2019-09

IDENTIFYING HONEYPOTS SIMULATING INTERNET-CONNECTED INDUSTRIAL-CONTROL SYSTEM DEVICES

Brown, Justin C.

Monterey, CA; Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

IDENTIFYING HONEYPOTS SIMULATING INTERNET-CONNECTED INDUSTRIAL-CONTROL SYSTEM DEVICES

by

Justin C. Brown

September 2019

Thesis Advisor: Neil C. Rowe
Second Reader: Robert Beverly

THIS PAGE INTENTIONALLY LEFT BLANK

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

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** September 2019	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE IDENTIFYING HONEYPOTS SIMULATING INTERNET-CONNECTED INDUSTRIAL-CONTROL SYSTEM DEVICES	**5. FUNDING NUMBERS**
6. AUTHOR(S) Justin C. Brown	
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)

Heuristic analysis can reveal honeypots (decoy computer systems doing intelligence gathering) among Internet-connected industrial-control sites. Detectability of honeypots is undesirable, as it enables a careful adversary to avoid them, thus inhibiting valuable intelligence. However, counting honeypots is crucial to cyber-security policy and planning activities. Using a data set that includes industrial-control sites and industrial-control honeypots on the public Internet, we tested three heuristics for their ability to detect instances of the Conpot honeypot. The heuristics searched for sites containing keywords from Conpot, for services on combinations of port numbers matching Conpot, and for industrial-control sites located in a public cloud service provider. Performance of each heuristic was tested by manual inspection of data returned by hosts to Shodan's probes, which we used to assess each host's status as an instance of Conpot or not. Testing showed mixed success of the three heuristics, highlighting presence of honeypots simulating Siemens STEP 7 devices. We also tested Honeyscore, a commercial product which tries to identify honeypots, and found it had good success but was not perfect. We show that no single tool detected all honeypots, and that multiple tools can be complementary. Suggestions are offered for increasing detection rates, as well as potential additional heuristics to test.

14. SUBJECT TERMS honeypot, cybersecurity, cyber-deception, industrial-control	**15. NUMBER OF PAGES** 85
	16. PRICE CODE

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

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

IDENTIFYING HONEYPOTS SIMULATING INTERNET-CONNECTED INDUSTRIAL-CONTROL SYSTEM DEVICES

Justin C. Brown
Civilian, Department of the Navy
BS, Colorado Technical University, 2013

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 2019

Approved by: Neil C. Rowe
Advisor

Robert Beverly
Second Reader

Peter J. Denning
Chair, Department of Computer Science

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Heuristic analysis can reveal honeypots (decoy computer systems doing intelligence gathering) among Internet-connected industrial-control sites. Detectability of honeypots is undesirable, as it enables a careful adversary to avoid them, thus inhibiting valuable intelligence. However, counting honeypots is crucial to cyber-security policy and planning activities. Using a data set that includes industrial-control sites and industrial-control honeypots on the public Internet, we tested three heuristics for their ability to detect instances of the Conpot honeypot. The heuristics searched for sites containing keywords from Conpot, for services on combinations of port numbers matching Conpot, and for industrial-control sites located in a public cloud service provider. Performance of each heuristic was tested by manual inspection of data returned by hosts to Shodan's probes, which we used to assess each host's status as an instance of Conpot or not. Testing showed mixed success of the three heuristics, highlighting presence of honeypots simulating Siemens STEP 7 devices. We also tested Honeyscore, a commercial product which tries to identify honeypots, and found it had good success but was not perfect. We show that no single tool detected all honeypots, and that multiple tools can be complementary. Suggestions are offered for increasing detection rates, as well as potential additional heuristics to test.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

List of Acronyms and Abbreviations

API	application programming interface
ASN	autonomous system number
AST	aboveground storage tank
ATG	automatic tank gauge
BMC	baseboard management controller
CCTV	closed-circuit television
DCS	distributed-control system
DNP3	Distributed Network Protocol
DOD	Department of Defense
EC2	Elastic Cloud Compute
FTP	File Transfer Protocol
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communication
HMI	human-machine interface
HTTP	Hyper Text Transfer Protocol
HTTPS	secure HTTP
ICCP	Inter-control Center Communications Protocol
ICMP	Internet Control Message Protocol
ICS-CERT	Industrial Control Systems Cyber Emergency Response Team

ICS industrial-control system

IEC-104 International Electrotechnical Commission 60870-5-104

IP Internet Protocol

IPMI Intelligent Platform Management Interface

ISP Internet service provider

JSON JavaScript Object Notation

KMP Kamstrup Meter Protocol

NAT network address translation

NIST National Institute of Standards and Technology

NPS Naval Postgraduate School

NSE Nmap Scripting Engine

OS operating system

OSINT open-source intelligence

PAT port address translation

PLC programmable logic controller

RAM random access memory

RSH Remote Shell

RTU remote terminal unit

S7Comm STEP 7 Communications

SCADA Supervisory Control and Data Acquisition

SNMP Simple Network Management Protocol

SSH Secure Shell

TCP Transmission Control Protocol

TFTP Trivial File Transfer Protocol

UDP User Datagram Protocol

VM virtual machine

XML Extensible Markup Language

XMPP Extensible Messaging and Presence Protocol

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

My deep appreciation goes out to my family and friends who supported my effort, and to Magda for providing insights on Poland. Additionally, thanks to the librarians, baristas, and food service workers near the libraries of Naval Postgraduate School and California State University, Monterey Bay, for providing much-needed support.

Additional heartfelt thanks to the Graduate Writing Center team at Naval Postgraduate School, to Dr. Neil Rowe for guiding my work at a high standard, and to Dr. Rob Beverly and Dr. Mark Gondree for providing the critique needed to deliver substantive work.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

Efforts to identify and characterize industrial-control system devices connected to the Internet have been undertaken for roughly the last decade [1]–[4]. Work of this nature often aggregates data collected from open-source intelligence (OSINT) sources such as the public scan databases Shodan.io and Censys.io, which compile scans of industrial-control system devices across the IPv4 Internet. The data needed for counting can otherwise be generated by actively probing remote hosts to identify industrial-control system services, as was done in [1].

Industrial-control system devices on networks receive attention from adversaries and defenders alike because they could enable inputs from the cyber domain to effect kinetic changes in the physical domain [5]–[7]. Examples of such effects include moving a robotic arm, adjusting the flow rate of a fluid, or powering off a system. Some industrial-control system devices critically lack built-in security controls, which increases the potential for an adversary to identify and exploit a vulnerability. Statistics show that the number of industrial-control system devices directly reachable via the Internet is increasing [1]–[4].

Accurately characterizing a remote network device poses technical challenges. For example, during an Internet-wide survey, if a device being characterized changes its Internet Protocol (IP) address during counting, it may be counted twice or not counted at all. Counting the same device additional times may also occur when it uses multiple IP addresses, as no convention limits the quantity of addresses on a device. Intermediate network nodes such as firewalls, network address translation (NAT) devices, port address translation (PAT) devices, and network proxies may alter IP packet header data between the remote network device and the sensor, for both deceptive [8], [9] and benign purposes [10]. Such devices potentially affect characterization because they can make targets unreachable for scanning databases such as Shodan, which is discussed in Section 2.5. Intermediate nodes can also aggregate network services into a single Internet address [11]. From an industrial-control perspective, we generally do not expect a controller to change its address because doing so will affect its integrations with other systems. Nor do we expect a controller to have multiple network interfaces and IP addresses as they might have physical size constraints.

Industrial-control system devices are particularly difficult to identify via operating system fingerprinting, because it does not typically find programmable logic controllers (PLCs) and remote terminal units (RTUs), important subtypes of industrial-control system devices. The difficulty of differentiating between control-device fingerprints and those of other devices is great enough that machine learning is believed to be necessary for producing meaningful findings [12]. Therefore, honeypots designed to masquerade as legitimate devices can fool efforts to characterize industrial-control devices [9], [10].

Cyber-security planning activities recommended by the Industrial Control Systems Cyber Emergency Response Team (ICS-CERT) require accurate counts of industrial-control systems [13], [14], and it is therefore necessary to distinguish legitimate devices from honeypots. While Project SHINE mentions the problem of double-counting devices on different IP addresses [3], surveys of Internet-connected industrial-control system devices often do not identify honeypots [2]–[4]. However, results from [1] found that honeypots comprised 5% of the Siemens S7 services and 0.5% of the Modbus services identified.

Terms such as those searched in [4] aid in identifying industrial-control devices in the Shodan database, and this can also identify honeypots. This study investigates the ability to identify instances of the industrial-control honeypot Conpot using the searching technique in [4] and similar techniques. The use of Shodan means that the work is repeatable at scale without excessively impacting legitimate industrial-control devices. Specific goals of the current research included:

- Discover detectable properties and behaviors of industrial-control system honeypots using Conpot as a model.
- Test ability of tools to detect instances of Conpot by searching a large-scale scanning database.

The contributions offered by this research include the following:

- Validation of the ability of three heuristic tools to count instances of the Conpot honeypot.
- Improving characterization of legitimate industrial-control devices by better recognizing honeypots.
- Suggesting additional behaviors to improve efforts to count honeypots.

1.1 Thesis Structure

The following topics are discussed in the chapters ahead:

- 2: Background
  - Motivators for counting industrial-control system devices, including targeting and vulnerabilities
  - High-level overview of cyber-deception
  - Examples of honeypots simulating industrial-control systems
  - Considerations when choosing between using active scanning and network scan databases as the data source
  - An example study that counted industrial-control devices by protocol using data from Shodan
- 3: Design and Methodology
  - Notes from tests probing industrial-control honeypot services using Conpot as a model
  - Scope and limitations of this work
  - Components of three hypotheses: H1, H2, and H3
  - The services and corresponding port numbers of probe responses to search in Shodan data
  - Querying Shodan for IP addresses whose responses matched the port numbers
  - Querying Shodan for all responses to probes by an IP address, and its Honeyscore
  - Reduction of the byte count of the data
  - Steps to generate a 8,127-host sample data set from the 122,678-host sample data set
  - Populating the analysis environment using Splunk
  - Steps used to identify honeypots via manual inspection
  - Country location of hosts in the data samples
- 4: Testing
  - Steps to measure the performance of each of H1, H2, H3, and Honeyscores 0.5, 0.8, and 1.0
  - Findings from a system of water and sewage-processing service honeypots
  - Performance, error sources, and lessons learned from H1, H2, H3, and Honeyscore

  - Agreement and disagreement of detections between the detection methods
  - Challenges encountered in our methodology
- 5: Summary and Potential Improvements
  - Performance metrics for each of H1, H2, H3, and Honeyscores greater than zero
  - Performance metrics for combinations of H1, H2, H3
  - Performance for a combination that maximized recall
  - Indicators that could be added to H1, H2, and H3 to increase detections
- 6: Conclusion and Future Work
  - Review of the goals and findings
  - Contributions made by this study
  - Percentage of honeypots identified by manual inspection
  - Opportunities for future work to broaden scope, compare data sources, validate detections, and add more heuristics

CHAPTER 2:
Background

This study draws upon National Institute of Standards and Technology (NIST) definitions for industrial-control systems [15] and Supervisory Control and Data Acquisition (SCADA) systems [16], which are as follows:

Industrial Control System (ICS)
An information system used to control industrial processes such as manufacturing, product handling, production, and distribution. Industrial control systems include supervisory control and data acquisition systems used to control geographically dispersed assets, as well as distributed control systems and smaller control systems using programmable logic controllers to control localized processes.

Supervisory Control and Data Acquisition (SCADA)
A generic name for a computerized system that is capable of gathering and processing data and applying operational controls over long distances. Typical uses include power transmission and distribution and pipeline systems. SCADA was designed for the unique communication challenges (e.g., delays, data integrity) posed by the various media that must be used, such as phone lines, microwave, and satellite. Usually shared rather than dedicated.

2.1 Industrial-Control System Targets

When industrial-control systems are extended to allow remote access, the security controls used to appropriately restrict access should be increased. Failing to sufficiently secure access to critical systems allows attackers to cause harm, as was illustrated famously by attacks using the Stuxnet malware on centrifuges at nuclear enrichment facilities [17]. Later attacks using the CrashOverride malware against a Kiev electrical power transmission station [18] offer additional examples of targeting industrial-control system (ICS) devices. Responses to

such attacks included advice from ICS-CERT that Internet-connected SCADA, distributed-control-system, and human-machine-interface devices have their access restricted, especially since such devices are often operated with default or unchangeable username and password combinations [19].

