Creating Effective Industrial-Control-System Honeypots

Neil C. Rowe
U.S. Naval Postgraduate School

Thuy D. Nguyen
U.S. Naval Postgraduate School

Marian M. Kendrick
U.S. Naval Postgraduate School

Zaki A. Rucker
U.S. Naval Postgraduate School

Dahae Hyun
U.S. Naval Postgraduate School

Justin C. Brown
U.S. Naval Postgraduate School

Cyberattacks on industrial control systems (ICSs) can be especially damaging. Honeypots are valuable network-defense tools, but it is difficult to simulate the specialized protocols of ICSs. This research compared the performance of the Conpot and GridPot honeypot tools for simulating nodes on an electrical grid with live attacks. We evaluated their success by observing their activity patterns and by scanning them. GridPot received a higher rate of traffic than Conpot, and many visitors to both, as well as scanners, did not realize they were honeypots. This is good news for collecting useful attack intelligence with ICS honeypots.

Keywords: industrial control systems, honeypots, testing, Conpot, Gridpot, traffic, network protocols, deception

INTRODUCTION

Industrial control systems (ICSs) are important concerns for cybersecurity even though they are not often attacked, because an attack on critical ICS infrastructure such as a power grid can have catastrophic effects on the operation of business and government [1]. ICSs are vulnerable due to their proprietary software and protocols, legacy devices, and outdated operating systems. Furthermore, many are difficult to update because they must provide continuous operation.

Honeypots (decoy digital systems) are useful defensive tools to investigate cyberattack threats and other kinds of malicious activity. It is useful to develop honeypots for ICSs to collect intelligence on their kinds of attacks. However, creating an effective honeypot for ICSs is more difficult than for most network nodes because they must simulate a wide variety of real-time industrial processes with proprietary protocols as well as standard protocols like TCP/IP. If ICS honeypots are not simulated accurately, attackers may realize they are being deceived and go away, failing to provide useful intelligence.

We installed two open-source industrial-control-system honeypots, Conpot [2] and GridPot [3], and studied the live traffic to them to evaluate their effectiveness. GridPot is an extension of Conpot designed to simulate different electric-grid models. Conpot is simpler and served as a control experiment to see if GridPot was a significant advance. While they have been assessed separately, no previous experiments have compared their performance in the same environment. We also checked whether a commonly used network-scanning tool could identify these and similar honeypots.

BACKGROUND

Electrical Grids

Our experiments focused on ICSs for electrical grids. A grid (or bulk electric power system) includes generation, transmission, distribution, and end use [4]. The generation of electricity occurs in many ways including coal-fired plants, natural-gas plants, solar farms, wind turbines, and hydroelectric plants. Generated electricity is passed through transformers to step up voltage to a high level for transmission lines which deliver electricity to substations. Substations use a transformer to step down the voltage to a low level before it is distributed to end users. Switch devices direct the flow of current by opening and closing circuits. Regulators ensure a constant and safe voltage level is maintained throughout the power system.

Software now allows operators to monitor and control portions of an electrical grid remotely. SCADA (Supervisory Control and Acquisition) devices, a subset of ICSs, allow operators to monitor many devices over a wide area. An IED (Intelligent Electronic Device) like a controller or a digital relay can send or receive data or control to or from an external source. ICSs are a subtype of cyber-physical systems, which integrate physics and logic to allow interaction between digital, analog, physical, and human components.

Electrical grids and related infrastructure have been targeted by a number of types of malware [5]. Well-reported cases involved Crashoverride or Industroyer, Stuxnet, Blackenergy, and Havex. In December 2016, a transmission-level substation was attacked in the Ukraine using Crashoverride [6]. Proof that grid operations can be affected by a cyberattack was demonstrated in a U.S. Department of Energy test which caused the self-destruction of a replica power plant generator by means of a cyberattack [7].

Cyber threats that target the distribution portion of the bulk power system can exploit load shedding, advanced metering infrastructures, and demand-side management [8]. The U.S. has seen load-shedding incidents in recent years that have caused cascading power outages. In 2007, Tempe, Arizona experienced large-scale load shedding which affected 98,700 customers for almost an hour.

Honeypots for ICSs

Our research explored the use of honeypots to collect data of cyberattacks on electric utilities. Honeypots can be most useful for gathering data when they entice attackers into revealing a rich set of information about their attacks [9]. Honeypots that conceal their purpose through deception are more productive because attackers do not want to interact with honeypots, since attackers are unlikely to subvert them and they try to learn attack methods to better foil them [10]. High-interaction honeypots especially can confuse attackers through program-based or scripted interaction designed to encourage further exploration. However, some botnets can detect honeypots [11] by recognizing firewalls and filters on outbound traffic. Since honeypots try to prevent their launching an attack on a third party, they often have an intrusion-detection system that filters for outbound activity. A bot that is prevented from propagating an attack would notice that and recognize it is on a honeypot.

