SCADA Honeypots – An In-depth Analysis of Conpot

By:

Arthur F Jicha III

A Master's Paper Submitted to the Faculty of the

DEPARTMENT OF MANAGEMENT INFORMATION SYSTEMS

ELLER COLLEGE OF MANAGEMENT

In Partial Fulfillment of the Requirements

For the Degree of

MASTER OF SCIENCE

In the Graduate College

THE UNIVERSITY OF ARIZONA

2016

STATEMENT BY AUTHOR

This thesis has been submitted in partial fulfillment of requirements for an advanced degree at the University of Arizona.

Brief quotations from this thesis are allowable without special permission, provided that an accurate acknowledgement of the source is made. Requests for permission for extended quotation from or reproduction of this manuscript must be obtained from the author.

SIGNED: Arthur F Jicha III

APPROVAL BY MASTERS PAPER ADVISOR

This Masters Paper has been approved on the date shown below:

_____ MM/DD/2016

Dr. Hsinchen Chen Date

Advisor Title of Management Information Systems

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

SCADA honeypots are key tools in determining not only threats which pertain to SCADA devices in the wild, but also as an early detection mechanism of potential malicious tampering within a SCADA device network. An analysis of one such SCADA honeypot, Conpot, will be conducted to determine its viability as an effective SCADA emulating device. A long term analysis is conducted and a simple scoring mechanism is leveraged to evaluate Conpot.

1 INTRODUCTION

In a world where the value of information is ever increasing, hackers are consistently targeting governments, corporations, and individuals to obtain valuable secrets, proprietary data, and personally identifiable information (PII). To better understand the landscape of where these attacks are originating, honeypots can be used to not only conduct research on threats within the wild, but also notify if a potential threat is within one's network. Supervisory Control and Data Acquisition (SCADA) Systems have since also become a target and with the advent of SCADA honeypots, early notification of SCADA devices being tampered with or new knowledge of threats pertaining to said devices can be determined and analyzed.

2 PROBLEM DEFINITION / BACKGROUND

2.1 Literature Review

2.1.1 Background Area 1 (Honeypots)

Within Honeypots, there are three different types: low-interaction, high-interaction, and hybrid. The low-interaction honeypots serves the purpose to "simulate only basic network services (or only a base part of the basic network services)" where as a high-interaction honeypot simulates

much more complex services (Buza et al., 2014). Furthermore, Buza et al. expands on the primary advantage of low-interaction honeypots being much easier to design and maintain and tend to be much more stable. High-interaction honeypots on the other hand are much harder to implement because of far more complexities but also are much harder to detect and make great targets for a potential SCADA system hacker. Hybrid honeypots attempt to bridge the advantages between the high and low interaction honeypots while minimizing on their downfall. A summary of the primary differences can be seen in *Table 1*.

Low-interaction	High-interaction
Solution emulates operating systems and services	No emulation, real operating systems and services are provided
Easy to install and deploy. Usually requires simply installing and configuring software on a computer.	Can capture far more informaiton including new tools, communications, or attacker keystrokes.
Minimal risk, as the emulated services control what attackers can and cannot do.	Can be complex to install or deploy (commercial versions tend to be much simpler).
Captures limited amounts of information main transactional data and some limited interaction	Increased risk, as attackers are provided real operating systems to interact with

Table 1: Honeypot Type Summary (Wade et al., 2011)

2.1.2 Background Area 2 (SCADA Specific Honeypots)

SCADA Specific Honeypots serve the purpose to effectively simulate a SCADA system. A typical SCADA system is composed of four parts: “a central computer (host), a number of field-based remote measurement and control units known as Remote Terminal Units (RTUs), a wide area telecommunications system to connect them, and an operator interface to allow the operator to access the system” (Wade et al., 2011). One of the first attempts at emulating a PLC device was conducted by members of the Cisco Critical Infrastructure Assurance Group (CIAG) in March of 2004. The use of Honeyd to simulate services typically used by a PLC included:

- TCP/IP Stack of the PLC

- Modbus/TCP server implementation
- FTP server
- Telnetd server
- HTTP server

Cisco's CIAG ended support for the project in 2005 and the scripts utilized with Honeyd were incomplete. Also, services were "only partially implemented and the realized functionality is not realistic nor interactive" and various bugs and mistakes were present within the Python code that was developed.

