

6-19-2014

Distinguishing Internet-facing ICS devices using PLC programming information

Paul M. Williams

Follow this and additional works at: <https://scholar.afit.edu/etd>

Recommended Citation

Williams, Paul M., "Distinguishing Internet-facing ICS devices using PLC programming information" (2014). *Theses and Dissertations*. 524.
<https://scholar.afit.edu/etd/524>

This Thesis is brought to you for free and open access by the Student Graduate Works at AFIT Scholar. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of AFIT Scholar. For more information, please contact richard.mansfield@afit.edu.



**DISTINGUISHING INTERNET-FACING ICS DEVICES USING PLC
PROGRAMMING INFORMATION**

THESIS

Paul M. Williams, Major, USA

AFIT-ENG-T-14-J-41

**DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY**

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the United States Air Force, the Department of Defense, or the United States Government.

This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

AFIT-ENG-T-14-J-41

DISTINGUISHING INTERNET-FACING ICS DEVICES USING PLC
PROGRAMMING INFORMATION

THESIS

Presented to the Faculty
Department of Electrical and Computer Engineering
Graduate School of Engineering and Management
Air Force Institute of Technology
Air University
Air Education and Training Command
in Partial Fulfillment of the Requirements for the
Degree of Master of Science in Cyber Operations

Paul M. Williams, B.S.E.E.T, M.A.I.T.M

Major, USA

June 2014

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

DISTINGUISHING INTERNET-FACING ICS DEVICES USING PLC
PROGRAMMING INFORMATION

Paul M. Williams, B.S.E.E.T, M.A.I.T.M
Major, USA

Approved:

<hr/> <i>//signed//</i> <hr/> Maj Jonathan Butts, PhD (Chairman)	<hr/> 28 May 2014 <hr/> Date
<hr/> <i>//signed//</i> <hr/> Lt Col David J. Robinson, PhD (Member)	<hr/> 28 May 2014 <hr/> Date
<hr/> <i>//signed//</i> <hr/> Juan Lopez Jr., (Member)	<hr/> 28 May 2014 <hr/> Date

Abstract

The Shodan search engine reveals Industrial Control System (ICS) devices around the globe are directly connected to the Internet. After Shodan's inception in 2009, multiple news reports have focused on the increased threat to infrastructure posed by Shodan. While no attacks to date have been directly attributed to Shodan searches, its existence provides an anonymous reconnaissance platform that facilitates ICS targeting for those actors with both a desire and capability to carry out attacks. Recent research has demonstrated that simple search queries return thousands of ICS devices indexed by Shodan, and the number of newly indexed ICS devices is growing. This research discusses the method used to distinguish the Internet-facing ICS devices indexed by the Shodan search engine. PLC code is obtained by sending specifically crafted CIP request messages to the devices, capitalizing on the fact that authentication is not built in to the CIP application layer protocol. This data allows categorization of Internet-facing devices by comparing PLC code attributes. The results of this research show PLC code can be collected from Internet-facing ICS devices with no significant impact to task execution times. Also, this research demonstrates a method to distinguish Internet-facing ICS devices by function and by Critical Infrastructure sector. This capability develops an understanding of the function and purpose of ICS devices that are being connected to the Internet.

I dedicate this to my wife and children for tolerating me in this endeavor.

Acknowledgments

I would like to thank my two Academic Advisors, Lt Col Robinson and Maj Butts along with Juan Lopez and Steven Dunlap for their leadership and motivation during the course of this research. I would also like to thank the faculty and students at AFIT who helped me think through some of these problems. Finally, I would like to thank Dr. Nathaniel Davis who took the time to ensure I had the tools necessary to compete and succeed in this program. Lion Brigade Sir!

Paul M. Williams

Table of Contents

	Page
Abstract	iv
Dedication	v
Acknowledgments	vi
Table of Contents	vii
List of Figures	x
List of Tables	xiii
List of Acronyms	xiv
I. Introduction	1
1.1 Problem Statement	1
1.2 Scope, Assumptions and Limitations	2
1.3 Approach	3
1.4 Overview of Subsequent Chapters	4
II. Background	6
2.1 Critical Infrastructure and Industrial Control Systems	7
2.1.1 SCADA network architectures	8
2.1.2 ICS Components	11
2.1.3 Allen-Bradley Industrial Control Systems	14
2.1.4 ICS Protocols	19
2.2 The Shodan Search Engine	21
2.2.1 Discovering Indexed ICS Devices	24
2.2.2 Project SHINE	26
2.3 Security Concerns and ICS Attacks	26
2.3.1 TrendMicro Reports on ICS Attacker	28
2.3.2 Shodan Impacts on ICS devices Pre/Post Indexing	31
2.4 Related Work	32
2.5 Conclusion	33

	Page
III. Methodology	34
3.1 Problem Definition	34
3.2 Approach	35
3.2.1 Evaluating Collection Impact on PLCs	36
3.2.2 PLC Code Collection Script Development	37
3.2.2.1 EtherNet/IP Connection Setup	38
3.2.2.2 CIP Attribute Requests	38
3.2.3 Detecting Process Control Terms in PLC Programming Information	40
3.3 Environment	42
3.3.1 Architecture and Hardware	42
3.3.2 Software	42
3.3.3 PLC Code collection tools	44
3.3.4 Performance Analysis Tools	44
3.3.5 Implementation	44
3.4 Experiment Design	45
3.4.1 Experiment Setup	45
3.4.2 PLC Execution Times	46
3.4.3 PLC Impact Experiments	47
3.4.4 Visual Inspection of PLC Code	49
3.5 Evaluation	50
3.6 Conclusion	51
IV. Results and Implementation	52
4.1 Exploratory Testing	52
4.1.1 Reverse Engineering Allen-Bradley PLC Code CIP Requests	52
4.1.2 Conclusion	59
4.2 Performance Analysis Results	59
4.2.1 Data	59
4.2.2 Non-parametric Statistical Analysis	62
4.2.3 Analysis	64
4.2.3.1 L61 CPU	64
4.2.3.2 L71 CPU	65
4.2.3.3 L23E CPU	65
4.2.3.4 L32E CPU	67
4.2.4 Discussion	70
4.3 Implementation Results	73
4.3.1 Identifying Internet-facing ICS devices	73
4.3.2 Collecting Diagnostic Web Page Data	75
4.3.3 Collecting PLC Code	76
4.3.4 Distinguishing ICS Devices	76

	Page
4.3.5 Analysis	77
4.4 Conclusion	77
V. Conclusions	79
5.1 Conclusions	79
5.2 Impact	79
5.3 Recommendations	80
5.4 Future Work	80
5.4.1 Machine Learning and Process Control Term Matching	80
5.4.2 Distinguishing Internet-facing ICS Devices by Sector	81
5.4.3 Determining Methodology Portability	81
5.5 Summary	82
Appendix A: 1: Process Control Terms	83
Appendix B: 2: List of Boxplots	88
Bibliography	95

List of Figures

Figure	Page
2.1 ICS Operation Function Blocks [20].	8
2.2 Example SCADA Architecture [1].	9
2.3 NIST SCADA Communication Topologies [20].	10
2.4 NIST Guide to ICS Security SCADA System Implementation Example [20]. . .	11
2.5 Allen-Bradley ControlLogix PLC Architecture [25].	13
2.6 Allen-Bradley EtherNet/IP Architecture [25].	15
2.7 EtherNet/IP Module Web Server.	16
2.8 Allen-Bradley CompactLogix PLC Architecture [27].	17
2.9 RSLogix Task Monitor.	18
2.10 Allen-Bradley Firmware Feature Comparison.	19
2.11 Common Industrial Protocol - Object Oriented Formatting.	20
2.12 Common Industrial Protocol - Identity Object Attributes.	21
2.13 Common Industrial Protocol Object Model [6].	22
2.14 Leverett search query results [19].	25
2.15 Comparison of Leverett search query results obtained in 2011 and 2013 [5]. . .	29
3.1 Process to Distinguish Internet-facing ICS Devices.	36
3.2 PLC Code Program-Specific Instance Request and Response.	38
3.3 Flowchart depicting Global and Program-specific code collection.	40
3.4 ControlLogix Experiment Architecture.	43
3.5 CompactLogix Experiment Architecture.	43
3.6 Task Monitor Graphical Interface shown with Wireshark CIP packet capture. .	48
3.7 Measuring PLC Task Execution Times.	50
4.1 Hardware configuration for exploratory testing.	53

Figure	Page
4.2 EtherNet/IP Packet.	54
4.3 PLC Code Request for Global Instance Values.	55
4.4 PLC Code Global Instance Request and Response.	56
4.5 PLC Code Request for Program-Specific Instance Values.	57
4.6 PLC Code Program-Specific Instance Request and Response.	57
4.7 R Q-Q Plot of PLC Task Execution Times L61 v16.56.47.	60
4.8 Boxplots of PLC Task Execution Times L61 v16.56.47.	62
4.9 Run B2 Scatter Plot for PLC Task Execution Times L61 v16.56.47.	63
4.10 Boxplot for L61 v19.11.56.	65
4.11 Boxplot for L23E v17.07.63.	68
4.12 Boxplot for L32E v17.12.64.	70
4.13 Boxplot for L32E v20.13.81.	71
4.14 Process to Distinguish Internet-facing ICS Devices (Black boxes contain redacted information).	74
4.15 Registered User Details view for an ICS device cataloged by Shodan (IP addresses redacted).	75
4.16 Sector/Industrial Category Distribution of Process Control Devices.	78
B.1 Boxplot for L61 v16.56.47.	88
B.2 Boxplot for L61 v19.11.56.	89
B.3 Boxplot for L61 v20.11.59.	89
B.4 Boxplot for L71 v20.11.59.	90
B.5 Boxplot for L71 v20.12.79.	90
B.6 Boxplot for L71 v20.13.81.	91
B.7 Boxplot for L23E v17.7.63.	91
B.8 Boxplot for L23E v18.12.57.	92