Research has tested the security of controllers, including Beresford's [20] analysis of vulnerabilities related to session handling, authentication bypass, memory access, and Transmission Control Protocol (TCP) replay in implementations of the STEP 7 Communications (S7Comm) protocol in the S7-300 and S7-1200 models of the SIMATIC S7 family of controllers. The work in [20] produced exploit modules for Metasploit Framework penetration-testing platform. To an adversary, the possibility of remote exploitation of a SIMATIC S7 device enhances the attractiveness of any S7 service, and the presence of vulnerable targets on the Internet amplifies opportunities for a successful attack. Therefore, accurately identifying controllers exposed to the Internet is needed to create a realistic view of an organization's security posture.

2.2 Cyber Deception

Cyber deception is a deception using cyberspace. The purpose of cyber deception varies depending on the motivations of the deceiver, who is either offensive or defensive in nature, and attempts to alter behavior to achieve a perceived good or prevent a perceived harm [21]. An example is the common practice of requesting a password at the logon prompt, even when the provided user-name is known to be invalid; this deception is desirable because it complicates password guessing. Cyber deception simulates nonexistent resources and conceals true resources, emphasizes "desired observables" and de-emphasizes "undesired observables" [22]. Cyber deception strategically facilitates adversary misperceptions to degrade their effectiveness and enhances detection and analysis on the part of the defender.

One type of cyber deception is network deception, which simulates the presence of network resources [23]. A motivation for network deception can be altering the target selection from an adversary's reconnaissance phase [23]. The LaBrea tarpit is an example that affects target selection by occupying 100% of the unused portion of the IP address space and delaying its response to probes [24]. A recent tarpit called Greasy modifies TCP window sizes and types to reduce the tarpit's susceptibility to fingerprinting tools such as Degreaser [9] while

enabling variable occupancy of available IP address and port resources, increasing the cost to discern real networks from fake ones [25]. The DeTracer project introduced a network-topological deception which returns traces of fake network infrastructure topologies of arbitrary size and complexity when probed via traceroute [26].

Another type of cyber deception is a honeypot. Its purpose is to appear to be an interesting and useful target for attack [27]. They also "induce signals to cause the attacker to find false targets," a diversion effect [28]. Honeypots can be built with personal-computer, mobile-device, Web-application, electronic-mail, and terminal (e.g., secure shell) designs among others. A honeypot offers network defenders better malicious-activity detection [29]–[31], malicious-binary capture [32], intrusion-detection-system signature generation [33], [34], better steering of adversary behavior [35], [36], and better capture of IP addresses for ingestion by distributed IP reputation systems [37]–[39].

2.3 Embedded and Industrial-Control System Honeypots

Honeypots resembling embedded, industrial-control system, and SCADA host-devices are starting to appear [40]–[46]. Benefits provided by embedded and ICS honeypots include malicious activity detection and malicious binary capture [40], [41], [47], [48]. Such honeypots allow industrial-control network defenders to be proactive.

A feature differentiating an industrial-control system honeypot from other honeypots is the incorporation of a real or emulated programmable logic controller. While using real hardware controllers to complement a software honeypot can be costly [10], proxy devices can operate hybrid hardware/software honeypots [8], [10], [48], [49], following the architecture used in the Honeynet Project, which separates externally-facing services and monitoring services onto different hosts [50]. The Winn honeypot used a network proxy to make a single physical controller to appear as 75 different physical controllers [10]. A hybridized technique was proposed to model the physical effects of honeypot interactions in [51], tracking the physical state of a notional valve or relay in the emulation. Emulating changes in state indicators on industrial-control system could extend the CryPLH honeypot [44] discussed in Section 2.3.7.

Software-only honeypots can be cheaper than hardware honeypots, and may run in a virtualized environment, adding benefits of scalability, portability, and virtualization functions

such as virtual hardware, virtual networks, cloning, and virtual machine (VM) snapshots. Virtualization and cloud computing facilitate honeypots such as [42]–[46], so that networks composed of industrial-control system honeypots can be deployed rapidly [52] at large scale [41]. Public cloud infrastructure can increase a honeypot's geographical distribution [41], [48].

An ICS honeypot may appear different from a legitimate device in several ways. For example, unlike a hardware PLC, a honeypot usually runs on a general-purpose computer which provides network interface virtualization and allows numerous network interfaces and IP addresses. Therefore, the likelihood an ICS service is a honeypot increases with every secondary address it uses. An ICS honeypot is also not necessarily constrained by the same processing and memory limits as a hardware controller, and may respond to probes more quickly than a hardware controller.

2.3.1 RUGGEDTRAX

An extension of project SHINE summarized in Section 2.5 called RUGGEDTRAX sought to find threats to critical infrastructure by remote adversaries [11] seeking vulnerable targets. A Siemens RuggedCom RS910 was configured with the software of a water pump plugged directly to the Internet as a hardware honeypot, and monitored for three months. It responded to requests to Secure Shell (SSH) (TCP 22), the Hyper Text Transfer Protocol (HTTP) (TCP 80), secure HTTP (HTTPS) (TCP 443), and Distributed Network Protocol (DNP3) (TCP 20000) services. Its Modbus, Telnet, Trivial File Transfer Protocol (TFTP), Remote Shell (RSH), and Simple Network Management Protocol (SNMP) listeners were disabled [11]. The device firmware name and version were displayed on its HTTPS Web page, alongside a fictitious system name indicative of a water well in Geneva, Illinois, U.S. The firmware produces a variant of the `GoAhead` embedded Web-server banner. Some of the honeypot's services were indexed by Shodan on October 15, two days after being directly connected to Internet [3].

While RUGGEDTRAX was online it received 140,403 SSH authentication attempts which were geolocated to six primary countries: China, France, United States, Germany, Korea, and Singapore [11]. While some services were disabled during configuration and after the factory reset [11], it was unstated whether the or not the project left the default usernames

in use. The data showed 3% of attempts to authenticate used the default accounts in the Rugged operating system by default [53], [54], which implies that some of the attackers might have used insights from the Web services to identify the operating system in their attempts to gain access [11].

2.3.2 Bodenheim

Another experiment tested Shodan's ability to identify Internet-connected industrial-control system devices [8]. For 55 days, four Allen-Bradley ControlLogix 1756-L61 hardware programmable logic controllers were connected to the Internet as honeypots. The controllers hosted HTTP (TCP 80) and Ethernet/IP (TCP 44818) services, though TCP 44818 was not scanned since Shodan did not scan this port at the time of the experiment [8]. Two controllers connected with an unmodified "Standard" HTTP banner, one with an "Obfuscated" HTTP banner (replacing the usual `Server: GoAhead-Webs` header string with `Server: KCCo2013_$h09mo]`), and one with an "Advertised" HTTP banner (replacing the header string with `Server: Allen Bradley ControlLogix 1756`). The controllers had the `Connection: Close` string removed from their HTTP banners [8]. The banner modifications were performed by an inline proxy which rewrote packets in transit.

After roughly one day of deployment, the obfuscated controllers and one standard controller each received a probe from Shodan despite having never provided their IP addresses to Shodan for scanning [8]. All controllers were probed within four days of deployment, and data from all four was visible on the Shodan website within 19 days of deployment, with the earliest appearance within four days of deployment. This demonstrates that Shodan can quickly find a controller upon connection to the Internet [8]. However, it was unclear whether Shodan had data from prior historical interactions with the IP addresses used.

The Bodenheim project also investigated the relative difficulty of discovering the standard, obfuscated and advertised controllers via crafted keyword searches on Shodan [8]. Queries were created by a researcher uninformed of the banner modifications from two sets of search terms related to basic knowledge of Allen-Bradley ControlLogix controllers (i.e., `Allen`, or `Bradley`, or `ControlLogix`, or controllers, or `port:80`), and terms related to specific knowledge of the controllers (i.e., `GoAhead-Webs`, or `Connection: Close`, or `index.html`). Querying Shodan with combinations of the basic knowledge search

terms in most cases found the advertised controller, and in several cases, only the advertised controller [8]. The only search term which returned both standard and obfuscated controllers was `port:80`, and it returned more than 170 million devices. Querying Shodan with combinations of the specific knowledge search terms, in all cases, returned the standard and advertised controllers, with the broadest search returning 102 million devices and the narrowest search returning 490 devices. The only search term that returned the obfuscated device was `index.html`, and it returned more than one million devices. We conclude that an adversary using a keyword search to identify instances of a specific make and model controller will likely miss an obfuscated device.

2.3.3 Conpot

Conpot [55] uses the framework developed for the Honeynet Project to provide a free open-source software industrial-control system honeypot requiring no specialized equipment and little manual configuration, and supporting customizable templates and extensibility [45]. Conpot is primarily written in Python and provides network interactions intended to emulate real devices. Responses to remote probes are configured via templates defined in Extensible Markup Language (XML) configuration files.

The default template of Conpot activates several protocol emulators for S7Comm (TCP 102) and HTTP (TCP 80) mimicking a Siemens SIMATIC S7-200 controller and its embedded Web server [45]. At the application layer, Conpot provides a banner string via HTTP, and a basic web user-interface. Conpot is a low-interaction honeypot since its emulation is not deep.

However, it does emulate Modbus (TCP 502), BACnet (UDP 47808), SNMP (UDP 161), and Intelligent Platform Management Interface (IPMI) (UDP 623) services. The IPMI emulator mimics a baseboard management controller (BMC), supporting functions such as "chassis status" and "user list," and can be interacted with via the utility `ipmitool` [56], [57]. The IPMI emulator also permits manipulation of system power via the virtual BMC [56]. A tarpit feature in Conpot delays responses to remote requests, feigning the limited processing that a real-time device may provide to non-realtime services, such as SNMP [58]. Conpot's tarpit option is different from network tarpits such as LaBrea, which forges replies from a nonexistent system [24], [50].

2.3.4 GasPot/guardian_ast

GasPot is a honeypot intended to emulate the Veeder-Root Guardian aboveground storage tank (AST) monitoring system [59]. Its release follows recent attention given to liquid-fuel tank monitoring devices found on the Internet, which were counted in 2015 to be roughly 5,800. Most devices are of the Veeder-Root brand, and over 5,300 are IP-geolocated to United States cities [60]. Veeder-Root tank monitors accessible via the Internet have been identified and targeted by remote attackers [47]. Logs from GasPot revealed unauthorized reads and writes, defacement, and denial of service attacks. Operators of a GasPot instance can tune the temperature standard and range, station name, and other parameters to reflect a realistic configuration.

GasPot was subsequently integrated into Conpot as the `guardian_ast` template [42], providing an emulated automatic tank gauge (ATG) service on TCP 10001. A test instance the service was generated for this study, as described in Section 3.1.

2.3.5 Smart Meter/kamstrup_382

The `kamstrup_382` template provided by Conpot mimics a Kamstrup model 382 smart electrical meter [43] that provides an electrical power metering service on TCP port 1025 and a management service on port 50100. The Kamstrup 382 hardware on which this template is based measures electrical circuits up to three-phase. It allows remote access for TCP/IP networking over Wi-Fi, the Global System for Mobile Communication (GSM), and General Packet Radio Service (GPRS) connectivity, and enables local interaction via optional serial and infrared interfaces [61]. We analyzed properties of this template and suggest some possible heuristics for detection in Section 3.1.

2.3.6 The SOY Honeynet

Other research deployed a multi-region distributed industrial-control system network of honeypots for threat research [62]. Their architecture used geographically dispersed nodes hosted on Amazon Elastic Cloud Compute (EC2), with emulation for the protocols DNP3, the Inter-control Center Communications Protocol (ICCP), the International Electrotechnical Commission 60870-5-104 (IEC-104), Modbus, SNMP, TFTP, and the Extensible Messaging and Presence Protocol (XMPP) [63]. Each node was configured to emulate some subset of these protocols. The TFTP and XMPP protocols functioned less to mimic

controllers and more to provide a set of general "control" services. The honeypot performed event logging, avoiding the need for a proxy for logging.

The SOY Honeynet [62] tried to measure unsolicited interaction after connecting industrial-control system devices to the Internet. The researchers observed that the Shodan search performed unsolicited interactions with five of the six honeypots comprising their honeynet, and that follow-on interactions from new sources began only after each honeypot was listed on Shodan. This suggests that deeper interactions are always preceded by reconnaissance via Shodan [41]. This suggests that requests for industrial-control services result from the presence of some desirable target protocol rather than some combination of protocols. For example, the SNMP services, particularly version 2c and version 1, were targeted during initial reconnaissance more than the industrial-control system services [41]. This further suggests that the Shodan search engine could be proactively monitored by defenders as an early warning of unintended visibility of critical systems to adversaries via the Internet.

2.3.7 CryPLH

CryPLH is a honeypot developed and deployed in 2014 for identifying malicious activities against a Siemens S7Comm 300 controller [44]. It accepts authentication attempts for the protected functions on its http (TCP 80), https (TCP 443), and S7Comm (TCP 102) services, and rejects all credentials. Parameters in `/proc/sys/ipv4` are modified to mirror the controller's `nmap` operating-system fingerprint, appearing to succeed in being matched to the fingerprint of the reference controller when scanned from a remote network, but not matching the controller when scanned from a host on the same local area network [44].