Several projects have used honeypots to monitor attacks on ICSs. One project deployed a large-scale cloud-based low-interaction honeypot system for 28 days using Amazon's EC2 cloud environment [12]. This experiment monitored the protocols DNP3, ICCP, IEC-104, Modbus, SNMP, TFTP, and XMPP. The researchers concluded that reconnaissance occurred more often than actual attacks, and reconnaissance targeted single protocols rather than combinations of protocols. They also identified a positive correlation between Modbus reconnaissance and discovery by the Shodan network-scanning tool of Modbus-enabled devices.

HoneyPhy [13] addressed the problem that ICS honeypots could be unrealistic in modeling device physics and device-actuation times and therefore could be identifiable. One honeypot they designed provided general structure-modeling processes and devices implementing a simple heating-ventilation system. Another modeled a simplified water-treatment system.

The GasPot honeypot simulated the Veeder-Root Guardian aboveground storage-tank monitoring system [14]. Logs revealed unauthorized reads and writes, defacement, and denial of service attacks . GasPot was subsequently included with Conpot as the guardian_ast template. Similarly, the kamstrup_382 template provided by Conpot mimics a Kamstrup model 382 smart electrical meter, providing an electrical power metering service on port 1025 and a management service on port 50100. The hardware on which this is based provides measurement of electrical circuits up to three-phase ones, allows remote access by way of optional modules for TCP/IP networking over Wi-Fi, GSM, and GPRS connectivity, and enables local interaction via optional serial and infrared interfaces.

Another honeypot architecture used geographically dispersed nodes hosted on Amazon Elastic Cloud Compute with emulation support for the protocols DNP3, ICCP, IEC 104, Modbus, SNMP, TFTP, and XMPP [15]. In experiments, the Shodan network scanner provided the first unsolicited interaction with five of the six honeypots, and attacks began only after each honeypot was listed on Shodan. This suggests that attackers are exploiting network-scanning databases.

Other similar honeypot projects were CryPLH [16] and the Digital Bond SCADA Honeynet [17].

Network Scanning

Honeypots may be detectable to by distinctive clues they provide to network scanners. Transport-layer scanners send TCP, UDP, and ICMP packets to a remote host and analyze the responses. Nmap (nmap.org) is popular transport-layer scanner. Protocol scanners interact with proprietary communications protocols. For instance, Digital Bond's Redpoint uses Nmap's NSE tool and the s7-info.nse script for simple interactions with Siemens PLC devices using the S7Comm protocol on port 102 [18]. Some scanners focus on specific protocols like zmap (zmap.io), and others scan more broadly such as ZoomEye (www.zoomeye.org).

The Shodan scanner (www.shodan.io) scans Internet-connected hosts continuously [19]. It appears to pick network addresses randomly and is more successful in the IPv4 address space. The Shodan website provides a service called Honeyscore which uses a proprietary algorithm to identify honeypots. Project SHINE for two years queried Shodan for selected search terms, starting with manufacturer names from trade magazines and blogs, and continuing by query terms derived from the results of searches on manufacturer names [20]. Eventually they sampled 2,186,971 devices from which they derived 578 unique search terms for traditional industrial-control system devices and 349 search terms for other devices with physical controls of some kind. The protocols studied were S7Comm, Modbus, DNP3, EtherNet/IP, and BACNet. Scanning could include possible duplicate devices and network-address-translation connections.

In an experiment, a Siemens RuggedCom RS910 was configured to respond as a water pump [21]. This RUGGEDTRAX honeypot was configured to run SSH, HTTP, HTTPS, and DNP3 services. The device firmware name and version were displayed on its HTTPS web page, alongside a fictitious system name indicating a water well in a specific location. The firmware sent a variant of the "goahead" embedded web-server banner. This honeypot was indexed by Shodan two days after being connected to Internet.

Similar results were observed in [19] in identifying industrial-control-system devices with Shodan. For 55 days they connected four Allen-Bradley ControlLogix 1756-L61 controllers to the Internet as honeypots. Two controllers connected with an unmodified standard HTTP banner, one with an obfuscated banner, and one with an "advertised" HTTP banner. All four were probed within four days of deployment, and two within a day; data from all four was visible on the Shodan website within 19 days of deployment, despite never having provided their addresses to Shodan.

OUR EXPERIMENTS WITH CONPOT

We first tested a low-interaction ICS honeypot, Conpot from conpot.org [2], with live attackers. It simulates an ICS like a power plant and collects information on cyberattacks. It acts as a master server for commonly used ICS network protocols and provides multiple templates that simulate simple forms of them. Conpot served as our control experiment for the subsequent experiments with GridPot. More details are in [22].

Methodology

Our experiments used a laptop computer with a Linux Ubuntu 16.04.3 LTS operating system. A Linux virtual machine was installed using Oracle VM Virtualbox 5.1.20, and Conpot 0.5.1 was installed in it. A local network was set up outside of our school's firewall to make it easier for external attackers to discover the honeypot without advertising it. Both the host and virtual machines used statically assigned IPv4 addresses and communicated with internal bridged networking. While our local network could not be mistaken for a major ICS installation, it could simulate a small node on an electrical grid.

