



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

DEPLOYING AND ANALYZING CONTAINERIZED HONEYPOTS IN THE CLOUD WITH T-POT

by

Alexander D. Washofsky

September 2021

Thesis Advisor:

Co-Advisor:

Thuy D. Nguyen

Neil C. Rowe

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC, 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2021	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE DEPLOYING AND ANALYZING CONTAINERIZED HONEYPOTS IN THE CLOUD WITH T-POT			5. FUNDING NUMBERS	
6. AUTHOR(S) Alexander D. Washofsky				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) <p>Honeypots (decoy systems) are effective tools to monitor cyberattack and intrusion attempts, but it is challenging to deploy enough of them to catch a sufficient amount of such activity. With cyberattacks on the rise, specifically those targeting critical infrastructure, better suspicious-traffic collection methods must be developed. This thesis explores the deployment and use of cloud-based honeypots within an open-source honeypot management framework, T-Pot. Instances of T-Pot ran honeypots that simulated a web server and an electrical-power distribution system, and their traffic was compared to previous local and cloud-based standalone honeypot deployments. The results showed that the cloud deployments received more traffic than local deployments and that the use of T-Pot did not discourage intrusions or attacks. T-Pot bundles security analysis tools and services for analyzing cloud-scale data, enabling more robust cyber defense for critical infrastructure and Department of Defense networks.</p>				
14. SUBJECT TERMS honeypot, cloud, cloud server, T-Pot, deception, cyber defense			15. NUMBER OF PAGES 83	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**DEPLOYING AND ANALYZING CONTAINERIZED HONEYPOTS IN THE
CLOUD WITH T-POT**

Alexander D. Washofsky
Lieutenant Commander, United States Navy
BA, The George Washington University, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

**NAVAL POSTGRADUATE SCHOOL
September 2021**

Approved by: Thuy D. Nguyen
Advisor

Neil C. Rowe
Co-Advisor

Gurminder Singh
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Honeypots (decoy systems) are effective tools to monitor cyberattack and intrusion attempts, but it is challenging to deploy enough of them to catch a sufficient amount of such activity. With cyberattacks on the rise, specifically those targeting critical infrastructure, better suspicious-traffic collection methods must be developed. This thesis explores the deployment and use of cloud-based honeypots within an open-source honeypot management framework, T-Pot. Instances of T-Pot ran honeypots that simulated a web server and an electrical-power distribution system, and their traffic was compared to previous local and cloud-based standalone honeypot deployments. The results showed that the cloud deployments received more traffic than local deployments and that the use of T-Pot did not discourage intrusions or attacks. T-Pot bundles security analysis tools and services for analyzing cloud-scale data, enabling more robust cyber defense for critical infrastructure and Department of Defense networks.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	MOTIVATION	1
B.	RESEARCH PLAN	2
C.	THESIS OUTLINE.....	2
II.	BACKGROUND AND RELATED WORK.....	3
A.	CLOUD COMPUTING.....	3
B.	HONEYPOTS	4
C.	PREVIOUS HONEYPOT RESEARCH.....	4
III.	METHODOLOGY AND DESIGN	7
A.	DIGITALOCEAN CLOUD ENVIRONMENT	7
B.	T-POT	7
C.	HONEYPOTS USED IN THIS THESIS	10
D.	PROTOCOLS USED IN CONPOT AND GRIDPOT	11
E.	OTHER TOOLS USED.....	12
IV.	EXPERIMENT IMPLEMENTATION	13
A.	CONFIGURATION AND IMPLEMENTATION.....	13
1.	TEST ENVIRONMENT	13
2.	CUSTOM T-POT TEMPLATES	13
3.	EXPERIMENT 1: CONPOT	14
4.	EXPERIMENTS 2 AND 3: SNARE AND TANNER	14
5.	EXPERIMENTS 4–7: GRIDPOT	15
B.	DATA COLLECTION	16
C.	PROBLEMS ENCOUNTERED	17
V.	ANALYSIS	21
A.	EXPERIMENT 1	21
1.	PROTOCOL DATA AND RESULTS	21
2.	OVERALL STATISTICS	27
B.	EXPERIMENTS 2 AND 3.....	29
1.	HTTP Data and Results.....	29
2.	Overall Statistics	33
C.	EXPERIMENTS 4–7	33
1.	Session Data	35
2.	SIMILARITIES BETWEEN EXPERIMENT TRAFFIC	41

3.	BEHAVIORAL ANALYSIS	43
4.	Shodan.....	44
D.	T-POT	45
VI.	CONCLUSIONS AND FUTURE WORK.....	51
	APPENDIX A. CUSTOM T-POT INSTALLATION.....	53
	APPENDIX B. COSINE SIMILARITY DATA.....	55
	LIST OF REFERENCES.....	59
	INITIAL DISTRIBUTION LIST	65

LIST OF FIGURES

Figure 1.	Screenshot of Cockpit Taken during Our Experiments, Showing Running Docker Containers.....	8
Figure 2.	Screenshot of Tanner Attack Map from Kibana Dashboard, Taken during Our Experiments, Showing a World Map of Geolocated Attack Sources.	9
Figure 3.	Screenshot of Suricata Alert Signature from the Kibana Dashboard, Taken during Our Experiments, Showing the Top 10 Alerts Generated by Suricata.	10
Figure 4.	Experiment 1 Architecture.....	14
Figure 5.	Experiments 2–3 Architecture	15
Figure 6.	Experiments 4–7 Architecture	16
Figure 7.	Experiment 1 HTTP Version Distribution.....	21
Figure 8.	Experiment 1 HTTP Request Method Distribution	22
Figure 9.	Experiment 1 MODBUS Function Codes.....	23
Figure 10.	Experiment 1 MODBUS Valid Function Codes.....	24
Figure 11.	MODBUS Activity by Country Count	25
Figure 12.	Experiment 1 EtherNet/IP Command Code Distribution	26
Figure 13.	S7Comm Activity by Country Count	27
Figure 14.	Experiment 1 Protocol Packet Count by Date	28
Figure 15.	Experiment 1 Protocol Distribution.....	28
Figure 16.	Experiments 2 and 3 HTTP Requests by Date (Logarithmic Scale)	31
Figure 17.	Experiments 4–7 IEC 104 Traffic.....	38
Figure 18.	Experiments 4–7 IEC 104 Session Statistics	38
Figure 19.	Example Hex Dump of IEC 104 Error Frame Showing Encapsulated HTTP Packet.....	40
Figure 20.	Example of IEC104 Error Packet Containing SIP Data	43

Figure 21.	Screenshot of Elastic Database of Experiment 2, Showing Multiple Data Fields from an HTTP Request.....	46
Figure 22.	Screenshot of Elastic Database of Experiment 4, Showing Entire HTTP Request in One Data Field	46
Figure 23.	Screenshot of Kibana Dashboard of Experiment 2, Showing an Errant HTTP Path	46
Figure 24.	Screenshot of Suricata Dashboard of Experiment 5, Showing One CVE.....	47
Figure 25.	Processor Use in Experiment 2	48
Figure 26.	Processor Use in Experiment 3	48
Figure 27.	Memory (RAM) Use in Experiment 2	49
Figure 28.	Memory (RAM) Use in Experiment 3	49
Figure 29.	Disk Use in Experiment 2	49
Figure 30.	Disk Use in Experiment 3	50

LIST OF TABLES

Table 1.	Experiment Data Collection Dates.....	17
Table 2.	Comparison of Experiment 1 S7COMM Activity to Hyun’s Data.....	26
Table 3.	Cosine Similarity between Experiment 1 and Hyun’s Data	29
Table 4.	Experiments 2 and 3 HTTP Versions	30
Table 5.	Experiments 2 and 3 HTTP Methods.....	30
Table 6.	Experiments 2 and 3 Top Countries Comparison	31
Table 7.	Experiment 2 and 3 Alleged Countries	32
Table 8.	Experiment 2 and 3 Top 11 Overall Count of HTTP Paths	33
Table 9.	Overall Traffic Statistics for Experiments 4–7	34
Table 10.	Experiments 4–7 IP Address Session Data	35
Table 11.	Experiments 4–7 HTTP Country Data.....	36
Table 12.	Experiments 4–7 HTTP Path Data.....	37
Table 13.	Experiments 4–7 IEC 104 Methods	39
Table 14.	Experiments 4–7 Daily IEC 104 Traffic	39
Table 15.	Experiments 4–7 IEC 104 Alleged Country Data.....	40
Table 16.	Experiments 4–7 Cosine Similarity for All Traffic	41
Table 17.	Experiments 4–5 Measure of Significance of Total Traffic Comparisons	42
Table 18.	Experiments 4–7 Cosine Similarity for IEC 104 Traffic	42
Table 19.	Experiments 4–7 Measure of Significance of IEC 104 Traffic	42
Table 20.	Experiments 4–7 Traffic from Specific Subnet	44

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

AWS	Amazon Web Services
BACnet	Building Automation and Control Network
CVE	Common Vulnerability and Exposure
DOD	Department of Defense
FTP	File Transfer Protocol
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
ICS	industrial control system
IDS	intrusion-detection system
IEC	International Electrotechnical Commission
IPMI	Intelligent Platform Management Interface
IPS	intrusion-prevention system
JSON	JavaScript Object Notation
NIST	National Institute for Standards and Technology
NPS	Naval Postgraduate School
PaaS	Platform as a Service
PCAP	packet capture
PDU	power distribution unit
PLC	programmable logic controller
S7Comm	S7Communications Protocol
SaaS	Software as a Service
SCADA	supervisory control and data acquisition
SECNAVINST	Secretary of the Navy Instruction
SNMP	Simple Network Management Protocol
SSH	Secure Shell
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TTP	tactics, techniques, and procedures
UDP	User Datagram Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First, thank you to all of the professors and staff at the Naval Postgraduate School. It was a pleasure to learn from you in person, and the efforts you went through to ensure our remote-learning quarters during the pandemic were still valuable did not go unappreciated. Thank you for always sharing your passion.

I would like to thank Dr. Rowe and Professor Nguyen for their guidance, assistance, and mentorship; this definitely would not have been possible without you. Thanks for introducing me to such an interesting research area and for giving me the room to explore it.

To my cohort, and especially the other two of the Three Amigos—I could not have asked for a better team. Thanks for the memories, discussions, study sessions, and, of course, the hilarious commentary and memes. You all kept me sane even when we could not meet in person.

Most importantly, I would like to thank my wife, Erica. You have always been my rock, and that was never truer than during this crazy time when we were both remote students. Thank you for all the support you gave me and continue to give me every day. I love you so much!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

This thesis addresses network defense for the National Cyber Strategy (National Security Council, 2018), specifically, the priorities to “Secure Federal Networks and Information” and “Secure Critical Infrastructure,” and the Interim National Security Strategic Guidance (National Security Council, 2021), which made cybersecurity a “top priority” and “elevate [d] cybersecurity as an imperative across the government.” This research is also guided by the “National Security Memorandum on Improving Cybersecurity for Critical Infrastructure Control Systems” (White House, 2021), which called for “deploying systems and technologies that can monitor control systems to detect malicious activity and facilitate response actions.”

This thesis developed methods to identify threats to servers by operating honeypots (decoy servers) in a cloud-computing environment, allowing later analysis of their saved attack data. It also examined threats to industrial control systems. The work explored how specific deployment strategies affected attacks. We analyzed data to determine whether particular strategies could help deployment and real-time analysis by providing a different “attack surface,” potentially encouraging more attack patterns by malicious actors.

This thesis evaluates the feasibility, advantages, and disadvantages of deploying honeypots in the cloud on a specific honeypot management platform, T-Pot (Telekom Security, n.d.). This platform promises a simplified installation process as well as greater data collection and analytical capabilities compared to using single honeypots.

A. MOTIVATION

Cyberattacks are not new. Malicious attempts to gain access, steal information, or damage systems have a long history. Even before computers were networked, malicious code was spread through other media (Middleton, 2017). The more systems were interconnected, the more opportunity for malicious attempts grew, as cybercriminals began using cyberattacks for illicit gain, and foreign governments began using cyberattacks for espionage (Kaplan, 2017). Today, governments and businesses must defend their systems against a rising threat of cybercrime, which cost \$3 trillion in 2015 (Huang et al., 2018)

and should double to \$6 trillion in 2021. Furthermore, the threats to industrial control systems (ICS) are “among the most significant and growing issues confronting our Nation” (Cybersecurity and Infrastructure Security Agency, 2021), as these systems control critical infrastructure. For example, oil and natural gas pipelines have recently been targeted by state-sponsored cyber actors (Cybersecurity and Infrastructure Security Agency and Federal Bureau of Investigation, 2021). Using honeypots to detect anomalies in network traffic is a recognized approach to reduce threats to ICS networks (Hurd & McCarty, 2017).

The U.S. Department of Defense relies heavily on Internet-facing servers for routine and wartime operations. These systems are identified by SECNAVINST 3501.1D (Secretary of the Navy, 2018) as critical infrastructure, and some of these are ICSs. Increasing our threat analysis and defensive capabilities can better defend these servers and make them more resilient to attack. By observing adversarial cyber actions against honeypots, we can learn adversary tactics, techniques, and procedures (TTPs) to better defend the real servers.

B. RESEARCH PLAN

Our research plan had three phases. Phase 1 tested a single honeypot deployed on a cloud server within a T-Pot honeypot platform and compared its results to that of a similar deployment on a local server. Phase 2 compared two instances of honeypots on cloud servers, one in our test T-Pot platform and one as a standalone server and compared the results. Phase 3 deployed and compared four instances of a modified honeypot previously used by our research group (Bieker & Pilkington, 2020), two in our T-Pot platform and two on standalone systems. Each instance in the pairs had a different location.

C. THESIS OUTLINE

Chapter II examines previous work on honeypots and cloud computing. Chapter III describes the architecture of our experiments and the honeypots used. Chapter IV discusses our research methodology, the specific configurations for each experiment, and the data-analysis. Chapter V summarizes and discusses the experiment results. Chapter VI states our conclusions and suggests future work.

II. BACKGROUND AND RELATED WORK

A. CLOUD COMPUTING

Cloud computing originates from time-sharing on large servers in the early days of computing (Surbiryala & Rong, 2019). Customers would send computing jobs to a shared computer to execute and output results, eliminating the need for each customer to have their own machines. Today cloud computing environments are provided by Amazon Web Services, Microsoft Azure, Google Cloud, and others (Borges et al., 2018).

The National Institute for Standards and Technology (NIST) defines three standard cloud-computing service models: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS) (Mell & Grance, 2011). IaaS uses virtual machines and emulation to provide a user with access to a private computing environment on a remote server. PaaS, mainly used for application development, provides the user with an environment including toolkits, databases, and services. SaaS provides the user with access to programs provided by the cloud-services provider.

NIST also defines three cloud-deployment models (Simmon, 2018). A private cloud is deployed by an organization, allowing that organization the most control over the implementation and security. A public cloud is accessed and provides services through the Internet. A hybrid cloud involves some elements of private and public clouds. For example, an organization may have a private cloud hosted on-premises but use public cloud resources through a provider when demand on the private cloud exceeds capacity.

Security is important in cloud computing (Basu et al., 2019). Depending on the service model, a cloud services provider may have significant access to the organization's data. This is a greatest concern with SaaS, where the user runs the provider's applications and must trust their security and safeguards. However even with IaaS where a user controls their own computing instance, the provider still controls the infrastructure behind it, which potentially gives them access to the computing instance's data. Also, security vulnerabilities in the cloud provider's system are passed to the user. For instance, a vulnerability discovered in Amazon's Elastic Compute Cloud (EC2) in 2009 allowed

attackers to intercept and modify messages that were assumed to be secure (Gruschka & Iacono, 2009).

B. HONEYPOTS

A honeypot is a computer security tool that collects data on unauthorized access and access attempts to a sacrificial computer or device (Campbell, Padayachee, & Masombuka, 2015). A honeypot emulates the first steps of legitimate services as it interacts with an attempted attacker. These interactions are logged for later analysis and can provide the basis for security alerts.

Honeypots can be categorized by their purpose and interaction level. They can be deployed as production honeypots or research honeypots (Zhang et al., 2003). Production honeypots serve as tripwires, telling personnel of unauthorized access attempts. Research honeypots collect as much information as possible from attackers to see trends, and are used in business, government, and academia.

Honeypots can be either low-interaction or high-interaction (Alata et al., 2006). Low-interaction honeypots emulate a few services, provide basic responses, but may not respond to a more complex attack as a real system would. Low-interaction honeypots are easy to deploy but can be easily identifiable as honeypots. High-interaction honeypots look as real as possible and can even include an entire virtual system for an attacker to interact with. While they can capture more complex attack data than low-interaction honeypots, they are more complex and resource-intensive to deploy. Medium-interaction honeypots have characteristics between those of low and high-interaction honeypots.