CryPLH initially was deployed in an Internet-facing university IP address space over eight days and then 30 days. In the first deployment, remote probes searched for vulnerable SSH, closed-circuit television (CCTV), and proxy services, but yielded no observable industrial-control interactions [44]. The second deployment saw interactions with the honeypot's S7Comm service (TCP 102), which were attributed to the Shodan search engine. A third deployment of 12 days saw results similar to the first deployment [64]. However, the experimenters were unable to discover it in the Shodan database, which may have meant reduced honeypot exposure to passive reconnaissance activities such as those later observed in [41].

2.3.8 Digital Bond's SCADA Honeynet

The Digital Bond SCADA Honeynet is a system emulating Modicon Quantum controller and a "Honeywall" proxy that logs remote interactions [46]. Logging included network-packet capture, network statistics, remote operating-system fingerprinting, and intrusion detection, enabled by an installation of Snort intrusion-detection system with Digital Bond QuickDraw signatures [46].

Default configuration emulates Modbus (TCP 502), SNMP (UDP 161), HTTP (TCP 80), Telnet (TCP 23), and File Transfer Protocol (FTP) (TCP 21) services. While Modbus is the only industrial-control system service on the controller, the remaining services are customized to appear as part of the same controller [46]. From a controller perspective, the service's low-interaction property is cited by [44] as limiting the potential for adversarial interest. An experimental Internet-facing SCADA Honeynet deployment found results consistent with this belief, as attacks detected against the emulated controller were aimed strictly at non-industrial-control system vulnerabilities [52].

2.4 Active Network Scanning

Network scanning is necessary for asset tracking and detecting misconfiguration. Active scanning techniques are performed by sending a probe, often using the Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) protocols. Tools for active scanning may include transport-layer scanners and protocol-layer scanners. A transport scan sends Internet Protocol (IP) packets to test communications. If the transport scan finds that communications are possible, a protocol scan may then test whether or not meaningful communications can be performed. A network scan database can subsequently store, query, and copy the results of each scan for purposes of research or reconnaissance.

When planning active scanning, one must consider adverse effects against the targets. If the targets are industrial controllers, then the consequences from adverse effects of scanning can be substantial since controllers interact with the physical domain under time constraints. A strategy employed in [1] to reduce load on the target was to probe two of Siemens's six source and destination address combinations possible for the STEP 7 SZL command. The result was believed to provide less responses than a complete set of probes as is employed by Shodan.

A transport-layer scanner sends some combination of TCP, UDP, and Internet Control Message Protocol (ICMP) packets to a remote host, and waits an assigned timeout length for responses. No response could mean the host is powered off, is not accepting incoming connections, is unreachable (e.g., behind a firewall), or does not exist at that address. A variety of transport-layer scanners exist, with two popular examples being netcat and Nmap. Like many transport-layer scanners, netcat and Nmap can also be used as a protocol scanner, with different intended use cases.

Protocol scanners build on the capabilities of transport layer scan engines by interacting with hosts via application-specific protocols. Without the protocol scanner such interactions would require substantial subject matter expertise with the application. The Nmap Scripting Engine (NSE), for example, is a library which enables active protocol scanning, and can be extended to interact with new applications. Digital Bond's Redpoint scripts use NSE to probe industrial controllers [65]. For example, the `s7-info.nse` script probes Siemens S7 devices over TCP 102 [66]. Another script, `shodan-api.nse`, allows Nmap to interface with Shodan [67] to learn about a remote target without being noticed by the target.

2.5 Network Scan Databases

Network scan databases provide data from scans of Internet address spaces. These scans may be narrow and leverage a specific protocol, as with the Carna Botnet which performed a net-wide scan of approximately 660 million IP addresses using nmap TCP "syn" scans from hosts compromised via default telnet credentials [68]. Others network-scan databases obtain data by a slower and more continuous horizontal method of scanning. One example, ZoomEye, obtains IPv4 host data from both continuous active probes and third-party data sets [69], [70]. Censys and Shodan appear to use address selection and scanning algorithms instead of the derivative data points ingested by ZoomEye [71], [72]. For attackers, scan databases are very useful as they collect large amounts of data in one place. Defenders can benefit from scanning databases by inspecting their own resources from the same external vantage point available to adversaries.

Censys

Censys [71] is a scanning database that provides a Web-based interface to search an IPv4 Internet-wide data set. It started with the Zmap project and is now continuously updated.

Zmap used single-packet probes to survey a single port across the IPv4 Internet, achieving better than 97% coverage even with no possibility of retransmission. By randomizing the order of destination addresses to probe, along with transmitting directly to Ethernet (bypassing the TCP/IP stack), its scans completed IPv4 Internet-wide surveys in under one hour [73]. Additional insights are provided by ZGrab, ZTag, and ZDNS which are maintained as open-source tools by the ZMap project [74].

Shodan

The Shodan search engine is a popular tool for scanning Internet devices [75]–[77]. Shodan performs continuous scanning of Internet-connected hosts, and provides results via Web interface and an application programming interface (API). Shodan's scanning captures data from known hosts in the IPv4 and IPv6 address spaces, and also probes for new hosts by selecting IP addresses randomly [8], [41], which is a viable algorithm only in the IPv4 address space. It is unclear if there is a host discovery algorithm for IPv6, though Shodan can enumerate some IPv6 hosts [78]. The Shodan interface enables user requests for Shodan to scan an arbitrary IP address or CIDR block [79]. Shodan provides free access to its primary data set and API for academic use.

The Shodan website links to a service called Honeyscore [80], which uses an unpublished algorithm to rate honeypot likelihoods. Honeyscore's capabilities and its use are described more in Section 3.6.3.

Project SHINE

Project SHINE sought to investigate whether industrial-control systems are directly exposed to the Internet using data from Shodan. It queried Shodan for 20 months, starting its search process with manufacturer names picked up in trade magazines, and adding search terms from findings discovered in the previous results [3], [81]. Table 2.1 shows counts of their host discovery.

Table 2.1. Quantities of industrial-control hosts counted during SHINE.

Protocol	Port	Quantity Hosts Discovered
Siemens SIMATIC / ICCP	TCP 102	3,477
DNP3	TCP 20000	625
BACnet	UDP 47808	11,553
MODBUS/TCP	TCP 502	16,066
Ethernet/IP	TCP 44818	4,522

Eventually SHINE sampled 2,186,971 devices, from which they obtained 578 terms for traditional industrial-control system devices and 349 terms for "non-traditional" industrial-control system devices—those suggesting physical controls, such as building-automation, building-environmental, and serial-ethernet-adapter-devices. The protocols and ports investigated were S7Comm (TCP 102), Modbus (TCP 502), DNP3 (TCP 20000), Ethernet/IP (TCP 44818), and BACnet (UDP 47808). SHINE's counting methodology was to search for all hosts containing a matching search term, track the results, and optionally follow links to Web services on matching hosts. To deduplicate, the team used a reverse DNS lookup to identify hosts having a single domain name and multiple addresses. A weakness of this methodology is that it can count a duplicate host in the case that the reverse DNS record of the previous and current addresses do not resolve to the same forward DNS record. Excepting the RUGGEDTRAX honeypot, which is discussed in Section 2.3.1, industrial-control system honeypots were not explained in findings from SHINE.

CHAPTER 3:
Design and Methodology

From a high-level perspective our test had four parts:

1. Obtain and evaluate a honeypot, noting behaviors and properties observed.
2. Hypothesize behaviors or properties which distinguish the honeypot from real systems.
3. Obtain data on interactions with Internet-facing industrial-control services, and search for interactions which allow us to test our hypotheses.
4. For each interaction which is consistent with a hypothesis, inspect services on the device, gather data on the same IP address, and judge whether or not the device is a honeypot.

3.1 Honeypot Overviews

Several industrial-control honeypots are available for use, including Digital Bond Honeynet and Conpot. An initial walk through of deployment of Conpot illustrated that it was easy to deploy and required only a single virtual machine or Docker container. Selecting Conpot as the object of this study appeared to support our first goal of discovering detectable properties of a subject honeypot.

We elected to study a single honeypot model of Conpot to keep the project scope manageable. That means that when inspecting a controller to find it as a honeypot or a non-honeypot, we judged whether it was an instance of Conpot. Future work could study other ICS honeypot models.

We interacted with each of Conpot's basic industrial-control emulators to discover their functionality. This included the S7Comm, Modbus, GasPot, and Kamstrup services, which are the Conpot control services for which we could find functioning clients. For testing, we installed Conpot version 0.5.1 from its Github.com repository atop a basic virtual machine running Debian Linux 3.16.0-4-amd64 (jessie) with Python version 2.7.9.

We used PLCscan [82] to probe the S7Comm service (TCP 102) on the default template

of our Conpot installation. The response returned by Conpot showed the strings illustrated in Figure 3.1, which reflected emulated registers from the S7 service on a Siemens S7-200 Micro PLC. The values assigned to these emulated registers are configurable by the Conpot operator. However, with real PLCs some of the registers may be configurable, such as the names, and some like Serial Number are not [83]. Some registers returned a null value, indicating that Conpot was not intended to be probed for those registers, suggesting that Conpot is designed for a limited number of scanners. Among the non-null values returned, the PLC name, Plant identification, and Serial number were curious in that they are static values set by the developers of Conpot, and Conpot's installation script made no mention that custom values should be assigned to these registers.

```
user@honey-debian ~ % python plcscan/plcscan.py --ports=102 192.168.214.10
Scan start...
192.168.214.10:102 S7comm (src_tsap=0x100, dst_tsap=0x102)
  Module                    :  v.0.0
  Name of the PLC           : Technodrome
  Name of the module        : Siemens, SIMATIC, S7-200
  Plant identification      : Mouser Factory
  Copyright                 : Original Siemens Equipment
  Serial number of module   : 88111222
  Module type name          : IM151-8 PN/DP CPU
  OEM ID of a module        :
  Location designation of a module:
Scan complete
user@honey-debian ~ %
```

Figure 3.1. Output from the S7Comm service (TCP 102) of `default` template in Conpot.

We probed the Modbus service (TCP 502) of the default template of Conpot with PLCscan, using a command line switch which instructed the client to exhaustively search all the $2^8 - 9$ (247) Modbus Unit numbers. Response from the probe revealed two notional units occupying Unit IDs 1 and 2 containing operator-configurable strings, illustrated in Figure 3.2. Remaining Unit numbers through 247 returned `Device info error: SLAVE`

DEVICE FAILURE. These were good clues for this honeypot.

```
user@honey-debian ~ % python plcscan/plcscan.py --ports=502 --brute-uid
     192.168.214.10 2>/dev/null | head -n 12
192.168.214.10:502 Modbus/TCP
  Unit ID: 255
    Device info error: SLAVE DEVICE FAILURE
  Unit ID: 1
    Device: Siemens SIMATIC S7-200
  Unit ID: 2
    Device: Siemens SIMATIC S7-200
  Unit ID: 3
    Device info error: SLAVE DEVICE FAILURE
  Unit ID: 4
    Device info error: SLAVE DEVICE FAILURE
  Unit ID: 5
user@honey-debian ~ %
```

Figure 3.2. Output from the Modbus service (TCP 502) of `default` template in Conpot.

We probed the AST service of the `guardian_ast` template on our Conpot host via the Digital Bond ATG-info script for NSE [65], and obtained output containing user-configurable strings for station identification, tank configuration, and product name, as illustrated in Figure 3.3. These values match the defaults provided by Conpot, though some are not static. Also returned were randomized Volume, TC Volume, Ullage, Height, Water, and Temp values. A review of the configuration for this template shows that the station name of `STATOIL STATION` is a value which is statically assigned and therefore a likely indication of a Gaspot instance. The names and order of the product selection of SUPER, UNLEAD, DIESEL, and PREMIUM was another good clue.

```
user@honey-debian ~ % sudo  nmap -Pn --script atg-info.nse -sT -p10001
                      -oN atg-info 192.168.214.10

Starting Nmap 7.40 ( https://nmap.org ) at 2019-09-03 06:26 PDT
Nmap scan report for 192.168.214.10
Host is up (0.00031s latency).
PORT      STATE SERVICE
10001/tcp open   Guardian AST
| atg-info: I20100
| 07/18/2016 05:13
|
| STATOIL STATION
|
|
|
| IN-TANK INVENTORY
|
| TANK PRODUCT VOLUME TC VOLUME ULLAGE HEIGHT WATER TEMP
|   1 SUPER 5656 5734 3172 52.62 9.01 57.63
|   2 UNLEAD 3205 3215 6761 67.25 8.21 58.95
|   3 DIESEL 4937 5011 7506 52.52 2.69 50.57
|_  4 PREMIUM 3114 3142 7506 51.08 7.37 55.14

Nmap done: 1 IP address (1 host up) scanned in 1.39 seconds
user@honey-debian ~ %
```

Figure 3.3. Output from the ATG service (TCP 10001) of guardian_ast template in Conpot.