Conpot is another option for a low-interactive SCADA honeypot and serves the purpose of being extremely easy to implement. "It supports the simulation of protocols such as HTTP, Modbus, and SNMP, as well integrating real PLC devices" (Serbanescu et al., 2015). The Conpot project by The Honeynet Project was released in May 2013. This particular honeypot utilizes a logging system to monitor changes that are potentially made. The honeypot logs events of HTTP, SNMP and Modbus services with millisecond accuracy and offers basic tracking information such as source address, request type, and resource requested in the case of HTTP (Buza et al., 2013).

One of the most advanced SCADA honeypots created is the SCADA Honeynet which is maintained by Digital Bond. The Honeynet uses two virtual machines, one of which is a Generation III honeywall extended with IDS signatures from Digital Bond's Quickdraw (Buza et al., 2013). The honeywall acts as a mechanism to monitor and log every malicious attack that may occur against the PLC. The second virtual machine simulates a popular PLC that runs five services which include FTP, Telnet, HTTP, SNMP, Modbus, and TCP. "The FTP, HTTP and Modbus

services are implemented by different Java applications while the Telnet and SNMP services are realized by Python scripts" (Buza et al., 2013)

Previous research can be found summarized in Table 2.

Study	Focus	Testbed	Methods	Findings
Serbanescu, et al. (2015)	Honeynet	Amazon EC2, Snort	Honeynet utilizing Modbus, DNP3, ICCP, IEC-104, SNMP (v1/2/3), TFTP, XMPP	Listings on SHODAN led to non-SHODAN peer interactions, positive correlation between discovery on SHODAN and interactions
Buza, et al. (2014)	CryPLH	Vmware	HTTP, HTTPS, SNMP, ISO-TSAP	Effective simulation of Siemens PLC via Linux
Scott, et al. (2014)	Conpot	Syslog, Splunk	SNMP, MODBUS, HTTP	Utilized Conpot in pre-existing airgapped SCADA network
Buza, et al. (2013)	Custom Linux Based	Vmware	HTTP, HTTPS, SNMP, ISO-TSAP	Effective simulation of HTTP, HTTPS, ISOTSAP, SNMP from Siemens PLC while also allowing for logging of attacker actions in high interaction environment
Wilhoit, et al. (2013)	Custom	Snort	HTTP	Accurately detected attacker locations utilizing BeEF, focused on targeted attacks versus drive-by or automatic approaches
Wade, et al. (2011)	Digital Bond Honeynet	Vmware, Snort	Telnet, FTP, SNMP, HTTP, MODBUS, VxWorks Debugger	Active probing of honeypot setup, however limited interactions while deployed on university network

Table 2: Summary of Prior SCADA Honeypot Studies

2.1.3 Research Gaps

After conducting the literature review of SCADA honeypots, a gap was identified regarding the analysis of the effectiveness of various honeypots. Various studies were found that detailed the interactions that occurred with a given honeypot, i.e. Digital Bond Honeynet and Conpot, however the actual effectiveness of any given honeypot has not been conducted. The closest approach to this field of study was *Evaluating Low Interaction Honeypots and On their Use against Advanced Persistent Threats* by Fronimos, et. al. Fronimos's paper however analyzed various general honeypots, not *only* SCADA honeypots, and provided an extremely high level analysis of the

various facets of the honeypots reviewed whereas this Masters Paper looks more in depth on the use of Conpot.

3 EXPERIMENT / METHODOLOGY / CONTENT

3.1 Introduction / Approach

In order to conduct a full analysis of the SCADA honeypot Conpot, an image was created and then used to create multiple instances across Amazon Web Services' zones. The logs were analyzed on April 11th after the honeypots had been running since March 25th. For the purposes of consistency in long term up-time, the honeypots were further analyzed on April 27th. The following section outlines the steps for setup and process for creating instances of Conpot.

3.2 Conpot Setup

3.2.1 Image Creation

In order to expedite the process of the Conpot setup, Amazon Web Services was utilized for deployment across the globe. Installation of Conpot is quite simple, however certain dependencies are necessary for it to fully function. Due to the age of some of the packages necessary, certain repositories must be manually added. Ubuntu 12.04 was used as the base operating system for a micro-instance within AWS and after configuring basic settings and conducting updates, the following commands were used to finish the Conpot install:

- sudo apt-get install git
- sudo vim /etc/apt/sources.list
  - Add this line
    "deb http://us.archive.ubuntu.com/ubuntu precise main multiverse"
- sudo apt-get update
- sudo apt-get install git
- sudo apt-get install gcc-4.9 build-essential autoconf libtool pkg-config
- sudo apt-get install python-dev