C. PREVIOUS HONEYPOT RESEARCH

One honeypot project (Kelly, Pitropakis, Mylonas, McKeown, & Buchanan, 2021) installed a standard version of the T-Pot (Telekom Security, n.d.) honeypot platform on multiple cloud service providers in different geographic locations. This standard version of the platform includes the following honeypots that are relevant to this thesis: Conpot (Jicha et al., 2016), Cowrie (Oosterhof, n.d.), Dionaea (Dionaea, 2015), and Snare and Tanner (Rist et al., n.d.). Their results demonstrated that the market share of the cloud service

providers did not affect the rate of attacks received; for instance, Google Cloud, the provider with the lowest market share, was targeted just as much as their other deployments. The researchers also noted that the T-Pot platform provided them real-time situational awareness into attacks with its visualization capabilities.

Bove (2018) set up multiple T-Pot instances in cloud services and analyzed the results. He saw secure shell (SSH) attacks to the Cowrie honeypot (Oosterhof, n.d.) and malware captured by the Dionaea honeypot (Dionaea, 2015). He concluded that no significant differences occurred between regular servers and cloud-based systems for SSH attacks as recorded by Cowrie, or the attempted malware executables captured by Dionaea.

Chapendama (2019) deployed a standard instance of T-Pot on the Google Cloud Platform in Europe. Results showed that Cowrie was the most often attacked honeypot, and most attacks targeted both the Telnet and SSH services. The most common attack source IP country was China, followed by the United States, but he noted that attacks came from around the world.

A honeypot project at the University of Wisconsin – Madison (Brown, Lam, Prasad, Ramasubramanian, & Slauson, 2012) deployed multiple honeypots into the cloud, but not as part of an overall honeypot platform such as T-Pot. Their data showed mostly attacks from China and the United States, and that attacks received were not significantly different between different cloud-service providers.

Another project (Sochor & Zuzcak, 2014) ran Dionaea and SSH honeypots over three months on both a local network and a cloud deployment. Their results suggested attacks detected with Dionaea did not differ with where the honeypot was hosted. The largest alleged source of attacks against these honeypots was India, which had eight times the number of attacks from China.

Serbanescu et al. (2015) deployed an ICS honeynet, a collection of honeypots, on a cloud server. This honeynet simulated seven protocol services in different combinations. The researchers concluded that attackers' interest in the honeypots depended more on the availability of certain protocols. They also noted that overall attacker interest in their

project was low and recommended that future work should involve “higher interaction” ICS honeypots.

A honeypot project at the Naval Postgraduate School (NPS) (Chong & Koh, 2018) ran HTTP and SSH honeypots in separate experiments; these honeypots were Snare & Tanner (Rist et al., n.d.) and Cowrie (Oosterhof, n.d.) respectively. The Snare and Tanner services cooperated to give an attacker a seemingly vulnerable HTTP server, while Cowrie presented a vulnerable SSH server. Their results demonstrated that most attacks on these services were automated, and changes to the honeypots to further obfuscate them seemed to have little effect on the attacks.

Another honeypot project at NPS (Hyun, 2018) ran a low-interaction ICS honeypot called Conpot (Jicha et al., 2016) within a virtual machine on a local internet-facing server for four months. It serviced multiple protocols, including Hypertext Transfer Protocol (HTTP), EtherNet/IP, MODBUS, S7Communications (S7Comm), Simple Network Management Protocol (SNMP), Building Automation and Control Networks (BACnet), and Intelligent Platform Management Interface (IPMI). HTTP, a non-ICS protocol, accounted for 56% of the attack traffic to this honeypot, and the MODBUS ICS protocol, accounted for 35% of traffic. This research demonstrated that even a low-interaction honeypot like Conpot could provide useful data.

Another ICS honeypot project (Dougherty, 2020) at NPS that we have built upon deployed a modified version of Conpot called GridPot that emulates an electrical distribution system. GridPot replaced Conpot’s low-interaction handling of the International Electrotechnical Commission (IEC) 60870-5-104 (IEC 104) communications protocol with a more interactive version that communicated to a power-grid simulator called GridLAB-D (Chassin et al., 2008), providing a more convincing server to attackers. This setup was later deployed as standalone honeypots in the cloud (Bieker & Pilkington, 2020). The results demonstrated that this was a workable platform for more detailed experiments.

III. METHODOLOGY AND DESIGN

This chapter discusses the design of our honeypot experiments. We also describe the cloud hosting of our honeypots, and how we constructed, deployed, and operated them. More specific details of our configurations and implementations are in Chapter IV. Our experiments used both low-interaction and high-interaction research honeypots in a cloud-computing environment. The goals were to compare the data collected by these honeypots to previous honeypot deployments and to contrast functional differences between being deployed as standalone honeypots and as honeypot daemons within the T-Pot honeypot platform.

A. DIGITALOCEAN CLOUD ENVIRONMENT

We chose to use a cloud IaaS service so we could have the most control over honeypot operation. Previous honeypot research at NPS had examined multiple providers including Amazon Web Services (AWS). However, DigitalOcean (n.d.) was found to be an ideal service, as it did not discourage security applications such as honeypots and was priced well for the services we would use.

DigitalOcean enables creation of virtual servers that they call “droplets.” Droplets can have either shared or dedicated hardware, with various options including the number of processors and amount of memory. Other droplet installation options include a pre-installed operating system, block storage, a datacenter region, and automatic backup. Initial configuration of the droplets is done through DigitalOcean’s remote console, and once firewall rules and the SSH remote-shell parameters are set up, the rest of the configuration can be done over SSH.

B. T-POT

T-Pot (Telekom Security, n.d.) is a honeypot deployment platform. Deployers can choose from multiple templates. For example, the “medical” template includes the honeypots Dicompot (Keri, Lechthaler, & Ochse, n.d.) and Medpot (Schmall, Vorbach, & Ochse, n.d.) providing attack surfaces for healthcare-related protocols, while the ICS

template includes Conpot and Cowrie. The templates also include tools for analysis and real-time monitoring such as Cockpit (Red Hat, n.d.), Elastic Stack (Elasticsearch B.V., n.d.), and Suricata (Open Information Security Foundation (OISF), n.d.).

T-Pot uses Docker (Combe, Martin, & Di Pietro, 2016), a program that can run software in virtual operating systems in instances called containers. These containers can easily be managed and configured on the host machine, and a single machine can run many Docker containers. T-Pot has each honeypot and tool containerized and running in separate Docker containers, which provides modularity as well as better safety compared to running software directly on the machine.

Cockpit is a Web-based graphical user interface for Linux servers. It monitors and manages the system on which T-Pot runs. Some of its information is duplicated in the DigitalOcean dashboard including live graphs of processor use, memory, and disk I/O. Cockpit also monitors running Docker containers (Figure 1), system services, and applications. It can also update software, create new user accounts, and provide a Web-based terminal session. Overall, Cockpit allows easier management of a T-Pot installation than SSH alone.

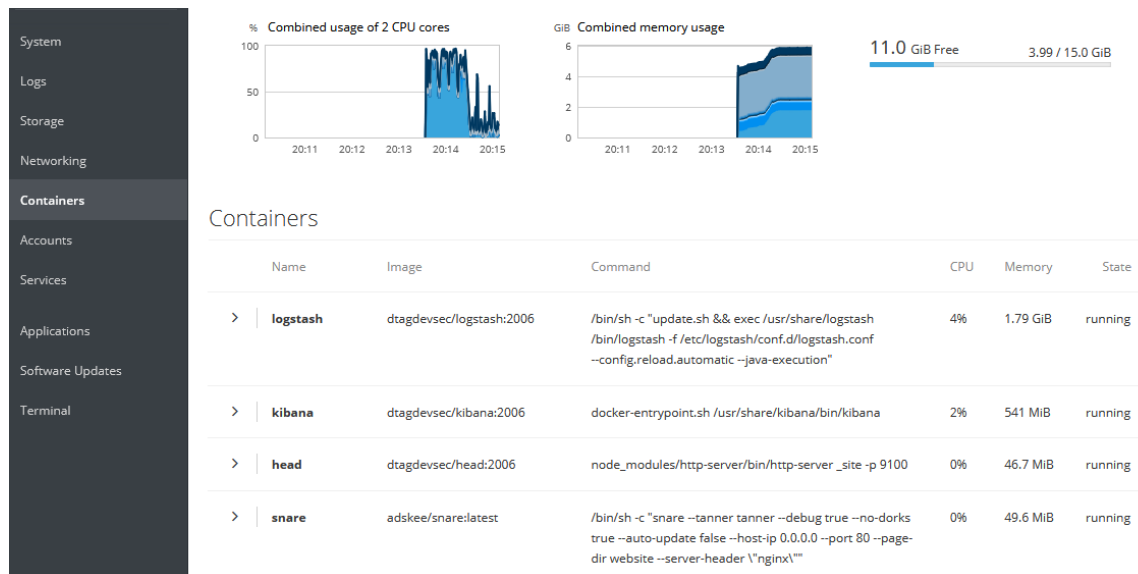


Figure 1. Screenshot of Cockpit Taken during Our Experiments, Showing Running Docker Containers

The Elastic Stack has three programs: Elasticsearch, Logstash, and Kibana. Logstash sends data from the honeypot and tool logs to the database of Elasticsearch. Elasticsearch searches and analyzes using JavaScript Object Notation (JSON). Kibana is a data visualization tool for the Elasticsearch database. Kibana has predefined dashboards that allow a user to view data from individual honeypots (Figure 2), individual tools like Suricata, or an overall “T-Pot Dashboard” that includes data from the honeypots and tools.



Figure 2. Screenshot of Tanner Attack Map from Kibana Dashboard, Taken during Our Experiments, Showing a World Map of Geolocated Attack Sources.

Suricata is a threat detection engine that can act as both an intrusion-detection system (IDS) and intrusion-prevention system (IPS). Within T-Pot, Suricata only detects malicious activity, logging alerts with their associated CVE (Common Vulnerabilities and Exposures) codes. Suricata provides real-time analysis of attacks, with details on their severity, category, and signature. Suricata examines all packets received, beyond what honeypots capture. For example, Suricata detects Transmission Control Protocol (TCP) packets with the SYN (synchronize) and URG (urgent) flags set that do not complete the TCP 3-way handshake and lack an HTTP payload, so they do not interact with HTTP honeypots. Suricata identifies these packets as related to critical vulnerabilities in VxWorks (Seri et al., 2019), a real-time operating system used in devices such as industrial control systems. These additional data points are similarly fed into the Elastic Stack and viewable on the Kibana dashboard (Figure 3).

Suricata Alert Signature - Top 10		
Export		
ID	Description	CNT
2210045	SURICATA STREAM Packet with invalid ack	4,201
2210029	SURICATA STREAM ESTABLISHED invalid ack	4,198
2210020	SURICATA STREAM ESTABLISHED packet out of window	3,252
2210048	SURICATA STREAM reassembly sequence GAP -- missing packet...	2,628
2001978	ET POLICY SSH session in progress on Expected Port	73
2030387	ET EXPLOIT Possible CVE-2020-11899 Multicast out-of-bound r...	70
2210051	SURICATA STREAM Packet with broken ack	19
2210041	SURICATA STREAM RST recv but no session	17
2029054	ET SCAN Zmap User-Agent (Inbound)	11
2221014	SURICATA HTTP missing Host header	10

Figure 3. Screenshot of Suricata Alert Signature from the Kibana Dashboard, Taken during Our Experiments, Showing the Top 10 Alerts Generated by Suricata.

C. HONEYPOTS USED IN THIS THESIS

Conpot (Jicha et al., 2016) is a low-interaction honeypot that can simulate several protocols important to us, including HTTP for a basic ICS-like Web interface, MODBUS, EtherNet/IP, S7Comm, and IEC 104. Its templates can specify combinations of these protocols. Conpot logs all interactions on the ports for these protocols and outputs them to a log file.

Snare and Tanner (Rist et al., n.d.) are two components of T-Pot. Snare accepts HTTP connections and sends HTTP requests to Tanner. Tanner analyzes Snare events and identifies an attacker trying to exploit a vulnerability. Tanner logs events in JSON format. Snare also includes a tool to clone (duplicate) existing websites and supply them to attackers. The version of Snare that comes with T-Pot includes ten cloned websites, and upon installation, one is randomly chosen to present to attackers.

GridPot (Dougherty, 2020) modifies and extends Conpot, simulating an HTTP server that hosts an ICS electric-grid interface. Attackers see an interface showing variables of an ICS system. Instead of the low-interaction Conpot implementation of IEC 104,

GridPot includes GridLAB-D (Chassin et al., 2008), an electric-grid simulator, to add realism to the IEC 104 protocol interactions. Dougherty modified GridPot to accept incoming IEC 104 requests, send them through the Conpot honeypot to GridLAB-D, and return the results of the request.

D. PROTOCOLS USED IN CONPOT AND GRIDPOT

Most traffic captured by our experiments was of the Hypertext Transfer Protocol (HTTP) (Leach, et al., 1999) for communication between clients and Web servers over TCP port 80. Most commonly, a client will send a “GET” request to retrieve a specific Web page. Other HTTP methods seen during our experiments were PUT, POST, HEAD, CONNECT, PROPFIND, and OPTIONS. Snare/Tanner allowed a user to interface with a specified copy of a Web page.

MODBUS (Swales, 1999) is an application-level protocol for industrial control systems, designed in 1979. It allows a “client” controlling device to send function codes to controlled “server” devices. The protocol defines function codes as between 1 and 255, with 128 and higher being reserved for exception responses from the controlled devices. One client device can communicate with a maximum of 247 server devices. Originally, MODBUS was used only for direct serial connections; today, MODBUS can be done by TCP packets, typically using port 502. Conpot can handle MODBUS in a limited capacity that only checks for proper formatting of requests; GridPot does not support MODBUS.

EtherNet/IP (Brooks, 2001) is an adaptation of the Common Industrial Protocol (CIP) for use with Ethernet at the link layer. EtherNet/IP is used in industrial control systems and includes explicit and implicit messaging. Explicit messages are sent by TCP port 44818 between a client and server and can change parameters and programs. Implicit messages sent by User Datagram Protocol (UDP) port 2222 and are for monitoring and basic data. The EtherNet/IP handler in Conpot only recognizes explicit messages on port 44818 and checks for their proper formatting; GridPot does not support EtherNet/IP.

S7 Communication (S7Comm) (Eigner, Kreimel, & Tavalato, 2018) is a proprietary protocol developed by Siemens for communication between their manufactured programmable logic controllers (PLCs) and supervisory control and data

acquisition (SCADA) systems. S7Comm uses TCP port 102. S7Comm is handled by Conpot, and proper formatting is checked, including power distribution unit (PDU) type, data length, and request ID. GridPot does not support S7Comm.

IEC 60870-5-104 (Matousek, 2017), often shortened to simply IEC 104, is an application-level protocol used in industrial control systems, and specifically electrical engineering and power systems. IEC 104 uses three kind of data frames. I-format frames transfer information, S-format frames send supervisory commands and acknowledgements, and U-format frames send control commands. Conpot checks if IEC 104 traffic on port 2404 is properly formatted as one of the three data frame types, and whether the frames contain valid control functions. GridPot as modified by Dougherty parses these data frames and sends the commands to the GridLAB-D simulation, which executes them and returns the results to the attacker through GridPot and then Conpot.

E. OTHER TOOLS USED

Shodan (n.d.) is a search engine designed to inspect Internet-connected devices and servers. It automatically crawls (searches for) publicly accessible IP addresses, and indexes them based on metadata provided by responses at those addresses. Shodan classifies addresses as cloud, honeypot, server, webcam, router, and more. Honeyscore is a tool within Shodan that judges whether an address is a honeypot. We used Shodan's search and scan capabilities, as well as Honeyscore, to determine how realistic our honeypots appeared.

GeoIP (MaxMind, Inc., n.d.) is a service that provides IP geolocation data. Depending on the geographic region, it can locate an IP address within a radius of several kilometers or several hundred kilometers. We used GeoIP to help identify the country to which an IP address allegedly belongs.

IV. EXPERIMENT IMPLEMENTATION

This chapter discusses further details of the implementation of our experiments. The research had three sets of experiments described in Section IV.A. Data analysis methods are described in Section IV.B, and the results are in Chapter V.

A. CONFIGURATION AND IMPLEMENTATION

We did seven experiments with three deployment methods.

1. TEST ENVIRONMENT

Our experiments used DigitalOcean virtual machines, i.e., “droplets,” with the same configuration. The droplets were built using DigitalOcean’s pre-built Debian Linux 10 x64 image.