We probed the electrical-metering service of the kamstrup_382 template on our Conpot host using the Kamstrup Meter Protocol (KMP) with a TCP-enabled variant of the PyKamstrup client [84]. The response provided static numeric values representing electrical measurements on a meter (including Voltage, Current, and Power) at each electrical phase. Also provided were values representing power input and output, as depicted in

Figure 3.4.

```
Energy in 7183.5 kWh
Energy out 0.0 kWh
Energy in hi-res 7183.5887 kWh
Energy out hi-res 0.0 kWh
Voltage p1 228.0 V
Voltage p2 229.0 V
Voltage p3 224.0 V
Current p1 5.11 A
Current p2 4.22 A
Current p3 1.44 A
Power p1 1.0 kW
Power p2 5.499 kW
Power p3 0.895 kW
```

Figure 3.4. Output from the metering service (TCP 1025) of `kamstrup_382` template in Conpot.

Some of the emulated registers in the `kamstrup_382` suffer from partial implementation, for example date and time. Two additional registers `0x3ea` (1002) and `0x03eb` (1003) give the current time and date on the device according to KMP specification [85]. However, when probed the returned values `203513.0` and `140727.0` indicated a time in the past (08:35:13 PM, 2014 July 27). A second probe after a delay of several minutes returned the same time, revealing that this is a hard-coded value.

We concluded that the default template of Conpot seemed a useful object to probe for generating indicators of honeypots. Its S7Comm service in particular had distinctive default system name, plant ID, and serial number values.

3.2 Scope and Limitations

This study examines only on Conpot, and for analysis derives its data from Shodan. These choices dictate that we could not see hosts and services that were missed by Shodan, and

that our manual inspection will likely miss honeypots which responded to probes from Shodan differently than Conpot. Shodan's data, as with data from any third-party data source, may be incomplete or have errors. By not actively scanning objects, this study could not distinguish a probe not being sent from a probe being sent and a response not being received. This study analyzed data that is 3+ years old, hence a refresh of the data set is needed for a current count of industrial-control honeypots. Finally, we did not derive ground truth from the operator of each device on its status as a honeypot or not, which could have affirmed or negated our findings even if only for a subset of the honeypots.

3.3 Hypotheses

Based in these preliminary experiments, we tested three clues to industrial-control system honeypots. The clues could increase the accuracy of counting of ICS devices in network scans without needing a proprietary tool like Honeyscore.

We developed the following hypotheses, which we call H1, H2, and H3. All three hypotheses are applicable to the Conpot honeypot, and H3 is applicable to honeypots other than Conpot. A fourth clue was considered, that a host running both an embedded system service and a general-purpose service is a honeypot; but this appeared to require active scanning, unlike the other three hypotheses, which we judged to exceed the conditions of our goals.

A potential benefit of H1 and H2 for detecting honeypots is they are not limited to publicly visible Internet sites. H1 and H2 can be tested on the local network, making them suitable for honeypots that are developed or modified after initial use, a use that is not enabled by H3 and Honeyscore.

3.3.1 H1

A service responding via the S7Comm protocol on TCP 102 to probes of its System Name, Plant Identification, and Serial Number with the terms "Technodrome," "Mouser Factory," or "88111222," is a Conpot instance.

The likelihood of the default S7 System Name and Plant ID values being returned from probes of a Conpot instance is high because services probed by Shodan need only to be visible on the Internet for a few days to be picked up by Shodan, as was found in [3] and [8].

The default serial number “88111222” is likely to be seen in responses from Conpot, and is unlikely to be seen in responses from a production controller because Siemens assigns STEP 7 serial numbers only once and does not allow assigning a custom value [83]. Unless an installation of Conpot is customized prior to responding to probes on the IPv4 Internet, these values will be present in its responses.

3.3.2 H2

A device providing the same industrial-control services as Conpot's default template is running a Conpot instance. Additional services beyond the services of Conpot do not lessen this finding, and neither does absence of additional services increase the finding.

Conpot's `default` template listens with services on six ports, as mentioned in Section 2.3.3: HTTP tcp/80, S7Comm tcp/102, SNMP udp/161, Modbus tcp/502, IPMI udp/623, and BACnet udp/47808. It is unusual to see these services running together on the same device. As an exception, tcp/80 is discounted as it is not strictly a control protocol, and it is common to see HTTP as it provides many network services.

3.3.3 H3

A device providing industrial-control services from a public cloud location is running a honeypot.

Hardware industrial-control system devices such as programmable logic controllers interact with the physical domain [5]–[7] and are thus need a location supportive of this hardware. Public cloud-computing rarely provides the resources necessary to run industrial-control systems [86].

We identified 29 public cloud locations (Table 3.1) from the three sources, those selling infrastructure-as-a-service and platform-as-a-service hosting:

1. Provider names discovered by sampling the provider name of hosts matching H1 in the data captured from Shodan described in Section 3.6. The source for the data is the internet service provider name given by Shodan at the time of query.
2. Provider names in our Shodan database having the keywords “cloud” or “hosting.”

3. Provider names on a “Most Reliable Hosting Company Sites” page at Netcraft.com [87].

Table 3.1. Cloud Internet service provider (ISP) list, used to identify hosts in the database matching H3.

Alibaba	InMotion Hosting
Amazon	Iomart
Atlantic.net	Kattare Internet Services
Choopa	Kvchosting.com LLC
Cloud Builders	KW Datacenter
CloudRadium	Linode
CloudVPS	LiquidNet US LLC
Codero	OVH Hosting
Datapipe	Rackspace
DigitalOcean	Reliablehosting.com
Digital Ocean	SimplerCloud
DinaHosting S.L.	TheNebulaCloud
EGIHosting	UOL Cloud Computing
GMO Cloud	Windstream Hosted
GoDaddy	

The names were extended to variants found in our data set, e.g., for DigitalOcean and Digital Ocean. Suffixes which were not necessary to identify the provider, such as US, LLC, LTD, Services, Data, and Hosting, were removed for searching, since some providers applied the suffixes inconsistently. For example, our query for Amazon within our sample is "Amazon*", and it returns data from hosts whose ISP names are any of AMAZON, Amazon Data Services Ireland Ltd, Amazon Technologies, and Amazon.com. An exception we made was OVH Hosting since removing "Hosting" from its query found also "OVH SAS" which appears to be a traditional Internet service instead of a cloud provider.

A coming problem with H3 is that there are efforts underway to develop integrations between PLCs and cloud-hosted SCADA systems like those studied in [88]. Similarly, cloud services may provide industrial-control system data in historian nodes and event management nodes for auditing and automation purposes.

3.4 Internet-Facing Industrial-Control Services

To test our hypotheses, we needed to obtain data regarding interactions with Internet-facing industrial-control services. Use of the Shodan scanning database enabled us to obtain the needed data in a standardized way without our actively scanning targets.

We needed to figure out what information to request from Shodan. Searching all of the protocols simulated by Conpot is difficult, as needed for testing H1 and H2, as the SNMP (tcp/161) and HTTP (tcp/80) protocols simulated by Conpot represented a large amount of data in our testing.

Another approach considered was to search for protocols which can be probed manually with readily available tools. Digital Bond's Redpoint [65] can probe S7Comm and Modbus services. The industrial-control protocols supported by Redpoint, which include the protocols used by Conpot, ultimately inspired the list of services to search in Shodan. These services and the port numbers associated with them are shown in Table 3.2.

Table 3.2. Industrial-control protocols supported by Digital Bond's Redpoint.

Name	Port
S7Comm	TCP/102
Modbus	TCP/502
CODESYS	TCP/1200
Niagara Fox	TCP/1911
PCWorx	TCP/1962
OMRON FINS	UDP,TCP/9600
ProconOS	TCP/20547
Ethernet/IP	TCP/44818
BACnet	UDP/47808

3.5 Shodan Functionality

Shodan offers three search methods which were useful to us: `count`, `search`, and `host`. Shodan's `count` function searches Shodan's database and returns the number of devices matching a query [89]. Table 3.3 shows examples from April 7, 2016, which include nine port numbers that are searchable via Shodan. It is notable that the numbers of hosts responding to Shodan over industrial-control port numbers increased over the 2014 numbers found by Project SHINE.

Table 3.3. Quantities of hosts per Shodan count function, compared with SHINE.

Query via api.count	Response via api.count	Quantity prev. reported by SHINE
port:102	3,537	3,447
port:502	16,340	16,066
port:1200	28,792	n/a
port:1911	19,981	n/a
port:1962	15,040	n/a
port:9600	27,622	n/a
port:20547	23,499	n/a
port:44818	4,663	4,522
port:47808	11,815	11,553

Shodan's `search` function provides summary information about hosts [79]. Search does not return all of data about a host. For example, Shodan's database may contain responses to multiple probes at an IP address over port 102, and some additional probing over 502, but the search returns only the latest probe of port 102. We thus use the search function to discover relevant IP addresses.

In contrast, Shodan's `host` function searches Shodan's database for more complete device data on each address. It provides the responses made by all services with which Shodan interacted, as well as the date-time stamp, autonomous system number (ASN), port numbers, and approximate geolocation, the source for which is not documented publicly. It also provides an ISP name, which Shodan appears to have derived from Whois. Table 3.4 lists the 21 primary fields returned by the host function.

Table 3.4. Data fields returned via Shodan's `host` function.

area_code	dma_code	longitude
asn	hostnames	org
city	ip	os
country_code	ip_str	ports
country_code3	isp	postal_code
country_name	last_update	region_code
data	latitude	tags

The `data` field has a set of fields that varies based on the application protocol Shodan used to probe each service. The protocol used in a given probe is tracked in the host or search data as `data{}._shodan.module`. Example protocols include `s7` and `modbus`. The protocol specified by Shodan was not used to perform this study, as the port number was used instead.

3.6 Building the Database

The database was established using the following procedure.

1. For each port number in the list to be given in Section 3.6.1, query Shodan for a list of the IP addresses with a service responding on that port.
2. Combine all the IP addresses found into a set.
3. For each address in the set, query Shodan for the data returned from interactions with services at that address.

The data capture environment was built via Python version 3.4.3 on 64-bit Ubuntu Linux, with 3.9 GB random access memory (RAM) and dual-core processor. Python was extended with the Shodan software development kit via the `setup.py` script provided on GitHub.com and all prerequisite packages [90]. Notes from steps 1, 2, and 3 above are as follows.

3.6.1 IP Address Discovery

The port numbers used in Step 1 were generated by combining the numbers for services in the default template of Conpot (102, 502, and 47808) with those of additional services used

by Digital Bond's Redpoint (1200, 1911, 1962, 9600, 20547, 44818). The combined list of nine port numbers used is as follows:

- 102
- 502
- 1200
- 1911
- 1962
- 9600
- 20547
- 44818
- 47808

The query syntax used with Shodan's software development kit in Step 1 was: `api.search( "port:%d" % ( port_number ) )`. Initial results from the search function were limited by Shodan to 100 results, but more could be obtained by incrementing to the next "page" of 100 results, until we had obtained all the results.

3.6.2 Host Data Query

Using these addresses we queried data from each host individually with Shodan. The history option for the `host` function was assigned to true. A few errors occurred during queries. The errors were printed to stdout, and thanks to some error handling logic the jobs continued. An alarm was assigned at a duration of 5 seconds to assist with what seemed like very long timeout conditions, terminating the request if not completed in 5 seconds. Basic error checking was enabled in the handler via Python's `try` and `except` logic, enabling a single retry in case of exceptions such as `Unable to parse JSON response(address)` or `No information available for that IP`. If data for the host was not captured after one retry the host was skipped. The query syntax used in Step 3 was `api.host( ip_address, history = True)`.

In total, data was captured for 122,678 IPv4 hosts. Additional hosts were found in IPv6 address spaces, but the data for them was not helpful for our hypotheses. Additionally, our experiments extracted Honeyscore values, and Honeyscore did not appear to be scanning IPv6 at the time of the experiments. The date of discovery of the IP addresses of relevant

hosts from Shodan was April 14, 2016. The dates of collecting host data for each IP addresses were April 17, 2016, through April 20, 2016. The number of sites Shodan returned, based on the port number of the service, is as follows in Table 3.5:

Table 3.5. Shodan host queries used, and quantities of data returned.

Query term	Count
port:102	3638
port:502	16301
port:1200	30387
port:1911	20052
port:1962	17682
port:9600	31120
port:20547	26487
port:44818	7436
port:47808	12100

The services with which Shodan interacted on these port numbers are not all industrial control services. Inspecting the data on each service was necessary to establish that the service responded to Shodan's probe in a manner consistent with an industrial-control service.