Figure	Page
B.9 Boxplot for L23E v19.11.16.	92
B.10 Boxplot for L32E v16.23.15.	93
B.11 Boxplot for L32E v17.12.64.	93
B.12 Boxplot for L32E v20.13.81.	94

List of Tables

Table	Page
2.1 Ports and services indexed by Shodan [4].	23
3.1 Firmware Versions used in performance analysis testing.	45
3.2 Listing of Experiment Runs for the 1756-L61 CPU.	47
4.1 Data Type Values found in Attribute 2 CIP Instance Responses.	58
4.2 L61 Baseline vs. Treatment p-values (Bold indicates slower execution time). . .	66
4.3 L71 Baseline vs. Treatment p-values (Bold indicates slower execution time). . .	67
4.4 L23E Baseline vs. Treatment p-values (Bold indicates slower execution time). .	69
4.5 L32E Baseline vs. Treatment p-values (Bold indicates slower execution time). .	72

List of Acronyms

Acronym	Definition
NIST	National Institute of Standards and Technology
ICS	Industrial Control System
SCADA	Supervisory Control and Data Acquisition Systems
DCS	Distributed Control Systems
PLC	Programmable Logic Controllers
IP	Internet Protocol
SHINE	SHodan INtelligence Extraction
CIP	Common Industry Protocol
ICS-CERT	Industrial Control System - Computer Emergency Response Team
ASCII	American Standard Code for Information Interchange
IEC	International Electrotechnical Commission
I/O	Input/Output
HTTP	Hypertext Transfer Protocol
CPU	Central Processing Unit
CIP	Common Industrial Protocol
EtherNet/IP	Ethernet / Industrial Protocol
OSI	Open Systems Interconnection
TCP	Transmission Control Protocol
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
SHINE	Shodan Intelligence Extraction
S4	SCADA Security Scientific Symposium
SSH	Secure Shell

Acronym	Definition
CIA	Central Intelligence Agency
FBI	Federal Bureau of Investigation
DoD	Department of Defense
NAT/PAT	Network Address Translation or Port Address Translation

DISTINGUISHING INTERNET-FACING ICS DEVICES USING PLC PROGRAMMING INFORMATION

I. Introduction

THE Shodan search engine maintains a database of devices connected to the Internet. It works by indexing the response messages for a variety of protocols for each public Internet address. While there have been many reports in the press that Shodan increases risk to Critical Infrastructure [22], no recent research has attempted to distinguish the Industrial Control System (ICS) that Shodan indexes. This research examines a non-invasive method to distinguish ICS devices based on Programmable Logic Controllers (PLC) programming information. Allen-Bradley RSLogix 5000 PLCs use tags in PLC code to label variables. These tags can be obtained by sending Common Industrial Protocol (CIP) requests for code attributes and parsing the results. This research demonstrates the ability to use tags to distinguish ICS devices indexed by Shodan based on function, and using industry experts, further distinguish those devices by industrial sector. This chapter describes the problem statement, scope, and methodology used to determine the metrics and methods required to obtain the necessary information from Internet-facing ICS devices that permits a classification based on function. This chapter concludes with an overview of the subsequent chapters included in this thesis.

1.1 Problem Statement

The National Institute of Standards and Technology (NIST) has published regulations regarding the safe and secure implementation of ICS networks [20]. In the NIST guidelines, it is clear that no ICS devices should be connected to the Internet, either by means of a

public Internet Protocol (IP) address or by forwarding public address space to a private address using Network Address Translation or Port Address Translation (NAT/PAT). Many industry experts have developed a false sense of security, believing their networks are safe from attack because they are not connected to the Internet [22]. However, Shodan has demonstrated that ICS devices are, in fact, connected to the Internet. Recent research has sought to enumerate ICS devices indexed by Shodan, and the results show Shodan continues to index more ICS devices as time continues [22]. To date, no research has been conducted that attempts to determine the function of those Internet-facing ICS devices. The goal of this research is to distinguish Internet-facing ICS devices indexed by Shodan.

To accomplish the research goal, data collection methods are tested in a controlled environment to ensure that Internet-facing ICS devices are not interrupted by the collection process. Reverse engineering techniques are used to write scripts in the Python language that craft application layer request messages for each PLC. The responses to those requests contain PLC code that contains Task, Program, Routing and Tag names along with tag data types. Next, testing on four different Allen-Bradley PLC Central Processing Units (CPUs) is conducted to measure PLC performance during the request/response process. Finally, the PLC code is visually inspected for process control terms to indicate if an ICS device is used to control a process. The results of this inspection are ICS devices classified as Process Control or Indeterminate.

1.2 Scope, Assumptions and Limitations

The scope of this research is limited in the type of ICS device tested and data sets available for analysis. This research builds on recent work using Allen-Bradley PLCs to detect changes to PLC code execution times and to detect changes in interaction with a PLC after it is indexed by Shodan. For this reason, a Shodan search query is used to obtain a pool of Allen-Bradley CompactLogix and ControlLogix family PLCs. From this device

pool, controlled testing is conducted on four different Allen-Bradley CPUs: 1756-L61, 1756-L71, 1769-L23E, and 1769-L32E.

The implementation of this research collects PLC code from Allen-Bradley PLCs that are connected to the Internet. The results from those collections are analyzed individually. This research does not attempt to look at similarities in PLC code among the sample population or look at contiguous IP space in order to identify systems of systems.

A set of municipal wastewater PLC project files are used for static analysis, providing a set of Program, Routine, and Tag data representing PLCs currently in use in Critical Infrastructure. These project files are from one small geographic area within the United States and from one Critical Infrastructure sector, and therefore not representative of a larger population.

Limitations to this research effect the types of Allen-Bradley PLCs available for testing and the data available for static analysis of PLC code. Allen-Bradley RSLogix5000 PLCs are used in this research as the RSLogix500 PLCs such as the MicroLogix family do not use tags to handle PLC code variables. Static analysis is also impacted by the small, geographically localized set of available PLC project files.

1.3 Approach

Collecting PLC code form Internet-facing ICS devices must not impact the device's operation. Impact is defined as a statistically significant increase in task execution time. Experimentation on ICS devices in a controlled environment provides the means to develop non-invasive data collection methods and measure performance metrics to determine impacts those collection methods have on ICS devices. The goal of this research is to distinguish Internet-facing ICS devices based on PLC programming information. This research accomplishes this goal by obtaining a list of Internet-facing ICS devices, collecting PLC code, and classifying devices by matching process control terms with the names used by ICS engineers to write PLC code.

The Shodan search engine indexes devices in the same manner that Google indexes web pages. Shodan requests service information for a specific IP and port, and indexes the response message. Carefully crafted Shodan search queries are used to obtain IP addresses for devices with matching service response messages. This is the method this research uses to obtain a list of Allen-Bradley PLCs indexed by Shodan.

Collecting PLC code from ICS devices makes use of CIP Get Attribute List messages that return Task, Program, and Routine names along with names and data types for Global and Program-specific tags. The RSLogix5000 software uploads PLC code from the device by making several CIP requests for class and instance values for Task, Program, Routine, and Tag values. Reverse engineering these requests using Wireshark captures, it is possible to create CIP requests replicating certain parts of the upload process, obtaining PLC code without using RSLogix5000. Once pilot testing confirms that the Python scripts are able to collect PLC code from an ICS device, testing is conducted measuring the task and system process execution times to ensure ICS device performance is not negatively impacted. During this testing, four different PLC CPUs are tested, each with three firmware versions. The firmware versions are selected according to firmware versions obtained during exploratory testing. For each CPU, it is tested with three firmware versions ranging from oldest to newest found on Internet-facing devices. Once PLC code is collected, each response is categorized based on attributes found in the PLC code.

Finally, a visual inspection of the PLC code collected from Internet-facing PLCs is conducted to match the names of Tasks, Programs, Routines, and Tags with a list of process control terms common across multiple Critical Infrastructure sectors.

1.4 Overview of Subsequent Chapters

The remainder of this thesis describes in detail the background, testing, and results of distinguishing Internet-facing ICS devices indexed by Shodan. Chapter 2 discusses the types of ICS devices and networks in use today along with the relevant research into

Shodan and Internet-facing ICS devices. Chapter 3 details the methodology used to develop methods to obtain data from Internet-facing ICS devices, ensuring PLC task execution is not negatively impacted. Chapter 4 describes the results of testing and implementing the research methodology. Finally, Chapter 5 states the research conclusions and future work that can be conducted to further classify Internet-facing ICS devices.

II. Background

INDUSTRIAL control systems (ICS) are used by corporations and municipalities to operate and maintain critical infrastructure providing essential services such as electric power distribution and wastewater management [20]. ICS systems operate by means of an interconnected network of devices to make automated and human assisted decisions affecting the operation of attached mechanical actuators and sensors. Despite governmental and industry efforts to standardize secure ICS implementations, compliance with security recommendations and industry best practices like those cited in the National Institute of Standards and Tehcnology (NIST) Guide to Industrial Control Systems are still voluntary [20]. The critical nature of these systems combined with a lack of security focus make ICS networks interesting and vulnerable targets for attackers.

Security managers believe segregating their ICS networks from the Internet provided a sufficient level of security [22]. In 2009, however, the Shodan search engine showed that against NIST security guidelines, ICS devices are in fact connected to the Internet. The Shodan search engine indexes these devices and provides a reconnaissance platform to passively identify ICS devices.

Recent research has shed new light on the ICS *attack surface* by indexing and cataloging ICS devices connected to Internet-facing IP addresses [19][22]. The purpose of this chapter is to define the components and methods used to implement ICS systems and also review current research demonstrating ways and means to interact with Internet-facing ICS devices.

