NPS-CS-20-003

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

IDENTIFYING ANOMALOUS NETWORK FLOW ACTIVITY

USING CLOUD-BASED HONEYPOTS

by

Neil C. Rowe, Thuy D. Nguyen, and Jeffrey T. Dougherty

October 2020

Approved for public release; distribution unlimited
Prepared for: U.S. Fleet Forces Command

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From-To)*
14-10-2020	Technical Report	01-10-2019 – 14-10-2020

4. TITLE AND SUBTITLE
Identifying Anomalous Network Flow Activity Using Cloud-Based Honeypots

5a. CONTRACT NUMBER

5b. GRANT NUMBER
NPS-20-N033-B-NRP

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Neil C. Rowe, Thuy D. Nguyen, Jeffrey T. Dougherty

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AND ADDRESS(ES)
Naval Postgraduate School
1 University Circle
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER
NPS-CS-20-003

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Naval Criminal Investigative Service, 27130 Telegraph Road, Quantico, VA 22314

10. SPONSOR/MONITOR'S ACRONYM(S)
NCIS

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
Distribution Statement A: Unlimited distribution

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This work addressed efficient and effective implementation of honeypots (decoy devices) in cloud services. Honeypots are essential tools for detecting new attacks on computers and networks, and cloud services are distributed processing systems that can be used to provide great flexibility in software deployment. The particular subtype of honeypot we investigated was for industrial control systems (ICS) that manage electrical-power systems. Starting with two existing software frameworks called Conpot and GridPot, we added new obfuscation techniques, new simulated features of a fake electric grid, and new interfaces that looked like real power-plant controls to increase their deceptive power. These deceptions were effective in our first experiments with a standalone honeypot, as we were attacked twice by a sophisticated adversary as well as by many other less sophisticated attackers. In our second experiments, not yet complete, we deployed the same honeypot configurations at two cloud sites in the U.S. and in Asia. We saw clear differences between all three deployments, showing that context is very important in deceiving attackers and collecting useful data about their attacks. We were concerned deployment in the cloud could be detected by attackers and discourage their investigation, but we saw no evidence of that; apparently enough real electric-generation systems are deployed in the cloud today that they are not suspicious. We conclude that honeypots for industrial control systems using cloud services are a useful tool for information security.

15. SUBJECT TERMS
honeypots, cloud services, industrial control systems, power plant, electric grid, cyberattacks, deception, Conpot, GridPot, obfuscation, deception, adversaries

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Neil Rowe
Unclassified	Unclassified	Unclassified	None	26	**19b. TELEPHONE NUMBER** ***(include area code)*** 8316562462

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ann E. Rondeau
President

Rob Dell
Acting Provost

The report entitled "Identifying Anomalous Network Flow Activity Using Cloud Honeypots" was prepared for Naval Criminal Investigative Service and funded by the NPS Naval Research Program.

Further distribution of all or part of this report is subject to the Distribution Statement appearing on the front cover.

This report was prepared by:

Neil C. Rowe
Professor of Computer Science

Thuy D. Nguyen
Research Associate

Jeffrey T. Dougherty
Student

Reviewed by:

Gurminder Singh
Chairman of Computer Science

Released by:

Jeffrey D. Paduan
Dean of Research

ABSTRACT

This work addressed efficient and effective implementation of honeypots (decoy devices) in cloud services. Honeypots are essential tools for detecting new attacks on computers and networks, and cloud services are distributed processing systems that can be used to provide great flexibility in software deployment. The particular subtype of honeypot we investigated was for industrial control systems (ICS) that manage electrical-power systems. We started with two integrated software frameworks called Conpot and GridPot, and added new obfuscation techniques, new simulated features of a fake electric grid, and new interfaces that looked like real power-plant controls to increase their deceptive power. These deceptions were effective in our first experiments with a standalone honeypot, as we were attacked twice by a sophisticated adversary as well as by many other less sophisticated attackers. In our second experiments, not yet complete, we deployed the same honeypots at two cloud sites in the U.S. and in Asia. We saw clear differences between all three deployments, showing that context is very important in deceiving attackers and collecting useful data about their attacks. We were concerned deployment in the cloud could be detected by attackers and discourage their investigation, but we saw no evidence of that; apparently enough real electric-generation systems are deployed in the cloud today that they are not suspicious. We conclude that honeypots for industrial control systems using cloud services are a useful tool for information security.

Keywords: *honeypots, cloud services, industrial control systems, power plant, electric grid, cyberattacks, deception, Conpot, GridPot, obfuscation, deception, adversaries*

I. INTRODUCTION AND PREVIOUS WORK

In recent years, critical-infrastructure systems have become increasingly complex, interdependent, and reliant on computerized industrial control systems (ICS) (Stouffer et al., 2015). As our dependence on these control systems has grown, so has the frequency and complexity of cyberattacks conducted against them. During the initial development of ICS systems from the 1950s, little attention was paid to identifying and managing risk from potential security flaws because the systems were not connected to large networks. This has changed (Hawk and Kaushiva, 2014). However, ICS systems are often constructed with a planned lifetime of 20 to 30 years, meaning that legacy systems with well-known vulnerabilities will continue to be used for some time.

This work studied electrical-grid systems in particular. These have become major users of ICSs to control power generation, transmission, and distribution. While this has increased efficiency, it has also left power grids increasingly vulnerable to cyberattack. An example occurred in December 2015, when hackers believed to be linked to the Russian government used malware called CrashOverride to cut electrical power service to over 230,000 people in the Ukraine (SANS ICS, 2016). Due to the inherent difficulties in securing the legacy systems used in many power systems, researchers are examining novel ways to enhance their security. One such method is the deployment of honeypots (decoy devices).

We have deployed honeypots for over 15 years at the Naval Postgraduate School (Rowe & Rrushi, 2016), and have derived much useful information about cyberattacks from them. The technology for implementing them is increasingly mature, and a variety of deceptions can be used to make them more effective (Rowe, 2018). In the past two years, we have explored honeypots for industrial-control systems, such as those in power plants. The Navy has many such systems in operation on ships, and also uses them ashore on bases similarly to civilian utilities, but they are especially vulnerable to cyberattacks because of the difficulty of updating their software. Honeypots for these systems are also difficult to implement because they must simulate specialized processes like energy delivery, rather than well-known network communications methods as implemented by most honeypots.

Cloud services would seem a good way to implement honeypots since many cyberattack campaigns choose targets and methods randomly, and having many targets permits seeing a more complete spectrum of attacks (Atadika, Burke, & Rowe, 2019). Industries are increasingly using cloud services for supervisory control of manufacturing, power plants, and other industrial control systems. However, they may not be convincing to cyberattackers because cloud usage for these purposes is relatively recent, and detection of the use of cloud services is often easy through identification of the owner of its Internet site, so many cyberattackers would be suspicious of sites in the cloud claiming to be industrial control systems. At least in theory – so many cyberattacks are automated today that rarely do humans attackers inspect the characteristics of a site anymore. The only reliable way to assess the suspiciousness of a cloud honeypot is to do experiments

with real cyberattackers. Fortunately, cyberattackers need not be recruited, as putting any site up on the Internet usually attracts attackers within an hour, and at a higher rate for industrial control systems than for traditional computer systems (Hyun, 2018; Kendrick and Rucker, 2019).