- sudo apt-get install libmysqlclient-dev
- git clone https://github.com/glastopf/conpot.git
- cd /opt
- sudo git clone https://github.com/glastopf/conpot.git
- cd conpot
- sudo apt-get install gcc
- sudo apt-get install libsmi2ldbl snmp-mibs-downloader python-dev libevent-dev libxslt1-dev libxml2-dev
- sudo python setup.py build
- sudo python setup.py install
- sudo conpot

After adding the repository "deb http://us.archive.ubuntu.com/ubuntu precise main multiverse", the necessary apt-get packages were made available for final install. Finally getting to the point where Conpot actually worked, however, took some time due to discrepancies between different installation instructions. Through a process of trial and error, the aforementioned command list was finalized for a fully working installation of Conpot. Successful installation of Conpot will yield the following menu after entering the command *sudo conpot*:

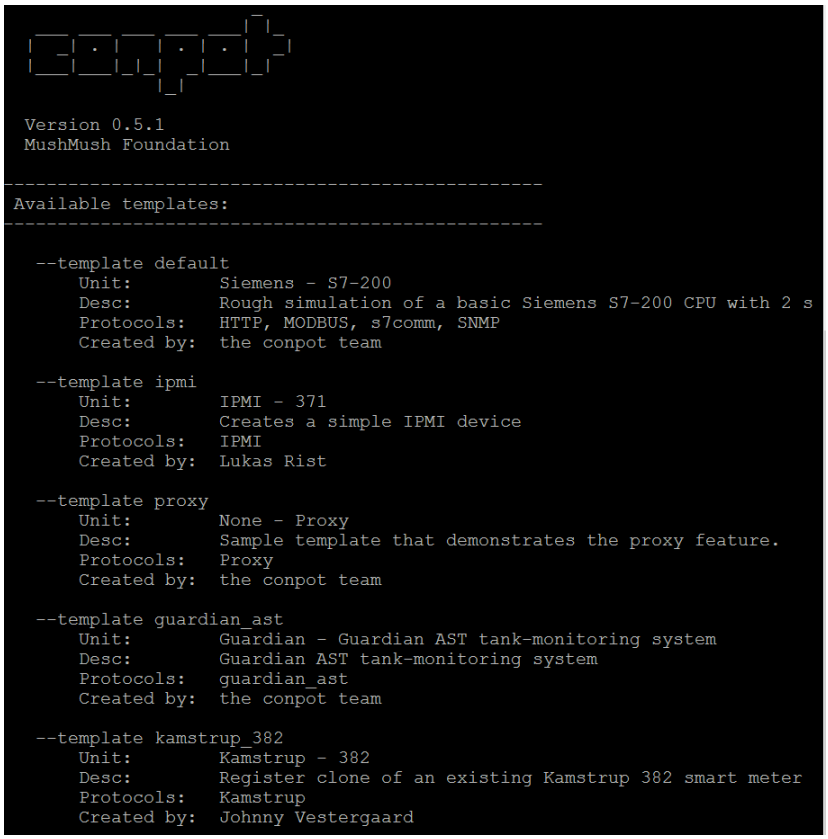

Figure 1: Conpot initial start screen

The aforementioned output displays the various templates that are available to the current iteration of Conpot at the time of this writing (*version 0.5.1*). Currently, five different "templates are available to the honeypot user which includes:

- Default: Siemens S7-200 ICS
- IPMI: Simple IPMI (intelligent platform management interface) device
- Proxy: proxy functionality demo
- Guardian AST: Guardian AST gas tank sensor
- Kamstrup 382: Emulation of a Kamstrup 382 Smart meter

During the analysis portion, the default template (Siemens S7-200 ICS) and Guardian AST template (guardian_ast) will have an in-depth review while the IPMI and Kamstrup devices will have a more surface level overview.

3.2.2 Amazon Web Services Deployment

After successfully obtaining the Conpot start screen, the AWS micro-instance was shut down so that an image could be created:

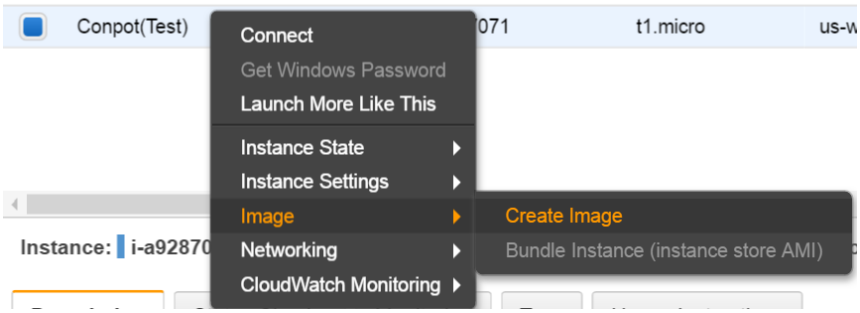

Figure 2: AWS image creation

Utilizing the "Create Image" function within AWS, the image was then added to the Images – AMI folder for deployment. This image is then also able to propagate to various deployment zones within the AWS infrastructure. After deploying the image twice in various zones (see *Figure 5* for an example), the SCADA honeypots were then accessed via SSH to finalize their deployment once the instances were in a "running" state.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS	Public IP
Conpot	i-da071759	t1.micro	sa-east-1a	running	2/2 checks ...	None	ec2-54-207-96-59.sa-ea...	54.207.96.59
Conpot(GasT...	i-dd07175e	t1.micro	sa-east-1a	running	2/2 checks ...	None	ec2-54-232-248-38.sa-e...	54.232.248.38

Figure 3: Amazon Web Services Dashboard for Sao Paulo

Another advantage to leveraging Amazon Web Services is its key management and port security options. Each instance of the Conpot which was to be deployed was setup to allow all ports to be accessible for the most accurate review of port information when running any given template

within the honeypot. Furthermore, the key pair options made obtaining access to each instance extremely easy. After obtaining the private key necessary to create a connection, each instance was generated using the same public key information so that merely using the private certificate combined with the password during creation allowed easy access to the various deployments of Conpot that were used.

After accessing each honeypot, the following command was utilized to start the Conpot with the designated template:

- sudo conpot --template [template name]

If a template name is not selected, the default option of "default" is used. For the purposes of the honeypot analysis, an in-depth review of both the Guardian AST and default Siemens S7-200 ICS, however a brief analysis of the IPMI - 371 and Kamstrup – 382 smart meter will also be conducted.

The following table summarizes the deployed honeypots by their location, IP and details of which template was utilized.

Location	Name	IP	Details	Zone
us-east-1a	Conpot1	52.23.225.126	Default template	US East (N. Virginia)
us-east-1a	Conpot2	54.86.249.160	Emulation of gas tank level	US East (N. Virginia)
us-west-2b	Conpot3	52.36.62.44	Default template	US West (Oregon)
us-west-2b	Conpot4	52.32.45.32	Emulation of gas tank level	US West (Oregon)
eu-west-1b	Conpot5	52.30.167.154	Default template	EU (Ireland)
eu-west-1b	Conpot6	52.19.95.69	Emulation of gas tank level	EU (Ireland)
ap-northeast-1c	Conpot7	52.192.20.179	Default template	Asia Pacific (Tokyo)
ap-northeast-1c	Conpot8	52.196.47.205	Emulation of gas tank level	Asia Pacific (Tokyo)
ap-southeast-1b	Conpot9	54.254.141.38	Default template	Asia Pacific (Singapore)
ap-southeast-1b	Conpot10	54.254.140.52	Emulation of gas tank level	Asia Pacific (Singapore)
sa-east-1a	Conpot11	54.207.96.59	Default template	South America (Sao Paulo)
sa-east-1a	Conpot12	54.232.248.38	Emulation of gas tank level	South America (Sao Paulo)

Table 2: AWS Conpot deployment zone information

An issue was noticed regarding the execution of the Python scripts to run Conpot. After disconnecting the terminal session, Putty was used in this case, the Python code would stop being executed. The program "screen" was therefore used to prevent sessions from terminating. Using screen is simple enough, after using the command *screen* and going through the necessary text, typing in the newly provided terminal and using the follow commands fixed the issue:

- Ctrl + a, Command tells screen to be used in the active window
- Ctrl + d, Command tells screen to disconnect from the active window, while still keeping the session alive in the background
- Screen –r, Command connects to a session that is residing in the background, however if multiple sessions are currently in the background, a number must be used to specify which session to rejoin

3.2.3 Work completed (Process)

3.3 Data or Results

This section goes over the data collected. During the following section, the discussion, the results will be analyzed in better detail. The following table lists the SCADA honeypots and the ports that should be found open within each one respectively.