We used droplets with two dedicated processors, eight gigabytes of random-access memory (RAM), and a 25-gigabyte solid-state drive (SSD) for main storage. For comparison purposes, these hardware options were the same as the previous projects’ droplet hardware configurations. Each droplet had a single unique public IPv4 address. For simpler analysis, we did not enable IPv6 connectivity on the droplets. During initial set-up and installation, we remotely accessed the console of each droplet using SSH from our local machine. DigitalOcean’s firewall allows each droplet to have individual rulesets, and this made it easy to isolate the droplets during installation and testing, during which time we limited access only to our local machine’s IP address. When conducting experiments, the specific firewall ruleset for the honeypot used in the experiments was applied to the droplets, allowing access from the outside world.

2. CUSTOM T-POT TEMPLATES

As previously mentioned, T-Pot has six built-in templates for installation. For each experiment deployment, we modified the T-Pot installation script to use a custom template that we named “Washofsky” to install the honeypot and tools required for each experiment (see Appendix A). This template used a custom docker-compose.yml configuration file,

modified from the default T-Pot configuration file, to set up the containers, services, and networks required

3. EXPERIMENT 1: CONPOT

Experiment 1 compared Hyun’s standalone Conpot (Hyun, 2018) with a similar Conpot instance running within T-Pot (Figure 4). Hyun deployed a default configuration of Conpot in a virtual machine on a server outside the Naval Postgraduate School firewall. Our Experiment 1 used a custom T-Pot template to install a Conpot honeypot running on a DigitalOcean droplet in the U.S., with Suricata for network management, and Cockpit, Elasticsearch, Kibana, and Logstash for management, visualization, and analysis. Experiment 1 acquainted us with the tools that T-Pot provided, especially understanding the underlying code and installation scripts.

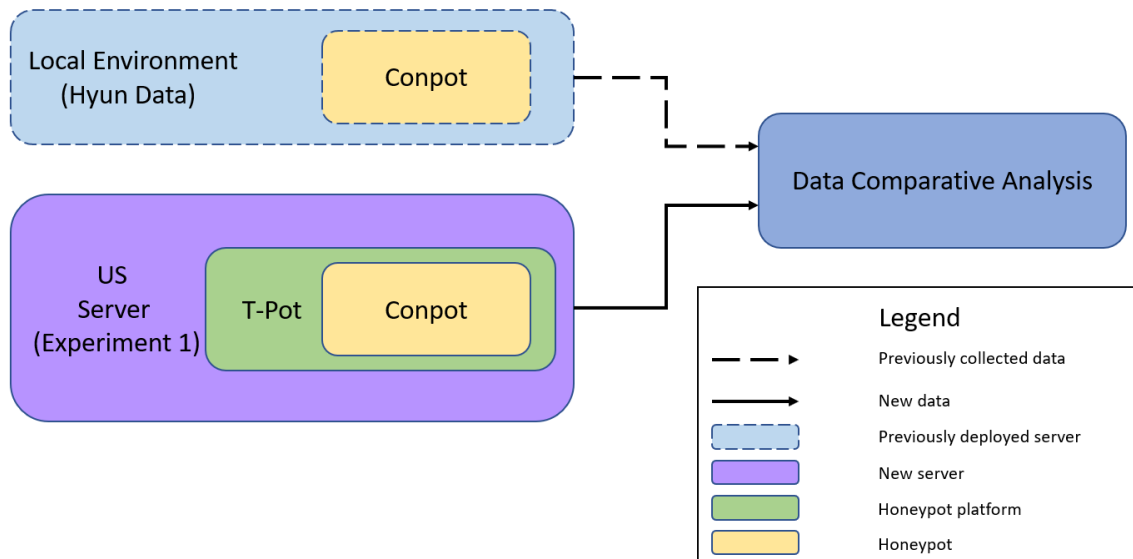


Figure 4. Experiment 1 Architecture

4. EXPERIMENTS 2 AND 3: SNARE AND TANNER

Experiments 2 and 3 compared to Chong and Koh’s work (Chong & Koh, 2018) with the Snare and Tanner honeypots in T-Pot (Figure 5). They deployed Snare, Tanner, and Cowrie honeypots in a virtual machine outside the Naval Postgraduate School firewall.

Our Experiment 2 installed Snare and Tanner on a cloud droplet, cloning the same website that Chong and Koh used. Our Experiment 3 installed Snare and Tanner using a custom T-Pot template and the tools listed in IV.A.3, with the same website content with minor name changes. Experiment 3 required a new Docker container to run Snare with our cloned website since the Snare containers that come with T-Pot are incomparable to Chong and Koh’s previous work.

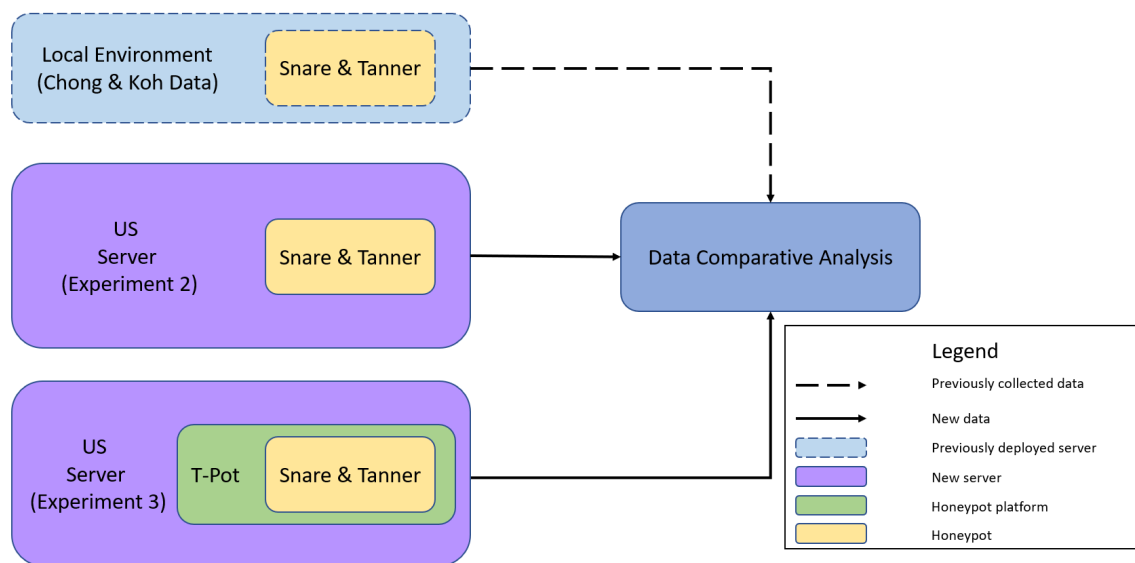


Figure 5. Experiments 2–3 Architecture

5. EXPERIMENTS 4–7: GRIDPOT

Experiments 4–7 compared our experiments to Bieker and Pilkington’s work with GridPot (Bieker & Pilkington, 2020), as modified by Dougherty (Figure 6). Their experiments deployed two GridPot honeypots, obfuscated to better simulate a real ICS environment, in different locations. Our Experiments 4 and 5 installed GridPot and the additional tools listed in IV.A.3 in T-Pot on two droplets in Asia and the U.S. Experiments 4 and 5 required some preparation since no support for GridPot was provided in T-Pot. We first created a new Docker container to host GridPot, and then tested it to ensure correct functionality in the containerized environment. Experiments 6 and 7 installed GridPot on

two additional droplets in the same two locations as Experiments 4 and 5, but without using the T-Pot platform.

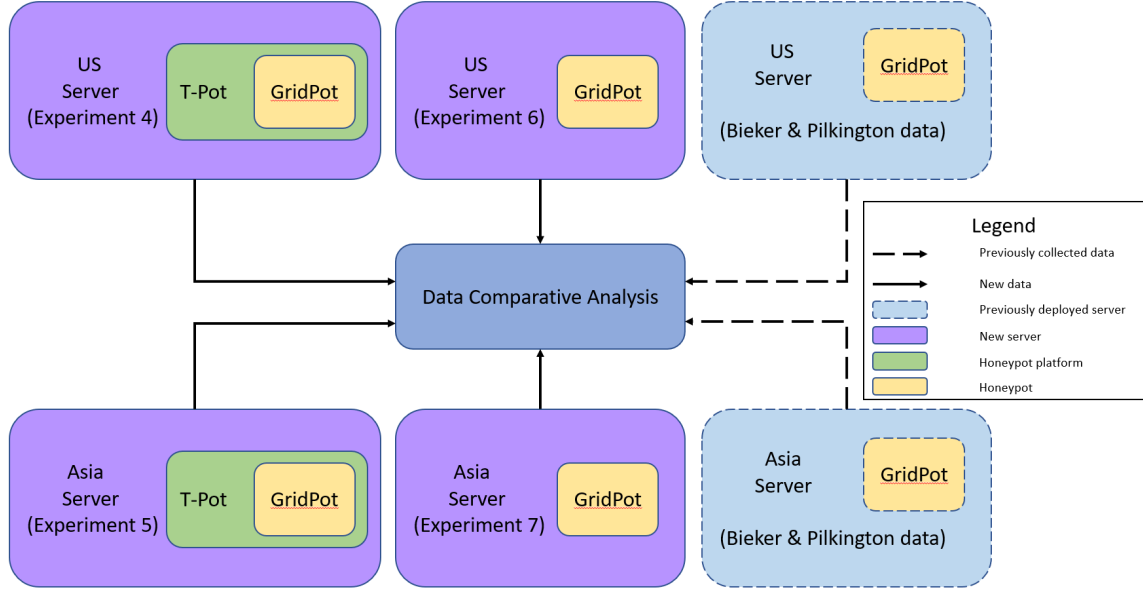


Figure 6. Experiments 4–7 Architecture

B. DATA COLLECTION

Data from each experiment was network traffic captured using packet-capture (PCAP) software and activity logs generated by the honeypot. On each droplet, an automated Linux service ran TShark (Wireshark Foundation, n.d.) to collect traffic on the TCP ports. These PCAP files were then pushed to a central repository hourly. Table 1 shows the data collection dates for each experiment.

Table 1. Experiment Data Collection Dates

Experiment Set	Date Range
Experiment 1	December 4, 2020 – January 29, 2021 (56 days)
Experiments 2 and 3	February 11, 2021 – May 12, 2021 (90 days)
Experiments 4–7	May 1, 2021 – June 1, 2021 (31 days)

Experiment 1 also collected Conpot log files to compare with Hyun’s data. These are timestamped text files recording sessions and connections, and some details of packets sent.

For each experiment, packet captures were merged using Mergecap (Renfro & Guyton, 2021). Parsers were created in Python for each type of data, including statistics about the source IP addresses, daily traffic, and the protocol-specific statistics used in each experiment. Only Conpot logs were analyzed in Experiment 1 since the previous work did not report captured traffic data.

C. PROBLEMS ENCOUNTERED

A packet capture script using TShark originally captured traffic in Experiment 1 from the virtual network adapter of the Conpot Docker container. However, this adapter was not persistent, and upon system reboot (which T-Pot scheduled daily), the network adapter had to be re-created. The packet capture script was rewritten to capture on the persistent “eth0” adapter, filtering for the ports used for the experiment. Also, T-Pot’s automatic maintenance removes honeypot logs older than 30 days. This caused partial loss of Conpot logs for Experiment 1.

Initially, Conpot in Experiment 1 did not respond to packets on UDP ports 161 (SNMP), 47808 (BACnet), and 623 (IPMI). The T-Pot template configuration file did not specifically forward that traffic to the honeypot using UDP, and Docker defaulted to forwarding using TCP. This misconfiguration was also in the default T-Pot configuration files, limiting any instance of Conpot to listening for these protocols using TCP. After we

submitted a bug report, this issue was corrected in Issue #781 of the T-Pot project. Experiments 2–7 did not use protocols requiring UDP traffic.

A problem with the Conpot log was that it recorded malformed HTTP traffic on port 80 as HTTP 0.9 packets. It did not check for a malformed packet before handing the data to the “BaseHTTPRequestHandler” Python module, which defaulted to HTTP 0.9 unless a version was detected. A comparison of the packet-capture data with Conpot log entries revealed HTTP requests recorded by TShark that were missed entirely in the Conpot log. This incompleteness was also identified by Hyun and Dougherty. Furthermore, timestamps for some MODBUS and FTP packets were inaccurate, as many cached ones appeared in the log during Conpot’s shutdown, indicating they were flushed before the program exited.

In Experiment 3, a change to the Docker configuration caused T-Pot and the Snare and Tanner honeypots to pause for about a week. This was only noticed when we could not view the Kibana dashboard. The droplet was still pushing packet-capture data to the repository, so we incorrectly assumed that the droplet was healthy. After the misconfigurations were fixed, we changed our “proof of life” procedure to routinely log into the droplet to ensure services are running, and view running Docker containers using Cockpit.

Experiment 3 revealed a bug in the Tanner software that logged certain HTTP requests incorrectly, which resulted in unexpected data fields in the Elastic Search database. This bug was also reported but is yet unresolved.

Experiment 7 received significantly more traffic than Experiments 4–6, and analysis of the traffic captures and a reverse IP lookup revealed an IP address that appears to be the sole address in the Domain Name Service (DNS) record for a Korean-based website. This website’s record was created in 2017 and last updated in 2020, well before Experiment 7 started. The registered domain is like a real website and could relate to a typo-squatting attack (Moubayed, Injadat, Shami, & Lutifyya, 2018), where attackers present a similar website to the one a user intended to visit but with a slightly different domain name. In our case, visitors to this website would only see the basic ICS website

provided by GridPot. (It is also possible that this website previously existed, and the DNS record has not changed though the IP address was recycled.) Bieker and Pilkington (2020) observed similar DNS issues in their experiments.

Experiments 6 and 7 saw an unexpected error which caused the honeypots in both experiments to stop responding to HTTP requests about seven days into data collection within hours of each other. The problem did not affect the handling of IEC 104 requests, and the honeypots continued to collect IEC 104 data for the entire duration of the experiments. It is unlikely that this error was caused by the DNS issue observed in Experiment 7, since Experiment 6 experienced the same error within hours of Experiment 7's failure.

THIS PAGE INTENTIONALLY LEFT BLANK

V. ANALYSIS

A. EXPERIMENT 1

Conpot ran from 4 December 2020 to 28 January 2021. Its droplet captured PCAP activity, but T-Pot’s default was to keep only 30 days of honeypot logs. The PCAP capture showed that the Conpot log does not record every event. Hyun’s thesis used the Conpot log only, so the comparison to Hyun’s thesis only used the Conpot log data.

1. PROTOCOL DATA AND RESULTS

a. HTTP

Conpot reported 5,656 HTTP requests and responses in Experiment 1. These mainly used HTTP Version 1.1 and 1.0, with counts of 4,498 and 890 respectively (Figure 7). Conpot also mistakenly reported 268 activities using HTTP Version 0.9, but packet analysis revealed these packets were malformed.

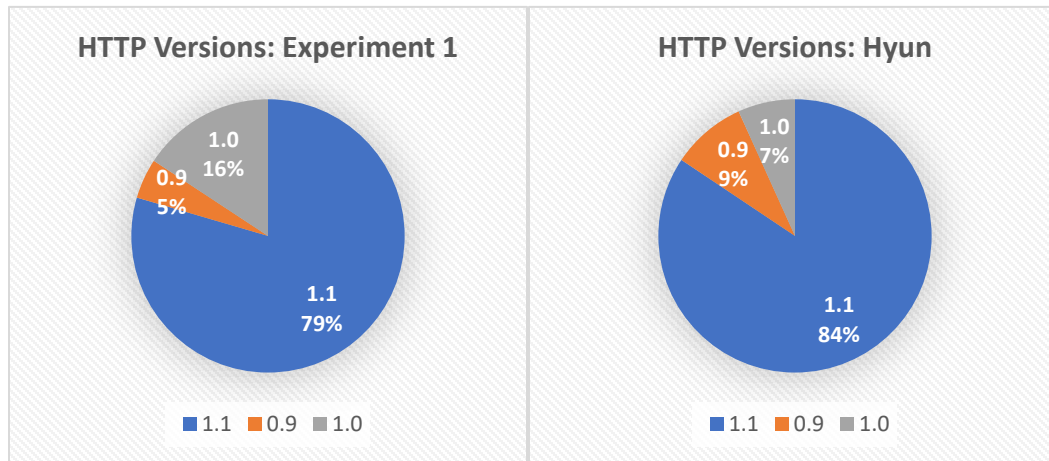


Figure 7. Experiment 1 HTTP Version Distribution

Conpot observed five HTTP methods, reporting a method of “None” for an invalid request (Figure 8). This distribution was like Hyun’s results.