Section 2.1 describes in detail the hardware, software, and communications networks that make up Industrial Control Systems. Section 2.2 discusses the Shodan search engine and recent research focusing on enumerating ICS devices indexed by Shodan. Section 2.3 describes the risks posed to ICS networks by attackers and provides examples of recent

attacks on Critical Infrastructure. Finally, section 2.4 covers recent research methods to measure CPU performance in ICS devices under test.

2.1 Critical Infrastructure and Industrial Control Systems

The Department of Homeland Security (DHS) defines Critical Infrastructure as the “assets, systems, and networks, whether physical or virtual, so vital to the United States that their incapacitation or destruction would have a debilitating effect on security, national economic security, national public health or safety, or any combination thereof.” DHS assumes the responsibility of protecting what it calls Critical Infrastructure and Key Resources (CIKR) [8][9]. DHS divides CIKR into 16 different sectors divided by functionality: financial, chemical, commercial, communications, manufacturing, dams, defense industrial base, education, emergency services, energy, food and agriculture, healthcare, national monuments and icons, nuclear reactors and materials, transportation, and water.

Industrial Control Systems can be divided into three components as shown in Figure 2.1: Human-Machine Interface (HMI), Remote Diagnostics and Maintenance Utilities, and the Control Loop consisting of Controllers, Actuators, Sensors, and a Controlled Process [20].

The control loop refers to the sensors, transmission methods, and controllers that operate at the lowest level of the ICS. Controllers such as Programmable Logic Controllers (PLCs) receive information from sensors and make decisions based on set points programmed into the PLC. For example, a PLC receives a specific voltage level from a temperature sensor and, based on that information, causes an actuator to open a cooling valve. The PLC also sends information to an HMI and a data historian for human observation and logging. HMIs are the devices that monitor and display information for a ICS which allows a central control room staffed by human operators to monitor data and

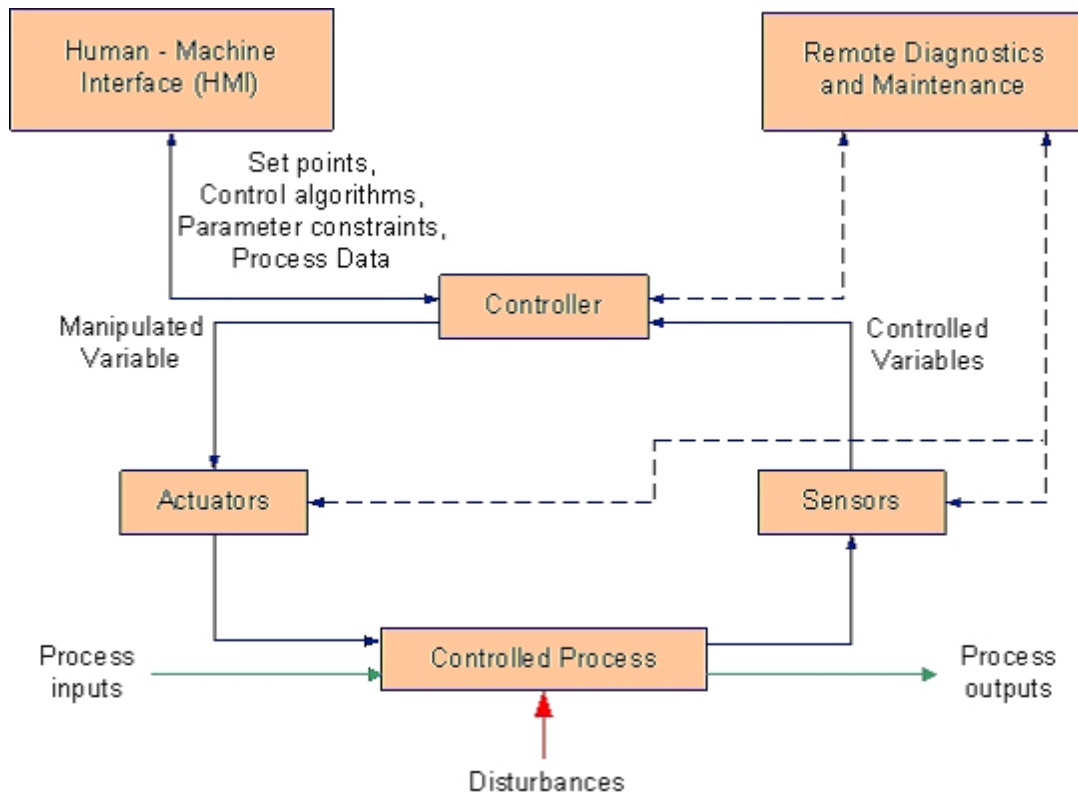


Figure 2.1: ICS Operation Function Blocks [20].

configure the PLC's parameters when required. Remote maintenance systems are used in ICS for preventing and recovering from equipment failures.

2.1.1 SCADA network architectures.

There are many different network topologies that support Supervisory Control and Data Acquisition Systems (SCADA) software and hardware design. This research will focus on an architecture using IP-based communications protocols and replicates SCADA architectures recommended by industry and governmental agencies.

Figure 2.2 is just one example of a SCADA architecture that generalizes how SCADA control and communications devices are interconnected to form a SCADA network [1]. The common components in any SCADA architecture are the topology, transmission systems, and control components.

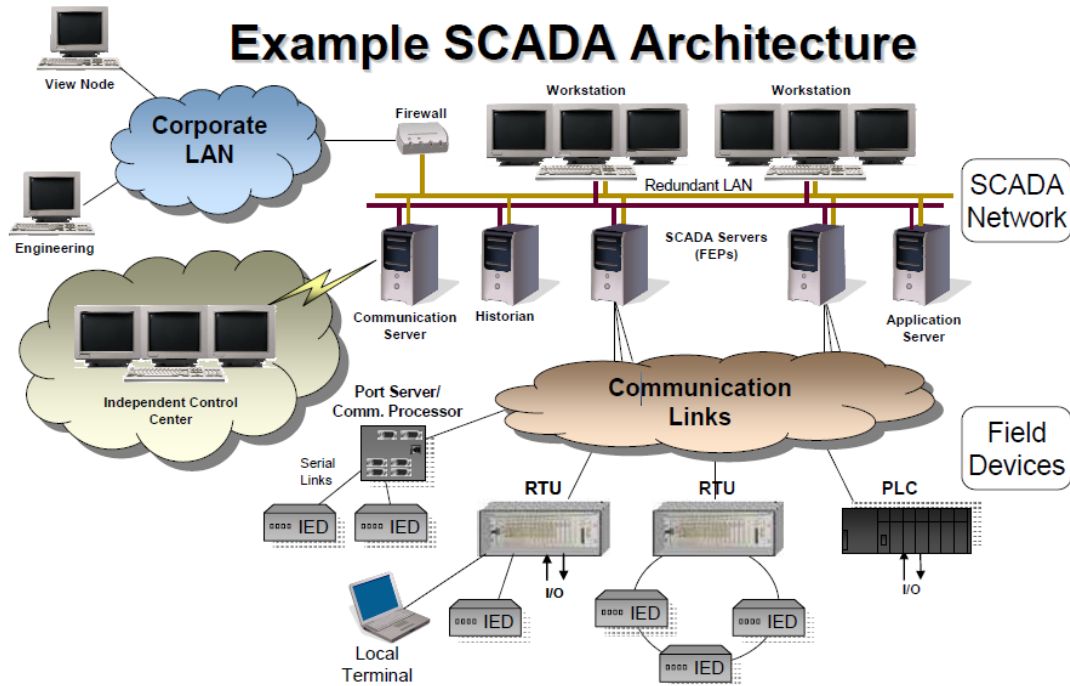


Figure 2.2: Example SCADA Architecture [1].

There are varying terminologies used to describe a SCADA topology which are depicted in Figure 2.3. The NIST Guide to ICS Security provides an example of four SCADA topologies: point to point, series, series-star, and multi-drop [20]. Point-to-point topologies typically use serial communications instead of the IP-based communications focused on in this research. Series, series-star, and multi-drop are used to network multiple field devices on a shared medium with the SCADA server.

Rockwell Collins defines their topologies as point to point and point to multipoint (multi-drop). Rockwell Collins identifies point to multipoint as the main topology used in SCADA networks. The multipoint (multidrop) topology connects several field devices to a SCADA server through a central hub. The entire SCADA system is interconnected through IP-based Local Area Network (LAN) and Wide Area Network (WAN) links creating the

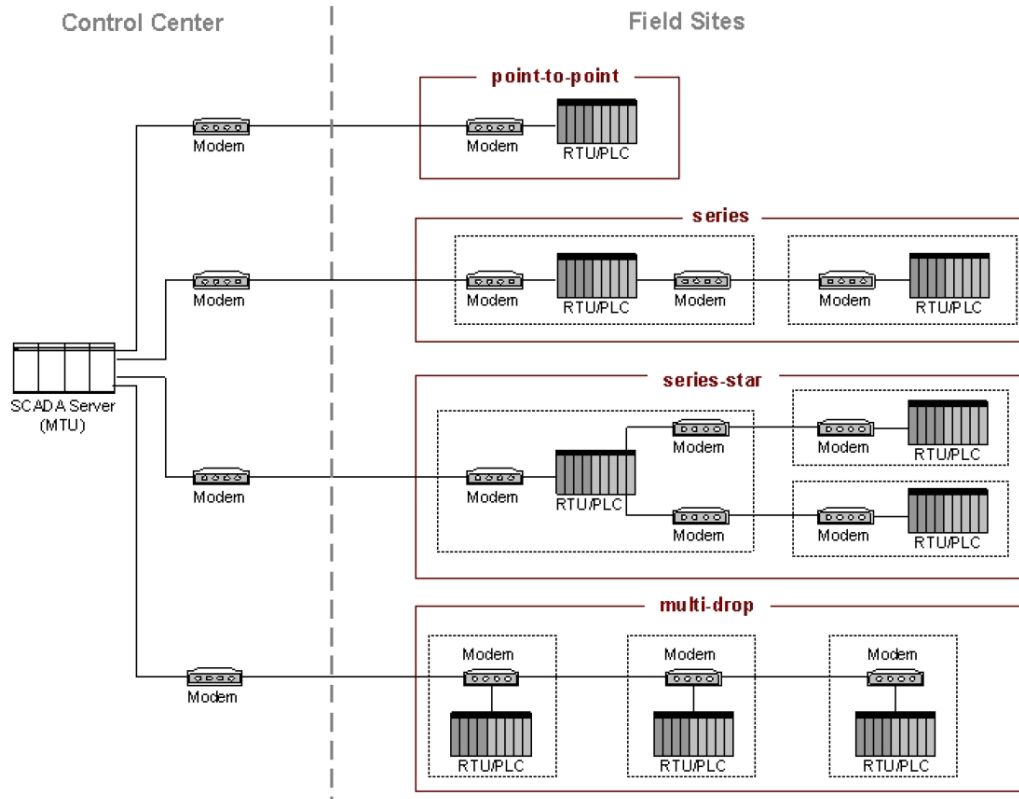


Figure 2.3: NIST SCADA Communication Topologies [20].

final design architecture as depicted in the NIST Guide to ICS Security shown in Figure 2.4.