Capture of the 122,678-host data sample from Shodan enabled summarization of the locations of the hosts by `country_name`, based on IP geolocation captured by Shodan. The top locations from total 232 countries are as follows, in Table 3.6:

Table 3.6. Distribution of countries in which host from our 122,678-host data set were located.

Country	Count	Country	Count
United States	48696	Netherlands	2412
Russian Federation	15690	Spain	2359
China	12919	Italy	1911
Canada	5127	Poland	1684
France	3292	Hong Kong	1531
Korea, Republic of	2723	Australia	1278
United Kingdom	2458	Other < 1%	18125
Germany	2419		

3.6.3 Honeyscore Data Query

In addition to capturing data from interactions with each host in step 4, we also captured the host's Honeyscore to enable testing it along with H1, H2, and H3. We queried Honeyscore using its Web interface in the Shodan API. We used the HTTP GET method in the Python Requests library against the generated URL `"https://api.shodan.io/labs/honeyscore/%s?key=%s" % (ip, apikey)`. Honeyscore returned a floating-point value between 0.0 and 1.0, which Shodan claims is the "probability that an IP is a honeypot" [91]. However, the derivation of each of Honeyscore's six values is undocumented. We suppose that a checklist of indicators was generated, and that each indicator is mapped to some classification and score. Out of 122,678 host records we successfully collected a Honeyscore for 122,554. Ten of the remaining 124 cases returned an error `No information available for that IP`, and 114 returned a `502 Bad Gateway` message.

3.6.4 Data Storage

Data received from the Shodan `host` function and from Shodan's Honeyscore were merged into a single data set using Python dictionary notation. The data was broken into pages of 100 hosts, and each page was committed to the file system, which was found to be necessary to prevent crashing the Python interpreter. Pickle was used to flush data from memory onto

the file system.

3.6.5 Data Reduction

The data set collected from Shodan was originally over 25 gigabytes, which was troublesome in our initial test environment in the Python interpreter. The data took several minutes to load, and then took several minutes to respond to our validation attempts, sometimes halting the interpreter instead. We looked at possible ways to reduce the size of the data set without negating its usefulness. We found that Shodan's `data` field often contained many values due to our use of Shodan's history option. The data can be useful, since a change in a site's responses can provide a clue to a honeypot. However, we opted for removal of the additional values, reasoning that sites with an industrial-control honeypot would likely continue to indicate honeypot throughout their lifecycle. A side benefit of its removal is that manually inspecting a host can be easier.

After elimination of data with these duplicate values, the data set became roughly 1.83 GB. The basic analysis functions then completed in a few seconds and never crashed the interpreter.

3.6.6 Data Sampling

Due to the number of hosts in the data set, a manual inspection of all hosts was impractical. Instead, we created a sample from the hosts. We use P to denote the primary data set containing 122,678 hosts and S for the sample data set. The procedure used to generate S from P is as follows:

1. From P, select randomly at 1 percent for roughly 1,200 hosts. Add the data from the selected hosts in P to S.
2. From P, identify hosts found by H1 and add them to S.
3. From P, identify hosts found by H2 and add them to S.
4. From P, identify hosts found by H3 and add them to S.
5. From P, identify hosts with Honeyscore of 0.5, 0.8, or 1.0, and add them to S.
6. Deduplicate contents of S on the IP address.

S is believed to be diverse enough to enable finding honeypots which were not identified

as a honeypot by H1, H2, H3 or Honeyscore. *S* is not random. However, *S* is complete in terms containing all detections from all available detection tools with the exception of Honeyscore 0.3, which they numbered 34,991, and was more than one out of every four hosts in our data set. When hosts from all selection methods were combined, and 74 hosts overlapping between methods were de-duplicated, the sample size was 8,127. The counts of each selection are as follows in Table 3.7.

Table 3.7. Counts from each source used in the sample data set.

Selection	Count
Random	1,190
H1	146
H2	2,456
H3	4,523
Honeyscore (0.5)	232
Honeyscore (0.8)	88
Honeyscore (1.0)	743

3.7 Analysis Environment

Analysis required inspection of host data captured by Shodan, and comparing values for hosts produced from H1, H2, H3, and Honeyscore with the host's manual inspection. The Python environment used to capture our data set was cumbersome for analysis because every search of the data would require that a different function to be developed. Instead, a data capture and analysis tool called Splunk was used. Its service was created inside a Docker container instance using the free Community Edition of Docker on a graphical Debian virtual machine. The Debian machine was installed on an instance of the trial VMware license on a Windows 10 laptop, and was provisioned with four virtual processors, 2 GB RAM, and a 30 GB storage partition on a solid-state drive.

3.7.1 Using Splunk

Data from the capture process was encoded to JavaScript Object Notation (JSON) and shipped to Splunk via `requests.post`, borrowing an example from the pyHEC function [92]. At the receiving end the Splunk interface was an HTTP Endpoint Connector on the

Splunk server. The data set indexed by Splunk is searchable via Splunk's Search Processing Language syntax.

The trial license for Splunk limits ingest of data to 500 megabytes per day, which we exceeded while transferring the data set into Splunk. However, at the time of use Splunk allowed up to three license violations in a rolling 30-day window before any enforcement [93], which was permissive enough for this use case. We checked that that the number of records in Splunk was equal to the number of original records. Data shipped into Splunk was stored in an index called `shodan_host`. The base search for all hosts in our data set was `index=shodan_host AND earliest=1`.

We also used the `spath` command which creates a data structure which can be more readily queried via `search` [94]. We checked whether or not fields in the JSON data appeared to be appropriately returned via `search`, and otherwise executed `spath` and piped the results into a `search` command.

Some data was initially found to be incomplete. Its size was 10,000 bytes, the default limit in Splunk. Since the largest data point was 5,997,986 bytes, the argument `TRUNCATE = 6000000` was added to configuration for `_json` sourcetype. The data set was re-ingested with this new byte limit, and a review showed that the largest value was present in its entirety.

3.8 Manual Inspection

Manual inspection of data from the sample data set S was used to assess whether sites were consistent with Conpot. The procedure we followed for each host continuing until the host was labeled, was:

1. Read the data available from all Shodan's interactions with the host's services.
2. For each controller service, parse a list of keywords. If one or more keywords present in the controller service are also present in keywords found in the Conpot model honeypot, and they are not a legitimate product identification (such as Siemens or S7-200 or SIMATIC) then label the host HONEYPOT.
3. For each controller service, if there is evidence of non-uniqueness in its control services, for example a serial number which is duplicated among more than two sites,

then label the host HONEYPOT.

4. If the host runs an industrial-control service and a non-control (e.g., FTP) honeypot, then label the host HONEYPOT.
5. If a host running an industrial-control service provides an SSH fingerprint which is duplicated among other hosts, and if one or more hosts duplicating the SSH fingerprint is a honeypot, then label the host HONEYPOT.
6. If a host provides implausible values, for example 0.0 in S7 Module version, or a serial number which matches a spare part model, then label the host HONEYPOT.
7. If none of the above behaviors were identified then label the host NOT HONEYPOT.

Manual inspection of the 8,127 hosts in *S* produced a list of 749 hosts labeled HONEYPOT, and 7,378 hosts labeled NOT HONEYPOT, which enabled a view of countries in which honeypot and non-honeypot devices in *S* were IP-geolocated at the time of Shodan's probe.

Honeypot locations are summarized as follows, in Figure 3.8:

Table 3.8. Distribution of countries in which 749 honeypots were located.

Country	Count	Country	Count	Country	Count
Poland	563	Korea, Republic of	5	Czech Republic	1
United States	51	Taiwan	5	Iran, Islamic Republic of	1
Japan	38	China	4	Ireland	1
Singapore	18	Australia	2	Italy	1
Canada	16	Hong Kong	2	New Zealand	1
Germany	11	India	2	Norway	1
Netherlands	9	Russian Federation	2	Romania	1
United Kingdom	7	Brazil	1		
France	5	Chile	1		

Hosts in *S* which were not honeypots were located across 83 countries, as follows in Figure 3.9:

Table 3.9. Distribution of countries in which 7,378 non-honeypots were located.

Country	Count	Country	Count
United States	4563	Italy	84
Canada	462	Malaysia	82
China	337	Czech Republic	81
Russian Federation	202	United Kingdom	77
Korea, Republic of	163	Germany	75
France	135	Poland	75
Spain	99	Other < 1%	848
Taiwan	94		

When comparing locations based on status, Poland's presence in S was more visible in the honeypots than in the non-honeypots. A similar increase in visibility in the honeypots is evident for hosts in Japan, Singapore, Taiwan, versus non-honeypots. Distribution of countries among non-honeypots in S appeared to be similar to those of P.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Testing

This chapter covers the testing of our detection tools against the host data in the sample data set. The process used for testing was to compare findings of each of the detection tools: H1, H2, H3, and Honeyscore. Except as otherwise noted, a positive Honeyscore means 0.5, 0.8, or 1.0, excluding Honeyscore 0.3 for the reason provided in Section 3.6.6. Detection of a honeypot by a given tool is denoted as True or False, e.g., H1=True. The correctness of the detection is compared against the host's status by manual inspection, as described in Section 3.8. For each test, performance is measured by the following values:

- Quantity of honeypots detected as honeypots (True Positives)
- Quantity of honeypots not detected as honeypots (False Negatives)
- Quantity of non-honeypots detected as honeypots (False Positives)
- Quantity of non-honeypots not detected as honeypots (True Negatives)
- Precision as: True positives / (True Positives + False Positives)
- Recall as: True positives / (True Positives + False Negatives)
- F-score as: 2 * (Precision * Recall) / (Precision + Recall)

Finally, number of hosts indicated as a honeypot by each of H1, H2, H3, and Honeyscore is compared with the number of hosts indicated by the other methods.

4.1 Estimated Number of Honeypots

The number of honeypots (positives) found in our sample data set *S* (8,127 hosts), for each of our tools, is provided in Table 4.1.

Table 4.1. Estimated number of honeypots based on each tool.

Tools	Count
H1	146
H2	2456
H3	4523
Honeyscore 0.5	232
Honeyscore 0.8	88
Honeyscore 1.0	743

The numbers of hosts having each Honeyscore are provided in Table 4.2.

Table 4.2. Number of hosts having each Honeyscore.

Honeyscore	Count
0.0	3108
0.3	3950
0.5	232
0.8	88
1.0	743
not captured	6
Total	8127

4.2 Czajka Replicas

During manual inspection a duplicated S7 serial number stood out, `6ES7 216-2AD23-0XB0`, whose usage spanned 554 S7 controllers, indicating that they are inauthentic. Of hosts using the duplicated serial number, 516 returned a duplicated SSH fingerprint. An SSH fingerprint is a hash of the public certificate, and its duplication suggests that some services on the host are copied from a common source. Thirty-eight hosts returned the duplicated S7 serial number and no SSH fingerprint. These replicas were labeled as "Czajka", from their S7 PLC name. We labeled the Czajka replicas that had empty responses to Shodan's S7 probes as honeypots, because they presented the same SSH fingerprint as a honeypot.

These SSH fingerprints and the above S7 serial number were present in 563 hosts in our

sample data set S, and in 592 hosts in our primary data set P. The 29 replicas that were not detected by any tool returned no S7 data, but had some FTP, SIP, MySQL, PortMapper, and Web services. Their services returned banners identical to those of the other hosts, except one FTP service which returned an indicator of Nepenthes honeypot, `-freeFTPd 1.0-warFTPd 1.65-`. Shodan reported 463 of the replicas being vulnerable to CVE-2015-0204 (FREAK) and CVE-2014-0160 (Heartbleed).

For Czajka replicas that returned S7 responses their responses were identical, and returned Module `v.0.0` consistent with Conpot. Responses from port 502 (Modbus) were identical. The replicas used 44 ISPs in Poland, and gave a water and sewage-processing company name in the S7 Plant identification. Honeyscore >= 0.5 detected 561 of the replicas, and H2 detected 549. Since H1, H2, and H3 do not look for duplicated S7 serial numbers or duplicated SSH fingerprints, additional heuristics are needed to detect them all.

4.3 H1 Performance

As described in Section 3.3.1, H1 looks for $Mouser \vee Technodrome \vee 88111222$ in response to the probe of the S7Comm service. Results for H1 against our sample data set are given in Table 4.3.

Table 4.3. H1 test performance compared with manual inspection.

H1	Manual Inspection	Count
False	HONEYPOT	603
False	NOT HONEYPOT	7378
True	HONEYPOT	146
True	NOT HONEYPOT	0
	Total	8127

Findings from H1 include 146 true positives, 0 false positives, and 603 false negatives. Precision is the quantity of true positives divided by the count in H1, which is 1. Recall is the ratio of true positives to honeypots in the sample, which is 0.2. The true positives in H1 suggested the honeypot customization was incomplete.