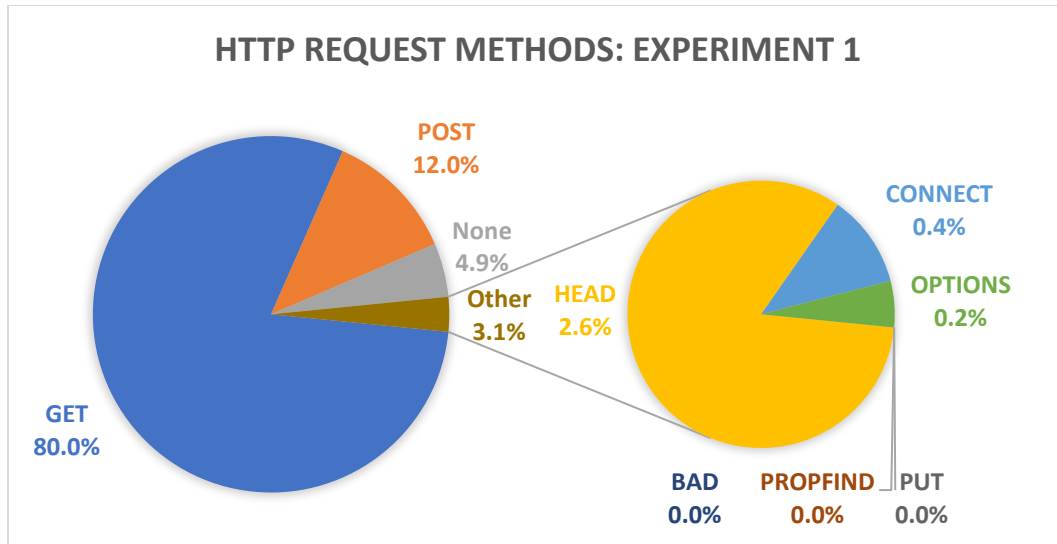


Figure 8. Experiment 1 HTTP Request Method Distribution

HTTP activity in Experiment 1 had alleged source IP addresses in 73 countries, as reported by GeoIP (MaxMind, Inc., n.d.), and the top three countries were the United States, Russia, and China. On average, our Conpot experiment saw 94 requests daily, with a single day high of 133 on 4 January 2021.

b. MODBUS

Conpot reported 1,079 MODBUS attempts, 609 connections and 470 traffic packets. A connection means a TCP connection was established on port 502, and the traffic packets counted were MODBUS-formatted. Only 20 of these had identifiable MODBUS function codes, 10 each of 0x2B (Read Device Identification) and 0x11 (Report Server ID) (Figure 9).

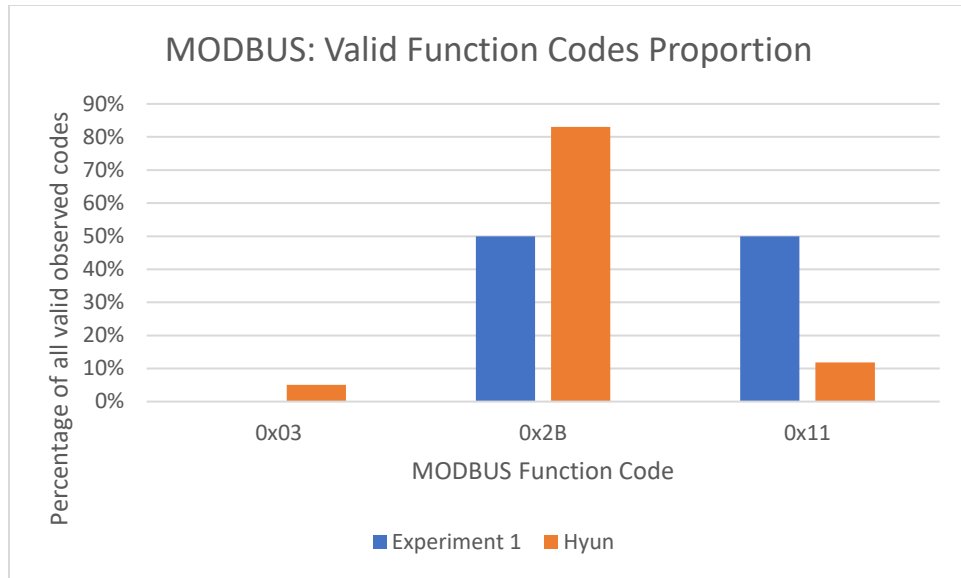


Figure 9. Experiment 1 MODBUS Function Codes

However, examination of the Conpot log and associated PCAP data showed that Conpot could not always identify a function code in a properly formatted MODBUS packet, even when correctly parsing the unit identifier and other data fields. In fact, packet capture during the same period as the Conpot log revealed 276 counts of code 0x2B and 205 counts of 0x11, more function codes than Conpot reported (Figure 10).

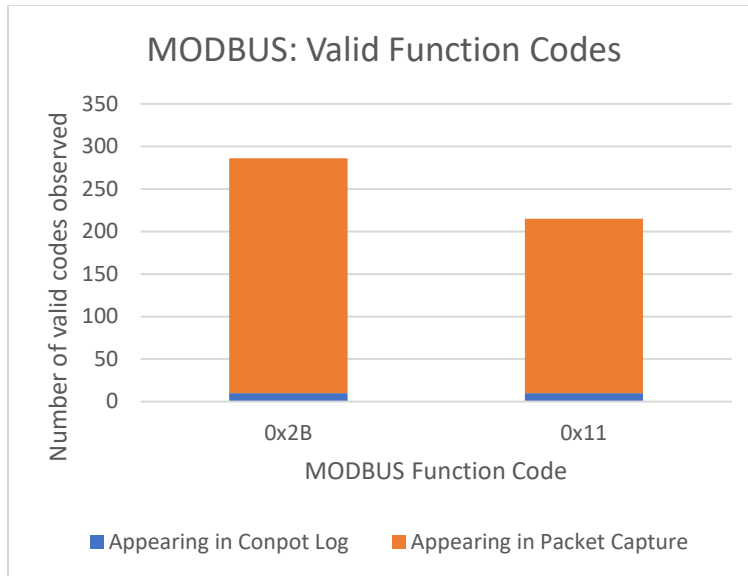


Figure 10. Experiment 1 MODBUS Valid Function Codes

Traffic varied considerably per day. Twenty-five of 30 days in the log had traffic counts of less than 5, but January 6th and 7th had counts of 68 and 303, respectively, accounting for 79% of all MODBUS traffic reported by Conpot. Three IP addresses showed a similar pattern of sending the 0x11 and 0x2B function codes to MODBUS units in increasing order from increasing source port numbers. On January 6th, an IP address in the Netherlands sent 46 such requests, and an IP address in the U.S. sent 21 similar requests. On January 7th, another IP address in the U.S. sent 300 requests in this pattern. Hyun’s experiment saw similar daily spikes.

Only five countries sent valid MODBUS traffic according to Conpot. The U.S. and the Netherlands accounted for 376 and 88 counts respectively, nearly 99% of valid traffic (Figure 11).

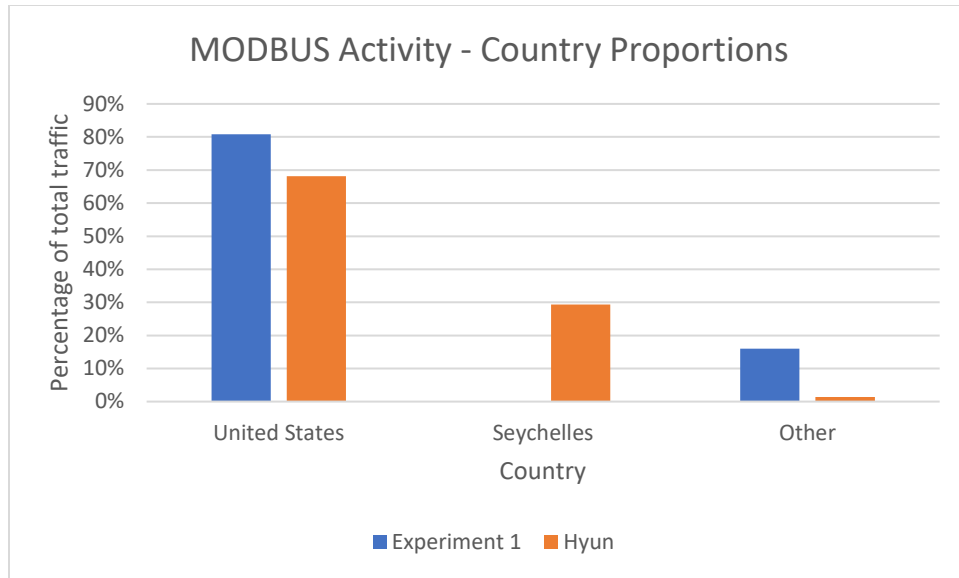


Figure 11. MODBUS Activity by Country Count

c. ETHERNET/IP

Conpot reported 79 EtherNet/IP packets of which 11 were properly formatted. Nine used the command 0x63 (List Identity), one used 0x65 (RegisterSession), and one did not use a valid command (0x01) (Figure 12). This is like Hyun's experiment, which found 20 valid commands in 154 packets.

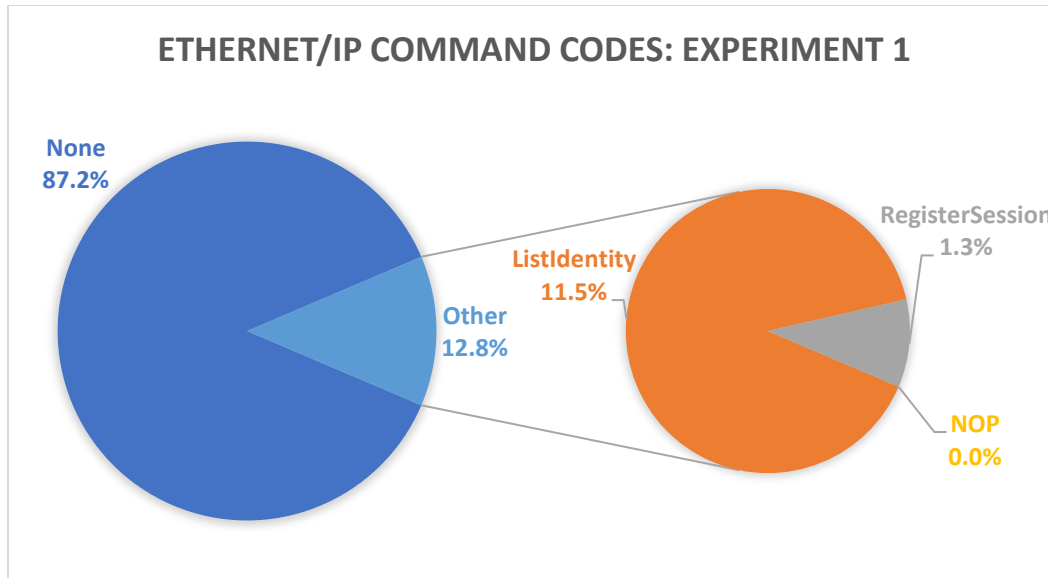


Figure 12. Experiment 1 EtherNet/IP Command Code Distribution

d. S7COMM

Conpot reported 651 S7Comm activities, including 286 connections, 157 sessions, and 208 packets (Table 2). All 208 packets were of PDU type 1 or 7, a data length of 0 or 8, and had a request ID of 0 or 1. Hyun's results showed about 150% more connections than our data, but 25% fewer packets sent and 62% fewer sessions. Since Hyun's experiment ran 134 days versus our 30 days, our experiment received a significantly higher rate of S7COMM attacks. Figure 13 shows the distribution of countries involved.

Table 2. Comparison of Experiment 1 S7COMM Activity to Hyun's Data

<i>S7COMM Activity</i>	<i>Experiment 1</i>	<i>Hyun</i>
Connections	286 (43.9%)	436 (67.6%)
Sessions	157 (24.1%)	60 (9.3%)
Packets Sent	208 (32%)	149 (23.1%)
Total	651	645

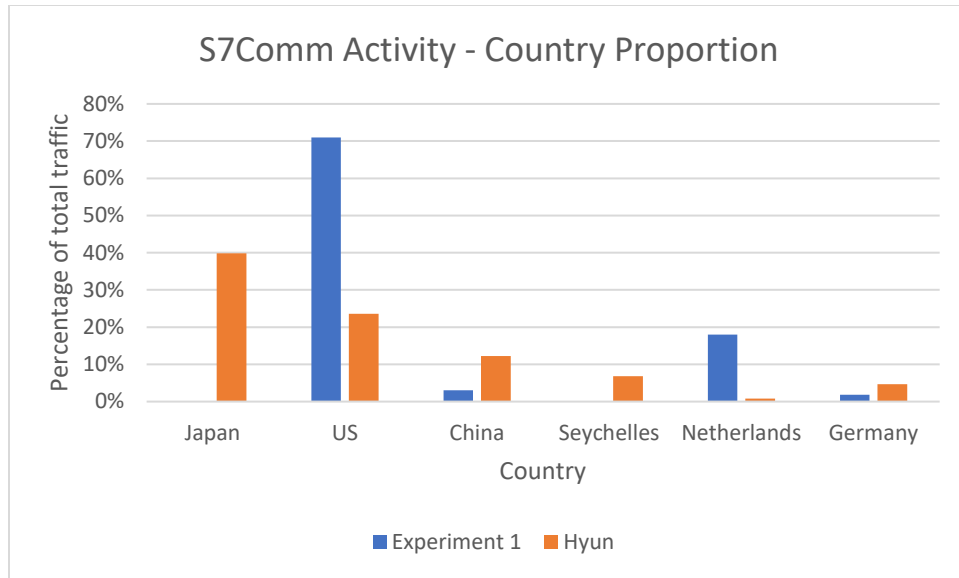


Figure 13. S7Comm Activity by Country Count

2. OVERALL STATISTICS

Over all protocols, Conpot was attacked from 74 countries, most often the United States. 76% of these focused on HTTP (Figure 15). Apart from the MODBUS activity spikes on January 6 and 7, activity levels across the experiment were relatively consistent (Figure 14).

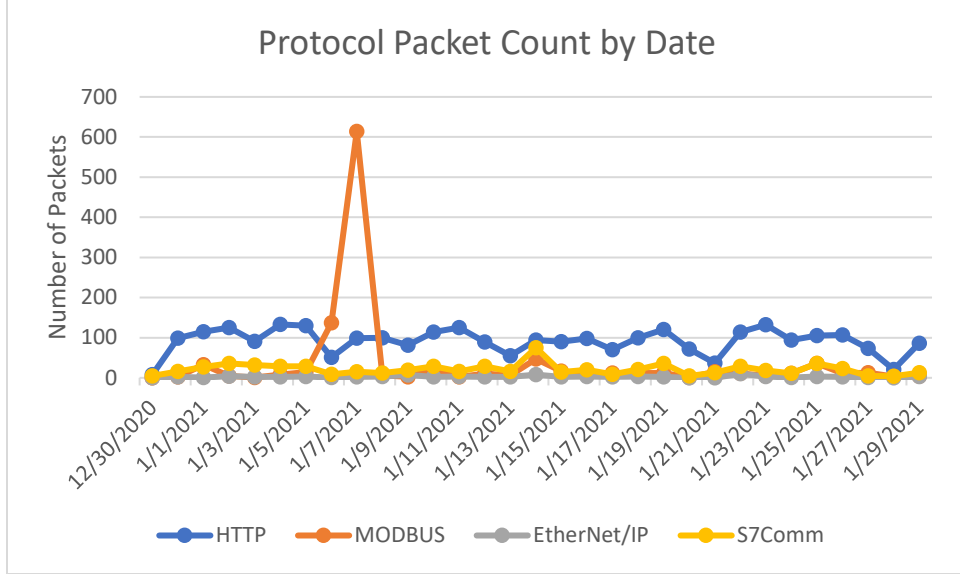


Figure 14. Experiment 1 Protocol Packet Count by Date

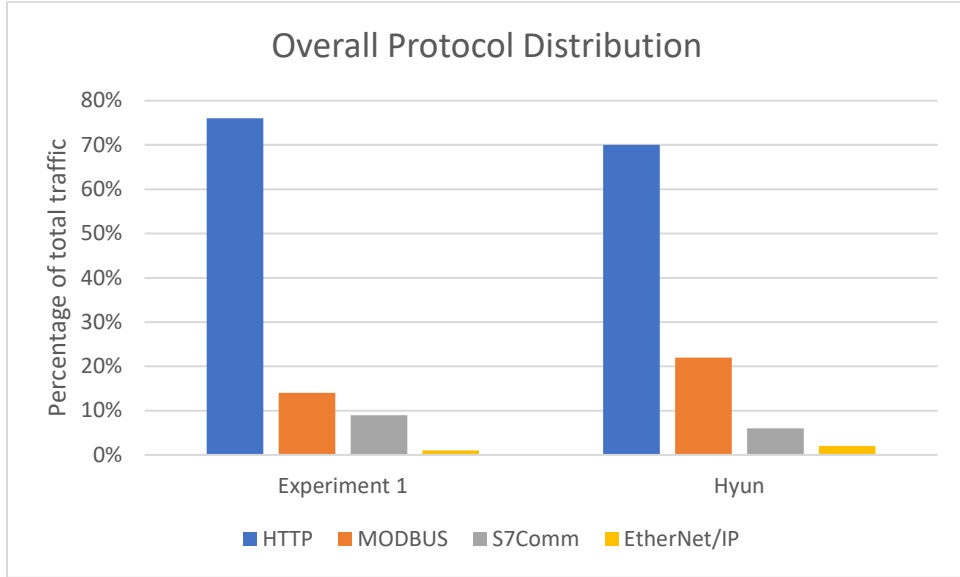


Figure 15. Experiment 1 Protocol Distribution

We used cosine similarity (Dangeti, 2017) to compare distributions between our experiment and Hyun’s experiment. Cosine similarity measures the cosine of the angle between two vectors on a scale of 0 to 1. We compared six aspects of the experiment, with vectors from 3 to 9 items (Table 3). By this metric, HTTP methods were the most similar,

while MODBUS methods were the least similar, though this could be due to the low counts. S7Comm activity also showed some differences.