II. EXPERIMENTAL SETUP

A. ELECTRICAL GRIDS AND PROTOCOLS

An electrical-power grid has four main functions: generation, transmission, distribution, and consumption (Blume, 2007). Power is generated at plants from coal, natural gas, nuclear, or solar collection. It is then transmitted from nearby electrical substations, at very high voltages until it reaches local distribution substations. There power is stepped down to lower voltages and delivered over local transmission lines to individual consumers in homes, office buildings, or factories. Typically, power is stepped down a final time just before delivery, minimizing losses in the distribution steps.

We simulated the power grid through the IEEE 13-Node Model, originally developed by the Test Feeder Working Group under the Analytic Methods for Power Systems technical committee of the IEEE's Power and Energy Society. This is an old, relatively simple, and well-characterized model of distribution from a central step-down substation to local customers. The advantages of using this model were that it was available in the open-source power-grid simulator GridLAB-D (GridLAB-D, 2020), required relatively few computational resources, and offered a variety of electrical devices to simulate in a fairly realistic environment. GridLAB-D was developed by Pacific Northwest Labs.

Modern electrical-grid ICSs mostly use either the Distributed Network Protocol 3 (DNP3), the IEC 60870-5 protocol, or the IEC 61850 protocol. The IEC 60870-5 standard assumes dedicated wired connections between devices, and thus does not route messages. The IEC 60870-5-104 companion standard, commonly called IEC 104, extends IEC 60870-5 to include routing by the Transfer Control Protocol/Internet Protocol (TCP/IP) protocol stack. When the project began our plan was to handle control messaging with IEC 61850. However, we later chose to use IEC 104 instead, because every device in a grid using IEC 61850 is described by a logical node (LN), which must be implemented for every device in the power grid. The IEC 104 protocol is much less prescriptive than IEC 61850, eliminating configuration of the data model as a possible source of error. Since our honeypot applications supported full IEC 104 messaging, we decided that it offered the best potential for simulating control messaging in our honeypot.

B. ICS AND SCADA HONEYPOTS

We wanted to create a honeypot that could imitate an ICS and attached Supervisory Control and Data Acquisition (SCADA) system, and was free of licensing issues. Few such honeypots exist for ICS systems. The honeypots we built in this research used two open-source software frameworks, Conpot (Antonioli et al., 2016) and GridPot (Redwood, 2016). Conpot is a low-interaction honeypot, meaning that it has limited simulation capabilities, but it does implement the first few steps in many common communications protocols to catch initial connection and subversion attempts. GridPot is

an extension of Conpot to provide more detailed interaction with attackers simulating electrical-power delivery systems, and uses GridLab-D.

GridPot work in 2019 installed both of these, but we encountered many problems and got only a few weeks of data; yet we did seem to fool attackers (Rowe et al., 2020). Overall, documentation and installation instructions were poor. Many required packages for GridPot had not been updated since 2016, and we had to find upgrades that would work with Conpot without major changes to GridPot. Once running, the Conpot portion did not impress attackers because of its limited simulation capabilities (Hyun, 2018). GridPot did more detailed simulation, but its implementation lacked parts for which we had to write our own code. We added additional camouflage in the names used in Conpot and GridPot since default names are easy clues for identifying honeypots.

Neither framework provided attackers with access to the physical devices of a power-grid ICS that we desired, so we added a Windows-based SCADA interface to the simulator, an open-source SCADA application that natively supports both IEC 104 and IEC 61850. With this application we could configure measurement and control points corresponding to the IEC 104 variables configured on Conpot, enabling receipt of updates and transmit commands to the virtual IEC 104 device running on Conpot. The name of this application is withheld to prevent fingerprinting. Although the project providing this application is open-source, we could not build from the available source code and had to rely on compiled binaries provided by the project maintainer. These binaries work only on Windows, forcing us to use a Windows virtual machine to host the SCADA system. The project maintainer created these binaries by compiling source with Visual Studio 6.0, an obsolete version that is difficult to find or run on modern systems.

III. EXPERIMENTS

Our experimentation had four phases. In Phase 1 we ran a standalone honeypot implementation without the SCADA interface and collected useful attack data. In Phase 2, we enabled the SCADA interface and the hosting Windows system was attacked twice by a sophisticated intruder or intruders, and many (but not all) logs were erased (Dougherty, 2020). One of the attacks witnessed an intruder trying to manufacture cryptocurrency. Clearly the interface or the Windows environment necessary for it were insufficiently hardened for Internet access In the third phase, we obtained a cloud account with DigitalOcean, and installed the same honeypot implementation, minus the SCADA interface, in a virtual machine (Bieker & Pilkington, 2020). We obtained baseline data. Then we obfuscated the honeypot and ran it at two sites, one in California and one in Asia, to test regional differences in attacks; the two instances will continue running until November.

A. EXPERIMENTAL DESIGN

Figure 1 shows our GridPot design. The remote user sends an IEC 104 message directed to Conpot's IEC 104 server, which calls a GridPot simulator method that passes the command to the associated GridPot information object. The GridPot simulator then uses the corresponding method to translate the IEC 104 command into an HTTP request and pass it to the GridPot HTTP server, from where it is passed to the GridLAB-D simulation which makes the change. The next time the GridPot simulator tells that GridPot object to poll its variables, the object passes that new value to the simulator, which notices the change. The simulator then informs the IEC 104 server of the available update, prompting it to send an IEC 104 packet telling the user of the new value.

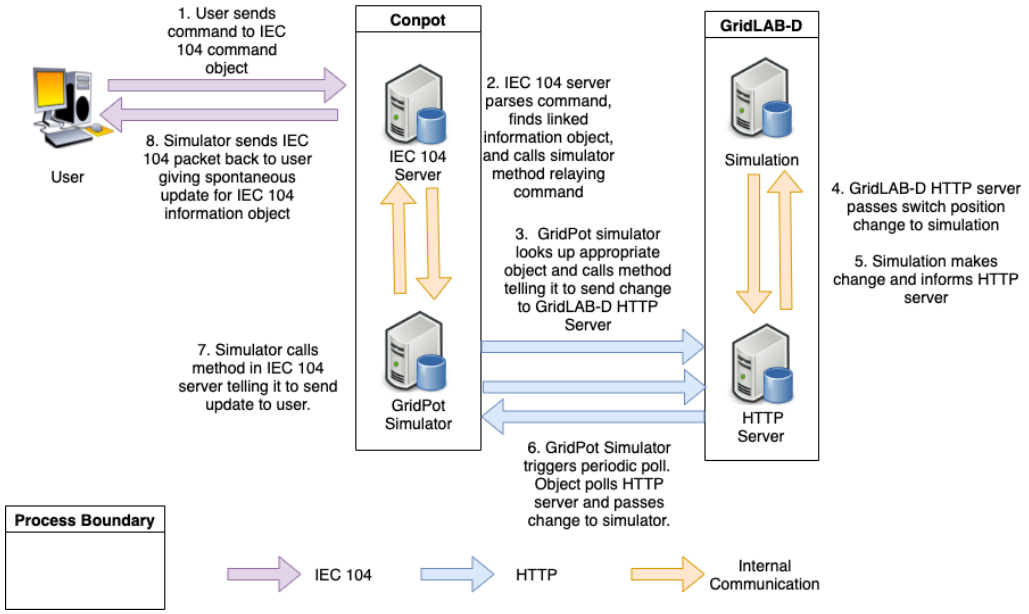

Figure 1: Phase 1 honeypot design.