SCADA architectures use a variety of transmission systems to connect the control center to field sites. These connections can be accomplished using serial connections, modem connections, WAN and LAN shared media connections, or wireless radio. Electric power distribution SCADA systems are geographically disperse and often utilize many different transmission methods such as Ethernet, dial-up/leased line serial connections, wireless IP and wireless radio [1]. Each of these systems use appropriate transmission protocols such as Modbus/TCP and DNP3 to format and send data over the transmission

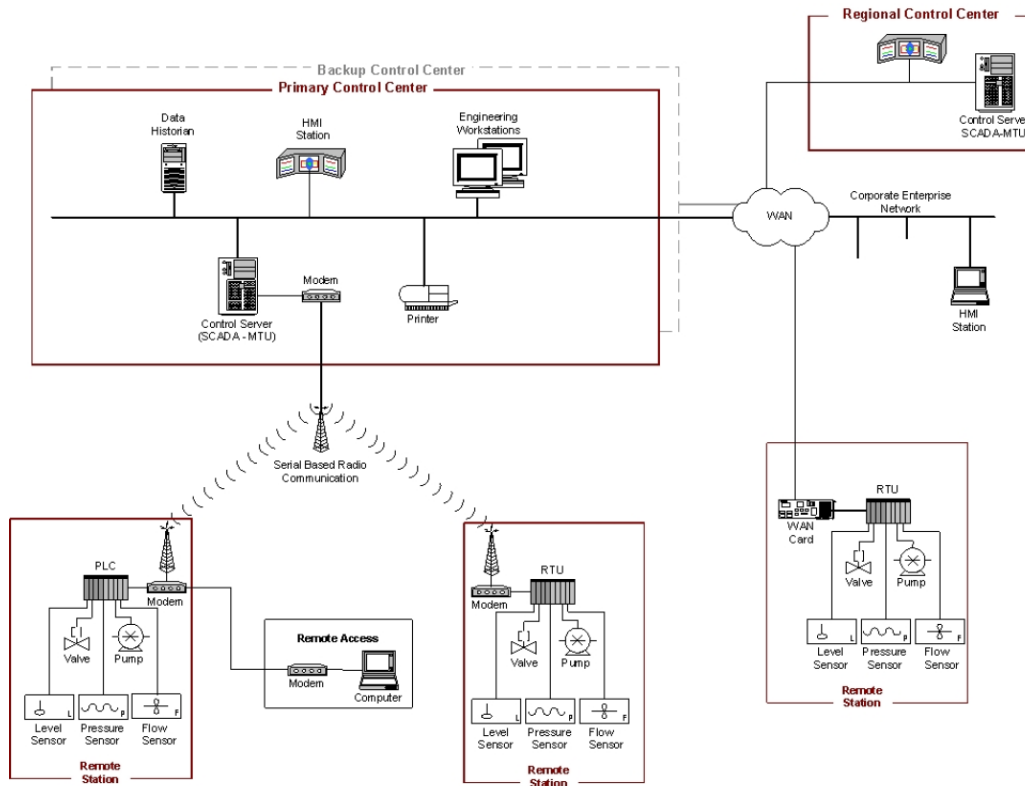


Figure 2.4: NIST Guide to ICS Security SCADA System Implementation Example [20].

medium used in a SCADA topology. The control components used in a SCADA system use the topology design and transmission medium to send and receive data within a network.

2.1.2 ICS Components.

ICSs are typically categorized as SCADA or Distributed Control Systems (DCS) based on their topology and purpose. SCADA systems are geographically dispersed systems with numerous field sites sending data to a central data historian, while DCS systems are normally configured to operate within a confined plant-centric area [20]. In SCADA systems, control devices such as PLCs are primarily used to supervise and monitor the state of an attached physical device, where PLCs in DCS systems typically control systems that execute mechanical operations.

SCADA components are separated into the control components and network components. These components allow SCADA systems to exchange real-time data between a centralized control center and field devices [18]. The control station is normally populated with any combination of control servers, Master Terminal Units (MTU), Human-Machine Interfaces (HMI), data historians, and Input/Output Servers [20]. The control server, Input/Output (I/O) server, MTU, and HMI can be separate systems or consolidated in a few physical devices. They make up the means for operators in the control center to monitor and remotely configure field devices. The *HMI* is comprised of software and hardware that allows operators to view the status of field devices, make changes to set points or algorithms, and override the commands of field devices when necessary. The *Control Server* hosts the control software that communicates with field devices, the *I/O Server* provides a method to communicate with field devices, and the *MTU* works with field devices in a master/slave configuration to implement changes made at the control center. The *Data Historian* is a centralized database server that logs all process information within the SCADA system for auditing, data analysis, and future planning.

These ICS field devices communicate with the control center and are responsible for monitoring and executing instructions based on their configured algorithms and set points. The PLC is a versatile control module that can be populated with special purpose modules to carry out a wide range of tasks. The PLC has the capability to logically control complex tasks by receiving feedback from an attached sensor and make decisions based on its programmed algorithm to maintain or change the state of an attached physical device.

PLCs used in SCADA systems are the most versatile and configurable of the field devices, capable of performing all supervisory control and data acquisition functions as other field devices [20]. Kalapatapu [18] describes how an RTU's use of communication protocols over wireless transmission systems is similar to PLC communications over wire medium using the same protocols. Siemens and Rockwell Collins are two of the

largest manufacturers of PLCs, with the Germany-based Siemens focusing on European markets and Rockwell Collins being the largest manufacturer based in the United States [30]. Rockwell Collins manufactures Allen-Bradley PLCs which are represented in nearly every Critical Infrastructure sector and have a 30 percent share of the North American electrical utilities market. The Allen-Bradley Logix 5000 series PLCs consist of a power supply, communications backplane, and 1756-A7 chassis with up to 7 slots for add-on modules. The PLC uses ARM processors for control and backplane communications. The control module houses the processors, memory, slot for flash memory and an RS-232 communications port. The remaining slots in the PLC are populated with I/O modules based on specific tasks the PLC is required to perform. Examples of these modules are analog and DC input modules used to read voltage levels from sensors which the PLC then converts into digital information used as input in execution of a task.

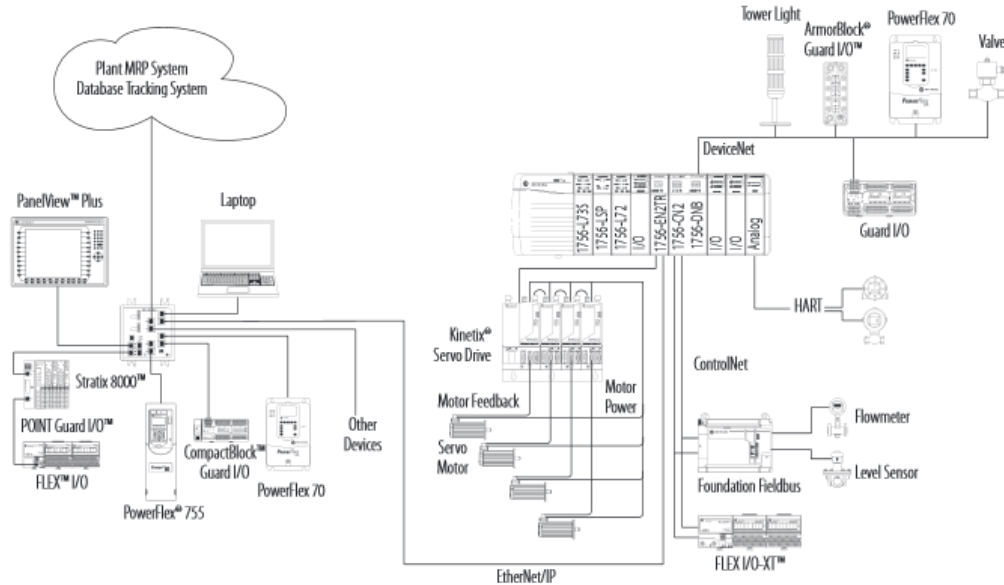


Figure 2.5: Allen-Bradley ControlLogix PLC Architecture [25].

2.1.3 Allen-Bradley Industrial Control Systems.

The Allen-Bradley ControlLogix family of PLCs shown in Figure 2.5 consists of a chassis, controller, communications module, and a number of optional modules providing additional functionality such as I/O [25]. The standard controllers for ControlLogix PLCs are the L6x and L7x series. These controllers can be populated in any chassis slot, and multiple controllers can be installed and operate simultaneously in a ControlLogix chassis. The primary difference between the L6x and L7x controllers is the built-in communications architecture. The L6x controller provides an RS-232 serial communications capability while the L7x controller has a built-in USB port. Allen-Bradley varies the amount of user memory within each series from 2MB on the L61 controller to 32MB on the L65 controller. Both controllers are used to execute PLC code and communicates with additional modules via the ControlLogix chassis backplane.