Table 3. Cosine Similarity between Experiment 1 and Hyun’s Data

<i>Vector</i>	<i>Cosine similarity</i>
Activity counts of HTTP, MODBUS, EtherNet/IP, S7Comm	0.9929
HTTP Version counts of 1.1, 0.9, 1.0	0.9923
HTTP Method counts of Bad, CONNECT, GET, HEAD, None, OPTIONS, POST, PROPFIND, PUT	0.9974
MODBUS Function Code counts of 0x03, 0x2B, 0x11	0.7985
EtherNet/IP Command counts of NOP, ListIdentity, None, RegisterSession]	0.995
S7Comm Activity counts of Connections, Sessions, Packets	0.918

Overall, Experiment 1’s Conpot (cloud-deployed within T-Pot using standard Conpot configuration) received more traffic than Hyun’s Conpot (locally-deployed using standard Conpot configuration). Our experiment had 340% more HTTP traffic, 200% more MODBUS traffic, 220% more EtherNet/IP traffic, and 450% more S7COMM traffic per day. However, Hyun’s results showed more HTTP methods, MODBUS function codes, and EtherNet/IP valid commands. Also, HTTP traffic was a larger proportion of our received attacks. This suggests that a cloud-based HTTP server is a more attractive target than a locally-deployed server.

B. EXPERIMENTS 2 AND 3

Data was analyzed for Experiments 2 and 3 from February 11, 2021, to May 11, 2021.

1. HTTP Data and Results

a. Overall Activities

Experiment 2 received 28,042 HTTP requests and Experiment 3 received 10,619. However, 13,199 (47%) of Experiment 2’s requests were from a single IP on February 27th,

making it a large outlier. A breakdown of HTTP Versions across experiments showed similar results (Table 4).

Table 4. Experiments 2 and 3 HTTP Versions

	<i>Experiment 2 without the outlier</i>	<i>Experiment 3</i>
<i>Total HTTP Requests</i>	14843	10619
<i>HTTP 1.1</i>	14035 (94.56%)	10040 (94.55%)
<i>HTTP 1.0</i>	791 (5.33%)	562 (5.29%)
<i>HTTP 0.9</i>	3 (0.02%)	2 (0.02%)
<i>None</i>	14 (0.09%)	15 (0.14%)

Experiments 2 and 3 both received HTTP requests that used the CONNECT, GET, HEAD, OPTIONS, and POST methods (Table 5), and Experiment 3 received three PROPFIND requests.

Table 5. Experiments 2 and 3 HTTP Methods

	<i>Experiment 2 without the outlier</i>	<i>Experiment 3</i>
<i>Total HTTP Requests</i>	14843	10619
<i>GET</i>	11372 (76.62%)	8976 (84.53%)
<i>POST</i>	3295 (22.2%)	1436 (13.52%)
<i>HEAD</i>	122 (0.82%)	129 (1.21%)
<i>CONNECT</i>	41 (0.28%)	48 (0.45%)
<i>OPTIONS</i>	13 (0.09%)	27 (0.25%)
<i>PROPFIND</i>	0 (0%)	3 (0.03%)

Experiments 2 and 3 saw 94 and 101 alleged countries respectively. The top alleged source for both experiments was the United States, consistent with Chong and Koh's results (Table 6). If the Experiment 2 outlier was kept in the data, the most common source country would be Germany.

Table 6. Experiments 2 and 3 Top Countries Comparison

	<i>Experiment 2 without the outlier</i>	<i>Experiment 3</i>	<i>Chong and Koh</i>
<i>United States</i>	28%	34%	28%
<i>China</i>	22%	17%	7%
<i>Russia</i>	14%	14%	-
<i>Malta</i>	0%	0%	19%

On average, Experiments 2 and 3 saw 167 and 119 daily requests respectively. The daily high for Experiment 2 was 1048 requests and for Experiment 3 was 533 requests, significantly higher than Chong and Koh's reported daily high of 73. Traffic spikes were similar between Experiments 2 and 3 (Figure 16).

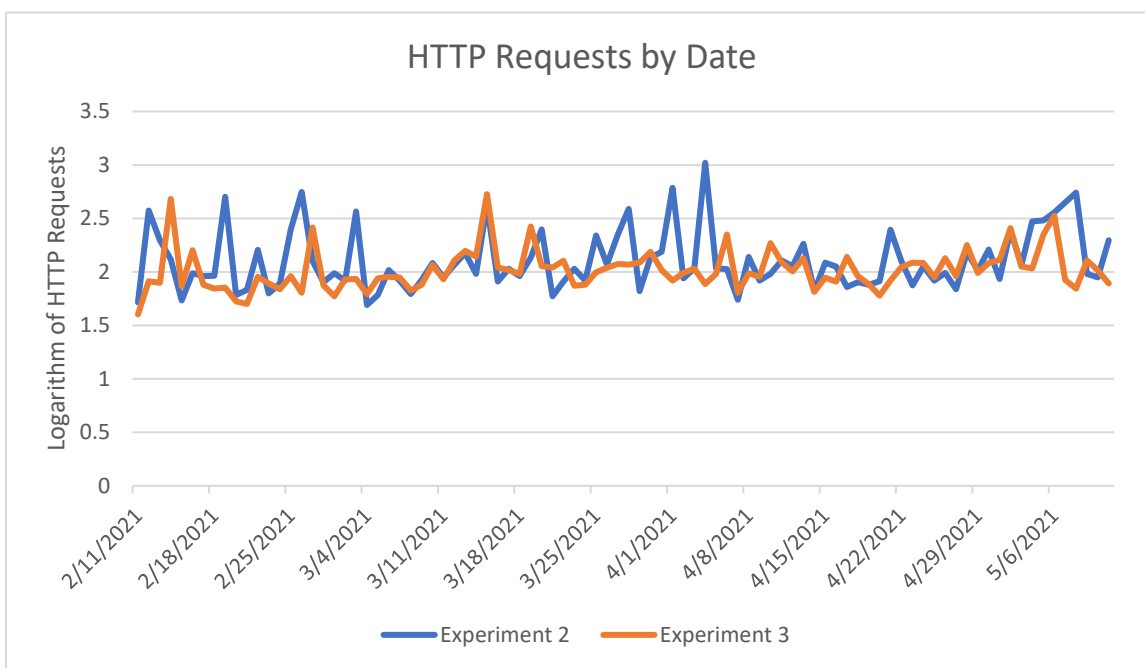


Figure 16. Experiments 2 and 3 HTTP Requests by Date (Logarithmic Scale)

Experiments 2 and 3 saw 2,644 and 2,615 different alleged IP addresses; 886 addresses accessed both experiments. Many addresses were on similar subnets, suggesting the same attacking entity. We also found that a Chinese IP address was more likely to be

unique to an experiment than a Russian IP address (Table 7), demonstrated by the number of unique IP addresses geolocating to China (29% and 34%) and Russia (2% and 4%).

Table 7. Experiment 2 and 3 Alleged Countries

	<i>Exp. 2 Overall</i>	<i>Exp. 3 Overall</i>	<i>Exp. 2 Common</i>	<i>Exp. 3 Common</i>	<i>Exp. 2 Unique</i>	<i>Exp. 3 Unique</i>
<i>United States</i>	28%	34%	38%	36%	21.3%	31.38%
<i>China</i>	22%	17%	2%	2%	34.9%	29.11%
<i>Russia</i>	14%	14%	29%	28%	4.17%	2.18%

b. Types of Activity

Experiment 2 received 2,237 unique HTTP path requests, while Experiment 3 received 1,765. The most common path requested was the root directory, or “/,” accounting for 20.88% and 26.46% respectively in the two experiments. This is consistent with Chong and Koh’s result of 24%. The second most common was “/.env” accounting for between 4.5% of requests in Experiment 2 and 7.5% of requests in Experiment 3.

Many HTTP path strings for both experiments contained the string “php,” 41.1% for Experiment 2 and 24.3% for Experiment 3. PHP services appeared to be attractive to attackers (Shiflett, 2006). Table 8 shows a breakdown of the observed paths. The “Crawler” category is defined as requests for “robots.txt,” “favicon.ico,” and “.env.” The “Files” category includes paths with .txt, .sh, .zip, and .tar extensions. The “Other .env” category includes paths that contain .env, except for root-level /.env. “Other” includes paths not easily categorized or one-off requests.

Table 8. Experiment 2 and 3 Top 11 Overall Count of HTTP Paths

<i>Path</i>	<i>Experiment 2</i>	<i>Experiment 3</i>
/	3099 (20.9%)	2810 (26.5%)
/_ignition/execution-solution	242 (1.6%)	229 (2.2%)
/manager/html	171 (1.2%)	145 (1.4%)
/config/getuser?index=0	167 (1.1%)	159 (1.5%)
/login	166 (1.1%)	134 (1.3%)
/Jenkins/login	127 (0.85%)	105 (1.0%)
index.html	2 (0.01%)	1 (0.01%)
<i>Category: PHP</i>	6097 (41.1%)	2578 (24.3%)
<i>Category: SQL</i>	182 (1.2%)	244 (2.3%)
<i>Category: Crawler</i>	1065 (7.2%)	1188 (11.2%)
<i>Category: .xml</i>	351 (2.4%)	336 (3.2%)
<i>Category: Shell commands</i>	209 (1.4%)	208 (2.0%)
<i>Category: JSON</i>	270 (4.3%)	268 (9.5%)
<i>Category: Top-level folders</i>	773 (5.2%)	655 (6.2%)
<i>Category: Files</i>	563 (3.8%)	474 (4.5%)
<i>Category: JavaScript</i>	211 (1.4%)	162 (1.5%)
<i>Category: Other .env</i>	161 (1.1%)	182 (1.7%)
<i>Other</i>	985 (6.6%)	740 (7.0%)

Most HTTP user-agents in Experiments 2 and 3 were some variation of “Mozilla/5.0,” 79.4% and 78.9% of requests. Most Web browsers start their user-agent string with Mozilla (Zhu & Desai, 2015) to receive Web pages with advanced features.

2. Overall Statistics

Overall, Experiment 3 received less traffic than Experiment 2. Experiment 3 did receive similar traffic to Experiment 2 in attack types, HTTP commands, and source countries. This suggests that use of T-Pot does not significantly affect the type of attacks received.

C. EXPERIMENTS 4–7

Experiments 4–7 ran from May 1, 2021, to June 1, 2021. Experiments 6 and 7 suffered an unexpected HTTP error and only collected seven days of HTTP traffic. As mentioned in Section IV.C, Experiment 7’s IP address was found to be registered to a Korean-based domain name, and consequently, collected some HTTP data from users

attempting to access it. Table 9 shows overall statistics for the four experiments. Generally speaking, the T-Pot deployments in Experiments 4 and 5 were accessed by more countries than the standalone honeypots, Experiment 7 received about four times the average HTTP traffic of the other three experiments, and IEC 104 message numbers were low throughout.

Table 9. Overall Traffic Statistics for Experiments 4–7

	<i>Experiment 4</i> <i>(US, T-Pot)</i>	<i>Experiment 5</i> <i>(Asia, T-Pot)</i>	<i>Experiment 6</i> <i>(US)</i>	<i>Experiment 7</i> <i>(Asia)</i>
<i>Number of unique countries</i>	75	81	44	74
<i>Number of unique IP addresses</i>	834	1034	305	1509
<i>Total number of requests</i>	5813	5951	879	8343
<i>Total HTTP requests</i>	5673	5830	8343	879
<i>Mean HTTP requests per day</i>	189.1	182.2	97.7	695.2
<i>Min HTTP requests per day</i>	50	66	15	1
<i>Max HTTP requests per day</i>	2480	1283	163	1883
<i>Total IEC 104 messages</i>	140	121	121	127
<i>Total IEC 104 malformed messages</i>	104	95	105	110
<i>Total IEC 104 valid messages</i>	36	26	16	17
<i>Mean IEC 104 messages per day</i>	10.8	8.1	7.6	7.5
<i>Min IEC 104 messages per day</i>	1	1	1	1
<i>Max IEC 104 messages per day</i>	85	88	83	91
<i>Mean valid IEC 104 messages per day</i>	3.6	2.2	1.3	1.7
<i>Min valid IEC 104 messages per day</i>	1	1	1	1
<i>Max valid IEC 104 messages per day</i>	9	3	2	3

1. Session Data

Dougherty (Dougherty, 2020) and Bieker and Pilkington (Bieker & Pilkington, 2020) defined a session as all packets exchanged by a socket pair on a day, and we follow this definition. We counted IP addresses that established only a single HTTP session, multiple HTTP sessions, only a single IEC 104 session, multiple IEC 104 sessions, and both HTTP and IEC 104 sessions (Table 10). To better compare to the limited HTTP dataset in Experiments 6 and 7, additional columns for Experiments 4 and 5 are given that match the seven-day data length.

Table 10. Experiments 4–7 IP Address Session Data

<i>Sessions Established with Unique IP Addresses</i>	<i>Exp. 4 (U.S., T-Pot)</i>	<i>Exp. 5 (Asia, T-Pot)</i>	<i>Exp. 4 (Only first 7 days of HTTP)</i>	<i>Exp. 5 (Only first 7 days of HTTP)</i>	<i>Exp. 6 (U.S.)</i>	<i>Exp. 7 (Asia)</i>
<i>Number of unique IP addresses</i>	834	1034	320	353	305	1509
<i>Established Single HTTP-only session</i>	516 (61.8%)	632 (61%)	192 (60%)	211 (60%)	192 (63%)	457 (30%)
<i>Established Multiple HTTP-only sessions</i>	298 (35.7%)	387 (37%)	108 (34%)	127 (36%)	92 (30%)	1030 (68%)
<i>Established Single IEC 104-only session</i>	10 (1.2%)	5 (0.5%)	10 (31%)	5 (1.5%)	7 (2%)	6 (0.4%)
<i>Established Multiple IEC 104-only sessions</i>	7 (0.8%)	6 (0.6%)	7 (2%)	6 (1.6%)	11 (4%)	14 (0.9%)
<i>Established HTTP and IEC 104 sessions</i>	3 (0.4%)	4 (0.4%)	3 (0.9%)	4 (1.1%)	2 (.6%)	2 (0.1%)

The distribution of HTTP traffic by country was similar across the four experiments (Table 11), although proportions of the top countries did vary. This was in contrast to

Bieker and Pilkington’s results, which had different top countries between their U.S. and Asia experiments

Table 11. Experiments 4–7 HTTP Country Data

	<i>Exp. 4</i>	<i>Exp. 5</i>	<i>Exp. 4 (7 days)</i>	<i>Exp. 5 (7 days)</i>	<i>Exp. 6</i>	<i>Exp. 7</i>
1	US (31.4%)	US (30.4%)	US (31.6%)	US (29.5%)	US (30.3%)	US (35%)
2	China (11.3%)	China (12.3%)	China (12.5%)	China (17.8%)	China (15.8%)	China (6.4%)
3	Germany (5.8%)	Germany (4.5%)	Germany (6.6%)	Germany (6.2%)	France (6.6%)	Russia (5.6%)
4	India (4.3%)	U.K. (4.3%)	Netherlands (5.3%)	Netherlands (5.4%)	Germany (5.9%)	Netherlands (5.2%)
5	Russia (4.1%)	Russia (4.0%)	India (4.4%)	UK (3.7%)	Russia (4.6%)	Germany (4.9%)
6	Netherlands (4.1%)	India (3.9%)	France (4.4%)	France (3.4%)	Netherlands (4.6%)	Singapore (4.4%)
7	U.K. (3.6%)	Netherlands (3.7%)	Russia (3.8%)	Russia (3.4%)	India (4.3%)	Canada (2.9%)
8	France (3.1%)	France (3.3%)	UK (3.8%)_	Singapore (2.8%)	Brazil (3.0%)	France (2.7%)
9	Brazil (2.9%)	Singapore (2.8%)	Singapore (2.8%)	Brazil (2.3%)	U.K. (3.0%)	India (2.3%)
10	Other (26.6%)	Other (25.4%)	Other (19.7%)	Other (20.1%)	Other (18%)	Other (26.9%)

HTTP path requests in Experiments 4–7 followed a similar pattern as Experiments 1–3, with large shares relating to either PHP or SQL, as well as the more regular root and index.html requests. Table 12 shows data on specific paths as well as defined categories. The “Crawler” category was defined as requests for “robots.txt,” “favicon.ico,” and “.env.” The “Files” category includes paths with .txt, .sh, .zip, and .tar extensions. The “.env” category includes paths with the string “.env” except for root-level “/.env.” “Other” includes paths not easily categorized or one-off requests, broadly like those seen in Experiments 2–3. For Experiment 7, “DNS redirect” meant paths including the registered domain discussed in Section IV.C as well as all paths beginning with “content/,” which were absent in Experiments 4–6 and were tries to reach content on the registered domain.