Phase 2 added another deception in a SCADA station with graphical user interface to the honeypot (Figure 2). Users could connect to a virtual machine running our open-source SCADA application on a Windows operating system through Microsoft's Remote Desktop Protocol. That virtual machine communicated with a backend Linux virtual machine running the Phase 1 GridPot. The backend system provided realistic feedback to the SCADA application on the Windows virtual machine. The desktop transmits the user's keystrokes and mouse clicks to the interface software, which translates them into IEC 104 messages and sends them over a virtual network to the Phase 1 GridPot running on the Linux virtual machine. This means that if the remote user captures packets on the Windows host's inward-facing virtual network interface controller, they will see only IEC 104 traffic of the types expected between a real SCADA control station and its associated IEC 104 server. Once the traffic reaches GridPot's IEC 104 server, it is processed identically to IEC 104 traffic in Phase 1. When the IEC 104 server sends its packet containing updates to the information object, the packet goes back over the virtual network to the SCADA interface software, which processes it and presents the results to the remote user. The changes in the graphical view are then transmitted back to the remote user's screen by the Remote Desktop Protocol.

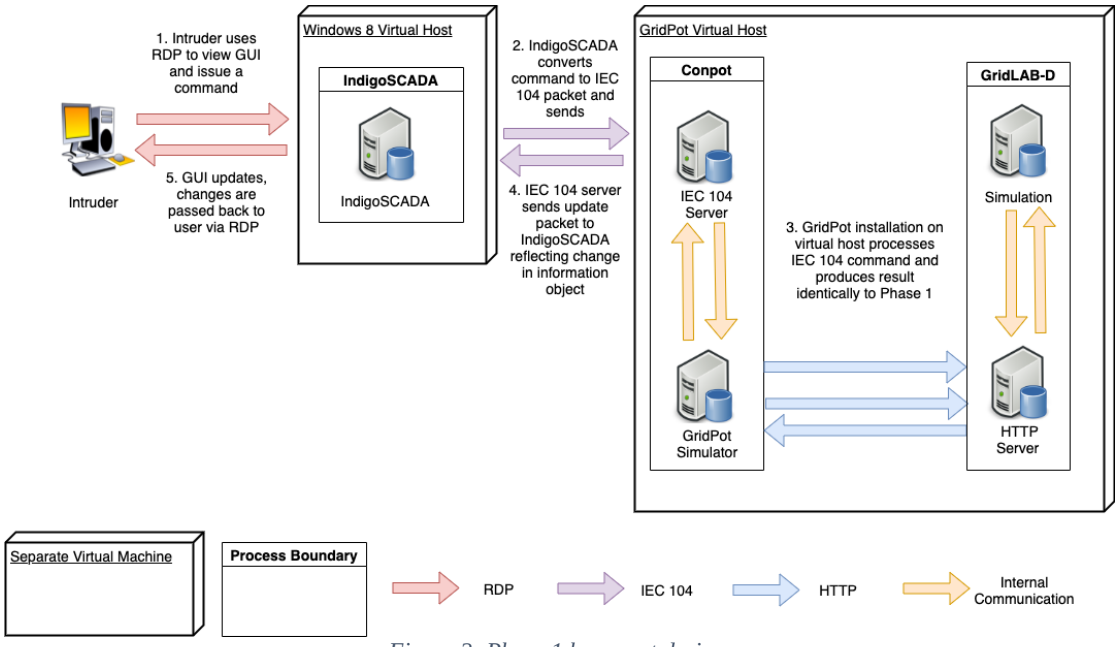

Figure 2: Phase 1 honeypot design.

B. MODIFICATIONS TO CONPOT IEC 104 SERVER

We added to Conpot the capability to optionally label IEC 104 information object addresses (IOAs) defined in the server's template as either "GridPot" or "Python" variables. "GridPot" variables map to a value in the GridLAB-D simulation. If the corresponding GridLAB-D value has changed when polled by its corresponding GridPot object, the change is automatically routed through the GridPot simulator to the IEC 104 server where the data is updated. Similarly, any commands sent to an IEC 104 command object are automatically passed to the GridPot simulator and on to GridLAB-D. "Python" variables, by contrast, map to variables in a GridPot object, not to variables in the GridLAB-D simulation. These variables are not polled by the simulator because they can only change from user commands. For example, a GridPot switch might have an "enable" command that must be set to True before the position of the switch can be changed.

Second, we modified the Conpot IEC 104 server to accept certain types of incoming commands not handled by the original GridPot, and to pass their values to the GridPot simulator for action if their IOAs mentioned GridPot or Python variables. We modified existing code for handling IEC 104 type-45 (set single-point Boolean), type-46 (set double-point Boolean), type-49 (set scaled value), and type-100 (general interrogation) commands to let them interface without requiring a SCADA application, and added the capability to handle type-63 (set floating-point value with time tag) commands as well. We chose to handle type-63 commands rather than modify the existing code for type-50 (set floating point) commands because our SCADA application sent type-63 commands when the user issued commands with floating-point numbers.

Third, in Phase 2, we enabled the Conpot IEC 104 server to send data updates to a remote user without prompting. When a remote socket interrogates an IEC 104 server, its session is now marked as "active" until the remote socket disconnects or times out. While a session is active, the GridPot simulator will continue to poll the objects it controls (such as switches) for changes to their variables, and will pass these changes to the IEC 104 server as with Phase 1. If the IEC 104 server receives updates, it generates packets with the new values, labels them as spontaneous updates, and sends them to the remote socket. When no active sessions remain, the IEC 104 server tells the GridPot simulator to stop polling its virtual devices.

Fourth, to aid handling of updates, we changed the IEC 104 server to provide a pointer to its “update handler” to the GridPot simulator using the Conpot data bus, and accept a pointer to the simulator’s update handler in return. When the IEC 104 server has completed startup, it pushes a pointer to its update handler onto the Conpot data bus and reads the pointer to the simulator’s update handler previously pushed during the simulator's startup. While in theory this might cause a crash if the server tries to retrieve a nonexistent pointer, testing showed this was not a problem. Once the server has a pointer to the simulator, it uses that pointer to call a simulator method that prompts the simulator to read the pointer the server just pushed to the data bus. Since both server and simulator have pointers to each other, they can communicate by calling each other's methods.

Finally, we changed the IEC 104 server to communicate with our SCADA application. Although not required for Phase 1, we implemented it to test it with the other IEC 104 server changes. IEC 104 specifies that to change a variable, the remote station must send an "activation" command, to which the server responds with an "activation confirmation" message. The user then sends a command which causes the server to change the variable and send back both an I-frame with the updated value and an "activation termination" message for the variable. By default, the Conpot IEC 104 server receives the command, changes the variable, sends back the updated value, then sends the activation termination message. However, our SCADA application expects the updated value to come after the activation termination message and will not listen for it until the termination message has been received. Since we could not determine which behavior was correct, we added a Boolean variable in the simulator allowing the GridPot user to choose at startup whether to use the default Conpot sequencing or interface-compatible sequencing.

C. MODIFICATIONS TO GRIDPOT SIMULATOR

Our changes to the GL_obj class in GridPot substantially expanded its functionality. We implemented methods for object variables not associated with the GridLAB-D simulation, similar to the "Python" variables implemented in the Conpot IEC 104 server. We also improved handling of variables by storing them as instances of a new class that stored mappings between IEC 104 IOA numbers and GridLAB-D or Python variables, and also contained methods for converting between the strings GridLAB-D uses to store data and IEC 104 values, including complex numbers. These changes substantially reduced the amount of device-specific code required for each subclass of GL_obj, since the only code needed for each device was what differentiated its behavior from others.