Ethernet / Industrial Protocol (EtherNet/IP) describes the Ethernet Industrial Protocol used by Allen-Bradley for PLC real-time messaging and I/O communications over IP-based networks as shown in Figure 2.6 [25]. The EtherNet/IP modules available for use in Allen-Bradley PLCs include an integrated web server that provides device configuration and communications data. Allen-Bradley recommends using the communications statistics provided by a Diagnostic Web Page shown in Figure 2.7. The web page is hosted by the EtherNet/IP module and is provided for troubleshooting PLC communication issues. The statistics provide a means to obtain current device state and communication capabilities by means of making a Hypertext Transfer Protocol (HTTP) GET request to the device's IP address.

The CompactLogix family of PLCs shown in Figure 2.8 provides a compact, integrated PLC solution that uses the same RSLogix common programming platform as the ControlLogix family [27]. The CompactLogix platform is built around an integrated

Example Configuration—EtherNet/IP Network

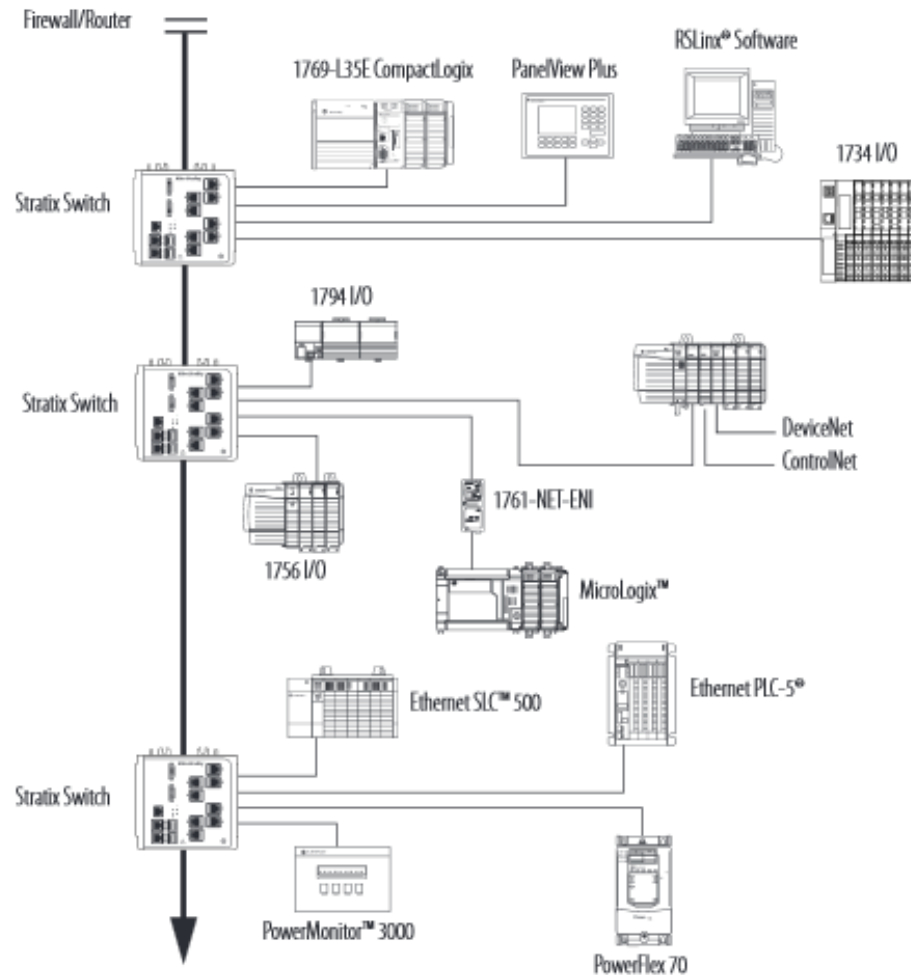


Figure 2.6: Allen-Bradley EtherNet/IP Architecture [25].

backplane, CPU, and communications module providing control and communications services for a range of I/O options. The 1769-L23x series provides embedded I/O functions in the device and limited expansion options. The 1769-L3x series increases options for onboard user memory and increases the number of possible add-on modules. Both use integrated serial communications or EtherNet/IP for communications with other devices or the RSLogix control software.

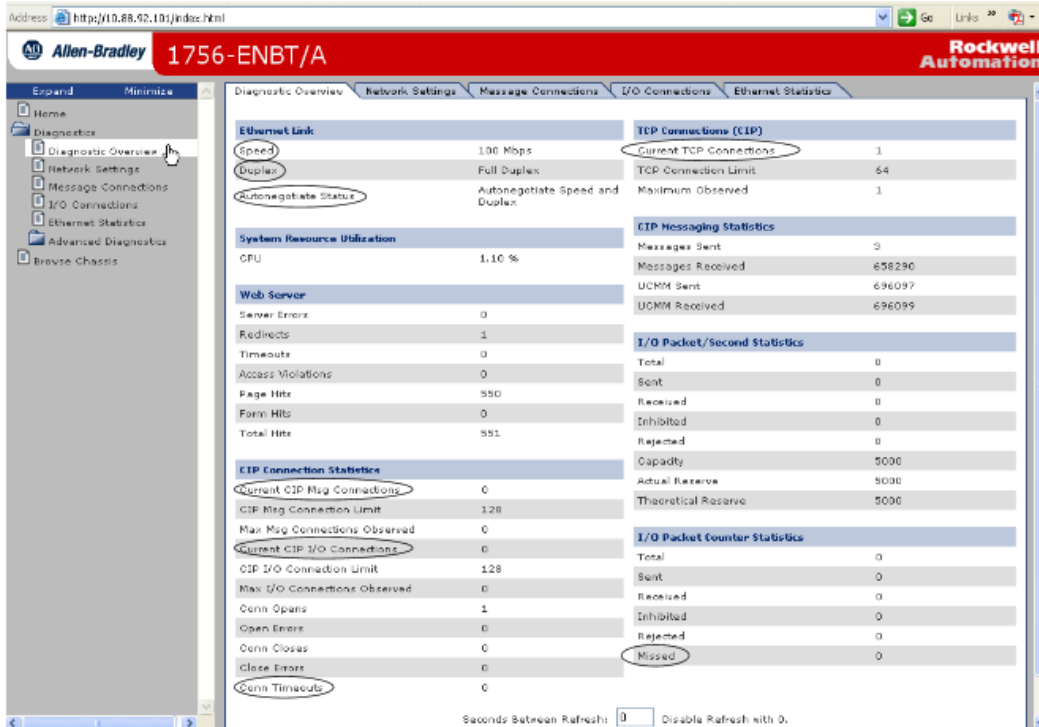


Figure 2.7: EtherNet/IP Module Web Server.

PLCs are programmed using one of the languages listed in the International Electrotechnical Commission (IEC) published standard IEC 61131-3. This standard describes the graphical and text-based programming languages which are used to provide logical decision-making for programmable controllers [20]. The Allen-Bradley Logix 5000 series PLCs conform to the IEC 61131-3 standard and uses a proprietary Studio 5000 Engineering and Design Environment to facilitate graphical program design and implementation [26].

RSLogix 5000 is a software application from Allen-Bradley that provides programming, control, and troubleshooting services for their PLCs [24]. RSLogix 5000 provides a graphical interface to build and monitor ladder logic, which is a type of PLC code supported by Allen-Bradley PLCs. RSLogix 5000 uses tags as variables within ladder logic. Tags can be thought of as any other programming variable, having assigned names and data

1768 CompactLogix System on an EtherNet/IP Network

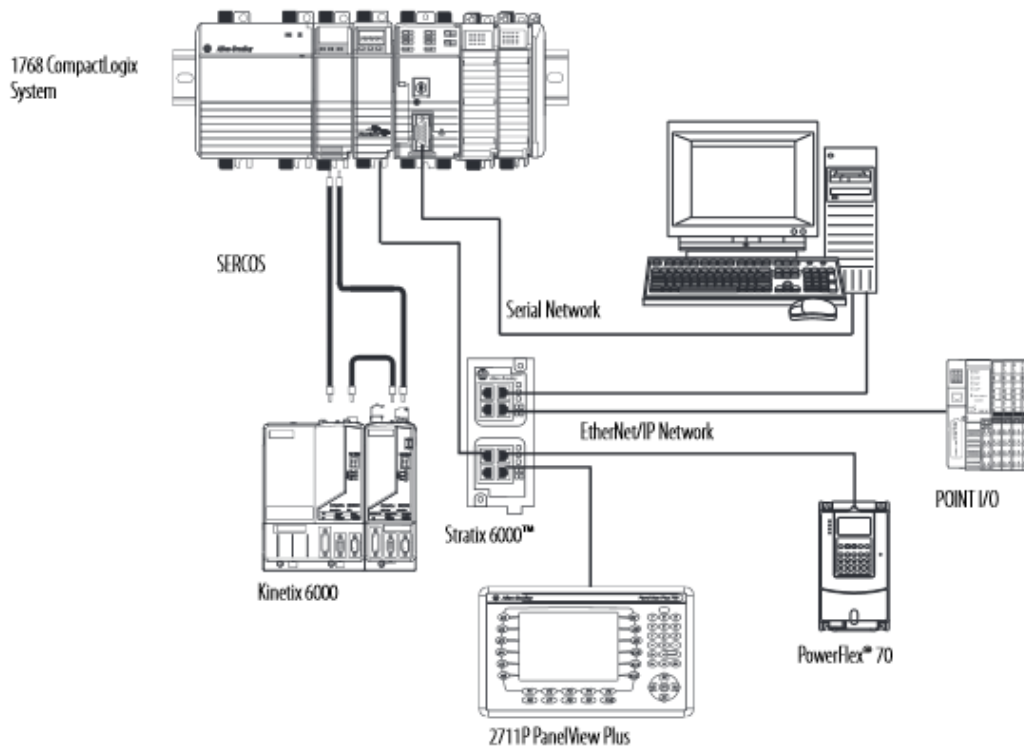


Figure 2.8: Allen-Bradley CompactLogix PLC Architecture [27].

types. Ladder logic instructions can use tags to make decisions, such as monitoring a tag as a setpoint.