Table 12. Experiments 4–7 HTTP Path Data

	<i>Exp. 4</i>	<i>Exp. 5</i>	<i>Exp. 6</i>	<i>Exp. 7</i>
<i>HTTP data</i>	31 days	31 days	7 days	7 days
/	948 (17.9%)	1028 (19.1%)	262 (32.5%)	1880 (23.3%)
/_ignition/execution-solution	46 (0.9%)	77 (1.4%)	14 (1.7%)	16 (0.2%)
/manager/html	15 (0.3%)	27 (0.5%)	3 (0.4%)	5 (0.1%)
/config/getuser?index=0	31 (0.6%)	22 (0.4%)	6 (0.7%)	6 (0.1%)
/login	11 (0.2%)	21 (0.4%)	2 (0.2%)	4 (0.1%)
/Jenkins/login	8 (0.2%)	18 (0.3%)	1 (0.1%)	4 (0.1%)
Index.html	389 (7.3%)	432 (8.0%)	102 (12.6%)	1781 (22.0%)
<i>Category: PHP</i>	2833 (53.4%)	2474 (46%)	110 (13.6%)	1100 (13.6%)
<i>Category: SQL</i>	5 (0.1%)	50 (0.9%)	1 (0.1%)	20 (0.3%)
<i>Category: Crawler</i>	213 (4.0%)	295 (5.5%)	67 (8.3%)	268 (3.3%)
<i>Category: .xml</i>	177 (3.3%)	113 (2.1%)	31 (3.8%)	18 (0.22%)
<i>Category: Shell commands</i>	58 (1.1%)	103 (1.9%)	8 (0.99%)	36 (0.45%)
<i>Category: JSON</i>	71 (1.3%)	80 (3.2%)	22 (2.7%)	23 (2.0%)
<i>Category: Top-level folders</i>	162 (3.0%)	224 (4.2%)	100 (12.4%)	55 (0.7%)
<i>Category: Files</i>	77 (1.5%)	124 (2.3%)	19 (2.4%)	49 (0.6%)
<i>Category: JavaScript</i>	81 (1.5%)	38 (0.7%)	0	9 (0.1%)
<i>Category: Other .env</i>	13 (0.2%)	82 (1.5%)	6 (0.7%)	0
<i>Other</i>	172 (3.2%)	162 (3.0%)	54 (6.7%)	252 (1.27%)
<i>DNS redirect</i>	0	0	0	2615 (32.35%)
<i>Total</i>	5305	5370	807	8081

Across all four experiments, IEC 104 traffic was lower than HTTP traffic. Experiments 4 and 5 running within T-Pot, received more valid IEC 104 messages than Experiments 6 and 7 (Figure 17), while Experiments 6 and 7 attracted more attackers as measured by IP addresses (Figure 18). Experiments 4 and 5 were more consistent with Bieker and Pilkington’s results for reasons that were unclear.

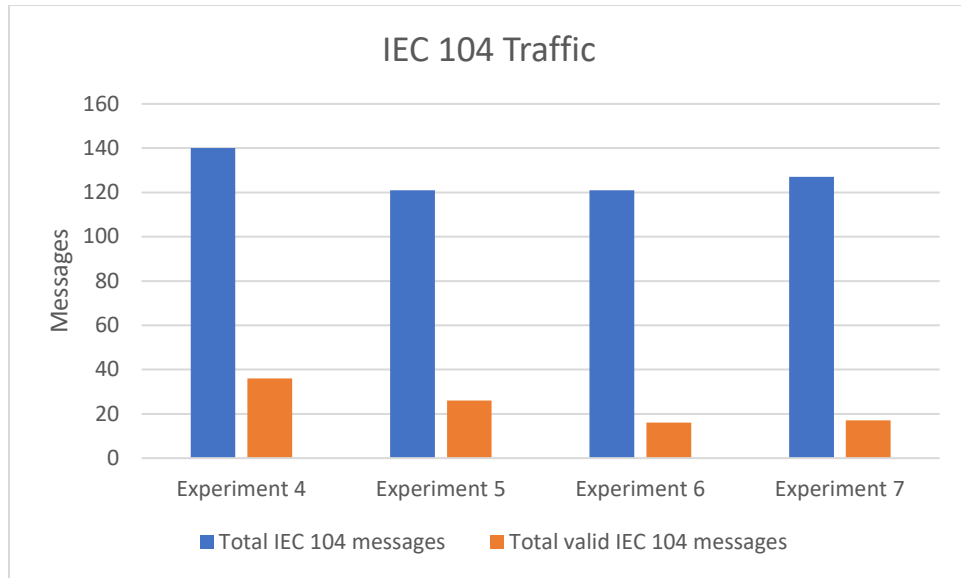


Figure 17. Experiments 4–7 IEC 104 Traffic

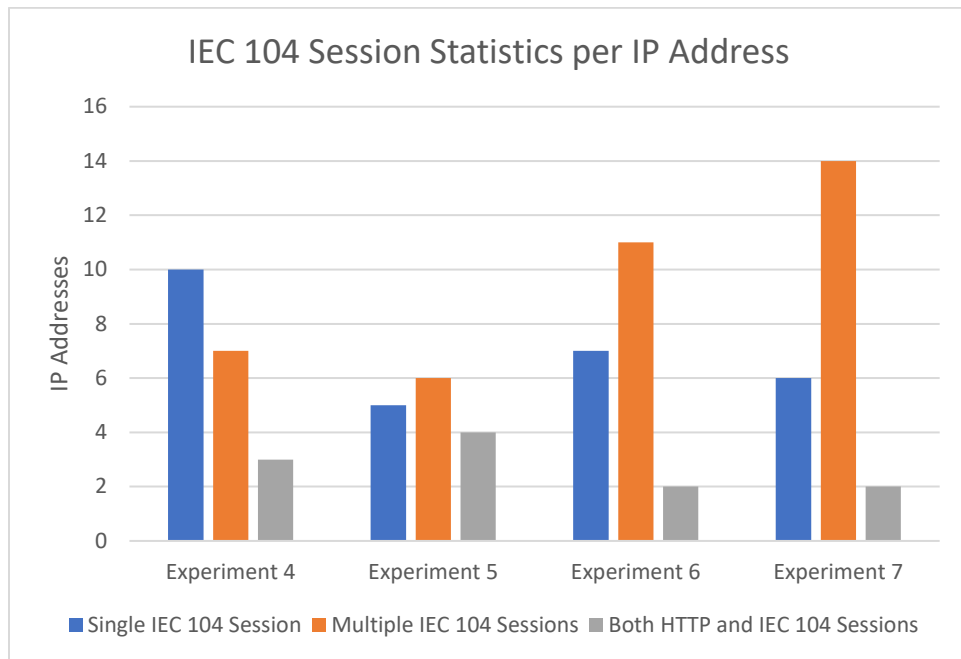


Figure 18. Experiments 4–7 IEC 104 Session Statistics

Table 13 shows the breakdown of valid IEC 104 messages in Experiments 4–7. Overall, Experiments 4 and 5 had more valid IEC 104 traffic and more diversity in traffic, receiving a mix of U-Format and I-Format IEC 104 frames. In contrast, Experiments 6 and

7 mostly received U-Format frames. All experiments received similar numbers of error frames, which were packets received on port 2404 in incorrect IEC 104 format. Erroneous IEC 104 traffic (Table 14) was less consistent day-to-day than valid traffic. Many of its error frames, rather than being malformed IEC 104 data, contained traffic for HTTP, FTP, or other protocols. Figure 19 shows an example error frame from Experiment 5.

Table 13. Experiments 4–7 IEC 104 Methods

Experiment	U-Format Frames	I-Format Frames	Error Frames	Total IEC 104 Traffic
<i>Experiment 4</i>	26	10	104	140
<i>Experiment 5</i>	20	6	95	121
<i>Experiment 6</i>	16	0	105	121
<i>Experiment 7</i>	16	1	110	127

Table 14. Experiments 4–7 Daily IEC 104 Traffic

<i>Experiment</i>	<i>Total IEC 104 Traffic</i>				<i>Valid IEC 104 Traffic</i>			
	<i>Mean</i>	<i>Min</i>	<i>Max</i>	<i>Std. Dev</i>	<i>Mean</i>	<i>Min</i>	<i>Max</i>	<i>Std. Dev</i>
<i>Experiment 4</i>	10.769	1	85	21.588	3.6	1	9	2.54
<i>Experiment 5</i>	8.066	1	88	21.446	2.166	1	3	0.986
<i>Experiment 6</i>	7.5625	1	83	19.609	1.33	1	2	0.471
<i>Experiment 7</i>	7.47	1	91	20.915	1.7	1	3	0.781

```

0000  ee 7d 86 b8 54 ed fe 00 00 00 01 01 08 00 45 08  }...T... ..E..
0010  01 03 50 e5 40 00 2c 06 3e 42 b9 b4 8f 94 80 c7  ..P@.,. >B.....
0020  f4 b5 95 46 09 64 10 f2 c9 0a af 5c b2 40 80 18  ...F.d... ..\..@..
0030  00 e5 db 90 00 00 01 01 08 0a 14 e0 cc 5e fd a8  .....^...
0040  f8 68 47 45 54 20 2f 20 48 54 54 50 2f 31 2e 31  .hGET / HTTP/1.1
0050  0d 0a 48 6f 73 74 3a 20 31 32 38 2e 31 39 39 2e  .Host: 128.199.
0060  32 34 34 2e 31 38 31 3a 32 34 30 34 0d 0a 55 73  244.181: 2404..Us
0070  65 72 2d 41 67 65 6e 74 3a 20 4d 6f 7a 69 6c 6c  er-Agent : Mozill
0080  61 2f 35 2e 30 20 28 57 69 6e 64 6f 77 73 20 4e  a/5.0 (W indows N
0090  54 20 31 30 2e 30 3b 20 57 4f 57 36 34 29 20 41  T 10.0; WOW64) A
00a0  70 70 6c 65 57 65 62 4b 69 74 2f 35 33 37 2e 33  ppleWebK it/537.3
00b0  36 20 28 4b 48 54 4d 4c 2c 20 6c 69 6b 65 20 47  6 (KHTML , like G
00c0  65 63 6b 6f 29 20 43 68 72 6f 6d 65 2f 36 36 2e  ecko) Ch rome/66.
00d0  30 2e 33 33 35 39 2e 31 31 37 20 53 61 66 61 72  0.3359.1 17 Safar
00e0  69 2f 35 33 37 2e 33 36 20 0d 0a 41 63 63 65 70  i/537.36 ..Accep
00f0  74 3a 20 2a 2f 2a 0d 0a 41 63 63 65 70 74 2d 45  t: */*.. Accept-E
0100  6e 63 6f 64 69 6e 67 3a 20 67 7a 69 70 0d 0a 0d  ncoding: gzip...
0110  0a

```

Figure 19. Example Hex Dump of IEC 104 Error Frame Showing Encapsulated HTTP Packet

Unlike HTTP traffic, the distribution of IEC 104 traffic by country was inconsistent across the experiments (Table 15). For example, IEC traffic from the U.S. ranged from 72.7% in Experiment 6 to 8.3% in Experiment 5. This inconsistency between geographically different experiments, however, is consistent with Bieker and Pilkington’s results.

Table 15. Experiments 4–7 IEC 104 Alleged Country Data

	<i>Experiment 4 (US T-Pot)</i>	<i>Experiment 5 (Asia T-Pot)</i>	<i>Experiment 6 (US)</i>	<i>Experiment 7 (Asia)</i>
1	Netherlands (34.3%)	Singapore (32.2%)	US (72.7%)	US (36.2%)
2	India (22.1%)	Germany (25.6%)	Canada (9.9%)	Netherlands (35.4%)
3	US (17.14%)	UK (9.9%)	Portugal (4.1%)	China (6.3%)
4	France (8.6%)	US (8.3%)	China (4.1%)	Vietnam (6.3%)
5	Germany (7.1%)	France (8.3%)	Netherlands (3.3%)	Portugal (3.9%)
6	China (5.7%)	China (5.0%)	France (3.3%)	France (3.1%)
7	Portugal (3.6%)	Portugal (4.1%)	UK (1.7%)	Japan (3.1%)
8	Russia (1.43%)	Netherlands (3.3%)	Russia (0.8%)	Russia (2.4%)
9		India (3.3%)		

2. SIMILARITIES BETWEEN EXPERIMENT TRAFFIC

We used cosine similarity to compare the results of our experiments and Bieker and Pilkington’s data. Our first comparison (Table 16) used a vector of four items: average weekly counts of HTTP GET commands, HTTP POST commands, other HTTP methods, and all IEC 104 traffic. The raw data for this comparison is in Appendix B. The vectors for Experiments 6 and 7 were affected by the missing HTTP data (after the unexpected HTTP error was seen) and DNS registration, so the more useful comparison is between Experiment 4 and Bieker and Pilkington’s Experiment 2 (both deployed in the US), and Experiment 5 and their Experiment 3 (both deployed in Asia). To assess how similar these experiments were, we took the cosine similarity of the average vectors of each experiment and divided it by the averages of the week-to-week cosine similarities. The absolute value of the logarithm of this value is a rough measure of significance, which could be refined using a table of the F distribution. Table 17 demonstrates the experiments are not significantly different from each other, using two standard deviations from the mean as the minimum for significance.

Table 16. Experiments 4–7 Cosine Similarity for All Traffic

<i>Experiment</i>	<i>Average cosine similarity of total traffic</i>
<i>Experiment 4 (with T-Pot)</i>	0.9833
<i>Experiment 5 (with T-Pot)</i>	0.8047
<i>Experiment 4 (7 days of HTTP)</i>	0.9350
<i>Experiment 5 (7 days of HTTP)</i>	0.7521
<i>Experiment 6</i>	0.7998
<i>Experiment 7</i>	0.6718
<i>Bieker & Pilkington’s Experiment 2</i>	0.774
<i>Bieker & Pilkington’s Experiment 3</i>	0.994

Table 17. Experiments 4–5 Measure of Significance of Total Traffic Comparisons

	<i>Experiment 4 versus Bieker & Pilkington’s Experiment 2</i>	<i>Experiment 5 versus Bieker & Pilkington’s Experiment 3</i>
<i>Measure of significance of total traffic</i>	$\log(0.992431) = 0.0033$	$\log(0.9873) = 0.0056$

Our second comparison examined IEC 104 traffic only. It used a vector of the average weekly counts of U-format frames, I-format frames, and error frames (Table 18). The measures of significance (Table 19) again demonstrates that geographically similar experiments have similar results.

Table 18. Experiments 4–7 Cosine Similarity for IEC 104 Traffic

<i>Experiment</i>	<i>Average cosine similarity of IEC 104 traffic</i>
Experiment 4	0.8304
Experiment 5	0.5048
Experiment 6	0.7167
Experiment 7	0.6020

Table 19. Experiments 4–7 Measure of Significance of IEC 104 Traffic

	<i>Experiment 4 versus Experiment 6</i>	<i>Experiment 5 versus Experiment 7</i>
<i>Measure of significance of IEC 104 Traffic</i>	$\log(1.2804) = 0.1073$	$\log(1.7976) = 0.2547$

Overall, the computed measures of significance do not demonstrate a significant difference between GridPots deployed within T-Pot (Experiments 4 and 5), and geographically-identical standalone GridPots (Experiments 6 and 7, and Bieker & Pilkington’s experiments).