In our first experiments we used Conpot's default template which simulated an electric-power plant using Siemens SIMATIC S7-200 Programmable Logic Controllers that communicate with at least two slaves. Conpot simulates the initial interactions with the protocols HTTP, Modbus over TCP/IP, S7Comm, SNMP, BACnet, IPMI, EtherNet/IP, and CIP. We created parsers to extract clues from log data for each protocol such as IP addresses, ports, and basic protocol-specific data. The IPMI emulator was special in that it mimics a baseboard management controller supporting functions such as "chassis status" and "user list", and permits manipulation of system power [23].

Conpot Results

Our honeypot collected live traffic over four months from October 2017 to February 2018 except when the main log was backed up. The network protocol analyzer Wireshark (www.wireshark.org) monitored and captured network traffic. Table 1 summarizes the traffic counts by protocol seen by Conpot and in the subsequent experiments with GridPot.

TABLE 1
TRAFFIC RATES IN LIVE-TRAFFIC TESTING OF OUR TWO HONEYPOTS

Protocol	Conpot count	Conpot percentage	GridPot count	GridPot percentage
HTTP	7,366	66.7%	9,641	93.0%
Modbus	2,316	21.0%	621	6.0%
S7Comm	645	5.8%	102	1.0%
BACnet	311	2.8%	0	0.0%
IPMI	262	2.4%	0	0.0%
EtherNet/IP	154	1.4%	0	0.0%
Total	11,054	100%	10,364	100%

Traffic to the honeypot was not particularly varied. Only 59 of the 2,316 Modbus activities had a valid Modbus function code of 0x03 (Read Holding Register), 0x2b (Read Device Identification), or 0x11 (Report Server ID). This suggests usage was for reconnaissance only. Spikes of activity occurred on November 13, November 23, December 8, and January 10. On November 16, ICS-CERT released a security advisory about Siemens SICAM equipment; this product supports Modbus, as does the honeypot, so that probably explains two of the activity spikes.

For the EtherNet/IP protocol, valid commands observed were NOP, RegisterSession, and ListIdentity; invalid and null commands were also observed, suggesting probing attempts. For S7Comm, all packets had a data length of 0 or 8 and a request ID of 0. For BACnet, some data apparently was sent for all 78 established connections using an invalid type, resulting in 78 decoding errors, but we could not identify where it came from. For IPMI activities, 129 had new traffic, 30 were returning traffic, and only 2 sessions were properly closed.

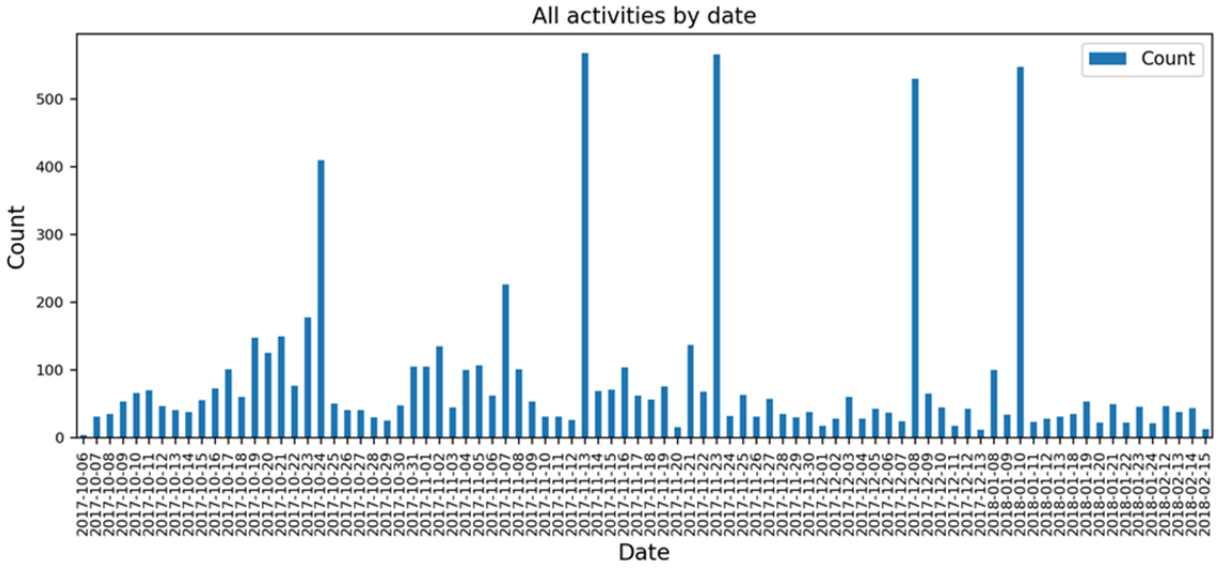
More overall activity occurred in October and November, and then it gradually declined (Figure 1), as is typical of new honeypots. The decline was predominantly due to HTTP and the spikes were predominantly due to Modbus activities. Claimed nationalities of the attacks were all over the world, suggesting multiple agents (Table 2).