Honeypot Type	Open Ports
Siemens S700	80, 102, 161, 502, 623, 47808
Guardian AST	10001
IPMI	623
Kampstrup Smart Meter	1025, 50100

Table 3: Conpot templates and corresponding ports

3.3.1 Nmap Scan Data

Nmap was utilized to check the ports which were open after running the Python scripts to start Conpot. To allow a better comparison, a "vanilla" installation of Ubuntu was deployed and scanned to show what ports by default were showing up. The follow nmap scanning commands were used:

- nmap -A -v [IP Address]
- nmap -A -v -Pn [IP Address]
- nmap -A -v -Pn -p- [IP Address]

nmap was used in a staged approach to show what different scanning techniques showed as the open port results. The flag -A results in nmap turning on, "version detection and other Advanced and Aggressive features" (nmap.org). Although this scanning technique is more intrusive and less stealthy due to its aggressive scanning and OS detection, it provides a good representation of what to expect for identification. Using the -Pn resulted in nmap not utilizing pings when conducting its scans to determine if the host was up. For the purposes of the analysis, the virtual machines were already known to be operational and in some cases, their configurations rejected pings. The -p- flag was also used to conduct a scan over the entire port range (ports 1-65535). Lastly, the flag -v was used to, although it was deemed not necessary due to the –A flag already including version detection.

Honeypot Type	**Result**
Siemens S700	22, 80
Guardian AST	N/A
IPMI	N/A
Kampstrup Smart Meter	N/A

Table 4: Nmap scanning (utilizing flags –v and –A)

Scanning with the -v and -A flags resulted in no results from the Guardian AST, IPMI, and Kampstrup smart meter due to pings being rejected.

Honeypot Type	Result
Siemens S700	22, 25, 80, 514, 6009, 8443
Guardian AST	22, 25, 514, 6004, 10001
IPMI	22
Kampstrup Smart Meter	22, 25, 514, 1025, 1068

Table 5: Nmap scanning (utilizing –v, -A, and -Pn flags)

After utilizing the -Pn flag to stop the ping option during scans, many more ports were identified across the various usable templates within Conpot.

Honeypot Type	Result
Siemens S700	22, 80, 102, 502, 514, 2000, 5060, 8008, 8020, 18556
Guardian AST	22, 514, 2000, 3826, 5060, 8008, 8020, 10001, 11190, 19116, 36123, 43787, 48191, 63790
IPMI	22, 2000, 5060, 8008, 8020
Kampstrup Smart Meter	22, 514, 1025, 2000, 4368, 5060, 8008, 32469, 50100, 52245, 57565
Vanilla Ubuntu Install	22, 514, 2000, 5060, 8008, 8020, 38051, 38093, 47785

Table 6: Nmap scanning (utilizing –v –A –Pn and -p- flags)

As a final scan to compare against, *all* ports were scanned to determine what a full nmap scan would show as open port results. On average the scans took around three to four hours to fully process due to the intensity of the scans.

3.3.2 SHODAN Scan Data

SHODAN was also used to determine which ports it saw as open within the various Conpot templates. Unfortunately, the IPMI and Kampstrup templates were never identified by SHODAN due to time constraints.

Honeypot Type	SHODAN Port Scan Result
Siemens S700	22, 80, 102, 161
Guardian AST	10001
IPMI	N/A
Kampstrup Smart Meter	N/A

Table 7: SHODAN Scan data results

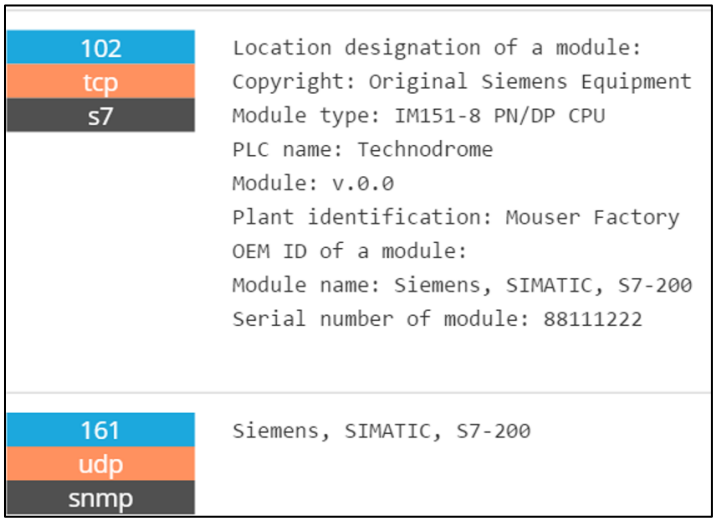

Figure 4: SHODAN Default template (Siemens S7-200) Scan Result (Not fully exhaustive, results for port 22 and 80 not displayed)

The previous screen shot shows two ports which SHODAN identified as open. Ports 22 and 80 are not shown due to being typical responses whereas the information from ports 102 and 161 of the Siemens S7-200 demonstrate effective emulation of the device. The banners associated with each respective port also demonstrate the template's alignment to the actual Siemens SIMATIC S7-200 device.