For example, when a switch is told to close, the HTTP command must read "status=CLOSED2" instead of "status=CLOSED" for the variable to change in the GridLAB-D simulation. Since this only applies to switches, sending "CLOSED2" instead of "CLOSED" is handled in the GL_switch class. This let us add basic power meters as GridPot devices by instantiating the base GL_obj class without any device-specific code.

A major limitation in the original GridPot simulator was the lack of integration with power-grid control protocols. Our changes to the GridPot simulator and our additions to the IEC 104 protocol server enabled the two to interact. The GridPot simulator periodically polls its devices for updated values and orders the IEC 104 server to send updates as needed by calling methods within the server, with mechanisms in place to prevent multiple updates for a single IOA to be queued simultaneously. Incoming IEC 104 requests for information or commands are passed to the simulator by the server calling its methods, with requests or commands being passed on by the simulator to individual objects and results being sent back. Since our changes to the GridPot simulator made it slower, we changed its architecture to try to improve its efficiency. We had the simulator poll its GridPot objects only when the IEC 104 server confirms an active connection. We also allowed the simulator to use multiple processors.

D. PHASE 2 IMPLEMENTATION

Creating a realistic experience for an intruder to see involved configuring a SCADA application and designing a realistic user interface within that application. Our SCADA application ran on a Windows virtual machine. We first created a virtual IEC 104 device and added it to the application's list of supported devices, then created IOAs within that virtual device for each information and command object from GridPot we wished to make visible at the SCADA interface. The SCADA application stores IOAs as integers, so we wrote code to convert IOAs from GridPot's format to integers.

We did not simulate the full IEEE 13-Node Model because testing showed it slowed GridPot's operation unacceptably. Instead, the Phase 1 configuration used a voltage regulator, a switch, and seven power meters for residential customers. We created two main displays within our SCADA application. The first display reported the state of all simulated devices in the system; the second showed the status of one specific simulated device, and allowed the intruder to send commands to it. The SCADA host's configuration enabled access to the system by the remote desktop Guest account, which by default lacks a password and is a popular target for attack (Boddy, Jones, and Stockley, 2019). Providing an account without a password did require several changes to the local machine's security policy and to Windows firewall rules.

E. PHASE 1 DEPLOYMENT

Phase 1 was deployed as a standalone honeypot. We used two environments for this experiment, a development environment and a live environment. The host for the live honeypot was a Dell XPS 8910 desktop computer with an Intel 3.40 GHz quad-core CPU, 16 gigabytes memory, and a 925 gigabyte hard disk running the Windows 10 Home operating system. The live honeypot host was connected to the Internet by an ISP outside

our School's firewall. However, the IP address is part of a block belonging to our institution, something that may have discouraged more sophisticated attackers.

We ran the honeypot as a virtual machine under VirtualBox 5.2.22 with two virtual CPU cores, 10,240 megabytes memory, and 50 gigabytes of storage in a virtual disk. The live-honeypot machine had a bridged adapter, and used the same hardware (MAC) address as that of the host machine's network-interface controller through which the virtual machine was bridged. The controller's IP address, subnet, and gateway were manually configured to match the host machine's configuration. This allowed Conpot to record the remote IP addresses connecting with its open ports.

1. Data Collection

Data was collected for Phase 1 between January 28th and May 14th and between June 17th and August 2nd of 2020. The Phase 1 GridPot, configured as described above, connected to public-facing external network and was left for Internet users to discover. Other than conducting a SHODAN scan and HoneyScore evaluation of the honeypot's IP address early in the first run, we did not publicize the honeypot's address or try to draw attention to it in any way.

We collected data in two ways. The first was with Wireshark, a network-protocol analyzer that captured incoming packets on GridPot's connection to the Internet. To keep the packet capture (PCAP) files to a manageable size, we only recorded packets for the two services provided by GridPot, HTTP (TCP port 80) and IEC 104 (TCP port 2404). We lost four days of Wireshark data when it was mistakenly set to capture on the wrong network interface, and ten days due to a power outage. We wrote a Python program using to generate IEC 104 packets from Conpot log data for periods in which we lacked PCAP data.

Our second data source was Conpot logs, specifically its text logger. These logs are continuously written to disk, so they gave information for periods in which we lost PCAP data. We had more trouble with Conpot data collection than with Wireshark collection; in some crashes, the logger would stop and suddenly output many events with timestamps corresponding to when it was stopped.

2. Data Analysis

Most analysis used the Python 3.6 Scapy packet handling library, and the Pandas data analysis library. To consolidate the data , we wrote a Python program to parse PCAP files and Conpot log files for incoming HTTP and IEC 104 traffic, and record the timestamp, remote IP address, target TCP port, and HTTP method or IEC 104 frame type. For IEC 104 packets, the program determined whether they were simply directed to port 2404 (called "traffic") or if they contained valid frames. A valid frame was defined as a TCP segment directed to port 2404 with a payload starting with a byte value of 0x68 and a second byte correctly specifying the length of the message.

To investigate which IP addresses contacted the honeypot more than once, we wrote a Python program to count the number of sessions started by each IP address. We defined a session as all packets exchanged by a socket pair on a given date

.

F. PHASE 2 DEPLOYMENT

Our Phase 1 GridPot virtual machine was deployed on VirtualBox 5.2.22 with two virtual processors, 4096 megabytes of memory, and 50 gigabytes of storage. One virtual network-interface controller attached to the machine, which was in turn attached to a VirtualBox internal network equipped with a DHCP server to allow communication with the Windows virtual machine hosting the SCADA application. The latter had four virtual processors, 6052 megabytes of memory, and 40 gigabytes of storage. It was equipped with two virtual network interface controllers: one attached to the same VirtualBox internal network as the Phase 1 GridPot virtual machine that provided communications with the GridPot virtual machine, and one a bridged adapter with its MAC address the same as the MAC address of the host machine network-interface controller that provided Internet connectivity. Since the Windows host could not respond to HTTP traffic, this controller forwarded incoming traffic addressed to TCP port 80 to the internal network-facing controller, and forwarded traffic from TCP port 80 on that controller back to the bridged adapter. This allowed the GridPot HTTP server to respond to these requests.

Data was collected for Phase 2 from May 26 to June 7 and from June 17 to June 29 in 2020. We analyzed three sources of data for Phase 1: PCAP packet captures, Conpot logs, and Windows event logs. Packet captures and Conpot logs were collected from the outset, while the Windows event logs were added later. We deployed Wireshark on the Windows virtual machine to collect packets from the Internet-facing bridged adapter. As with Phase 1, we collected packets to or from TCP ports 80 (HTTP) and 3389 (RDP). We did not collect data from port 2404 during this phase, since the Internet-facing virtual machine did not have that port open. Wireshark saved its PCAP packet data every three hours. After our data collection issues on our first deployment of Phase 1, we also ran Wireshark on the GridPot virtual machine. This secondary source of data was less useful since all traffic to the GridPot machine passed through the Windows virtual machine, and thus lost the remote IP address. Since the Microsoft Remote Desktop Protocol is an encrypted protocol, we used the security tool Mimikatz to extract the Windows host's encryption keys, allowing Wireshark to capture and store remote desktop traffic in the clear. Conpot's text logger on the GridPot host was secondary data source, although it too could not record the remote IP originating the traffic.