TABLE 2
COUNTRY CLAIMED BY CONPOT ATTACK TRAFFIC OBSERVED

Country	Percent	Country	Percent
U.S.	27%	Netherlands	5%
China	22%	Hong Kong	4%
Brazil	8%	France	4%
Russia	8%	Indonesia	3%
Egypt	6%	Indonesia	3%
India	6%	Japan	3%

As could be expected with a low-interaction honeypot, overall traffic was mostly reconnaissance. Conpot traffic concentrated on the most familiar protocol, HTTP, despite its limited capability on this honeypot. Modbus traffic appeared to be testing access permutations using many malformed packets. EtherNet/IP and Modbus activities showed that Conpot had difficulty distinguishing between real protocol activities and embedded protocol requests sent to the ports on which Conpot listened. Since Conpot only logs basic flow data, embedded payloads went unnoticed.

FIGURE 1
TOTAL CONPOT ACTIVITY COUNT OVER TIME



OUR EXPERIMENTS WITH GRIDPOT

We also tested the GridPot open-source high-interaction ICS honeypot framework (github.com/sk4ld/gridpot) with both artificial and live attackers. It uses GridLAB-D (gridlab.org), a simulation and analysis tool for power-distribution systems developed by the U.S. Department of Energy and Pacific Northwest National Laboratory to enable testing of power-distribution systems [24]. Our copy was run on the same local network as our Conpot experiments to enable a fair comparison. More details are in [25].

Setup

GridLAB-D Model objects used in our honeypot were the node, link, switch, transformer, and regulator. Object-node properties include phases, object connections, open/closed status, power flow, temperature, tap position, and configuration. Objects can include schedules of parameter values over time. Network protocols we supported with GridPot were HTTP, Modbus, S7Comm, SNMP, and IEC 61850.

GridPot uses a honeypot layer and a modeling layer to add electrical components and integration between GridLAB-D and Conpot, including IEC 61850 communication. GridPot's honeypot layer is derived from Conpot, adding an XML-formatted GridPot template that specifies to which GridLAB- D model to link. Additional Python-coded GridPot files are included in the honeypot layer to retrieve parameter values from the running model in real time using port 6267.

GridPot's primary modeling layer uses GridLAB-D's Powerflow module, adding GridPot-model (GPM) configuration files. Powerflow simulates voltage and current values across an IEEE 13-node grid model with 15 houses. GridPot code includes additional modeling features for "intelligent electronic devices" under an electric components subdirectory that contains code to simulate a GE Brick Merging Unit and a generic input/output switch-control device.

Our experiments used both a test environment and a live environment. The test environment altered the Conpot code to use its localhost IP address instead of the host environment's external IP address, which kept traffic internal to our machine. The live environment enabled external user access and attack-data collection from outside the school firewall. We used network-address-translation, host-only-adapter, and bridged-adapter network settings in both environments.

We used Oracle VM Virtualbox 5.2.22 to install a virtual machine in which to place GridPot. It ran the same operating system as the host. The honeypot layer initialized Conpot using the GridPot template and the modeling layer initialized the GridLAB-D model IEEE_13_Node_With_Houses. We updated the gridpotmodel_file field value to link with our custom GPM file for the latter. Four protocol servers were started upon launch as in the original code. Modbus was used on TCP/IP port 502, connecting to one client and two servers. The IEEE_13_Node_With_Houses model contained switch, transformer, and regulator objects used for the Conpot integration, and required minimal code modifications. A schedule based on local time was used to alter power flow readings across the switch. Real-time power-in and power-out simulated switch parameter values which were displayed on our web-based interface. We created a GPM file to link with the switch, transformer, and regulator objects specified in the IEEE_13_Node_With_Houses GLM file by modifying an existing GPM file.

Testing

We used the Wireshark network-protocol analyzer (www.wireshark.org) to confirm that the Conpot and Gridpot logs were complete; the Nmap (nmap.org) and Linux Netstat built-in network scanners to check which ports were open; the Nessus vulnerability-assessment tool (www.tenable.com) to check for obvious vulnerabilities; and the Metasploit penetration tester (www.metasploit.com) to test the logging of attacks. Nessus and Metasploit were used with SCADA plugins.

First we tested if our web-based interface display was accurate to the running model by pointing a web-browser to GridPot's HTTP server using localhost IP address and TCP port 80, and comparing the results to the GridLAB-D model instance that listened on port 6267. We used the Netstat tool to determine which ports were opened by GridPot. We then ran scans using Nmap, Nessus, and Metasploit against our honeypot. We focused these scans on open ports and probed for operational-technology devices using the Modbus protocol by running detection, discovery, and interaction scans. We focused on Modbus since it is the most common ICS protocol.