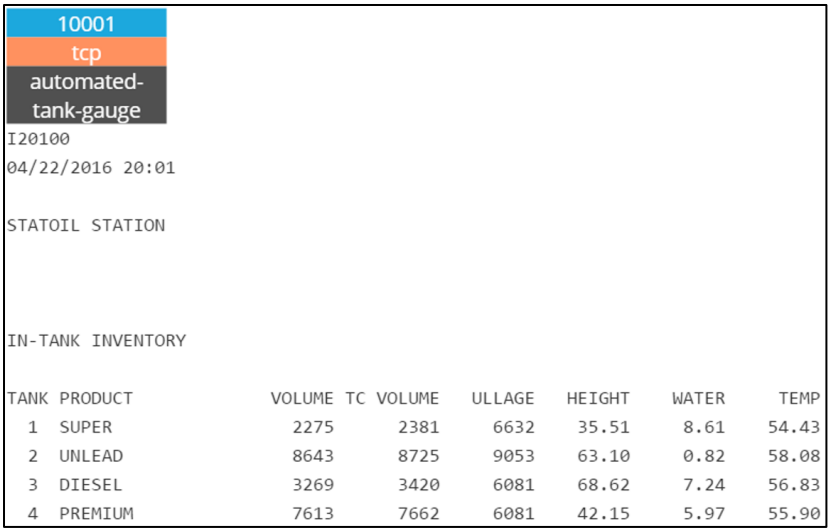

Figure 5: SHODAN Gas Tank template (Guardian AST) Scan Result

Figure 7 shows the full results of SHODAN's scan regarding the Conpot Guardian AST device. Only port 10001 was identified and the banner pulled shows actual data readouts from various gas products within the tank inventory.

3.3.3 Log Parsing

Conpot does not come pre-packaged with any built-in analysis mechanism, only a mere basic logging system. The following demonstrates what the log looks like:

```
2016-03-29 22:18:16,763 HTTP/1.1 response to ('91.196.50.33', 51895): 404.
06bf6ce0-e18c-4e9f-968f-22d8a18ae8f0
2016-03-30 00:28:26,877 SNMP Exception: This class is not converted to new
architecture
2016-03-30 02:04:59,010 New IPMI traffic from ('184.105.247.250', 33919)
2016-03-30 02:04:59,011 New IPMI session initialized for client (('184.105.
247.250', 33919))
2016-03-30 02:04:59,011 Connection established with ('184.105.247.250', 339
19)
2016-03-30 02:04:59,011 IPMI response sent to ('184.105.247.250', 33919)
2016-03-30 02:09:35,441 New http session from 89.248.174.4 (cfb8ca92-474b-4
675-a389-0a6b3fc6de53)
2016-03-30 02:09:35,441 HTTP/1.1 GET request from ('89.248.174.4', 55768):
('/xmlrpc.php', ['Host: 52.23.225.126\r\n'], None). cfb8ca92-474b-4675-a389
-0a6b3fc6de53
2016-03-30 02:09:35,441 HTTP/1.1 response to ('89.248.174.4', 55768): 404.
cfb8ca92-474b-4675-a389-0a6b3fc6de53
```

Figure 6: Example snippet of conpot.log for default template

To conduct a basic analysis of the aforementioned log, the following commands were used to parse IP information:

- grep -o '[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}' conpot.log > output.txt
- sort -u output.txt >> outputCleansed.txt

The first command uses the Linux grep program to parse the log and display any given IP address found within the Conpot log. The results were placed in a semi-processed file called output.txt which was then further processed by the second command which sorted IP addresses from smallest to greatest, numerically, and kept only unique IP addresses based on the -u flag. Further processing was also conducted to remove the Conpot's own IP address from the result set. These commands were used to obtain IP results to yield the following tables:

SCADA Interactions - GuardianAST (Gas Tank)	
Country	**Interaction Count**
Netherlands	4
United States	5

Table 8: Originating countries for Conpot Guardian AST interactions

SCADA Interactions - Siemens Siamatic PLC	
Country	**Interaction Count**
United States	446
China	130
Netherlands	59
France	36
Poland	26
Russian Federation	20
Germany	18
Taiwan; Republic of China (ROC)	11
Switzerland	10
United Kingdom	9
Brazil	7
Moldova Republic of	7
Ukraine	7
Korea Republic of	5
Lithuania	5
Iceland	4
Indonesia	4
Italy	4
Sweden	4

Table 9: Originating countries for Conpot Siemens SIAMATIC S7-200 interactions (only

Both Table 8 and Table 9 show basic connection information for each corresponding Conpot template that was reviewed in-depth, Guardian AST and Siemens SIAMATIC S7-200 respectively.