A term that was present in the Czajka replica S7 services, and is also present in Conpot's

S7 service, is the S7 Module version `v.0.0`. This `v.0.0` was present in 586 of the 603 false negatives produced by H1, and is the greatest source of error in H1. A term present in Conpot that H1 is missing is the default IPMI banner, which caused 23 false negatives. The banner matches the data from port 623 in the Conpot instance shown in Figure 4.2. These should be added to H1 as recommended in Section 5.2.

One of the hosts missed by H1 was an instance of the Gaspot service in Conpot. The honeypot was also missed by H3 and Honeyscore, only being detected by H2. Its indicator as a Conpot system is the default IPMI banner from Conpot. Its S7 service is customized, and on its own would have fooled manual inspection. The "Product" values in its ATG services are assigned dynamically, as shown in Figure 4.1. This shows that potential exists for a heuristic to identify an implausible product in automatic tank gauge (ATG) responses, as suggested in Section 6.2.

```
I20100
04/07/2016 07:50

STATOIL STATION



IN-TANK INVENTORY

TANK PRODUCT          VOLUME TC VOLUME ULLAGE HEIGHT   WATER    TEMP
  1  fbe9417a841df85fde91f06c5d8deb578857 8960   3244   29.85    2.07
     55.34
  2  2016-04-07T09:50:47+02001114 1197    8068   50.81    9.32   57.34
  3  DIESEL              7540     7587    3520   73.84    6.93   55.03
  4  PREMIUM             5000     5097    3520   43.25    6.21   53.32
```

Figure 4.1. An instance of Gaspot which was not detected by H1.

4.4 H2 Performance

As described in Section 3.3.2, H2 compares the port numbers of Conpot's emulated industrial-control services (102, 502, 623, 161, and 47808) to the ports listed in Shodan's `ports` for each host. When all ports in the above list had to be present on a host, no matching hosts were found in our full set of 122,678 hosts. But, requiring that all but one of the port numbers needed to be present, 3 hosts were identified out of 122,678 hosts. "All but three" seemed the best choice. Comparison of three variants of H2 by their F-score are as shown in Table 4.4:

Table 4.4. Performance of three variations in quantity of port matches of H2.

Detection Mechanism	True Positives	False Pos.	False Neg.	Precision	Recall	F-1 Score
H2 "all but one"	3	0	746	1	0.0040	0.0080
H2 "all but two"	92	102	657	0.47	0.12	0.20
H2 "all but three"	687	1769	62	0.28	0.92	0.43

The "all but three" variation was selected by F-score, and results from it against S are as follows in Table 4.5:

Table 4.5. H2 "all but three ports" test performance compared with manual inspection.

H2	Manual Inspection	Count
False	HONEYPOT	62
False	NOT HONEYPOT	5609
True	HONEYPOT	687
True	NOT HONEYPOT	1769
	Total	8127

H2 had 687 true positives, 1769 false positives, and 62 false negatives. Precision is the quantity of true positives divided by the count in H2, which is 0.28. Recall is the ratio of true positives to honeypots in the sample, which is 0.92.

4.5 H3 Performance

H3 looks at whether or not the ISP of a host matches any of the names on our Cloud ISP list in Table 3.1. The results are shown in Table 4.6.

Table 4.6. H3 test performance compared with manual inspection.

H3	Manual Inspection	Count
False	HONEYPOT	645
False	NOT HONEYPOT	2959
True	HONEYPOT	104
True	NOT HONEYPOT	4419
	Total	8127

Findings from H3 include 104 true positives, 4419 false positives, and 645 false negatives. Precision is the quantity of true positives divided by the count in H3, which is 0.02. Recall is the ratio of true positives to honeypots in the sample, which is 0.14.

Among the False Negative results in H3, the ISPs of hosts include some that appear to specialize in infrastructure-as-a-service and platform-as-a-service. These included the following names:

- ColoCrossing
- Host Sailor Ltd.
- Liquid Web
- PhoenixNAP LLC
- SingleHop
- Total Server Solutions L.L.C.
- Zappie Host LLC

These ISPs should be added to the Cloud ISP List to improve detection in any future iteration of H3.

4.6 Honeyscore Performance

Findings from Honeyscore were checked against manual inspection in the sample data set S as summarized in Table 4.7.

Table 4.7. Honeyscore test performance compared with manual inspection.

Honeyscore	Manual Inspection	Count
0	HONEYPOT	5
0.3	HONEYPOT	6
0.5	HONEYPOT	1
0.8	HONEYPOT	1
1	HONEYPOT	734
not captured	HONEYPOT	1
0	NOT HONEYPOT	3103
0.3	NOT HONEYPOT	3944
0.5	NOT HONEYPOT	231
0.8	NOT HONEYPOT	87
1	NOT HONEYPOT	9
not captured	NOT HONEYPOT	5
	Total	8127

Honeyscore was good at detecting Conpot, with only 11 instances of Conpot in S having a Honeyscore less than 0.5. The 11 which were missed by Honeyscore 0.5/0.8/1.0 are indeed instances of Conpot, and one example host instance is illustrated in Figure 4.2, in which data from the host's non-ICS services has been removed to simplify it.

```
>>> pprint(conhost)
[{'asn': 'AS46652',
  'data': [ [...] {'asn': 'AS46652',/
           'data': 'Location designation of a module: \n'
                   'Copyright: Original Siemens Equipment\n'
                   'Module type: IM151-8 PN/DP CPU\n'
                   'PLC name: Technodrome\n'
                   'Module: v.0.0\n'
                   'Plant identification: Mouser Factory\n'
                   'OEM ID of a module: \n'
                   'Module name: Siemens, SIMATIC, S7-200\n'
                   'Serial number of module: 88111222\n',
           'port': 102,
           'timestamp': '2016-04-08T03:16:46.697920'},
          {'asn': 'AS46652',
           'data': 'Siemens, SIMATIC, S7-200',
           'port': 161,
           'timestamp': '2016-04-04T18:20:34.112566'},
          {'asn': 'AS46652',
           'data': '\\x06\\x00\\xff\\x07\\x00\\x00\\x00\\x00\\x00\\x00\\
              x00\\x00\\x00\\x10\\x81\\x1cc '
                   '\\x008\\x00\\x01\\x80\\x04\\x02\\x00\\x00\\x00\\x00
                      !',
           'port': 623,
           'timestamp': '2016-02-25T11:52:48.129997'},
[...]     ],
  'honeyscore': '0.0',
  'ip_str': '82.196.4.78',
  'isp': 'Digital Ocean',
  'last_update': '2016-04-15T16:42:02.248095',
  'ports': [22, 80, 102, 161, 623, 443, 11211]}]
>>>
```

Figure 4.2. An instance of Conpot which was not detected by Honeyscore.

Without knowing how Honeyscore's detection mechanism works, we can only guess about what went wrong detecting the above instance of Conpot. Perhaps the indicators used by Honeyscore were incomplete at the time of testing this particular host, and were subsequently updated to detect similar hosts without updating prior results. This host is unique because it was detected by H1, H2, and H3, but not by Honeyscore. It appears that no single tool can be guaranteed to detect all honeypots.

4.7 Comparison of Tools

Comparing detections by Honeyscore with detections from our tools was tricky when Honeyscore's values were split six ways due to the many combinations. To remedy this, we grouped hosts with Honeyscores 0.5, 0.8, or 1.0 together. The detections by our tools, and their agreements and disagreements with Honeyscore, are shown in Table 4.8.

Table 4.8. Results of tests from all tools with results from all others.

H1	H2	H3	Honeyscore >= 0.5	Manual Inspection	Count
False	False	False	False	NOT HONEYPOT	1115
False	False	False	True	HONEYPOT	15
False	False	False	True	NOT HONEYPOT	82
False	False	False	not captured	NOT HONEYPOT	1
False	False	True	False	HONEYPOT	1
False	False	True	False	NOT HONEYPOT	4174
False	False	True	True	NOT HONEYPOT	234
False	False	True	not captured	NOT HONEYPOT	3
False	True	False	False	HONEYPOT	4
False	True	False	False	NOT HONEYPOT	1758
False	True	False	True	HONEYPOT	564
False	True	False	True	NOT HONEYPOT	2
False	True	False	not captured	HONEYPOT	1
False	True	False	not captured	NOT HONEYPOT	1
False	True	True	False	HONEYPOT	1
False	True	True	True	HONEYPOT	17
False	True	True	True	NOT HONEYPOT	8
True	False	False	False	HONEYPOT	2
True	False	False	True	HONEYPOT	41
True	False	True	True	HONEYPOT	3
True	True	False	False	HONEYPOT	2
True	True	False	True	HONEYPOT	16
True	True	True	False	HONEYPOT	1
True	True	True	True	HONEYPOT	81
				Total	8127

Fifteen honeypots were detected by Honeyscore and not by H1, H2, or H3. Among them, 12 had an implausible S7 Module Version `v0.0`, 12 had a duplicate S7 Serial Number, and 10 had an SSH fingerprint which overlapped with the SSH fingerprint of a host running a honeypot. These would all be good heuristics to add.

One honeypot was detected by H3 only. Four honeypots were detected by H2 only. One honeypot was detected by H1, H2, and H3, but not Honeyscore. The highest-numbered agreement between tools about honeypots exists between H2 and Honeyscore, which both identified 564 honeypots. All honeypots in S were identified by some tool.

4.8 Challenges Encountered

Challenges encountered during the data capture and analysis included the following:

1. While we had access to tools such as PLCscan and Redpoint's `modicon-info.nse` and `s7-enumerate.nse` scripts, we could not tell which underlying tool is used by Shodan's S7 and Modbus probes, nor whether S7 and Modbus services generally provide access to memory registers that were missing from Shodan's probes. Knowing something about the limitations in the probing mechanisms could have been useful.
2. Some services in Shodan's data responded to Shodan's probe with a null string. We can infer some possible causes of this, but absent a status code from the probe, we cannot be certain why.
3. When control service data was found, parsing that data into a format which facilitates searching and filtering was tricky. For example, Siemens SIMATIC device models may or may not provide a serial number in their S7 service, and they may or may not provide a serial number in their SNMP service. We used Splunk's search-time field extraction capability to parse fields when those fields were relevant, but automatic parsing would have facilitated quicker and perhaps more complete analyses.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Summary and Potential Improvements

5.1 Test Summary

The results of each test are summarized with True Positives, False Positives, False Negatives, Precision, Recall, and F-1 Score, as follows in Table 5.1:

Table 5.1. Comparison of results of detection tools.

Detection Mechanism	True Positives	False Pos.	False Neg.	Precision	Recall	F-1 Score
H1	146	0	603	1	0.19	0.33
H2	687	1769	62	0.28	0.92	0.43
H3	104	4419	645	0.023	0.14	0.039
H1 or H2	733	1769	16	0.29	0.98	0.45
H1 or H3	165	4419	584	0.036	0.22	0.062
H2 or H3	691	6180	58	0.10	0.92	0.18
H1 or H2 or H3	734	6180	15	0.11	0.98	0.19
Honeyscore 0.3	6	3944	5	0.0015	0.5	0.0030
Honeyscore 0.5	1	231	11	0.0043	0.083	0.0082
Honeyscore 0.8	1	87	12	0.011	0.08	0.020
Honeyscore 1.0	735	8	13	0.99	0.98	0.99
Honeyscore>=0.3	743	4270	6	0.15	0.99	0.2
Honeyscore>=0.5	737	326	12	0.69	0.98	0.81
Honeyscore>=0.8	736	95	13	0.89	0.98	0.93
H1 or H2 or H3 or Honeyscore 1.0	749	6182	0	0.11	1	0.20

H1 provided perfect precision albeit low recall, which may make H1 a valuable heuristic in the case that correct detection of some Conpot honeypots is prioritized over the risk of missing most honeypots. H2 provided the broadest identification of honeypots among H1, H2, and H3, which may make H2 desirable as a general-purpose heuristic, allowing

for some false positives. H3 provided poor precision and low recall; however it found a honeypot that was not detected by other tools. The best F-1 score among H1, H2, and H3 came from combining detections from H1 or H2, making it the most useful approach for searches that allow for trade-offs between completeness and correctness of detections. Shodan's Honeyscore, overall, offered near-perfect precision, and near-perfect recall. The S7 honeypots detected by H1 or H2 or H3 were 731, which left 12 S7 honeypots undetected, all of which returned the default S7 Module `v.0.0` from Conpot. It was possible to maximize recall from the criterion "H1 or H2 or H3 or Honeyscore 1.0".

Since Honeyscore's detections of instances of Conpot did not always agree with detections from H1, H2, or H3, it appears that combining detections from Honeyscore with detections from a heuristic should increase the detections. Using variations of these combinations can enable a researcher to emphasize correctness of detections (H1), or near-completeness of detections (H1 or H2), or completeness of detections with high false positives (H1 or H2 or H3 or Honeyscore 1.0).