We gathered the following logs for each live run:

- Windows System
- Windows Security
- Windows Firewall
- Microsoft-Windows-Terminal Services-LocalSessionManager

- Microsoft-Windows-Terminal Services- RemoteConnectionManager
- Microsoft-Windows-User Profile Service- Operational
- Microsoft-Windows-WindowsDefender-Operational
- Microsoft-Windows-User Profile Service-Operational

G. PHASES 3 AND 4

For Phases 3 and 4 we used the cloud provider DigitalOcean. We did not use our school-endorsed provider, Amazon Web Services, because it did not allow for installation of honeypots for security reasons, and it was more expensive. We installed the same honeypot setup with Conpot and GridPot used in Phase 1 without the SCADA interface because of the problems we had with it (Bieker & Pilkington, 2020). It was installed in a virtual machine to enable easier portability. We first obtained baseline data, then additionally obfuscated the honeypot and ran it at two sites, one in California and one in Asia, to test regional differences in attacks. The testing is currently incomplete and will continue running until November since the students responsible for it are graduating in December.

IV. RESULTS

A. PHASE 1

The Shodan Honeyscore evaluation of our live Phase 1 GridPot said it found a 0% probability of Phase 1 GridPot being a honeypot. A score of 1.0 (or 100% probability of being a honeypot) was received by the previous version of GridPot (Rowe et al, 2020), indicating that our version is substantially better at evading a commonly used automated honeypot detection tool. We also used the Shodan application programming interface to request information on our deployed GridPot as well as on the host on which the older version of GridPot was deployed. Shodan did not identify it as a honeypot regardless of whether or not the IP's history was considered. We conclude that our version of GridPot is better at evading this Shodan-based detection mechanism.

Phase 1's first run lasted 111 days and recorded traffic from 2,276 unique IP addresses from 121 countries; its second run lasted 46 days and recorded traffic from 588 unique IP addresses from 67 countries.

B. PHASE 2

The first Phase 2 run began on 5/26/20 and had an indicator of compromise on 6/7/2020 when we discovered a process called XMRig Miner using 100% of our CPU. The screenshot of the Guest desktop is shown in Figure 18. Further investigation established that packet capture had been ended, leaving only a few entries from 5/26 that recorded some HTTP traffic and one TCP SYN scan of the remote desktop port; and that a third-party software called Crack_by_NERO had been installed and used to password-lock the Guest desktop. We also found a new folder on the Guest desktop called "xmrig-5.1.1," containing the XMRig Miner executable, which appeared to be a crypto-currency miner, the desktop locker executable, and a then-unidentified executable called "SpoolerComp.exe." The password locker achieved persistence by adding itself to the Windows Registry at HKEY_USERS/<Guest User ID>/Software/Microsoft/Windows/CurrentVersion/Run, which specifies programs that run automatically on login.

Our initial analysis of the first run concluded that the malware was installed in the Desktop folder because a Guest user had write access to it. We therefore disabled Guest's write access to Desktop and other folders in their home directory, as well as setting our SCADA software to start at login to make it more apparent that the system was an ICS. Phase 1 second run started on 6/17/2020. When we checked the honeypot on 6/29, we observed that packet capture was no longer running, the Windows preinstalled Administrator account was visible despite it being set to invisible at startup, a new Administrator level account called "Admin" had been created. and a new executable called "opera_portable_56.0.3051.36.exe" had been downloaded to the C:\Program Files directory, as well as several executables labeled as Opera installers.

We conclude that there were at least two penetrations of the system on 6/25. The first exploit used the Admin account and installed the infected opera.exe file, but was

apparently thwarted by Windows Defender's anti-malware rules. This attack probably originated from the IP address 88.209.137.9. The second attack installed a different file in the same C:\ProgramData directory, which was again stopped by Windows Defender. Instead of giving up like the first time, someone or something deactivated Windows Defender and proceeded with the exploit. Although log erasure means we cannot say for certain whether these two attacks were related, the use of the same type of malware in the same unauthorized C:\ProgramData directory suggests it.

Because of the lack of data from Phase 2 and the possibility it was compromised, we did not include its data in the comparison that comprises the rest of this chapter.

C. COMPARATIVE ANALYSIS OF PHASE 1 AND 3

Phases 3 was done in the DigitalOcean cloud service. Phase 3 tested a copy of the Phase 1 setup without modification. The Phase 4 honeypot used an additionally obfuscated version of the Phase 3 honeypot, which was installed on two sites, locally and in Asia. Phase 4 is not complete, so we confine our analysis here to Phase 3 and compare it to all the data from both runs of Phases 1.

Preliminary results showed significantly more traffic on the cloud sites in Phase 3 than on the standalone non-cloud sites, so it is encouraging for intelligence gathering that attackers were not discouraged by the cloud implementation. We also saw significant differences between the U.S. and Asia traffic, which suggest some security advantages of international cloud services. These results confirm that cloud honeypots are feasible and effective for collecting intelligence on new cyberattacks on industrial control systems.

Table 1 shows the testing periods, the number of unique IP addresses that contacted the honeypots, and the number of countries with which those addressed were registered.

Table 1: Comparison of unique IP addresses and unique countries.

	Testing Period	**Unique IP addresses**	**Countries**
Phase 1	Run 1: 111 days (1/28 – 5/21/2020) Run 2: 46 days (7/5 – 8/2/2020)	2864	188
Phase 3	17 days (8/17/20 – 9/3/20)	542	67

Figure 3 shows the proportions of single HTTP sessions (Remote IP addresses that had only connected to the honeypot for a single HTTP session, in red), multiple HTTP sessions without ICS protocols (in yellow), and remote IP addresses that had connected to the honeypot for more than one HTTP session (in green). Here “Local” means Phase 1 (combined Run 1 and Run 2) and “Cloud-1” means Phase 3. Figure 1 shows the Phase 1 GridPot had more HTTP traffic than the Phase 3 GridPot, for both single and multiple sessions. It also shows that Phase 3 GridPot had more ICS traffic (shown as “Other”)

than Phase 1. The cosine similarity between the distributions for Phase 1and Phase 3 was 0.998.

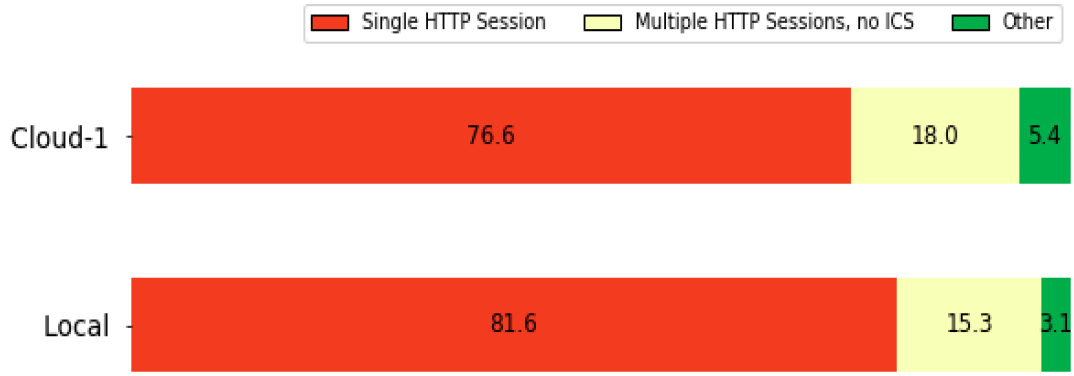

Figure 3: Percentage of IP addresses by number and type of session.