RSLogix 5000 also provides real-time monitoring for ladder logic execution, and provides a suite of tools useful for troubleshooting [23]. RSLogix Task Monitor shown in Figure 2.9 which monitors and logs the PLC state during ladder logic execution. Task Monitor tracks operating statistics such as CPU Utilization and ladder logic execution times. Task Monitor collects data from the PLC by sending requests for execution times to the PLC at a user-defined interval. The PLC returns in microseconds the execution times for system processes and all tasks executing in ladder logic. This allows Task Monitor to calculate CPU Utilization and track the load on system services, such as the communications service.

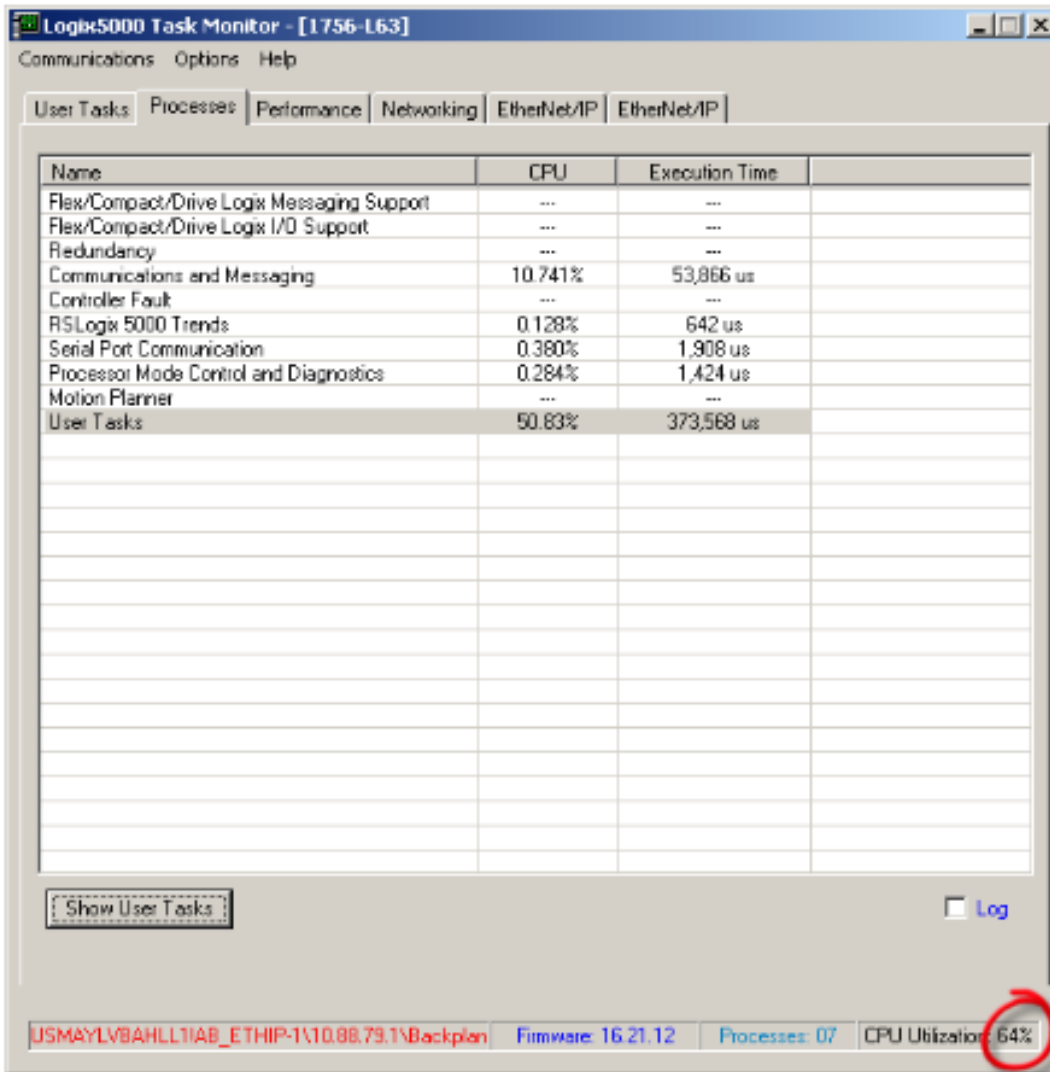


Figure 2.9: RSLogix Task Monitor.

The PLC firmware is the layer between logic and PLC hardware that implements logical operations in PLC code on the PLC itself. Allen-Bradley provides a support site where firmware feature sets and compatibility can be checked as in Figure 2.10, and firmware can be downloaded once an e-mail address is registered with Allen-Bradley through an online sign-up page.

Product Features by Version

1756-L61 (series B)							
ControlLogix Controllers							
Version	20.013	20.012	20.011	19.015	19.011	18.011	17.004
Downloads Information							
System Features							
General							
Auto-Save and Recovery	✓	✓	✓	✓	✓	✓	✓
Activation Key Software Grace Period	✓	✓	✓	✓	✓	✓	✓
System Install	✓	✓	✓	✓	✓	✓	✓
Start Page	✓	✓	✓	✓	✓	✓	✓
Online Help, Instruction Context Sensitive	✓	✓	✓	✓	✓	✓	✓
Integrated System Install	✓	✓	✓	✓	✓	✓	✓
Language Switching, Multi-lingual Project Documentation	✓	✓	✓	✓	✓	✓	✓
Sample Projects	✓	✓	✓	✓	✓	✓	✓
Electronic Data Sheet AOP	✓	✓	✓	✗	✗	✗	✗
Logix Tree, Module Discovery	✓	✓	✓	✗	✗	✗	✗
Logix Tree, Enhancements to Finding-Adding Devices	✓	✓	✓	✗	✗	✗	✗
Tool Windows, Auto-Hide	✓	✓	✓	✓	✓	✓	✗
Product Resiliency Improvement				✓	✓	✓	✗

Figure 2.10: Allen-Bradley Firmware Feature Comparison.

2.1.4 ICS Protocols.

A PLC makes decisions and executes code once all appropriate modules, firmware, and PLC code is loaded and configured. During operation, PLCs need to communicate to other field devices and HMIs for real-time information reporting and control. ICS protocols such as EtherNet/IP and CIP allow PLCs to communicate using an industry standard protocol which allows PLCs to communicate over any Transmission Control Protocol/Internet Protocol (TCP/IP) network.

CIP is an open industry standard application layer protocol managed by ODVA, Inc. that allows PLCs to communicate using a variety of ICS networking technologies such as DeviceNet, ControlNet, CompoNet, and EtherNet/IP [31]. CIP uses a producer-consumer model which allows PLCs to publish tag data and attributes to multiple consumers. CIP identifies messages by Connection IDs instead of source/destination address which allows multiple devices to make a single request for a Connection ID. Attribute messages are published for all consumers who have made a request to the Connection ID eliminating the need for multiple devices to initiate individual requests for tag attributes [2].

Figure 2.11 shows a CIP message requesting a list of attributes for an Identity Object using class and instance values. Identity Objects are objects with common attributes, separated by class [6]. CIP provides some pre-defined public classes along with the ability to define vendor specific classes in the Class ID range 100-199. An example of an Identity Object is shown in Figure 2.12.

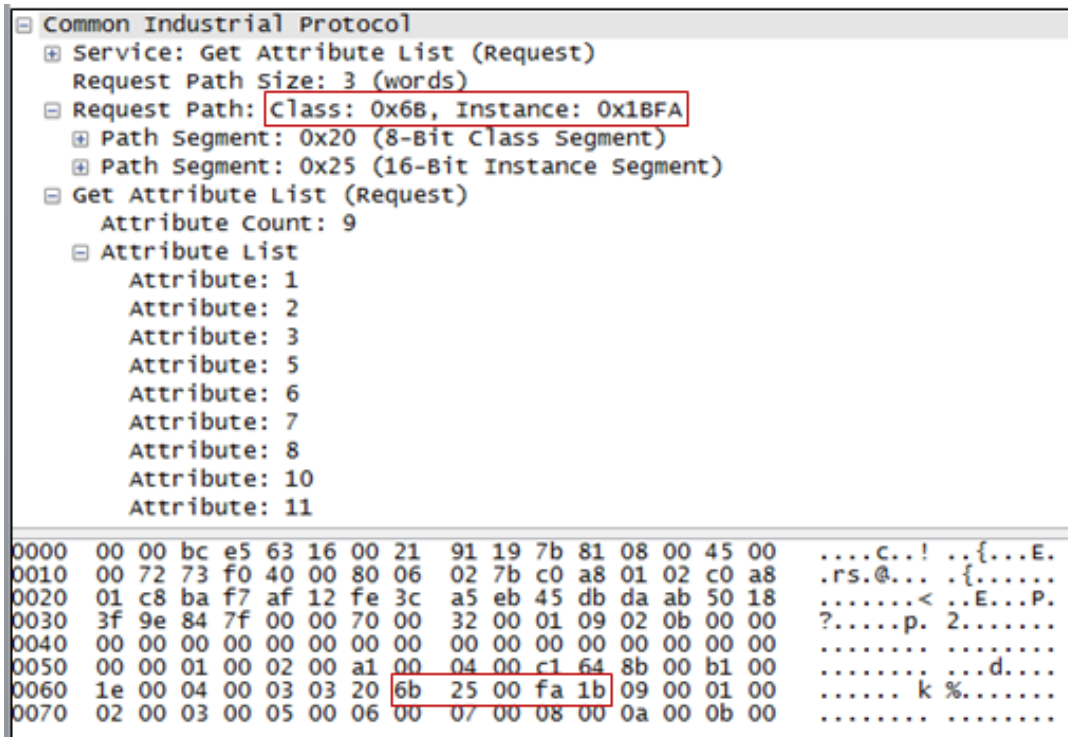


Figure 2.11: Common Industrial Protocol - Object Oriented Formatting.