3.4 Discussion

3.4.1 Scan Data Analysis

A very interesting finding in the nmap scan data is that while the Guardian AST, Kampstrup, and IPMI devices all denied pings, the Siemens SIAMATIC S7-200 did not (Table 4). When removing the ping option for the result set in Table 5, the results were more comprehensive and revealing. In every scan result, port 22 was shown as open, which of course would be the case due to utilizing

SSH to gain access to each honeypot via a terminal in Putty. When comparing what should have been seen as open ports from Table 3 for each respective template within Conpot to the results from Table 5, nmap failed to determine that the following ports were open on their respective devices:

- Siemens S7-200: 102, 161, 502, 623, 47808
- IPMI: 623
- Kampstrup Smart Meter: 50100

However, these ports may not have been found due to not being part of the top 1000 which nmap commonly scans without being directed to scan each and every port. To that point, nmap was eventually set to scan each and every port which results are shown in Table 6. After scanning all ports, not all ports which should have been open according to Table 3 were found. The results are as follows for ports which were not found:

- Siemens S7-200: 161, 623, 47808
- IPMI: 623

These findings are very perplexing due to, in the case of the Siemens device, SHODAN being able to find that port 161 was open and yielding a banner as shown in Figure 4. As previously discussed, all ports were left open in the AWS firewall settings, so these ports should have been found during the complete comprehensive scan of all ports. What was more surprising during the full comprehensive scan was the number of ports that were not expected to be open at all within Table 6. Due to the large variety of ports that were discovered to be open, the “vanilla” install of the Ubuntu image was deployed without running any Conpot template. Based on these initial results, it appears that more ports are being opened than was originally anticipated when running any given

Conpot template. Further analysis will need to be conducted to determine if there are indeed extra ports being opened that might be indicative of a honeypot instead of an effective emulation.

The results from the SHODAN scan were also very insightful in that they more accurately showed the Conpot instances as being SCADA devices. This is primarily because SHODAN focuses its scan results on a much smaller port set, which resulted in the results not showing the large amount of open ports that were shown in the all port scan of nmap. The most intriguing finding here, as previously mentioned, is that SHODAN found port 161 open on the Siemens device while nmap did not. The banner grabbed by SHODAN also showed that the device was a Siemens SIAMATIC S7-200 device. These findings may show that nmap is indeed not fully effective in determining ports that are actually open. Unfortunately, at the time of this writing, SHODAN had not discovered the IPMI and Kampstrup devices, so comparing SHODAN results of these devices with the nmap port scans was not available.

3.4.2 Conpot Overview Scoring

Conpot will be scored in six areas: setup, ease of use, documentation, logging, uptime, analysis. Within each of these areas, a qualitative measure will be used (Below Average, Average, and Above Average) to represent Conpot's effectiveness as a SCADA Honeypot.

Setup – Above Average

Overall setup for Conpot is extremely easy. Although there was not a single designated instruction set that was found to work 100%, an effective install was derived from using multiple guides on the internet which eventually resulted in a working Conpot deployment.

Ease of Use – Above Average

Activating Conpot is easy enough. After Conpot receives a short command with sudo privileges, it is able to execute the necessary Python scripts to begin emulating whichever device is chosen from its template list. One issue identified with Conpot is that after accessing the Honeypot and activating the Python scripts to run, the job ends up ending when disconnecting from the terminal. To remedy this issue, the program screen was used to force the session to persist after a terminal session with Putty was ended. Utilizing screens ability to keep commands persistent was quite simple and considering it is available in a lot of Linux installations, it is readily available.

Documentation – Average

Various pieces of documentation were found, however the final direction set utilizes was composed of multiple guides to have a product that worked. Dependency issues often occurred and some packages that were attempted to be installed were not able to be found due to not existing in the default repositories within Linux. This problem would have been easy to overcome if guides specified alternative repositories to use, but often times they did not as packages and software versions were deprecated. Conpot also has the ability for individuals to use their own templates if they so choose, however instructions for actually creating these templates are miniscule at best.