Figure 4 breaks down the "Other" in Figure 3 into Single ICS Sessions (remote IP addresses that had only connected to the honeypot for a single ICS session), Multiple ICS Sessions without HTTP (remote IP addresses that had connected to the honeypot for more than one ICS session), and Both HTTP and ICS Sessions (remote IP addresses that had connected to the honeypot for both HTTP and ICS sessions). Phase 1 handled more ICS-only sessions than Phase 3. The number of IP addresses that connected to the HTTP and ICS servers in Phase 3 was significantly higher than the number of IP addresses that connected to them in Phase 1. The cosine similarity between the Phase 1 and Phase 3 distributions was 0.402, so they were significantly different.

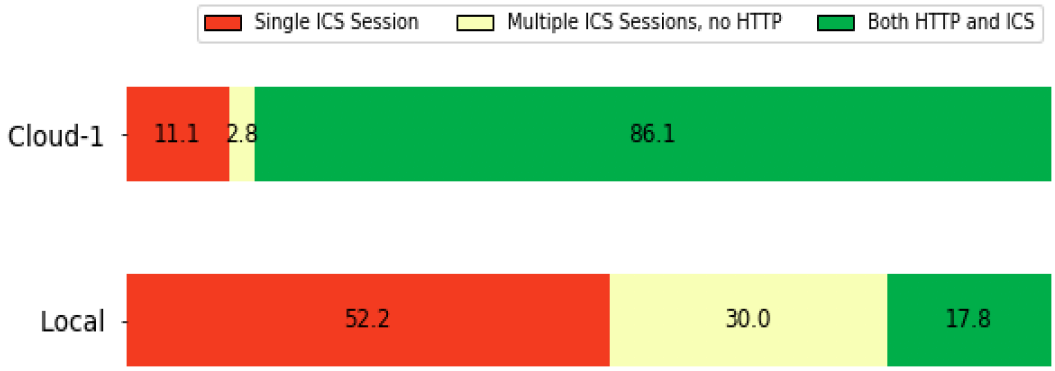

Figure 4: Percentages of non-HTML traffic.

Table 2 and Table 3 compare the number of requests made per session between IP addresses that only conducted a single session and IP addresses that conducted multiple sessions for HTTP and ICS, respectively. The table also shows the means and standard deviations, which were applied to a two-tailed, two-sample T-test to determine whether the means significantly differ. A p-value less than or equal to 0.05 is usually considered significant, indicating that there is a 5% probability that the variation between the two means could be random. The p-values shown in Tables 2 and 3 indicate that there was a

statistically significant difference in the number of requests per session, with IP addresses contacting the honeypot multiple times making more requests per session than IP addresses that only connected to the honeypot once. (Each p-value on Tables 2 and 3 refers to the comparison between a set of single-session IP addresses and the corresponding set of multiple-session IP addresses; thus we only show one p-value per pair.) This suggests that most single-session IP addresses conducted relatively simple port scans, while IP addresses that connected multiple times are more likely to perform a detailed investigation of the open port. The cosine similarity values of the comparison of the data between phases were 1.000 for single-session similarity and 0.689 for multiple-session similarity for HTTP (Table 2), and 0.990 and 0.999 respectively for IEC 104 (Table 3), so there were differences in the type of traffic.

Table 2: Requests per session for HTTP.

		Mean	Standard Deviation	Number of Addresses	T-Statistic	P-Value
Phase 1	Single-Session IP Addresses	1.97	33.67	2235	-9.47	.000
	Multiple-Session IP Addresses	38.35	169.0	573	--	--
Phase 3	Single-Session IP Addresses	1.37	2.96	399	-3.19	.001
	Multiple-Session IP Addresses	44.9	271.7	139	--	--

Table 3: Requests per session for ICS protocol IEC 104.

		Mean	Standard Deviation	Number of Addresses	T-Statistic	P-Value
Phase 1	Single-Session IP Addresses	1.70	0.93	40	-6.51	.000
	Multiple-Session IP Addresses	15.42	13.15	38	--	--
Phase 3	Single-Session IP Addresses	2.63	1.27	16.4	-3.50	.000
	Multiple-Session IP Addresses	7.68	5.50	19.4	--	--

Figure 4 and Figure 5 show the inter-session times by protocol for phases 2 and 3. Rapid sequences of session establishment from a single IP address might indicate scanning activity. Most HTTP sessions occurred within 10 seconds or less on both honeypots. Phase 3 saw a large number of HTTP sessions occurring between 10 and 100 seconds, and a spike between 100,000 and 1,000,000 seconds; Phase 1 saw a spike between 10,000 and 100,000 seconds. For ICS sessions, most sessions occurred between 100,000 and 1,000,00 seconds on Phase 3 whereas most sessions occurred within 10 seconds or less on Phase 1. Phase 3 also saw a large number of ICS sessions occurring between 1,000 and 10,000 seconds.

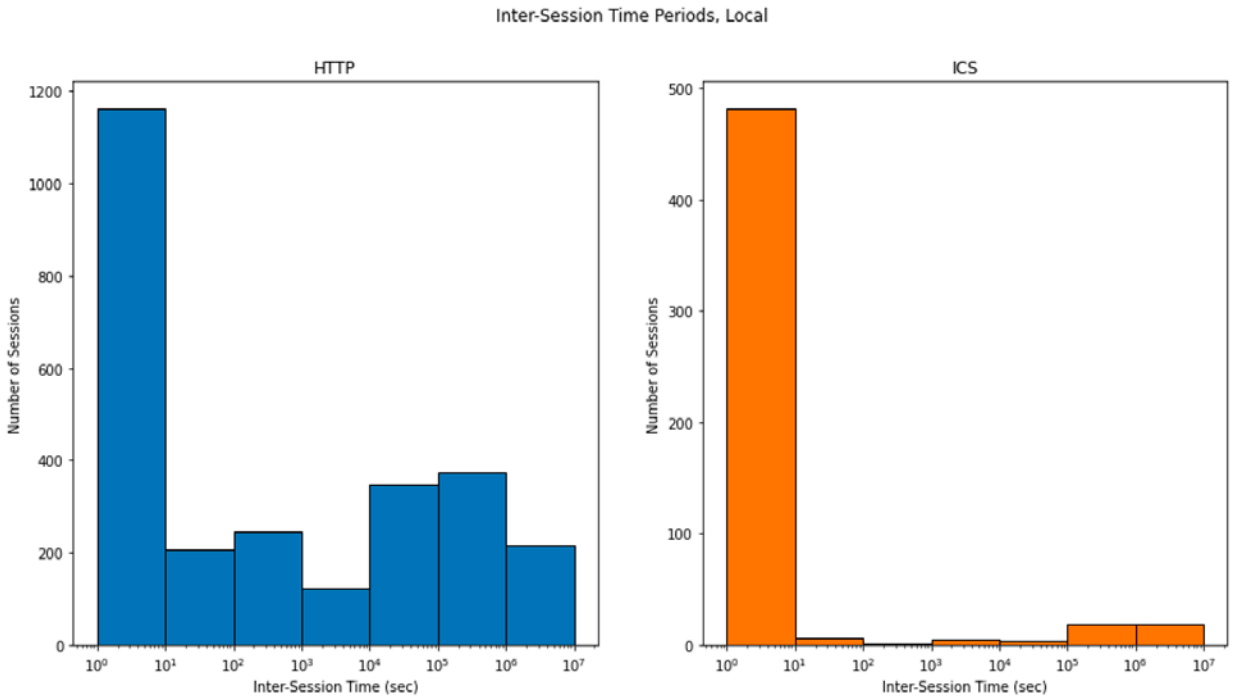

Figure 5: Inter-session times for Phase 1.