EtherNet/IP is an ICS communications protocol operating at layers 1-4 of the Open Systems Interconnection (OSI) model allowing CIP messages to route over IP networks. EtherNet/IP at the transport layer segments CIP messages using Transmission Control Protocol (TCP) or User Datagram Protocol (UDP) protocols based on the CIP message type. EtherNet/IP uses TCP for explicit CIP messages such as the Get Attribute List request shown in Figure 2.11 [2]. TCP is a connection-oriented protocol that ensures

IDENTITY OBJECT	
Mandatory Attributes	Optional Attributes
<ul style="list-style-type: none"> • Vendor ID • Device Type • Product Code • Revision • Status • Serial Number • Product Name 	<ul style="list-style-type: none"> • State • Configuration Consistency Value • Heartbeat Interval

Figure 2.12: Common Industrial Protocol - Identity Object Attributes.

delivery of the message. EtherNet/IP uses UDP to transmit real-time information such as I/O messages. UDP is connection-less, but it is faster and smaller than TCP and suitable for time-sensitive transmissions. EtherNet/IP also provides a mechanism to establish connections between devices through a Connection Manager shown in Figure 2.13 [6]. CIP connections define packets that are produced and transmitted over a given network. Explicit messaging handles the routine produce and consume messages, assigning Connection IDs for produced information and handling the messaging to consumers. Implicit messaging refers to the time-sensitive I/O information transmitted via UDP by EtherNet/IP. The Unconnected Message Manager handles communication requests for routine information such as Identity Object attributes without establishing Connection IDs. This reduces the communications overhead for infrequent or routine requests by eliminating the need to establish connections and reserve resources through the Connection Manager.

2.2 The Shodan Search Engine

Vice magazine in 2013 published an article called “Is Shodan really the world’s most dangerous search engine?” [7]. In 2009, the Shodan search engine was placed into operation and provided proof that ICS networks around the globe were directly connected to the Internet by means of PLCs using public-facing IP addresses [13]. Recent research has developed techniques to quantify the problem and continues to discover more public-

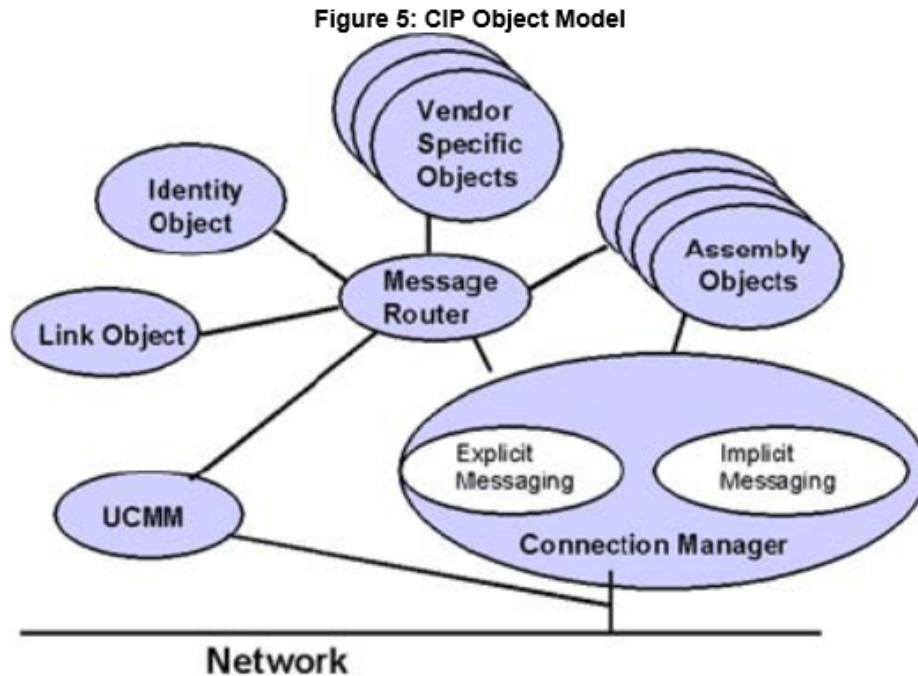


Figure 2.13: Common Industrial Protocol Object Model [6].

facing ICS devices. Much is being done to raise the alarm about the quantity of ICS devices connected to the Internet, but not much is being done to determine the function those devices perform.

Shodan was created by John Matherly in order to map software deployments across the internet to assist developers determine what systems were connected to the Internet [17]. The utility of Shodan became clear to researchers, security experts, and hackers alike [7] and Shodan became a tool to find all types of hardware and software using public-facing IP addresses. In 2009, Elian Leverett published his methodology using Shodan to determine the IP addresses of over 7,500 public-facing ICS devices.

Matherly started indexing banner messages on a Dell computer in his free time, cataloging messages at a rate of 10,000 per month [7]. His intent to conduct world-wide banner grabbing was to allow companies to track deployment of their hardware and

software. Shodan works by randomly selecting a public IP address, then interrogating a set of well-known ports for that IP shown in Table 2.1, indexing any banner messages returned.

Table 2.1: Ports and services indexed by Shodan [4].

Shodan Port Filters							
Port	Service	Port	Service	Port	Service	Port	Service
21	FTP	143	IMAP	1900	UPnP	6379	Redis
22	SSH	161	SNMP	2323	Telnet	7777	Oracle
23	Telnet	443	HTTPS	3306	MySQL	8000	Qconn
25	SMTP	445	SMB	3389	RDP	8080	HTTP
53	DNS	465	SMTP	5000	Synology	8129	Snapstream
80	HTTP	623	IPMI	5001	Synology	8443	HTTPS
81	HTTP	993	IMAP + SSL	5432	PostgreSQL	9200	ElasticSearch
110	POP3	995	POP3 + SSL	5560	Oracle	11211	MemCache
119	NNTP	1023	Telnet	5632	PC Anywhere	27017	MongoDB
137	NetBIOS	1434	MS-SQL	5900	VNC	28017	MongoDB Web

“I don’t consider my search engine scary...It’s scary that there are power plants connected to the Internet” [17]. Any user can query Shodan’s database and receive up to 10 results for free. Alternatively, users can register and pay a \$20 fee to have access to 10,000 results per search query. Some articles on Matherly have hinted that this provides a layer of insulation between Shodan and a black-hat hacker; however, Matherly states in the Vice article that the subscription service allows him to devote more assets to Shodan and now indexes “hundreds of millions a month.”

In 2010, Michael Schearer presented his work “Shodan for Penetration Testers” at DEFCON 18 [28]. Schearer began his talk by describing methods used to craft search

queries and utilize Shodan's filters to narrow results by port, protocol, service, or even geographic location. Schearer educated his audience in the art of using Shodan for network reconnaissance by finding networking devices with default credentials (or no authentication at all). Schearer was able to gain the highest access level (level 15) to an Internet Service Provider's (ISP) distribution switch and determine sensitive configuration information used by the ISP.

At DEFCON 20, Dan Tentler delivered a presentation on Shodan where he showed a city's traffic control system was indexed along with a hydroelectric power plant's control system [13]. This demonstrates Shodan's direct and potentially dangerous relationship with Critical Infrastructure and the ability for anyone to conduct passive reconnaissance on the ICS networks supporting CI.

2.2.1 Discovering Indexed ICS Devices.

In June 2011, Eireann Leverett completed his research titled "Quantitatively Assessing and Visualizing Industrial System Attack Surfaces" [19]. Leverett's motivation was to disprove the common assumption that ICS devices were safe from attack because they were only connected to internal, private networks. "Vendors say they don't need to do security testing because the systems are never connected to the internet; it's a very dangerous claim." [35]. Leverett crafted 29 Shodan queries in order to identify ICS devices by specific strings contained in the banner messages indexed by Shodan shown in Figure 2.14 from Leverett's work.

Leverett's research produced a tool that visualizes ICS devices by location to show that vulnerable Internet-facing ICS devices are connected worldwide. He presented his findings to several industry professionals including Industrial Control System - Computer Emergency Response Team (ICS-CERT), and continues to give talks on ICS categorization using Shodan.

Shodan Query	Connections	Category	Note
A850+Telemetry+Gateway	3	Telemetry	
ABB+Webmodule	3	Embedded Webserver	
Allen-Bradley	23	PAC	
/BroadWeb/	148	HMI	Known Vulnerabilities
Cimetrics+Eplus+Web+Server	6	Embedded Web Server	
CIMPLICITY	90	HMI	Zero Config Web View
CitectSCADA	3	PCS	
EIG+Embedded+Web+Server	104	Embedded Web Server	
eiPortal	1	Historian	
EnergyICT	585	RTU	Primarily Energy
HMS+AnyBus-S+WebServer	40	Embedded Web Server	
i.LON	1342	BMS	Primarily for energy
ioLogik	36	PLC	Small Vendor
Modbus+Bridge	12	Protocol Bridge	IP to Modbus
ModbusGW	11	Protocol Bridge	
Modicon+M340+CPU	3	Protocol Bridge	
Niagara+Web+Server	2794	HAN/BMS	Web server for EMS/BMS
NovaTech+HTTPD	1	Embedded Web Server	Substation Automation
Powerlink	257	BMS/HAN	
Reliance+4+Control+Server	10	SCADA	
RTS+Scada	15	SCADA	Runs on FreeBSD
RTU560	2	RTU	Web Interface
Simatic+HMI	9	HMI	Affected by Stuxnet
SIMATIC+NET	13	HMI	Affected by Stuxnet
Simatic+S7	13	PLC	Affected by Stuxnet
SoftPLC	80	PAC	Eastern Europe
TAC/Xenta	1880	BMS	Self Certs for HTTPS
WAGO	2	Telemetry	
webSCADA-Modbus	3	HAN	
Total	7489		

Figure 2.14: Leverett search query results [19].