3. BEHAVIORAL ANALYSIS

We performed additional analysis on IP addresses that sent IEC 104 traffic. In each experiment, two of eight IP addresses sent the same packet to port 2404, which registered as an error frame (Figure 20). The two IP addresses in each experiment did not appear related to each other, as they were on different subnets. Instead of properly formatted IEC 104 data, this packet held Session Initiation Protocol (SIP) traffic not intended for an IEC 104 device.

```

0040 94 17 4f 50 54 49 4f 4e 53 20 73 69 70 3a 6e 6d 00OPTION S sip:nm
0050 20 53 49 50 2f 32 2e 30 0d 0a 56 69 61 3a 20 53 SIP/2.0 Via: S
0060 49 50 2f 32 2e 30 2f 54 43 50 20 6e 6d 3b 62 72 IP/2.0/T CP nm;br
0070 61 6e 63 68 3d 66 6f 6f 0d 0a 46 72 6f 6d 3a 20 anch=foo From:
0080 3c 73 69 70 3a 6e 6d 40 6e 6d 3e 3b 74 61 67 3d <sip:nm@ nm>;tag=
0090 72 6f 6f 74 0d 0a 54 6f 3a 20 3c 73 69 70 3a 6e root To : <sip:n
00a0 6d 32 40 6e 6d 32 3e 0d 0a 43 61 6c 6c 2d 49 44 m2@nm2> Call-ID
00b0 3a 20 35 30 30 30 30 0d 0a 43 53 65 71 3a 20 34 : 50000 CSeq: 4
00c0 32 20 4f 50 54 49 4f 4e 53 0d 0a 4d 61 78 2d 46 2 OPTION S Max-F
00d0 6f 72 77 61 72 64 73 3a 20 37 30 0d 0a 43 6f 6e orwards: 70 Con
00e0 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 30 0d 0a tent-Len gth: 0
00f0 43 6f 6e 74 61 63 74 3a 20 3c 73 69 70 3a 6e 6d Contact: <sip:nm
0100 40 6e 6d 3e 0d 0a 41 63 63 65 70 74 3a 20 61 70 @nm> Ac cept: ap
0110 70 6c 69 63 61 74 69 6f 6e 2f 73 64 70 0d 0a 0d plicatio n/sdp
0120 0a

```

Figure 20. Example of IEC104 Error Packet Containing SIP Data

Except for one IP address in Experiment 5, all IP addresses that sent an I-Format also sent a U-Format frame. All experiments also had IP addresses that sent U-Format frames but no I-Format frames.

Twelve addresses of a subnet sent traffic to all four experiments in various ways (Table 20). Eleven addresses on the subnet sent IEC 104 traffic to some but not all experiments. One address only sent HTTP traffic but sent it to every experiment. Its HTTP requests were “GET /” and “GET /index.html” requests, which suggests a crawler and is not itself suspicious. Based on the GeoIP database, the IP range geolocated to mainland France. The days of activity were Mondays, Wednesdays, and Fridays, and the hours of activity were 0200–1800 Pacific Time, corresponding to 1100–0300 Central European Time. Interestingly, this leaves an eight-hour gap of inactivity. If this traffic required

human interaction, their waking hours are 0600–2200, then the originating time zone may actually be GMT-4 in Brazil.

Table 20. Experiments 4–7 Traffic from Specific Subnet

<i>51.254.49.0/20</i>	<i>Experiment 4</i>	<i>Experiment 5</i>	<i>Experiment 6</i>	<i>Experiment 7</i>
<i>.49.96</i>				1 U-Format
<i>.49.97</i>	2 U-Format, 1 I-Format			
<i>.49.98</i>		2 U-Format, 1 I-Format		
<i>.49.99</i>		3 U-Format, 1 I-Format	1 U-Format	
<i>.49.100</i>	2 U-Format, 1 I-Format			
<i>.49.101</i>	2 U-Format, 1 I-Format			
<i>.49.102</i>			1 U-Format	
<i>.49.103</i>		2 U-Format, 1 I-Format		1 U-Format
<i>.49.104</i>				1 U-Format
<i>.49.109</i>	2 U-Format, 1 I-Format		1 U-Format	
<i>.49.110</i>			1 U-Format	
<i>.59.113</i>	3 sets of 2 HTTP requests	3 sets of 2 HTTP requests	1 set of 2 HTTP requests	1 set of 2 HTTP requests
<i>TOTAL:</i>	8 U-Format, 4 I-Format, 6 HTTP	7 U-Format, 3 I-Format, 6 HTTP	4 U-Format, 2 HTTP	4 U-Format, 2 HTTP

4. Shodan

Only one IP address, in Experiment 4, was continuously scanned by Shodan; this scan occurred six days after the droplet was exposed to the Internet. We started a scan on the principal honeypot addresses in all four experiments, and subsequently, a search of these addresses showed not only the results from our scan, but also historical searches, including one four days after being exposed to the Internet. Experiment 5 was regularly scanned between May 5th and June 26th, and then the scanning stopped until our requested scan started. This gap might explain why “no results found” was returned for the history

searches prior to our requested scan. The IP addresses for Experiments 6 and 7 still showed “no results found” for no obvious reason, while all four experiments could be queried on Honeyscore.

Shodan correctly identified the relevant data for Experiments 4 and 5, including the cloud provider, cloud region, country, and open ports. It did not identify either as a honeypot in the scan results. However, out of the four experiments, Honeyscore did report Experiment 5 as a possible honeypot, while saying Experiments 4 and 6–7 “look like a real system.” Honeyscore does not report what factors went into the decision, nor does it give the user a score, so it is not clear why only one experiment was judged a honeypot.

D. T-POT

Initially, we tried to use the Kibana and ELK stack tools within T-Pot for our analyses; however, this proved more challenging than anticipated. First, not all events that Conpot handled were logged, and some of its timestamps were likely inaccurate when they were flushed from the cache before a shutdown. This incomplete and inaccurate dataset skewed the visualizations and activity databases of T-Pot.

Also, the ease of analyzing a honeypot depends on how it logs its data. For example: Snare and Tanner break down an attacker’s HTTP request into individual components (Figure 21), while Conpot leaves the entire request as one data field (Figure 22). This limited our ability to analyze specific data points. Errors in how Kibana parses a honeypot’s logs can also result in errant data as shown in Figure 23, where the location of the Tanner logs is presented as the 3rd-most requested HTTP path.



Figure 21. Screenshot of Elastic Database of Experiment 2, Showing Multiple Data Fields from an HTTP Request

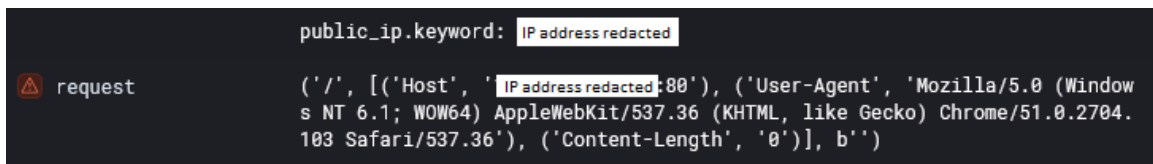
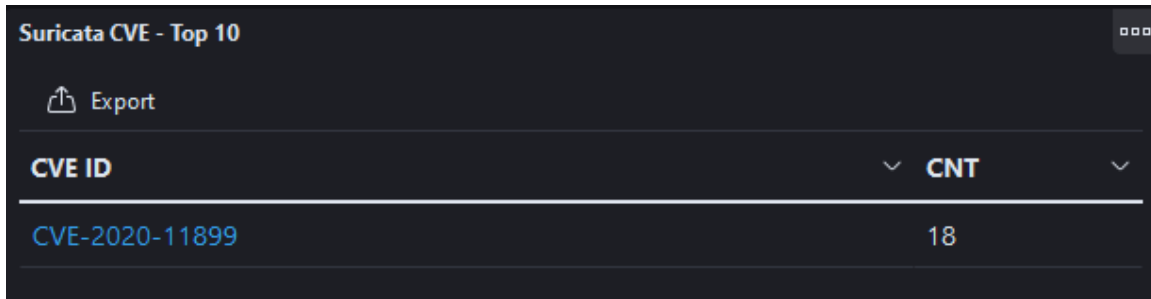


Figure 22. Screenshot of Elastic Database of Experiment 4, Showing Entire HTTP Request in One Data Field

Tanner URI - Top 10	
Export	
URI	CNT
/	813
/.env	294
/data/tanner/log/tanner_report.json	273

Figure 23. Screenshot of Kibana Dashboard of Experiment 2, Showing an Errant HTTP Path

The data that Suricata provides was more complete. Figure 24 shows one way Suricata’s data can be visualized, as a top-10 list of generated alert signatures with potential CVEs. This data supplements the honeypot data.



CVE ID	CNT
CVE-2020-11899	18

Figure 24. Screenshot of Suricata Dashboard of Experiment 5, Showing One CVE

Experiments 2 and 3 provided us the opportunity to directly compare a standalone honeypot to a honeypot running on T-Pot. Experiment 3 (with Snare and Tanner running on T-Pot) was easier to manage than Experiment 2 (Snare and Tanner installed on a bare server). The Cockpit with DigitalOcean’s tools allowed easy management of the droplet. The Kibana dashboard, Suricata threat analysis, and the HTTP data provided by Snare and Tanner provided a real-time monitoring capability of incoming attacks and a visual method to ensure the honeypot is operating, a capability lacking on a standalone honeypot.

During installation, T-Pot schedules daily restarts and weekly software updates. During these restarts, T-Pot rebuilds the Docker containers for the honeypots and tools while preserving previously collected data, ensuring a clean start to the 24-hour data collection period. This ensures that if an error occurs, as it did in Experiments 6 and 7, which ceased to respond to HTTP requests after seven days, we would lose at most only 24 hours of data. The installer also automatically migrates the server’s SSH capabilities to a different port than the default port 22, which allowed us full access to the T-Pot server without having to expose port 22 and thus change the intended outward presentation of the server.

T-Pot did use more computing resources than a standalone honeypot. Experiment 2 (standalone honeypot) used 20–25 times less processing power, six times less memory, and four times less disk space than Experiment 3 (T-Pot installation) (Figures 25–30). These two experiments ran on identically configured DigitalOcean droplets with 2 processors, 8 gigabytes of memory and 25 gigabytes of disk space.

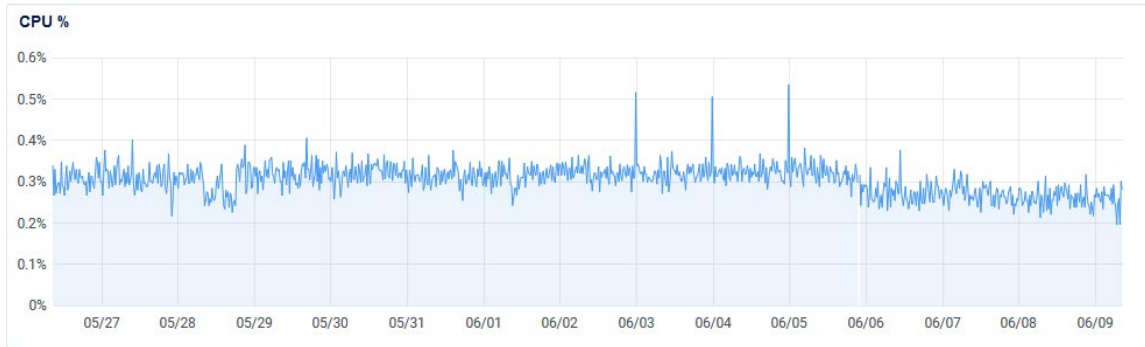


Figure 25. Processor Use in Experiment 2



Figure 26. Processor Use in Experiment 3

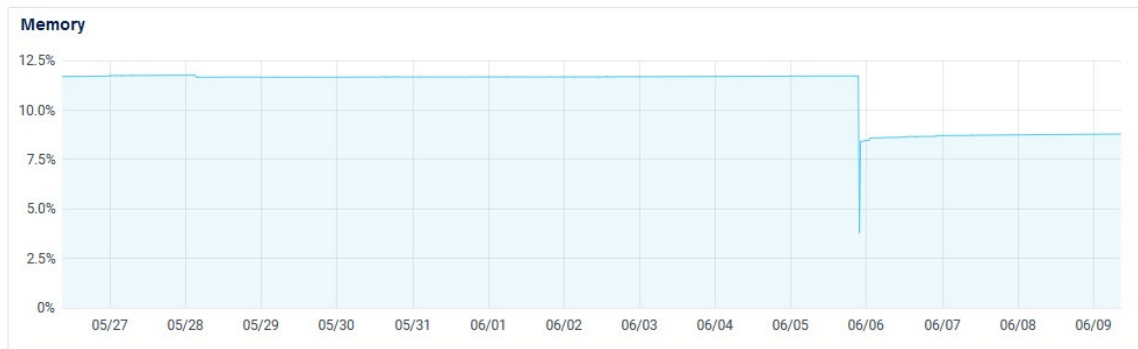


Figure 27. Memory (RAM) Use in Experiment 2

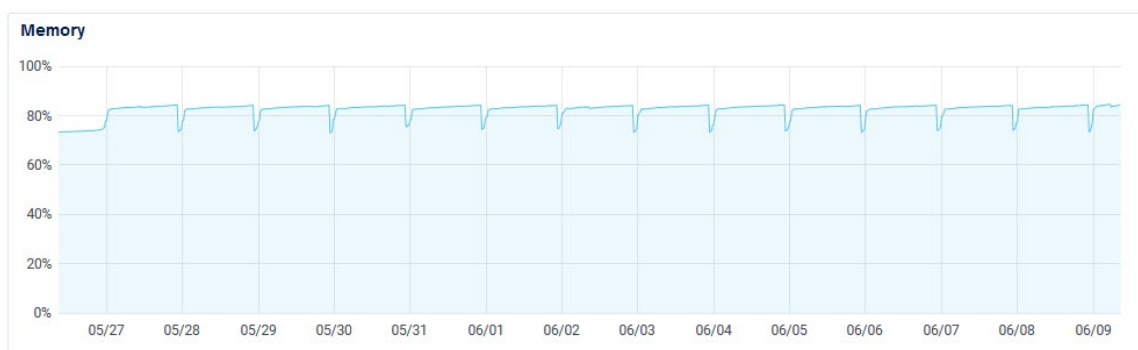


Figure 28. Memory (RAM) Use in Experiment 3

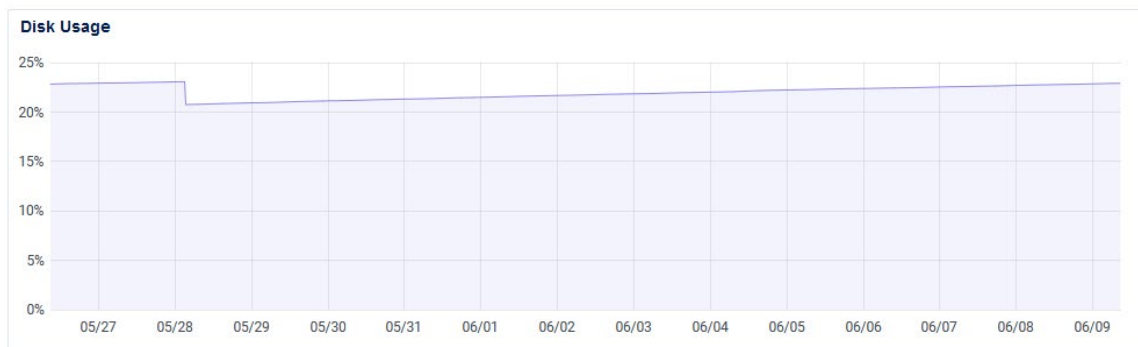


Figure 29. Disk Use in Experiment 2



Figure 30. Disk Use in Experiment 3

VI. CONCLUSIONS AND FUTURE WORK

This thesis explored the cloud deployment of honeypots within the T-Pot honeypot platform. Several kinds of implementations were compared. Our experiments did not show a significant difference in traffic received by honeypots deployed in the cloud, honeypots deployed in the cloud within T-Pot, and honeypots deployed on local servers. This is encouraging as it allows flexibility for future honeypot deployment schemes.

Experiment 1 received more daily traffic across the protocols used by Conpot, although our experiment received larger proportion of HTTP traffic compared to the other protocols. This suggests that cloud-deployed web servers are more popular for attackers, compared to the other ICS-related protocol handlers that Conpot provides.

Experiments 2 and 3 showed the honeypot deployed within T-Pot received less traffic than the one installed directly onto the droplet server, though the traffic distribution was similar. However, even though both experiments were deployed on the same cloud provider in the same geographic region, only a third of the source IP addresses were shared between them, although many addresses in the experiments were similar. Future work could run identical honeypots on the same servers to analyze the similarity among the received attacks. If those results were like ours, it would demonstrate that despite geographic co-location or IP address similarity, honeypots are not identically attacked, and that attackers are not uniformly crawling through and attacking IPv4 address space.