Logging – Average

Conpot's logging feature is quite in-depth in that it records literally every interaction it has. However this also creates a ton of noise with little to no ability to change what is logged or not for an individual's interest. Having no internal analysis software means that the logs must have parsers built to merely identify any given type of behavior. Conpot does have an aggregate collector which goes the creators of Conpot, called HPFeeds, however at the time of this writing the website was

not available. After a flag within Conpot is set to on, information the Conpot collects is sent to this aggregator for analysis, and details are actually made available to participants of HPFeeds.

Uptime – Above Average

Not all templates of Conpot were tested for longevity, however the templates which were, Guardian AST and Siemens SIAMATIC S7-200, proved to be quite resilient. In total the SCADA honeypots ran from March 25th until the default Siemens unit stopped serving webpages across port 80 on April 22nd. The cause for this was not determined as nothing was found noticeable within the logs, however simply stopping the Python script and restarting it resulted in the issue being corrected. At the time of this writing the Guardian AST was still consistently running when it was checked on April 29th. Overall, the uptime of the Conpot is actually quite incredible.

Analysis – Below Average

As previously mentioned, no internal analysis system is in place to analyze the data that Conpot collects. As such, analyzing the logs requires another program or scripts to be used for parsing. Conpot receives a below average score for this based on the fact that their logs are also difficult to work with. Had the logs been in a format such as .csv, it would be much easier to manipulate, however at this point in time, using a program such as Excel to read through the log means column breaks must be used.

4 CONCLUSION

In conclusion, Conpot is an extremely effective SCADA honeypot that is very versatile based on the various devices it is capable of emulating. However, Conpot also has a long way to come in

regards to being more beneficial in the analysis realm. It may actually be one of the easiest honeypots to not only install but also deploy, but the lack of analysis really hinders its ability to fully analyze the results. Future work that could be done would be to utilize a SIEM such as Splunk to conduct an analysis on the log data to see if better results could be leveraged from the conpot.log file. Lastly, to better analyze the Conpot templates for their authenticity of accurately emulating SCADA devices, further work could be to compare the actual devices versus their emulated version to see how they compare, especially at the packet level.

5 REFERENCES

Buza, D. I., Juhasz, F., & Miru, G. (2013). Design and implementation of critical infrastructure protection system. *Budapest University of Technology and Economics*, 1-58. Retrieved March, 2016, from http://tdk.bme.hu/VIK/DownloadPaper/Kritikus-infrastruktura-vedelmi-rendszer

Buza, D. I., Juhasz, F., Miru, G., Felegyhazi, M., & Holczer, T. (2014). CryPLH: Protecting Smart Energy Systems from Targeted Attacks with a PLC Honeypot. *Springer International Publishing Switzerland*, 1-12. Retrieved March, 2016, from https://www.crysys.hu/~mfelegyhazi/publications/Buza2014cryplh.pdf.

Chinn, R. (2015). *Botnet Detection: Honeypots and the Internet of Things* (Unpublished doctoral dissertation). University of Arizona.

Fronimos, D., Magkos, E., & Chrissikopoulos, V. (2014). Evaluating Low Interaction Honeypots and On their Use against Advanced Persistent Threats. *Proceedings of the 18th Panhellenic Conference on Informatics - PCI '14*, 1-6. Retrieved March, 2016, from http://dl.acm.org/citation.cfm?doid=2645791.2645850

Scott, C. (2014). Designing and Implementing a Honeypot for a SCADA Network. *SANS Institute InfoSec Reading Room*, 1-39. Retrieved March, 2016, from https://www.sans.org/reading-room/whitepapers/detection/designing-implementing-honeypot-scada-network-35252.

Serbanescu, A. V., Obermeier, S., & Yu, D. (2015). ICS Threat Analysis Using a Large-Scale Honeynet. *BCS Learning & Development Ltd.,* 1-11. Retrieved March, 2016.

Wade, S. M. (2011). SCADA Honeynets: The attractivenss of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats (Doctoral dissertation, Iowa State University) (pp. 1-67). Iowa State University.

Wilhoit, K. (2013). The SCADA That Didn't Cry Wolf. *Blackhat 2013,* 1-24. Retrieved March, 2016, from https://media.blackhat.com/us-13/US-13-Wilhoit-The-SCADA-That-Didnt-Cry-Wolf-Whos-Really-Attacking-Your-ICS-Devices-Slides.pdf