

Experiment 1 ran 192 hours. The observed attacks on SWS ranged from old techniques such as directory traversal to find the password file to more modern attacks exploiting the Log4j and Laravel vulnerabilities (Lacework Labs, 2022). These attacks were unsuccessful and did not find any vulnerabilities in SWS. Several were repeated despite their previous failures. 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. Each type of traffic increased over time, with legitimate traffic increasing the most quickly, followed by malicious traffic. The fewer addresses associated with scanning suggests that scanning services use consistent source IP addresses and run continuously. In this experiment, some scanning services visited our 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 classified in two categories, malicious and legitimate plus scanning. Most traffic came from large corporations such as cloud-service providers. 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.

5.2. Experiment 2 results

Attackers tried to exploit several vulnerabilities in 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. They also tried to use the resources of SWS by "tunneling" through the HTTP CONNECT method. The objective was 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 had the redirection worked.

We observed attempts 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 command and host 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 access protected resources.

Traffic patterns were similar to Experiment 1. Legitimate traffic was the most common and scanning traffic was the least. All three categories of traffic increased over time, indicating that the honeypot should continue to identify new IP addresses and new traffic patterns if it ran longer. The ratios of malicious to legitimate IP addresses by 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 in Experiment 2. The distribution of malicious and legitimate IP addresses by country resembled that of Experiment 1. The three largest organizations, by unique IP addresses, remained the same, but no malicious traffic came from the Hong-Kong-based organization observed in Experiment 1 according to CENSYS and Constantine Cybersecurity.

In Experiment 2, we saw no indication that SWS was compromised or that attempts to interact with the Web-based human-machine interface were made. The cosine of similarity between the three kinds of traffic in 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.

5.3. Experiment 3 results

Experiment 3 used a new cloud-based virtual machine with a different IP address. Experiment 3 lasted 338 hours and recorded 549 unique IP addresses. Despite the new IP address and extended run time, Experiment 3 demonstrated similar traffic patterns to the previous two experiments. The relationship of country to traffic type and the distribution of traffic over organization also matched the previous two experiments.

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 DigitalOcean recycles IP addresses, possibly our IP address had been assigned to a vulnerable site previously. The first step of the attack had the Web server connect to a remote 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, 2023). The consensus was that it was Mirai malware, 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.

We noticed several other attempts to compromise the Web server, and one IP address that sent over 400 HTTP messages to find information about the Web server or attempt exploitation of a potential vulnerability. The cosine similarity between the three kinds of traffic between Experiment 2 and Experiment 3 was 0.9991, suggesting a strong similarity in the HTTP traffic despite using a new IP address.

5.4. Discussion

Across all three experiments, we observed 933 unique IP addresses over 656 hours. Figure 5 shows the cumulative results of the classification; the vertical lines separate experiments. The graph shows that the increase in unique IP addresses seen over time was nearly constant even with a new IP address in Experiment 3.

The slope of the graphs in Figure 5 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.

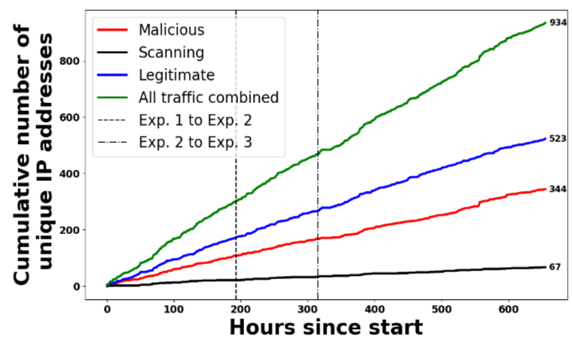

Figure 5. Cumulative unique IP addresses by traffic type over time across all three experiments.

We combined the daily traffic patterns of three experiments 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 6. No obvious patterns are apparent. Almost half of the IP addresses were linked to the United States, 455 of the 933 observed. Figure 7 shows the distribution by countries and Figure 8 shows unique IP addresses by organizations across all experiments.

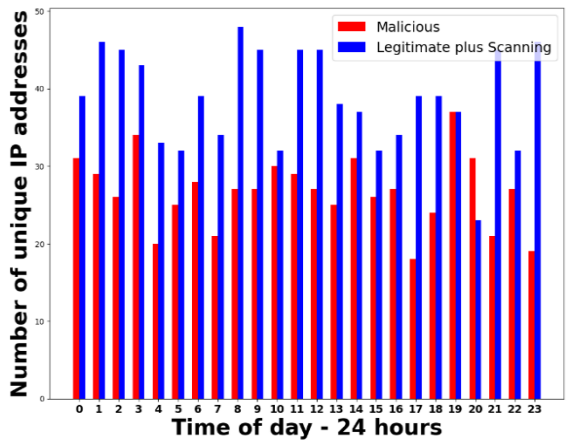

Figure 6. Unique IP addresses by traffic type by hour of day for all three experiments.