5.2 Improving Detection Rates

Opportunities to increase detection rates could occur by adding Conpot's default IPMI "banner" string as an indicator to H1, and addition of Conpot's default S7 Module Version `v0.0` to H1, since they were found in Conpot instances and were not found in real controllers. Their addition, when checked against our 8,127-host sample data set *S*, appear to change H1's F-score to 0.989, however a re-run of the data sampling must precede any test of H1's performance with these additions.

Other improvements to our hypotheses based on inspection of the sample data in *S* are:

1. H1 and H2 used the default template in Conpot as a model, and could both be extended to include the Gaspot template via addition of keywords and port numbers from that template.
2. H2 could be extended to include the Smart Meter template in Conpot.
3. Hosts that responded to probes from Shodan with no attempt to look like a PLC should be filtered from all H3. This would reduce false positives in H3 by 3,579 by ignoring hosts with ISPs named Reliablehosting.com, EGIHosting, or CloudRadium L.L.C., the services of which returned null responses to Shodan's probes.

4. Ten additional honeypots in S can be found by H3 by adding new ISPs cited in 4.5 to the Cloud ISP List, and 66 additional hosts, which may or may not be honeypots, can be found in P.

A non-indicator finding which was notable was a Modbus Slave Device error which appeared to be consistent across instances of Conpot. This adds support to an indicator if present. Contents of an example are shown in Table 5.2.

Table 5.2. Modbus error response common among Conpot findings in Shodan.

Port	Data
502	Unit ID: 0 -- Slave ID Data: Illegal Function (Error) -- Device Identification: Slave Device Failure (Error) Unit ID: 255 -- Slave ID Data: Gateway Target Device Failed To Respond (Error) -- Device Identification: Slave Device Failure (Error)

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 6:
Conclusion and Future Work

6.1 Conclusions

This study aimed to determine properties and behaviors of industrial-control system honeypots based on the Conpot honeypot, and to test ability of heuristics to identify instances of Conpot by searching a large-scale scanning database. Three heuristics were tested against a sample data set from Shodan, confirming ability of the heuristics to characterize honeypots at differing levels of precision and recall. Most of the honeypots detected were replicas in a system of Conpot instances related by S7 serial number and SSH fingerprint.

Shodan's Honeyscore was also tested with success, however neither our hypotheses nor Honeyscore found all the honeypots unless detections from multiple methods were combined. The study enabled identification of 743 S7 honeypots from our 8,127-host sample data set, which is 20% of the 3,638 S7 services in our 122,678-host primary data set. The remaining 2,895 devices were more suitable for counting as S7 controllers, because they were not detectable Conpot instances. This appears to support [1], which found at roughly the same time of search as this study that 5% of S7 services were honeypots. Caveats are that detections in [1] used active scanning versus Shodan and counted less S7 services than Shodan counted, and that the detection methods did not incorporate S7 Module, S7 Serial number, and SSH fingerprint as potential indicators of a honeypot.

This study illustrated that a STEP 7 honeypot simulated by Conpot can be detected when a single configurable value is left as default. Combinations of services matching Conpot were shown to indicate instances of Conpot. Use of cloud service providers by a controller was also shown to provide an indicator, but with errors. We showed that these heuristics could aid in identifying as much as 20% of STEP 7 services as honeypots, potentially narrowing STEP 7 device counts. In the big picture, this represents an incremental advancement toward counting legitimate industrial-control devices by removing obvious honeypots. Suggestions were made to improve the heuristics' precision and recall, and additional heuristics were highlighted to further improve counting.

6.2 Future Work

The foremost aim for an industrial-control systems survey is to track legitimate industrial-control devices open to the IPv4 internet. The answer to this could be more narrow than the set of all of hosts listening on TCP/102 that are not an instance of a known honeypot or otherwise fake. Using a focused methodology for counting legitimate devices would enable a researcher to perform a longitudinal study of how this population looks now, and how it is evolving over time, which would be valuable to all of the stakeholders of industrial-control systems security.

Opportunities for future work based on this study include:

- Generate ground truth on hosts' status as a honeypot or not, perhaps by reaching out to operators of devices found by a heuristic.
- Perform a quicker analysis of the data to inspect 100% of hosts.
- Add signatures to enhance H1, H2 and H3 from those identified in Section 5.2.
- Add heuristics to complement H1, H2, H3, and Honeyscore.
- Track legitimate industrial-control devices that are exposed to the internet.

For tracking honeypots it would be useful to re-run the tests with a current refresh of the data set retrieved from Shodan to enable detection of modifications to honeypots over time, identification of new honeypots, or trends with Conpot. Using a second provider such as Censys, whose scanning mechanisms are open-source, would provide a means to compare the quantities and industrial-control application data with those of Shodan.

Among possible new heuristics, the following were made evident by this work:

1. If the serial number of a control service duplicates a serial number of a control service in the same family (for example, S7) then at most one of the services sharing the number is authentic, and it is possible that all are inauthentic.
2. If the SSH fingerprint of a controller duplicates the SSH fingerprint of other controllers, then the status of its control service as honeypot or not is equal to that of all of the other controllers with that SSH fingerprint.
3. If the Product returned in an automatic tank gauge (ATG) response is implausible, then the service in an instance of Gaspot. Basic constraints could be used, for example checking if the product name changes between probes, or if the product exceeds a

character limit.

4. If the date and time registers (`0x3ea` and `0x03eb`) in a Kamstrup Meter Protocol (KMP) service do not increment between time-lapsed probes, then the service is an instance of Conpot.

Additional industrial-control honeypots, such as alternative templates in Conpot, Digital Bond SCADA Honeynet, and emerging ones such as Gridpot can be evaluated using our approach, making for new findings beyond the default template in Conpot. Evaluating the latest edition of Conpot could add new insights, such as potential findings from version 0.6.0's Ethernet/IP emulator, which returns what appears to be a static default serial number.

Additional approaches that may add value may include new tools and addition of subject matter expertise in the industrial-control domain. For example, use of machine-learning tools against data sets derived similarly from public scan databases such as Shodan or Censys may speed up analysis, and may provide additional observable indicators of honeypots. Some probes of legitimate S7 services from programmable-logic controllers on hand could help by providing look and feel from the perspective of interactions with real controllers.

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

[1] A. Mirian, Z. Ma, D. Adrian, M. Tischer, T. Chuenchujit, T. Yardley, R. Berthier, J. Mason, Z. Durumeric, J. A. Halderman, and M. Bailey, "An internet-wide view of ics devices," in *2016 14th Annual Conference on Privacy, Security and Trust (PST)*, Dec 2016, pp. 96–103.

[2] O. Andreeva et al. Industrial control systems and their online availability. [Online]. Available: https://kasperskycontenthub.com/securelist/files/2016/07/KL_REPORT_ICS_Availability_Statistics.pdf. Accessed 2016 July 19.

[3] B. Radvanovsky and J. Brodsky. Project SHINE What we discovered and why you should care. [Online]. Available: https://files.sans.org/summit/ics2015/PDFs/Project_SHINE_What_We_Discovered_and_Why_You_Should_Care_Bob_Radvanovsky_Infracritical.pdf. Accessed 2016 July 19.

[4] É. P. Leverett. (2011). Quantitatively assessing and visualising industrial system attack surfaces. [Online]. Available: https://www.cl.cam.ac.uk/~fms27/papers/2011-Leverett-industrial.pdf

[5] K. Stouffer, S. Lightman, V. Pillitteri, M. Abrams, and A. Hahn. (2015, May). NIST SP 800-82, Revision 2, Guide to Industrial Control System (ICS) Security. [Online]. Available: https://csrc.nist.gov/publications/detail/sp/800-82/rev-2/final

[6] Department of Homeland Security-National Cyber Security Division. (2009). *Recommended Practice: Improving Industrial Control Systems Cybersecurity with Defense-In-Depth Strategies*. [Online]. Available: https://ics-cert.us-cert.gov/sites/default/files/recommended_practices/Defense_in_Depth_Oct09.pdf

[7] R. J. Turk. (2005, Oct.). Cyber incidents involving control systems. [Online]. Available: https://inldigitallibrary.inl.gov/sti/3480144.pdf

[8] R. Bodenheim, J. Butts, S. Dunlap, and B. Mullins, "Evaluation of the ability of the Shodan search engine to identify internet-facing industrial control devices," *International Journal of Critical Infrastructure Protection*, vol. 7, no. 2, pp. 114–123, 2014. Available: http://dx.doi.org/10.1016/j.ijcip.2014.03.001

[9] L. Alt, R. Beverly, and A. Dainotti, "Uncovering network tarpits with degreaser," in *Proceedings of the 30th Annual Computer Security Applications Conference* (ACSAC '14). New York, NY, USA: ACM, 2014, pp. 156–165. Available: http://doi.acm.org/10.1145/2664243.2664285

[10] M. M. Winn. (2015). Constructing cost-effective and targetable ICS honeypots suited for production networks. [Online]. Available: https://inldigitallibrary.inl.gov/sti/3480144.pdf

[11] B. Radvanovsky. (2015, Nov.). Project RUGGEDTRAX SCADA/ICS analysis findings report. [Online]. Available: http://www.slideshare.net/BobRadvanovsky/project-ruggedtrax-findings-report-28nov2015

[12] M. Caselli, D. Hadziosmanovi, E. Zambon, and F. Kargl. On the feasibility of device fingerprinting in industrial control systems. [Online]. Available: http://link.springer.com/chapter/10.1007%2F978-3-319-03964-0_14. Accessed 2015 Sep. 12.

[13] T. Stillions. Navy Cybersecurity. [Online]. Available: https://web.archive.org/web/20170102092427/http://www.kmimediagroup.com/navy-air-sea-peo-forum/445-articles-npeo/navy-cybersecurity/6633-navy-cybersecurity. Accessed 2016 Aug. 13.

[14] M. Chipley and D. Haegley. (2013). Cybersecuring industrial control systems. [Online]. Available: http://themilitaryengineer.com/index.php/tme-articles/tme-magazine-online/item/261-cybersecuring-industrial-control-systems

[15] National Institute of Standards and Technology. ICS - Glossary | CSRC. [Online]. Available: https://csrc.nist.gov/glossary/term/ICS. Accessed 2019 Mar. 05.

[16] National Institute of Standards and Technology. Supervisory Control and Data Acquisition (SCADA) - Glossary | CSRC. [Online]. Available: https://csrc.nist.gov/glossary/term/Supervisory-Control-and-Data-Acquisition. Accessed 2019 Mar. 05.

[17] K. Zetter. (2011, July). How digital detectives deciphered stuxnet, the most menacing malware in history. [Online]. Available: https://www.wired.com/2011/07/how-digital-detectives-deciphered-stuxnet/

[18] Dragos. (2018). CRASHOVERRIDE, Analysis of the Threat to Electric Grid Operations. [Online]. Available: https://dragos.com/wp-content/uploads/CrashOverride-01.pdf

[19] DHS ICS-CERT. (2014, Dec.). Control system internet accessibility (Update A). [Online]. Available: https://ics-cert.us-cert.gov/alerts/ICS-ALERT-11-343-01A

[20] D. Beresford. (2011, July). Exploiting Siemens Simatic S7 PLCs. [Online]. Available: https://media.blackhat.com/bh-us-11/Beresford/BH_US11_Beresford_S7_PLCs_WP.pdf

[21] N. C. Rowe, and J. L. Rrushi, "Introduction to cyberdeception," in *Springer International Publishing*, 2016.

[22] F. Cohen, D. Lambert, C. Preston, N. Berry, C. Stewart, and E. Thomas. (2001). A framework for deception. [Online]. Available: http://courses. all.net/ Deception/deception/DeceptionFramework.pdf

[23] S. Trassare, R. Beverly, and D. Alderson, "A technique for network topology deception," in *Proceedings of the Military Communications Conference (MILCOM)*, Nov. 2013.

[24] T. Liston. LaBrea-Intro history. [Online]. Available: http://labrea.sourceforge.net/ Intro-History.html. Accessed 2015 Aug. 8.

[25] L. Shing, 2016. Available: https://calhoun.nps.edu/handle/10945/48595

[26] A. West, "Toward a robust method of presenting a rich, interconnected deceptive network topology," M.S. thesis, Naval Postgraduate School, Monterey, CA 93943, Mar. 2015.

[27] L. Spitzner, *Honeypots: tracking hackers*. Addison-Wesley Reading, 2003, vol. 1.