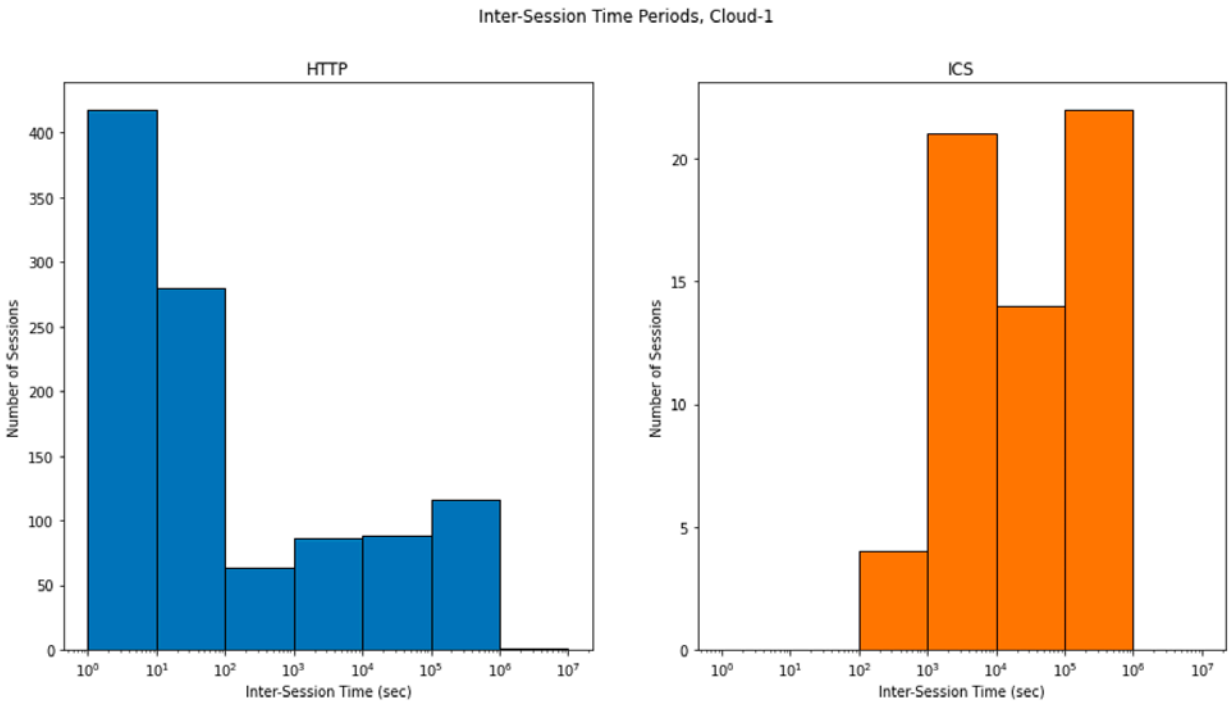

Figure 6: Inter-session times for Phase 3.

Table 4 shows the distribution of the most common countries of alleged source that visited Phase 1 and Phase 3 honeypots via HTTP. For both implementations, ten countries contributed 2% or more of the total number of IP addresses, although the percentages of their contributions were different. The cosine similarity of these country distributions was 0.969, suggesting that similar attackers occur on non-cloud and cloud sites.

Table 4: Major source countries seen in the honeypots.

Country	Percentage in Phase 1	Percent in Phase 3
United States	17.8	25.5
Brazil	8.3	7.2
China	8.0	9.2
Russia	6.0	5.4
India	3.3	4.2
Germany	2.9	7.0
Netherlands	2.3	4.8
Other	51.4	36.7

Figure 7 and Figure 8 show the logarithm of the number of incoming requests per day for both honeypots. IEC 104 traffic means all traffic using IEC 104's well-known TCP port (2404), and IEC 104 messages means only traffic with correctly formatted IEC 104 data. Phase 1 ran longer than Phase 3 and HTTP was observed on most days. Phase 3 saw continuous HTTP and IEC 104 probing. However, both phases saw gaps in IEC 104 traffic with valid IEC 104 messages because of its low overall frequency. For both phases, most traffic directed to the IEC 104 TCP port contained invalid messages, which most likely means basic port scanning.

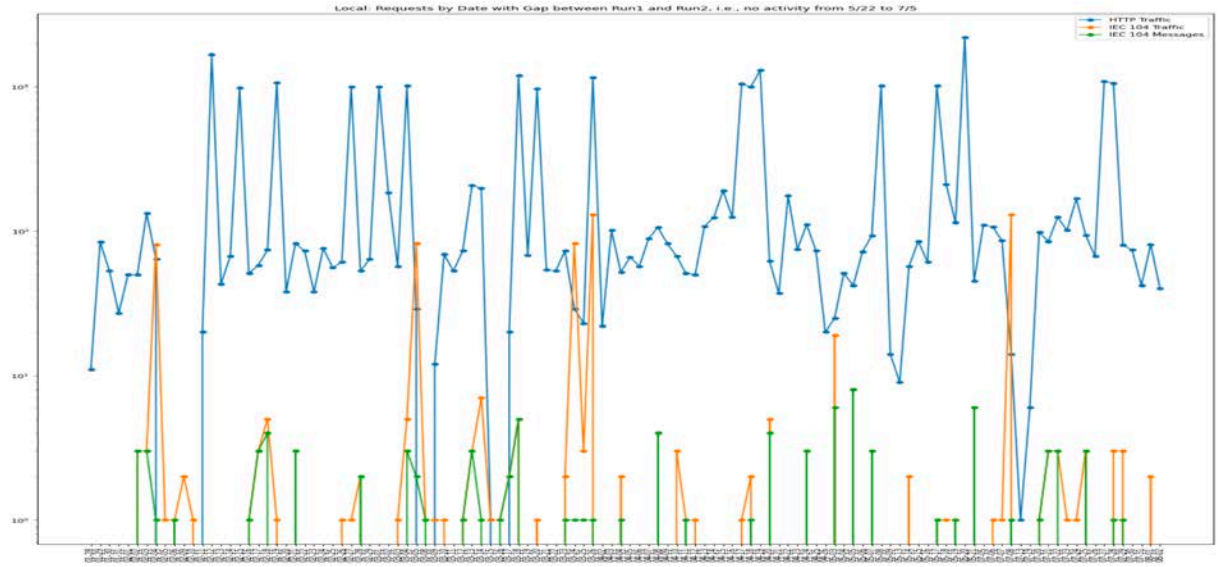

Figure 7: Requests per day for Phase 1.