Host-to-virtual-machine baseline testing first required altering our network connectivity from disabled-network status to host-only status. We tested the status using "ping" commands between our host and virtual machine, and confirmed receipt of a "network is unreachable" error when trying to ping an arbitrary IP address. The same scans using Nmap, Nessus, and Metasploit were then performed, changing the IP address to our GridPot virtual machine instead of the localhost address. To generate useful log data for comparison against live attacks, we conducted a denial-of-service scan using an auxiliary Metasploit module.

We modified portions of the default Conpot configuration in the template to eliminate well-known clues to Conpot and fool more attackers. Testing confirmed our web-based interface accurately displayed values of the running GridLAB-D model. Netstat results confirmed that our four protocol ports (HTTP, Modbus, S7Comm, and SNMP) were open. Results of the Nmap, Nessus, and Metasploit scans also saw these ports as open and that Modbus-enabled devices were running on our honeypot.

Our live honeypot collected data over 19 days from April 11-30, 2019. GridPot ran continuously except when we fixed a broken link. Conpot stopped logging twice, which could have been due either to bugs or malicious activity that we could not distinguish.

GridPot Results

Live GridPot traffic data collected by Wireshark totaled 1,525,059 packets and 165 MBs. This was a higher traffic rate of 545 interactions per day versus 92 with Conpot. The GridPot protocol distribution differed from that of Conpot (Table 1). BACnet, IPMI, and EtherNet/IP were not included in our GridPot template and were not logged, though there was likely a small amount of their traffic judging by the Conpot results. HTTP traffic was a larger percentage of traffic with GridPot. This is likely due to the additional HTTP deceptions beyond Conpot provided by GridPot, though a contributing factor could be the increasing numbers over a year of real electrical grids that use HTTP [1]. It thus appears that GridPot's additional deceptions are justified and effective.

The greatest number of packets (1,013,726) came from a California-based cloud-hosting corporation. It came from an address registered to Fastly, a content delivery network provider. GridPot exchanged 84,588 packets with just one Fastly address using MaxMind. Traffic from this address occurred throughout our collection, and contained over 26,000 retransmissions of nearly identical ACK messages, so this campaign was not intelligent. The second-highest source of packets was an IP address registered to an LLC in St. Petersburg, Russia, which was responsible for 56,280 packets of 3,221KB. Censys.io traced this address to a Debian-based SSH server in Amsterdam. 38,754 were SYN packets sent to GridPot, and there were also RST packets.

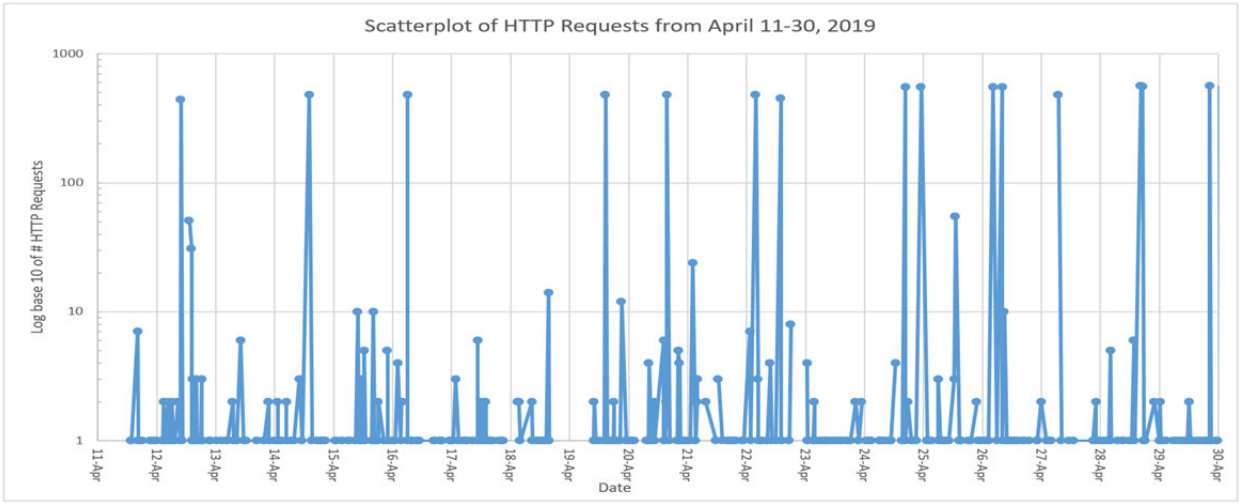
Seven different HTTP methods were seen in the HTTP requests, including 79 "None" and 78 "Bad". Significant spikes in the number of HTTP requests occurred almost daily (Figure 2). Each peak contained roughly the same number of GET and POST request methods in the same order with varying speeds. This suggests these attackers used a single HTTP scanning tool to conduct the attacks, and were not inspecting their results carefully. We compared this apparent scanning to our host-to-virtual-machine logs of recorded attacks and concluded that the real attacks did not match any recorded attacks because of, among other things, their quite different time patterns.