[28] F. Cohen. (2006). The use of deception techniques: honeypots and decoys. [Online]. Available: http://all.net/courses.all.net/Deception/deception/Deception_Techniques_ .pdf

[29] K. Cabaj and P Gawkowski. (2015, July). HoneyPot systems in practice. [Online]. Available: http://pe.org.pl/articles/2015/2/16.pdf

[30] L. Nolan, R. Beverly, and G. Xie, "Transport traffic analysis for abusive infrastructure characterization," Dec. 2012. Available: https://calhoun.nps.edu/handle/ 10945/25354

[31] V. Yegneswaran, P. Barford, and J. Ullrich, "Internet intrusions: global characteristics and prevalence," in *Proceedings of the 2003 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems* (SIGMETRICS '03). New York, NY, USA: ACM, 2003, pp. 138–147. Available: http: //doi.acm.org/10.1145/781027.781045

[32] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling. (2006). The Nepenthes Platform: An efficient approach to collect malware. [Online]. Available: http://link.springer.com/chapter/10.1007/11856214_9

[33] H. K. and B. Karp. (2004, Aug.). Autograph: Toward automated, distributed worm signature detection. [Online]. Available: http://static.usenix.org/event/sec04/tech/ full_papers/kim/kim_html/

[34] C. Kreibich and J. Crowcroft. (2004, Jan.). Honeycomb – Creating intrusion detection signatures using honeypots. [Online]. Available: http://dl.acm.org/citation.cfm?id=972384

[35] F. Cohen and D. Koike, "Leading attackers through attack graphs with deceptions," *Computers & Security*, vol. 22, no. 5, pp. 402–411, 2003.

[36] B. Cheswick, "An evening with berferd in which a cracker is lured, endured, and studied," in *Proc. Winter USENIX Conference, San Francisco*, 1992, pp. 20–24.

[37] Binary Defense Systems. BinaryDefense/artillery: The Artillery Project is an open-source blue team tool designed to protect Linux and Windows operating systems through multiple methods. [Online]. Available: https://github.com/BinaryDefense/artillery. Accessed 2016 Sep. 12.

[38] M. Andreolini, A. Bulgarelli, M. Colajanni, and F. Mazzoni. (2005). HoneySpam: Honeypots fighting spam at the source. [Online]. Available: https://www.usenix.org/legacy/event/sruti05/tech/full_papers/andreolini/andreolini_html/sruti_05.html

[39] Unspam Technologies, Inc. Frequently Asked Questions (FAQ) | Project Honey Pot. [Online]. Available: https://www.projecthoneypot.org/faq.php#g. Accessed 2016 Sep. 12.

[40] Y. M. Pa Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow. "IoTPOT: Analysing the rise of IoT compromises," in *9th USENIX Work-shop on Offensive Technologies (WOOT 15)*. Washington, D.C.: USENIX Associa-tion, Aug. 2015. Available: https://www.usenix.org/conference/woot15/workshop-program/presentation/pa

[41] A. V. Serbanescu, S. Obermeier, and D. Yu, "ICS threat analysis using a large-scale honeynet," in *Proceedings of the 3rd International Symposium for ICS & SCADA Cyber Security Research* (ICS-CSR '15). Swinton, UK: British Computer Society, 2015, pp. 20–30. Available: http://dx.doi.org/10.14236/ewic/ICS2015.3

[42] L. Rist. (2015, Sep.). Gas tank monitoring system honeypot | The Honeynet Project. [Online]. Available: https://www.honeynet.org/node/1269

[43] J. Verstergaard. (2014, Oct.). Outsmarting the smart meter. [Online]. Available: https://www.honeynet.org/node/1179

[44] D. I. Buza, F. Juhász, G. Miru, M. Félegyházi, and T. Holczer, *CryPLH: Protecting smart energy systems from targeted attacks with a PLC honeypot*. Cham: Springer International Publishing, 2014, pp. 181–192. Available: http://dx.doi.org/10.1007/978-3-319-10329-7_12

[45] L. Rist. (2013, May). Introducing Conpot. [Online]. Available: https://www.honeynet.org/node/1047

[46] Digital Bond Inc. (2011). SCADA honeynet. [Online]. Available: https://digitalbond.com/tools/scada-honeynet

[47] K. Wilhoit and S. Hilt. (2015). The GasPot experiment: Unexamined perils in using gas-tank-tonitoring systems. [Online]. Available: https://documents.trendmicro.com/assets/wp/wp_the_gaspot_experiment.pdf

[48] K. Wilhoit. (2013, Mar.). Who's really attacking your ICS equipment? [Online]. Available: http://www.trendmicro.com.hk/cloud-content/apac/pdfs/security-intelligence/white-papers/wp-whos-really-attacking-your-ics-equipment.pdf

[49] J. Verstergaard. (2014, Oct.). Conpot proxy. [Online]. Available: https://www.honeynet.org/node/1178

[50] L. Spitzner, "The honeynet project: Trapping the hackers," *IEEE Security & Privacy*, vol. 1, no. 2, pp. 15–23, 2003.

[51] S. Litchfield, D. Formby, J. Rogers, S. Meliopoulos, and R. Beyah. (2016). Poster: Re-thinking the honeypot for cyber-physical systems. [Online]. Available: http://www.ieee-security.org/TC/SP2016/poster-abstracts/20-poster_abstract.pdf

[52] S. M. Wade. SCADA honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats. [Online]. Available: http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3130&context=etd. Accessed 2015 Sep. 7.

[53] Siemens. (2014, Dec.). RUGGEDCOM ROS v4.1 User Guide. [Online]. Available: http://www2.schneider-electric.com/resources/sites/SCHNEIDER_ELECTRIC/content/live/FAQS/239000/FA239168/en_US/ROS_RMC30_User-Guide_EN.pdf

[54] infracritical. (2016, Jan.). SCADA/ICS analysis - results from direct connection to the internet. [Online]. Available: https://github.com/infracritical/ruggedtrax

[55] The Honeynet Project. Conpot. [Online]. Available: http://web.archive.org/web/20131010011249/http://conpot.org/. Accessed 2016 Aug 27.

[56] P. So(ò)ky. (2015). Extented functionality of honeypots. [Online]. Available: https://dspace.vutbr.cz/bitstream/handle/11012/52363/16127.pdf?sequence=2

[57] J. M. Audet. (2016, Jan.). IPMItool. [Online]. Available: https://sourceforge.net/projects/ipmitool/

[58] D. Haslinger. Tarpit: artificial delay for all services #67. [Online]. Available: https://github.com/mushorg/conpot/issues/67. Accessed 2016 Sep. 3.

[59] S. Hilt. (2016). GasPot. [Online]. Available: https://github.com/sjhilt/GasPot

[60] H. D. Moore. (2015, Jan.). The internet of gas station tank gauges. [Online]. Available: https://community.rapid7.com/community/infosec/blog/2015/01/22/the-internet-of-gas-station-tank-gauges

[61] Kamstrup A/S. Electricity meters for commercial and industrial applications. [Online]. Available: http://products.kamstrup.com/?goto=205. Accessed 2016 Sep. 20.

[62] A. V. Serbanescu, S. Obermeier, and D. Yu, "A flexible architecture for industrial control system honeypots," in *2015 12th International Joint Conference on e-Business and Telecommunications (ICETE)*, July 2015, vol. 04, pp. 16–26.

[63] XMPP Standards Foundation. XMPP Standards Foundation (XSF). [Online]. Available: https://xmpp.org/about/xmpp-standards-foundation.html. Accessed 2016 Sep. 1.

[64] M. F. T. Holczer and L. Buttyan, "The design and implementation of a plc honeypot for detecting cyber attacks against industrial control systems," in *Proceedings of International Conference on Computer Security in a Nuclear World: Expert Discussion and Exchange*. IAEA, 2015.

[65] Digital Bond Inc. (2016). Digital Bond's ICS enumeration tools. [Online]. Available: https://github.com/digitalbond/Redpoint

[66] Nmap.org. File s7-info. [Online]. Available: https://nmap.org/nsedoc/scripts/s7-info.html. Accessed 2016 May 26.

[67] Nmap.org. File shodan-api. [Online]. Available: https://nmap.org/nsedoc/scripts/shodan-api.html. Accessed 2016 May 26.

[68] anonymous. (2016, Mar.). Internet Census 2012. [Online]. Available: http://internetcensus2012.bitbucket.org/paper.html

[69] Black Hat Asia 2016. (2016). Zhong Chenming. [Online]. Available: https://www.blackhat.com/asia-16/presenters/Zhong-Chenming.html

[70] 404 Team from Knownsec. (2016). ZoomEye - Cyberspace Search Engine. [Online]. Available: https://www.zoomeye.org/help/manual

[71] Z. Durumeric, D. Adrian, A. Mirian, M. Bailey, and J. A. Halderman, "A search engine backed by internet-wide scanning," in *Proceedings of the 22nd ACM Conference on Computer and Communications Security*, Oct. 2015.

[72] J. Matherly. Complete Guide to Shodan. [Online]. Available: http://leanpub.com/shodan. Accessed 2016 Aug. 10.

[73] Z. Durumeric, E. Wustrow, and J. A. Halderman. (2013, Aug.). ZMap: Fast internet-wide scanning and its security applications. [Online]. Available: https://zmap.io/paper.pdf

[74] Censys. (2019). What does Censys scan? [Online]. Available: https://support.censys.io/en/articles/1287810-what-does-censys-scan

[75] R. O'Harrow Jr. (2012, June). Cyber search engine Shodan exposes industrial control systems to new risks. [Online]. Available: https://www.washingtonpost.com/investigations/cyber-search-engine-exposes-vulnerabilities/2012/06/03/gJQAIK9KCV_story.html

[76] N. Bogart. (2016, Jan.). Shodan: The search engine that lets you browse vulnerable baby monitors, webcams. [Online]. Available: http://globalnews.ca/news/2478442/shodan-search-engine-browses-vulnerable-baby-monitors-webcams/

[77] C. Baraniuk. (2016, May). I never imagined a nuclear plant's control system being online. [Online]. Available: https://www.newscientist.com/article/mg23030730-400-i-never-imagined-a-nuclear-plants-control-system-being-online/?cmpid=RSS|NSNS|2012-GLOBAL|magcontents

[78] L. Bruno. (2016, Jan.). shodan.io actively infiltrating ntp.org IPv6 pools for scanning purposes. [Online]. Available: http://seclists.org/oss-sec/2016/q1/219

[79] Shodan. REST API documentation. [Online]. Available: https://developer.shodan.io/api. Accessed 2016 July 19.

[80] Shodan.io. Honeypot or not? [Online]. Available: https://honeyscore.shodan.io/. Accessed 2016 July 7.

[81] B. Radvanovsky and J. Brodsky. (2014, Oct.). Project SHINE (SHodan INtelligence Extraction) findings report. [Online]. Available: http://www.slideshare.net/BobRadvanovsky/project-shine-findings-report-dated-1oct2014

[82] D. Efanov and unknown. plcscan. [Online]. Available: https://github.com/meeas/plcscan. Accessed 2016 July 7.

[83] Siemens AG. (2010). System Software for S7-300/400 System and Standard Functions Volume 1 and Volume 2. [Online]. Available: https://support.industry.siemens.com/cs/document/44240604/system-software-for-s7-300-400-system-and-standard-functions-volume-1-and-volume-2

[84] johnnykv. PyKamstrup. [Online]. Available: https://github.com/johnnykv/PyKamstrup. Accessed 2019 June 15.

[85] Stockshed. (2012, July). MULTICAL ® 62 Water Meter. [Online]. Available: http://kamstrupmeters.com/image/kamstrup/multical62_tech_desc.pdf

[86] P. Mell and T. Grance. (2011, Sep.). The NIST definition of cloud computing. [Online]. Available: http://dx.doi.org/10.6028/NIST.SP.800-145

[87] Netcraft Ltd. Most reliable hosting company sites in Apr. 2015. [Online]. Available: http://news.netcraft.com/archives/2015/05/06/most-reliable-hosting-company-sites-in-apr.-2015.html. Accessed 2016 Apr. 30.

[88] P. Church, H. Mueller, C. Ryan, S. V. Gogouvitis, A. Goscinski, and Z. Tari. (2017). Migration of a SCADA system to IaaS clouds – a case study. [Online]. Available: https://journalofcloudcomputing.springeropen.com/articles/10.1186/s13677-017-0080-5

[89] achillean. (2016, Sep.). Shodan-Python documentation, release 1.0. [Online]. Available: https://media.readthedocs.org/pdf/shodan/latest/shodan.pdf

[90] J. Matherly. The official Python library for Shodan. [Online]. Available: https://github.com/achillean/shodan-python. Accessed 2016 Jan. 15.

[91] Shodan. Honeypot or Not? [Online]. Available: https://honeyscore.shodan.io. Accessed 2016 July 22.

[92] J. Vlachogiannis. jonromero/pyHEC. [Online]. Available: https://github.com/jonromero/pyHEC. Accessed 2019 May 1.

[93] Splunk. About Splunk Free. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/7.3.0/Admin/MoreaboutSplunkFree. Accessed 2019 June 15.

[94] Splunk. spath. [Online]. Available: https://docs.splunk.com/Documentation/Splunk/7.2.6/SearchReference/Spath. Accessed 2019 May 1.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California