Experiments 4–7 showed some differences between the experiments, such as the GridPots within T-Pot received slightly more IEC 104 traffic. We also observed that results of experiments deployed in identical geographic areas were not significantly statistically different. The increase in ICS traffic, compared to the decrease seen in Experiment 1, could be due to GridPot showing a more realistic and high-interaction interface than Conpot. Future work could modify Phase II of Dougherty’s GridPot to run as a Docker container and integrate it with T-Pot. Other future work could use Snare and Tanner to handle HTTP traffic instead of GridPot’s HTTP server, which could provide more attack data when

combined with a sleeker user interface, besides stability and a more robust logging capability.

T-Pot proved a capable honeypot management platform with many built-in capabilities and support for customization. We observed some shortcomings in its analytical capabilities due to incomplete data from honeypot logging. We overcame this hurdle by running packet-capture software and primarily relying on its data for offline analysis. We recommend any future honeypot deployment, production, or research include packet capturing, since T-Pot does not natively provide it.

Our research did not show that a T-Pot deployment could be fingerprinted as such. Future work should analyze whether any distinctive characteristics of a server running a T-Pot exist.

Time constraints limited how much data we could collect for each experiment. However, these experiments are still running, and this larger collection of data could be analyzed at different time intervals, e.g., over six months or a year, to further determine if deployment on the cloud within T-Pot affects the rate of attacks or the honeypot data collected. Also, machine learning can be used on this dataset for further behavioral analysis, including determining if any penetration-testing software was used in attacks on the honeypots.

APPENDIX A. CUSTOM T-POT INSTALLATION

We used the following steps to customize our T-Pot installation for use in our experiments.

Step 1: Clone the T-Pot repository.

```
git clone https://github.com/telekom-security/tpotce.git
```

Step 2: Copy the custom T-Pot template into the installation folder.

```
cp Washofsky.yml tpotce/iso/installer
```

Step 3: Edit the installation script (tpotce/iso/installer/install.sh)

Step 3a: Add the custom template as a user-selectable choice.

Insert into install.sh Line 522:

```
"WASHOFISKY" "For experiments" \
```

Step 3b: Copy the custom template into the installed directory.

Insert into install.sh Line 713:

```
cp washofsky.yml /opt/tpot/etc/compose/
```

Step 3c: Add functionality for custom template

Insert into install.sh Line 745:

```
WASHOFISKY)
```

```
fuBANNER "WASHOFISKY"
```

```
ln -s /opt/tpot/etc/compose/washofsky.yml $myTPOTCOMPOSE
```

```
;;
```

Step 4: Run installation script as root

```
sudo ./tpotce/iso/installer/install.sh --type=user
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. COSINE SIMILARITY DATA

The tables below show the data used for the cosine similarity analysis in Section V.C.2. Tables B-1 through B-4 contain weekly counts of HTTP and IEC methods for all experiments, Table B-5 contains the values used for the vector elements in cosine similarity, and Tables B-6 through B-8 contain the calculated cosine similarity values.

Table B-1. Experiment 4 (US T-Pot) Weekly HTTP and IEC 104 Method Counts

Week	CONNECT	GET	POST	OPTIONS	HEAD	NONE	PROPFIND	Invalid IEC104	I- Frames	U- Frames
1	0	602	81	0	6	43	0	6	1	2
2	13	766	88	6	4	126	0	89	4	10
3	2	2787	49	1	4	20	0	3	2	6
4	0	550	63	0	2	50	0	6	3	8

Table B-2. Experiment 5 (Asia T-Pot) Weekly HTTP and IEC 104 Method Counts

Week	CONNECT	GET	POST	OPTIONS	HEAD	NONE	PROPFIND	Invalid IEC104	I- Frames	U- Frames
1	1	972	834	3	5	84	0	1	1	3
2	11	646	86	3	6	103	0	87	1	5
3	5	930	911	1	26	63	0	1	1	4
4	2	539	67	4	3	58	0	5	2	5

Table B-3. Experiment 6 (US) Weekly HTTP and IEC 104 Method Counts

Week	CONNECT	GET	POST	OPTIONS	HEAD	NONE	PROPFIND	Invalid IEC104	I- Frames	U- Frames
1	2	593	72	0	3	51	0	5	0	1
2	6	130	16	0	0	6	0	89	0	4
3	0	0	0	0	0	0	0	1	0	3
4	0	0	0	0	0	0	0	10	0	7

Table B-4. Experiment 7 (Asia) Weekly HTTP and IEC 104 Method Counts

Week	CONNECT	GET	POST	OPTIONS	HEAD	NONE	PROPFIND	Invalid IEC104	I- Frames	U- Frames
1	2	5887	793	7	114	167	0	8	0	1
2	3	1251	39	0	43	34	0	94	0	3
3	0	0	0	0	0	2	0	1	0	4
4	0	0	0	0	0	0	0	6	0	2

Table B-5. Average Counts used for HTTP and IEC 104 Cosine Similarity Comparisons

Experiment	GET	POST	Other Methods	IEC
4	1176.3	70.25	69.25	35
5	771.75	474.5	94.5	29
6	180.75	22	17	30
7	1784.5	208	93	29.75

Table B-6. Weekly HTTP and IEC 104 Cosine Similarity for Individual Experiments

Week	Experiment 4	Experiment 5	Experiment 6	Experiment 7
2	0.9871	0.8274	0.8222	0.9917
3	0.9712	0.7909	0.5773	0.0952
4	0.9915	0.796	1.0	0.9285

Table B-7. Weekly Average HTTP and IEC 104 Cosine Similarity per Experiment

Experiment 4	Experiment 5	Experiment 6	Experiment 7
0.9833	0.8047	0.79984	0.6718

Table B-8. Weekly IEC 104 Cosine Similarity for Individual Experiments

Week	Experiment 4	Experiment 5	Experiment 6	Experiment 7
2	0.9721	0.3564	0.9883	0.9957
3	0.5338	0.2921	0.3585	0.2734
4	0.9852	0.8660	0.8031	0.5370

Table B-9. Weekly Average IEC 104 Cosine Similarity per Experiment

Experiment 4	Experiment 5	Experiment 6	Experiment 7
0.8304	0.5048	0.7166	0.602

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Alata, E. et al. (2006). Lessons learned from the deployment of a high-interaction honeypot. *2006 Sixth European Dependable Computing Conference*, 39–46. doi:10.1109/EDCC.2006.17
- Basu, S. et al. (2019). Cloud computing security challenges & solutions—A survey. *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 347–356. doi:10.1109/CCWC.2018.8301700
- Bieker, M., & Pilkington, D. (2020). Deploying an ICS Honeypot in a cloud computing environment and comparatively Analyzing Results against Physical Network Deployment. (*Master's thesis, Naval Postgraduate School*). NPS Archive: Calhoun. Retrieved from <http://hdl.handle.net/10945/66586>
- Borges, L. et al. (2018). An evolutive hybrid approach to cloud computing provider selection. *2018 IEEE Congress on Evolutionary Computation (CEC)*, 1–8. doi:10.1109/CEC.2018.7477742
- Bove, D. (2018). Using honeypots to detect and analyze attack patterns on cloud infrastructures. (*Master's thesis, Friedrich-Alexander University*). Bavaria, Germany. Retrieved from <https://davidebove.com/files/thesis-bove-public.pdf>
- Brooks, P. (2001). *EtherNet/IP industrial protocol white paper*. Institute of Electrical and Electronic Engineers (IEEE).
- Brown, S., Lam, R., Prasad, S., Ramasubramanian, S., & Slauson, J. (2012, December 19). Honeypots in the cloud. Madison: University of Wisconsin - Madison. <http://pages.cs.wisc.edu/~sbrown/downloads/honeypots-in-the-cloud.pdf>
- Campbell, R., Padayachee, K., & Masombuka, T. (2015). A survey on honeypot research: Trends and opportunities. *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*, 208–212. doi:10.1109/ICITST.2015.7412090
- Chapendama, S. (2019). Analysing honeypot data using Kibana and Elasticsearch. *Towards Data Science*. <https://towardsdatascience.com/analysing-honeypot-data-using-kibana-and-elasticsearch-5e3d61eb2098>
- Chassin, D. et al. (2008). An open-source power systems modeling and simulation environment. *2008 IEEE/PES Transmission and Distribution Conference and Exposition*, 1–5. doi:10.1109/TDC.2008.4517260

- Chong, W., & Koh, C. (2018). *Learning cyberattack patterns with active honeypots*. (Master's thesis, Naval Postgraduate School). NPS Archive: Calhoun. <http://hdl.handle.net/10945/60377>
- Combe, T., Martin, A., & Di Pietro, R. (2016, November 11). To docker or not to docker: A security perspective. *IEEE Cloud Computing*, 3(5), 54–62. doi:10.1109/MCC.2016.100
- Cybersecurity and Infrastructure Security Agency. (2021). *Significant historical cyber-intrusion campaigns targeting ICS*. Arlington: CISA. <https://us-cert.cisa.gov/ncas/current-activity/2021/07/20/significant-historical-cyber-intrusion-campaigns-targeting-ics>
- Cybersecurity and Infrastructure Security Agency and Federal Bureau of Investigation. (2021). *Joint cybersecurity advisory: Chinese gas pipeline intrusion campaign, 2011 to 2013*. Arlington: Cyberstructure and Infrastructure Security Agency. Retrieved July 21, 2021, from <https://us-cert.cisa.gov/ncas/alerts/aa21-201a>
- Dangeti, P. (2017). *Statistics for machine learning: Techniques for exploring unsupervised, supervised, and reinforcement learning models with Python and R*. Birmingham, UK: Packt Publishing.
- DigitalOcean, LLC. (n.d.). *Droplets*. <https://docs.digitalocean.com/products/droplets>
- Dionaea. (2015). *Dionaea documentation*. <https://dionaea.readthedocs.io>
- Dougherty, J. (2020). *Evasion of honeypot detection mechanisms through improved interactivity of ICS-Based Systems*. (Master's thesis, Naval Postgraduate School). NPS Archive: Calhoun. Retrieved from <http://hdl.handle.net/10945/66065>
- Eigner, O., Kreimel, P., & Tavolato, P. (2018). *Identifying S7Comm protocol data injection attacks in cyber-physical systems*. 5th International Symposium for ICS & SCADA Cyber Security Research.
- Elasticsearch B.V. (n.d.). *Elastic (ELK) Stack*. <https://www.elastic.co/elastic-stack>
- Gruschka, N., & Iacono, L. (2009). Vulnerable cloud: SOAP message security validation revisited. *2009 IEEE International Conference on Web Services*, 625–631. doi:10.1109/ICWS.2009.70
- Huang, K. et al. (2018, Jul). Systematically understanding the cyber attack business: A survey. *ACM Computing Surveys*, 51(4), 1–36.
- Hurd, C. M., & McCarty, M. V. (2017). *A survey of security tools for the industrial control system environment*. Idaho Falls: Idaho National Laboratory.

- Hyun, D. (2018). *Collecting cyberattack data for industrial control systems using honeypots*. (Master's thesis, Naval Postgraduate School). NPS Archive: Calhoun. <http://hdl.handle.net/10945/58316>
- Jicha, A. et al. (2016). SCADA honeypots: An in-depth analysis of Conpot. *2016 IEEE Conference on Intelligence and Security Informatics (ISI)*, 196–198. doi:10.1109/ISL.2016.7745468
- Kaplan, F. (2017). *Dark territory: The secret history of cyber war*. New York City: Simon & Schuster.
- Kelly, C., Pitropakis, N., Mylonas, A., McKeown, S., & Buchanan, W. (2021). A comparative analysis of honeypots on different cloud platforms. *Sensors*, 21(2433). doi:10.3390/s21072433
- Keri, M., Lechthaler, B., & Ochse, M. (n.d.). *DICOM honeypot*. Retrieved July 15, 2021, from GitHub: <https://github.com/nsmfoo/dicompot>
- Leach, P. J., Berners-Lee, T., Mogul, J. C., Masinter, L., Fielding, R. T., & Gettys, J. (1999). *Hypertext Transfer Protocol - HTTP/1.1*. Internet Engineering Task Force. <https://tools.ietf.org/html/rfc2616>
- Matousek, P. (2017). *Description and analysis of IEC 104pProtocol*. Brno University of Technology.
- MaxMind, Inc. (n.d.). *GeoIPdatabases & services*. Retrieved July 15, 2021, from MaxMind: <https://www.maxmind.com/en/geoip2-services-and-databases>
- Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. Gaithersburg: National Institute of Standards and Technology. doi:SP 800–145
- Middleton, B. (2017). *A history of cyber security attacks: 1980 to present*. Boca Raton: Auerbach Publications.
- Moubayed, A., Injadat, M., Shami, A., & Lutfiyya, H. (2018). DNS typo-squatting domain detection: A data analytics & machine learning based approach. *2018 IEEE Global Communications Conference (GLOBECOM)*, 1–7. doi:10.1109/GLOCOM.2018.8647679
- National Security Council. (2018). *National cyber strategy of the United States of America*. Washington, DC: National Security Council.
- National Security Council. (2021). *Interim national security strategic guidance*. Washington, DC: National Security Council.
- Oosterhof, M. (n.d.). *Cowrie*. Retrieved July 15, 2021, from www.cowrie.org

- Open Information Security Foundation (OISF). (n.d.). *Suricata*. Retrieved July 15, 2021, from <https://suricata.io>
- Red Hat. (n.d.). *Cockpit Project*. Retrieved July 15, 2021, from <https://cockpit-project.org>
- Renfro, S., & Guyton, B. (2021, July 15). *mergecap*. Retrieved from Wireshark.org: <https://www.wireshark.org/docs/man-pages/mergecap.html>
- Rist, L. et al. (n.d.). *SNARE/TANNER*. Retrieved July 15, 2021, from www.mushmush.org
- Schmall, M., Vorbach, A., & Ochse, M. (n.d.). *medpot*. Retrieved July 2015, 2015, from GitHub: <https://github.com/schmalle/medpot>
- Secretary of the Navy. (2018). *Department of the Navy Critical Infrastructure Protection Program*. Washington, DC: Department of the Navy. doi:SECNAVINST 3501.1D
- Serbanescu, A., Obermeier, S., & Yu, D.-Y. (2015, September). ICS threat analysis using a large-scale honeynet. *3rd International Symposium for ICS & SCADA Cyber Security Research 2015*, 20–30. doi:10.14236/ewic/ICS2015.3
- Seri, B. et al. (2019). *Urgent/11: Critical vulnerabilities to remotely compromise VxWorks, the most popular RTOS*. Armis, Inc. https://info.armis.com/rs/645-PDC-047/images/Urgent11_Technical_White_Paper.pdf
- Shiflett, C. (2006). *Essential PHP security: A Guide to building secure web applications*. Sebastopol: O'Reilly Media, Inc.
- Shodan. (n.d.). *What is Shodan?* Retrieved July 15, 2021, from <https://help.shodan.io/the-basics/what-is-shodan>
- Simmon, E. (2018). *Evaluation of cloud computing services based on NIST SP 800-145*. Gaithersburg: National Institute of Standards and Technology. doi:SP 500–322
- Sochor, T., & Zuzcak, M. (2014, February). Study of internet threats and attack methods using honeypots and honeynets. *International Conference on Computer Networks*, 118–127. doi:10.1007/978-3-319-07941-7_12
- Surbiryala, J., & Rong, C. (2019). Cloud computing: History and overview. *2019 IEEE Cloud Summit*, 1–7. doi:10.1109/CloudSummit47114.2019.00007
- Swales, A. (1999). *Open MODBUS/TCP Specification*. Schneider Electric.
- Telekom Security. (n.d.). *T-Pot*. Retrieved July 15, 2021, from <https://github.com/telekim-security/tpotce>

- White House. (2021). *National Security Memorandum on improving cybersecurity for critical infrastructure control Systems*. Washington, DC: National Security Memorandum. Retrieved July 28, 2021, from <https://www.whitehouse.gov/briefing-room/statements-releases/2021/07/28/national-security-memorandum-on-improving-cybersecurity-for-critical-infrastructure-control-systems/>
- Wireshark Foundation. (n.d.). *tshark*. Retrieved July 15, 2021, from Wireshark.org: <https://www.wireshark.org/docs/man-pages/tshark.html>
- Zhang, F. et al. (2003). Honeypot: A supplemented active defense system for network security. *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 231–235. doi:10.1109/PDCAT.2003.1236295
- Zhu, J., & Desai, B. (2015). User Agent and privacy compromise. *C3S2E '15: Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, 38–45. doi:10.1145/2790798.2790803

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California