Heavy scanning with Modbus was seen twice. 39 unique source addresses sent packets to our honeypot multiple times, some of which demonstrated information had been learned from the first interaction. Most Modbus traffic used the "none" function code, and the remainder split between function codes 17 and 43, totaling 604 sessions. Protocol scanning using Modbus was visible in the form of incrementing slave ID numbers with each new request seen. We inferred that Nmap and the "modbus-discover.nse" script were used by attackers for this because of in the similarity to one of our recorded attacks in the sequencing of function code, the slave ID, the request values, and the response values.

We observed 20 new S7Comm sessions, 102 S7Comm connections, 13 COTP connection requests, and 19 S7 packets. S7Comm messages were only of types 1 and 7, with counts 6 and 13 respectively.

Configuring GridPot was difficult due its lack of updating for required dependencies in the open-source version we used, though there is now a commercial version that we assume is better. But once configured, it proved successful at collecting intelligence for threat analysis. It is apparent that attackers thoroughly explore Web-based vulnerabilities in ICS interfaces, as seen in the large amount of HTTP traffic captured. The Modbus scanning indicated that our simulated grid was realistic enough to encourage specialized-protocol reconnaissance.

FIGURE 2
GRIDPOT HTTP REQUESTS DURING LIVE TESTING



TESTING OF SCANNING FOR ICS HONEYPOTS

We also tested detection of honeypots with network scanning tools [26]. Most of our experiments used Shodan's Honeyscore, which rates a site from 0.0 (not a honeypot) to 1.0 (definitely a honeypot). The Honeyscore tool was sure that our final GridPot implementation was a honeypot when its history was taken into account, but was sure it was not a honeypot without the history. This is likely because we have reused the IP addresses for other honeypot projects [27, 28], so Shodan's automated scanning has found them many times, but the customization of the configuration of our GridPot implementation appeared to sufficiently disguise the honeypot when not knowing its history. In fact, the disguise may have been better than that of predecessor honeypots on the site because it received significantly more traffic, so not many visitors were inspecting records of its history.

To further explore these issues, we examined the records of 122,668 Internet sites in Shodan's database collected between April 22, 2016 and April 20, 2017 that had records for one of eight ports known to be specifically related to ICSs according to the Digital Bond ICS Enumeration plugin: Modbus, S7Comm, BACnet, CODESYS, Niagara Fox, OMRONFINS, ProconOS, and Ethernet/IP. For 114 of these, no Honeyscore was reported for reasons unexplained. Of the remainder, 1063 sites had a Honeyscore of 0.5 or larger for a rate of 0.87%. Shodan does not publish their criteria for Honeyscore, apparently to discourage honeypot developers from engineering easy countermeasures. However, some clues are obvious, so as a simple test we explored three heuristics:

- H1: A device which services the S7Comm protocol on TCP/102 and returns the terms "Technodrome", "Mouser", or "88111222" is a honeypot. These strings occur in the PLC Name, Plant ID, and Serial number fields in Conpot default implementations. These are implausible as values in a production S7Comm service.
- H2: A device providing the same ICS services as Conpot's default template, plus or minus one service, is a honeypot. Those services are HTTP (port 80), S7Comm (102), SNMP (161), Modbus (502), IPMI (623), and BACnet UDP (47808). It is unusual to see these services together otherwise.
- H3: A device providing industrial-control services from a public cloud location is running a honeypot. Cloud locations came from hosts identified as matching H1 in the Shodan data, names with the keywords "cloud" or "hosting", and names listed in a "Most Reliable Hosting Company Sites" page at Netcraft.com.

None of these heuristics applied to our GridPot site:

- H1 does not apply because we changed the default strings in our implementation. However, the earlier experiments with Conpot alone had the default strings [22].
- H2 does not apply because using Nmap against our GridPot honeypot, we found open ports 80, 102, 502, 6267, 8834, and 11211, so a scanner could match only 3 of the 6 target ports with 3 extra ports.
- H3 does not apply because our site did not offer any of those clues to directory services. It was only listed by our Internet Service Provider (AT&T) as being associated with our school.

To test the heuristics, we supplemented the 1063 high-Honeyscore sites with all sites matching either H1, H2, or H3 in the Shodan database, to get a test set of 8127 sites. We manually inspected other scanning data to estimate that 748 of these were honeypots (9.2%). This inspection used DNS node data, IP geolocations, and "who is" service data whenever there was a strong disagreement between Honeyscore and our heuristics. For some hosts we transmitted active probes using the Nmap OS detection method and observed the response. Among the sites found from manual inspection were 550 Conpot clones identified by duplicated ICS serial numbers, duplicated SSH fingerprints on controllers, and in one case, an FTP honeypot on the same site as the ICS honeypot.