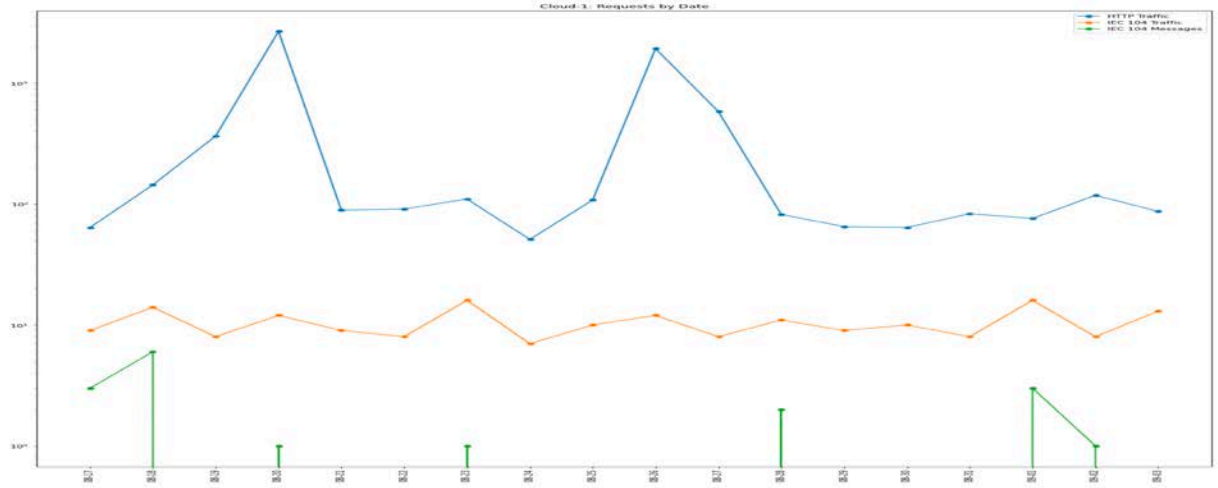

Figure 8: Requests per day for Phase 3.

Table 5 shows the counts of protocol requests per day. The cosine similarity for HTTP, overall IEC 104 traffic, and IEC 104 traffic with valid IEC 104 messages were 0.997, 0.759, and 0.992, respectively.

Table 5: Incoming protocols requests per day.

Phase	Protocol	Mean	Standard Deviation
Phase 1	HTTP	239.7	413.9
Phase 3	HTTP	376.9	704.0
Phase 1	All IEC 104 traffic	10.5	27.7
Phase 3	All IEC 104 traffic	10.4	2.7
Phase 1	Valid IEC 104 traffic	2.3	1.7
Phase 3	Valid IEC 104 traffic	2.4	1.7

Table 6 summarizes the different HTTP methods that were attempted on the honeypots. The PROPFIND method is an HTTP extension for the Web Distributed Authoring and Versioning (WebDAV) application. The cosine similarity of these distributions was 0.779 so there were important differences.

Table 6: HTTP methods seen.

	POST	GET	CONNECT	HEAD	OPTIONS	PROPFIND	None
Phase 1	13393	11651	76	66	31	3	1148
Phase 3	992	5561	13	7	4	0	208

Table 7 summarizes the different IEC 104 packets that were observed on the honeypots.

Table 7: IEC 104 packets seen.

	U Frame	I Frame	Malformed
Phase 1	77	19	557
Phase 3	13	4	171

Phase 1 had 91% malformed messages, 6.9% U-frame messages, and 2.1% I-frame message; Phase 3 had 85.3% malformed messages, 11.8% U-frame messages, and 2.9% I-frame messages. Hence, Phase 3 seemed to elicit more serious attempts to connect.

V. CONCLUSIONS

More experiments with cloud honeypots need to be conducted. Since we observed regional differences, hundreds of honeypots could be set up all over the world and their outputs compared. Cloud services make this much easier than setting up sites individually, and we saw no significant disadvantages to them, so they clearly should be the preferred implementation.

Having more detailed simulation capabilities for industrial control systems definitely increased cyberattacker interest and the amount of activity on our honeypots, so simulation features should be enhanced. However, a better SCADA interface is important because we were twice comprised. We plan to run the SCADA interface on a Linux machine using a Windows API translator like Wine to avoid the vulnerabilities of the Microsoft environment.

As for data analysis, more can be done to find new cyberattacks than we could do because there are a wide variety of methods we have not had time to try. Being able to compare attacks across many different sites should make it easier to see past the randomization of many attacks. After termination of this grant in October, three students will be continuing this work, including two students who graduate in December (Bieker & Pilkington, 2020), and one who graduates next September (Washofsky, 2021).

LIST OF REFERENCES

Antonioli, D., Agrawal, A., & Tippenhauer, N. (2016). Towards high-interaction virtual ICS honeypots-in-a-box. Proc. 2nd ACM Workshop on Cyber-Physical Systems Security and Privacy, pp. 13–22.

Atadika, M., Burke, K., & Rowe, N. (2019, August). Critical risk management practices to mitigate cloud migration misconfigurations. Proc. Intl. Conf. on Computational Science and Computational Intelligence, Las Vegas, NV, United States.

Bieker, M. & Pilkington, D. (2020, expected December). Deploying an ICS honeypot in a cloud computing environment and comparatively analyzing results against physical network deployment. [Master's thesis, Naval Postgraduate School].

Blume, S. (2007). System overview, terminology, and basic concepts, in *Electrical Power System Basics for the Nonelectrical Professional*, Hoboken: John Wiley & Sons, pp. 1–12.

Boddy, M., Jones, B., and Stockley, M. (2019). RDP exposed - the threat that's already at your door. Sophos, Inc, Sophos White Paper, 2019.

Dougherty, J. (2020, September). Evasion of honeypot detection mechanisms through improved interactivity of ICS-based systems. [Master's thesis, Naval Postgraduate School].

Electricity Information Sharing and Analysis Center (2016, March). Analysis of the cyber attack on the Ukrainian power grid: defense use case. SANS ICS, Washington, DC.

GridLAB-D Project, (2020, June 26). GridLAB-D Github page. [Online]. Available: https://github.com/gridlab-d/gridlab-d. [Accessed: 09-Aug-2020]

Hawk, C., and Kaushiva, A. (2014, October). Cybersecurity and the smarter grid. *The Electricity Journal*, vol. 27, no. 8, pp. 84–95.

Hyun, D. (2018, March). Extraction and analysis of IOCs using honeypots. [Master's thesis, Naval Postgraduate School]. NPS Archive: Calhoun. https://calhoun.nps.edu/handle/10945/58316

Redwood, W. (2016). Cyber physical system vulnerability research. [PhD Dissertation, Florida State University] FSU Digital Repository: DigiNole. https://diginole.lib.fsu.edu/islandora/object/fsu%3A360429

Rowe, N. (2019). Honeypot deception tactics. In Al-Shaer, E., Wei, J., Hamlen, K., & Wang, C. (Eds.), *Autonomous cyber deception: Reasoning, adaptive planning, and evaluation of Honey Things*, Springer, pp. 35-45.

Rowe, N., Nguyen, T., Kendrick, M., Rucker, Z., Hyun, D., & Brown, J. (2020, January). Creating effective industrial-control-systems honeypots. Proc. Hawaii Intl. Conf. on Systems Sciences, Wailea, HI, United States.

Rowe, N. & Rrushi, J. (2016). *Introduction to cyberdeception*. Springer.

Stouffer, K., Pillitteri, V., Lightman, S., Abrams, M., and Hahn, A., (2015, June). Guide to Industrial Control Systems (ICS) Security. National Institute of Standards and Technology, Gaithersburg, MD, NIST SP 800-82 Revision 2 [Online]

Washofsky, A. (2021, expected September). Promises and Perils: Deploying and Analyzing Multiuse Honeypots in the Cloud. [Master's thesis, Naval Postgraduate School].

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Research Sponsored Programs Office, Code 41
   Naval Postgraduate School
   Monterey, CA 93943