Prior research has shown that an industrial-control-system honeypot may take months to receive ICS-specific attacks (Hilt et al., 2020). This finding matched our experience as there were no ICS-specific attacks in any experiment. 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, if it continues running, it 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.

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. Attackers probably chose not to attack Gunicorn because it has no published vulnerabilities.

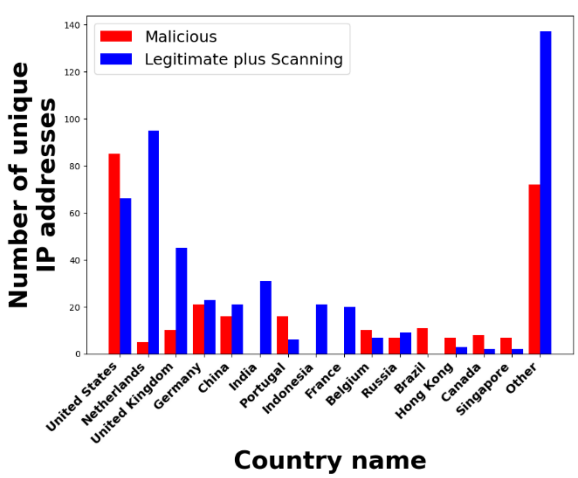

Figure 7. Types of unique IP address by traffic type by country across all three experiments.

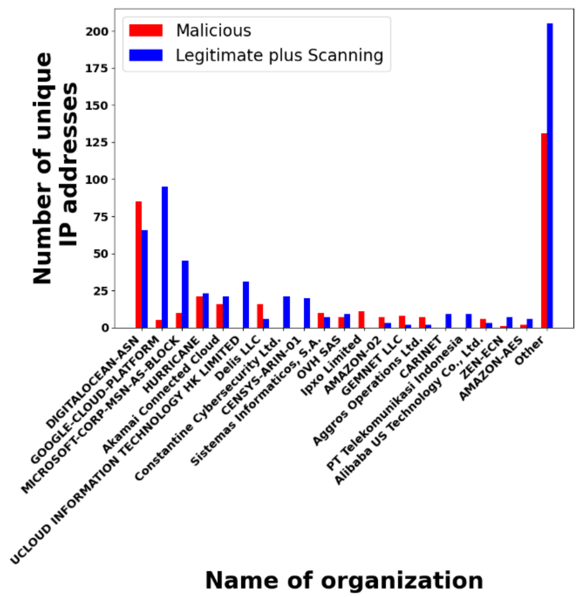

Figure 8. Unique IP addresses by traffic type over time across all experiments (non-cumulative).

During internal penetration testing by a cybersecurity SME, we found that DigitalOcean 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.

The SME also gathered enough information about SWS to identify its applications and the version of the underlying 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 Web server, and the latter were also protected by a firewall.

7. Conclusion

We successfully built a resilient dual-purpose Web server that provided a straightforward human-machine interface for an ICS honeypot and acted as a separate HTTP honeypot to collect attack data on Internet-exposed Web servers. We ran 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.

Due to the short test periods, data collected was limited and no ICS attack was observed. 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. Further research could use the data collected to train a supervised model to perform automated classification. We also plan to deploy our honeypot in Europe where IEC 104 is commonly used; we expect there would be a regional influence on the results as we had observed in our three experiments.

Disclaimer The views expressed in this material are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government .

8. References

Abrams, M. & Weiss, J. (2008). Malicious control system cyber security attack case study: Maroochy Water Services, Australia. http://www.mitre.org/sites/default/files/pdf/08_1145.pdf

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

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

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

Flask (2023, November 26). Welcome to Flask – Flask documentation (2.3.x). https://flask-docs.readthedocs.io/en/latest/

Google (2023, November 10). My Maps – About – Google maps. 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 (2023, November 11). Configuration Overview. 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

Hemsley, 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

International Electrotechnical Commission. (2023). International Standard IEC 60870-5-104. International Electrotechnical Commission.

Lacework Labs (2022). 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

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/

MushMush Foundation (2024, June 7). Conpot. https://github.com/mushorg/conpot

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

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

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 (2023, November 7). File 7c587a47eba7baa875790791ed49614a8ef49fb0c798dce570e93384b75e5851 https://www.virustotal.com/gui/file/7c587a47eba7baa875790791ed49614a8ef49fb0c798dce570e93384b75e5851/summary

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