Overall counts on the test set are shown in Table 3. We assumed a Honeyscore ≥ 0.5 as "ground truth" of a honeypot, and ignored the sites that were missing Honeyscore values. Then in the sample of 8127, H1 had a precision of 1.00, recall of 0.20, and an F-score of 0.33; H2 had a precision of 0.28, recall

of 0.92, and an F-score of 0.43; H3 had a precision of 0.03, recall of 0.14, and an F-score of 0.14. So H2 was the best and H3 was the worst. Note there is a tradeoff between precision and recall which will need to be best set based on information-security priorities.

TABLE 3
RESULTS OF TESTING A SAMPLE OF 122,068 SITES IN SHODAN'S DATABASE FOR HONEYPOT CLUES

H1	H2	H3	Honeyscore ≥ 0.5?	Count
YES	YES	YES	YES	81
YES	YES	NO	YES	16
YES	NO	YES	YES	3
YES	NO	NO	YES	41
NO	YES	YES	YES	25
NO	YES	NO	YES	566
NO	NO	YES	YES	234
NO	NO	NO	YES	97
YES	YES	YES	NO	1
YES	YES	NO	NO	2
YES	NO	YES	NO	0
YES	NO	NO	NO	2
NO	YES	YES	NO	1
NO	YES	NO	NO	1762
NO	NO	YES	NO	4175
NO	NO	NO	NO	115548
NO	YES	NO	Missing	2
NO	NO	YES	Missing	3
NO	NO	NO	Missing	109

As for identifying honeypots using Honeyscore alone, the precision for Honeyscore ≥ 0.5 was 748/1063 = 0.703. The recall was difficult to estimate, but it must be close to 1.0 because for H1 we found 7 sites with lower Honeyscores that we confirmed were honeypots; for H2 we found only 1 of 108 lower Honeyscores were honeypots; and for H3 we found no honeypots in a sample of 100 with lower Honeyscores. Thus the F-score is close to 0.70. Thus our heuristics can model some but not all of Honeyscore functionality.

Thus we conclude that Shodan can be fooled into thinking an ICS honeypot is a real ICS system with a few simple modifications to the default configuration that we can find by testing heuristics, provided Shodan is not examining site history. But on the other side of the coin, these results suggest that probes and attacks that do use history should be easy to fool with "fake honeypots" [29], real ICSs that have artifacts, services, and a history of honeypots; Shodan's outdated information will "scare away" attacks and help protect these sites. If attackers try to counter this by ignoring the historical data and just testing the current properties of the site with heuristics like H1, H2, and H3, some small modifications to the site like those of our GridPot implementation will cause a Shodan-like system to conclude a real honeypot is not a honeypot. The nice thing about this strategy is that the sort of attackers for which this will work best are the more sophisticated and intelligent attackers who gather thorough intelligence before focused attacks, so these sites can provide some sorely needed defensive techniques for attackers to which we are especially vulnerable.

CONCLUSION

Due to their real-time requirements and proprietary protocols, ICSs are more difficult to simulate with honeypots than other kinds of network nodes. The two ICS-honeypot frameworks we tested, Conpot and GridPot, did seem to be effective, however; we saw more traffic to them, and more varied traffic, than to our previous secure-shell and Web honeypots at the same addresses despite the rarity of ICS sites on the Internet. GridPot was definitely more successful at deception than Conpot because it saw a higher rate of traffic, mostly HTTP. Deception was effective for both honeypots because most traffic either did not recognize features of a honeypot or did not care. For the minority of attackers who either inspect sites or use scanning tools against them, our sites were probably easy to recognize as honeypots since they were not on a specialized subnetwork. However, this means that a different kind of deception of false honeypots could encourage these attackers to leave.

Future work will explore this as well as adding more simulated devices and services to ICS honeypots to keep attackers interested longer. Future work will also involve honeypots in the cloud. Our data is available for other researchers to use under restrictions.

ACKNOWLEDGEMENTS

This work was supported in part by the NPS Foundation and in part by the Naval Research Program at NPS. The views expressed are those of the authors and do not represent those of the U.S. Government.

REFERENCES

Blume, S. (Ed.) (2007). *Electric power system basics for the nonelectrical professional*. Wiley.

Bodenheim, R., Butts, J., Dunlap, S., & Mullins, B. (2014). Evaluation of the ability of the Shodan search engine to identify Internet-facing industrial control devices. *International Journal of Critical Infrastructure Protection*, 7(2), 114–123.

Bodungen, C., Singer, B., Shbeeb, A., Wilhoit, K., & Hilt, S. (2016). *Hacking exposed industrial control systems: ICS and SCADA security secrets and solutions*. McGraw-Hill.

Brown, J. (2019, September). *Identifying honeypots among Internet-connected industrial control devices* [M.S. thesis, U.S. Naval Postgraduate School]. Retrieved from http://Calhoun.nps.edu/handle/10945/63438.

Buza, D., Juhász, F., Miru, G., Félegyházi, M., & Holczer, T. (2014, February). CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot. *Proceedings of the International Workshop on Smart Grid Security* (pp. 181–192). Retrieved from http://www.crysys.hu/publications/files/ BuzaJMFH2014smartgridsec.pdf.

Chong, W., & Koh, C. (2018, September). *Learning cyberattack patterns with active honeypots* [M.S. thesis, U.S. Naval Postgraduate School]. Retrieved from http://faculty.nps.edu/ncrowe/oldstudents/ Chong_Koh_thesis.htm.

Digital Bond Inc. (2016). *Digital Bond's ICS enumeration tools*. Retrieved from http://github.com/digitalbond/ Redpoint.

Digital Bond, Inc. (2016). *SCADA honeynet*. Retrieved from http://digitalbond.com/tools/scada-honeynet.

Henderson, B., McKenna, S., & Rowe, N. (2018, December). Web honeypots for spies. *Proceedings of the International Conference on Computational Science and Computational Intelligence* (pp. 1-6). Las Vegas, NV, USA.

Hyun, D. (2018, March). *Extraction and analysis of IOCs using honeypots* [M.S. thesis, U.S. Naval Postgraduate School]. Retrieved from http://faculty.nps.edu/ncrowe/oldstudents/ Hyun_Dahae_ICS_thesis.htm.

Jicha, A., Patton, M., & Chen, H. (2016). SCADA Honeypots: An in-depth analysis of Conpot. *Proceedings of the 2016 IEEE Conf. on Intelligence and Security Information* (pp. 196–198). Tucson, AZ, US.

Joshi, R., & Sardana, A. (2011). *Honeypots: A new paradigm in information security*. CRC Press.
Kendrick, M., & Rucker, Z. (2019, June). *Energy-grid threat analysis using honeypots* [M.S. thesis, U.S. Naval Postgraduate School]. Retrieved from http://faculty.nps.edu/ncrowe/oldstudents/Kendrick_Rucker_thesis_full.htm.
Knapp, E., & Langill, J. (2015). *Industrial network security* (2nd ed.). Syngress.
Litchfield, S. (2017). *HoneyPhy: A physics-aware CPS honeypot framework* [M.S. thesis, Electrical. and Computer Engineering, Georgia Inst. of Technology]. Retrieved from http://smartech.gatech.edu/handle/1953/58329.
Lu, N., Taylor, Z., Chassin, D., Guttormson, R., & Studham, S. (2005, June). Parallel computing environments and methods for power distribution simulation. *Proceedings of the IEEE Power Engineering Society General Meeting* (pp. 215-219).
NCCIC. (2015). *CrashOverride malware*. NCCIC Alert ICS-ALERT-17-206-01. Retrieved from https://www.us-cert.gov/ics/alerts/ICS-ALERT-17-206-01.
Radvanovsky, B., & Brodsky, J. (2014, October). *Project SHINE (SHodan INtelligence Extraction) findings report*. Retrieved from http://www.slideshare.net/BobRadvanovsky/project-shine-findings-report-dated-1oct2014.
Radvanovsky, B. (2015, November). *Project RUGGEDTRAX SCADA/ICS analysis findings report*. Retrieved from http://www.slideshare.net/BobRadvanovsky/project-ruggedtrax-findings-report-28nov2015.
Redwood, W. (2015). *Cyber physical system vulnerability research* [Ph.D. Dissertation, Florida State University]. Retrieved from http://diginole.lib.fsu.edu/islandora/object/fsu%3A360429.
Rist, L. (2015, September). *Gas tank monitoring system honeypot*. Retrieved from http://www.honeynet.org/ node/1269.
Rowe, N. (2018). Honeypot deception tactics. In E. Al-Shaer, J. Wei, K. Hamlen, & C. Wang (Eds.), *Autonomous cyber deception: Reasoning, adaptive planning, and evaluation of HoneyThings* (pp. 35-45). Springer.
Rowe, N., & Rrushi, J. (2016). *Introduction to cyberdeception*. Springer.
Serbanescu, A., Obermeier, S., & Der-Yeuan, Y. (2015). *Threat analysis using a large-scale honeynet*. Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research.
Serbanescu, A., Obermeier, S., & Yu, D.-Y. (2015, July). A flexible architecture for industrial control system honeypots. *Proceedings of the 12th International Joint Conference on e-Business and Telecommunications*, 4, 16–26.
Soòky, P. (2015). *Extented functionality of honeypots* [B.S. thesis, Brno University of Technology]. Retrieved from http://dspace.vutbr.cz/bitstream/handle/11012/52363/16127.pdf.
Sridhar, S., Hahn, A., & Govindarasu, M. (2012, January). Cyber–physical system security for the electric power grid. *Proceedings of the IEEE*, 100(1), 210–224.
Theohary, C., & Harrington, A. (2015). *Cyber operations in DoD policy and plans: Issues for Congress*. CRS Report No. R43848. Retrieved from https://fas.org/sgp/crs/natsec/R43848.pdf.
Zou, C., & Cunningham, R. (2006). Honeypot-aware advanced botnet construction and maintenance. *Proceedings of the International Conference on Dependable Systems and Networks* (pp. 199–208).