When looking at Leverett’s research in the context of risk to Critical Infrastructure, it is important to note that Leverett’s research does not categorize ICS devices by function or industry sector. This leads some to conclude that the large number of ICS devices indexed by Shodan can be correlated to an increased risk to Critical Infrastructure. In 2012 during a talk at the Digital Bond SCADA Security Scientific Symposium (S4), Dale Peterson raised

a question about determining risk to Critical Infrastructure. Leverett's response showed the lack of research in this area: "Dale may be right here, the amount of truly 'critical' infrastructure is likely to be low in this data set" [21].

2.2.2 Project SHINE.

Project SHINE, the product of Rob Radvanovsky and Jake Brodsky, uses Shodan to catalog Internet-facing ICS devices in similar fashion to Leverett's work [15]. Shodan Intelligence Extraction (SHINE) began around the same time as Leverett and produced similar results. According to Radvanovsky, SHINE "began ingesting raw data mid-April 2012...to determine a baseline of just how many SCADA/ICS devices and software products are directly connected to the Internet" [22].

Where Leverett developed his thesis in an academic environment, the security professionals behind Project SHINE do not share Leverett's openness or willingness to share results. Radvanovsky states Project SHINE has categorized a list of over 1,000,000 unique IP addresses appearing to belong to SCADA/ICS devices and typically finds an additional 2,000-8,000 per day. He stated this in September 2013, implying he has discovered another half-million devices. Radvanovsky says SHINE is able to find such a large number of devices using "just shy of 700 searchable terms and (we) are adding more every week" [22].

Radvanovsky continues to collect data on Internet-facing ICS devices, and has stated that they "intend to perform our own...scans...through a new (undisclosed) method in the not-too-distant-future" [22]. A motive behind Radvanovsky's secrecy may be financial; "we may make available SHINE data to be used as part of a compliance service for asset owners, but we are struggling with an appropriate business model" [22].

2.3 Security Concerns and ICS Attacks

Recent events highlight the significant capabilities of state and non-state actors to carry out attacks on Critical Infrastructure. Stuxnet is a widely known example of malware

creating devastating physical effects. Stuxnet modified PLC code on a Siemens device and reports false information to the Siemens control software in an effort to mask the new PLC code that damages the targeted nuclear centrifuges [11]. Stuxnet is remarkable in the complexity of the code and the way it seeks out a very specific target. The Stuxnet developers have never been discovered. However, it is generally accepted to be part of a US program to intervene in the Iranian Nuclear program.

To demonstrate the desire to attack critical infrastructure using cyber exploits, an example emerged three decades before Stuxnet in the Trans-Siberian Pipeline [32]. The Central Intelligence Agency (CIA) used Soviet officer Colonel Vladimir I. Vetrov to facilitate a known Soviet plot to use a shadow company to illegally obtain banned technology. The CIA, Department of Defense (DoD), and Federal Bureau of Investigation (FBI) conducted an undercover operation which sold modified equipment to the Soviets through their shadow company. According to the CIA's report on The Farewell Dossier, "Contrived computer chips found their way into Soviet military equipment, flawed turbines were installed on a gas pipeline, and defective plans disrupted the output of chemical plants and a tractor factory" [32]. The resulting sale of sabotaged devices is believed to have caused an explosion on the Trans-Siberian pipeline in 1982. While some dispute the explosion was caused by CIA actions, it is a clear demonstration of an intent to carry out cyber attacks on critical infrastructure dating back thirty years.

The Darkreading.com IT security news portal has reported several recent exploits and attacks focusing on ICS devices and networks. In 2012, Kelly Higgins reported on backdoor exploits that target Siemens PLCs allowing the capture of passwords and ability to manipulate PLC code [14]. The same article described the addition of ICS exploits into the Metasploit Framework and an electric utility that had experienced unsuccessful brute force logon attempts through the Secure Shell (SSH) service.

Higgins has written several articles highlighting recent attacks on the Oil and Gas sector. In 2012, the Saudi Aramco oil and natural gas company had 30,000 computers on their corporate network infected and damaged by a piece of malware called Shamoon [3]. Shamoon is a version of W32.Disttrack destructive malware that writes over the Master Boot Records of hard drives, rendering the machine inoperable.

In 2014, Higgins wrote about a group called STTEAM using a “mix of hacktivist, nation-state, and pure cybercrime techniques” against oil and gas companies in the Middle East [16]. The STTEAM attacks show an increased sophistication from the Shamoon malware. Shamoon seemed to be poorly written and was able to be quickly removed from the Saudi Aramco network by replacing all 30,000 hard drives. STTEAM shows an intent to become a persistent threat to the systems they attack by using several powerful ASP backdoor scripts to allow attackers entry into the target network and remotely execute commands [12].

These recent attacks demonstrate the advanced capabilities available to exploit and attack critical infrastructure. This section has shown examples of a desire to attack critical infrastructure and increased exploits to carry out attacks. Shodan provides those with a desire to attack critical infrastructure an advanced reconnaissance capability to discover and collect targeting information on Internet-facing ICS devices. Since Leverett’s work in 2011, the number of Internet-facing devices indexed in his work has risen as shown in Figure 2.15 [19]. Along with the devices discovered by Leverett’s search terms, more specific terms are able to discover a larger number of Internet-facing ICS devices, all without having to interrogate the target devices directly.

2.3.1 TrendMicro Reports on ICS Attakers.

In 2013, SCADA security researcher Kyle Wilhoit published two papers with TrendMicro titled ‘Whos Really Attacking Your ICS Equipment?’ [33] and ‘The SCADA That Didnt Cry Wolf. Whos Really Attacking Your ICS Equipment? (Part 2)’ [34]. In the

Shodan Query	2011	2013	Category	Inc/Dec
A850+Telemetry+Gateway	3	34	Telemetry	1033%
ABB+Webmodule	3	3	Embedded Webserver	0%
Allen-Bradley	23	99	PAC	2533%
/BroadWeb/	148	352	HMI	6800%
Cimetrics+Eplus+Web+Server	6	16	Embedded Web Server	333%
CIMPLICITY	90	239	HMI	4967%
CitectSCADA	3	3	PCS	0%
EIG+Embedded+Web+Server	104	137	Embeddded Web Server	1100%
eiPortal	1	98	Historian	3233%
EnergyICT	585	2706	RTU	70700%
HMS+AnyBus-S+WebServer	40	121	Embedded Web Server	2700%
i.LON	1342	4643	BMS	110033%
ioLogik	36	184	PLC	4933%
Modbus+Bridge	12	99	Protocol Bridge	2900%
ModbusGW	11	94	Protocol Bridge	2767%
Modicon+M340+CPU	3	56	Protocol Bridge	1767%
Niagara+Web+Server	2794	34560	HAN/BMS	1058867%
NovaTech+HTTPD	1	0	Embedded Web Server	-33%
Powerlink	257	3121	BMS/HAN	95467%
Reliance+4+Control+Server	10	6	SCADA	-133%
RTS+Scada	15	28	SCADA	433%
RTU560	2	18	RTU	533%
Simatic+HMI	9	91	HMI	2733%
SIMATIC+NET	13	152	HMI	4633%
Simatic+S7	13	201	PLC	6267%
SoftPLC	80	1088	PAC	33600%
TAC/Xenta	1880	9165	BMS	242833%
WAGO	2	89	Telemetry	2900%
webSCADA-Modbus	3	6	HAN	100%
Total	7489	57409		

Figure 2.15: Comparison of Leverett search query results obtained in 2011 and 2013 [5].

first publication, Wilhoit describes his effort to identify SCADA attacks though the use of Internet-facing SCADA honeypots. Wilhoit used an actual PLC, PLC software running on Amazon Cloud, and a web server mimicking an ICS web server, all on different static IPs registered in the United States. Wilhoit provided a definition of threat in the paper:

‘We define an attack as anything that may be deemed a threat to Internet-facing ICS/SCADA systems. This includes unauthorized access to secure areas of sites, modifications on perceived controllers, or any attack against a protocol specific to ICS/SCADA devices like Modbus. In addition to classifying these attempts as attacks, we also consider any attempt to gain access or cause an incident to the server in a targeted fashion attacks [34].’

In 28 days, the honeypots Wilhoit identified 39 attacks 12 of which Wilhoit described as targeted.

Wilhoit’s second paper revisited the honeypot configuration. He developed a more robust honeypot architecture that closely replicated an actual SCADA system, including Modbus modules, PLCs, HMIs and other ICS systems. Wilhoit deployed 12 honeypots worldwide across 8 countries, taking care to use local language to make each deployment more realistic. In a four month period, Wilhoit’s honeynet saw 74 attacks, 10 of which Wilhoit classified as critical. Six of those critical attacks triggered Snort IDS signatures developed by DigitalBond to monitor unauthorized Modbus traffic.

Wilhoit’s research does not fully explain the nature of reported ICS attacks. Many of the generic attacks seen by Wilhoit were not targeting ICS field devices, but instead were targeting workstations used as HMIs. He did not describe what critical attacks are or if they were successful. In his second paper, he gives only one example of a targeted attack discovered during his earlier research. The targeted attack was a phishing attempt against an e-mail address found on an ICS honeypot. The phishing attempt targeted the Windows computer, not the ICS device and was designed to exfiltrate data if successful. Wilhoit called it a targeted ICS attack because it targeted an email address found on his honeypot, but he does not detail how attackers could only have found this email address by targeting his honeypot.

2.3.2 Shodan Impacts on ICS devices Pre/Post Indexing.

The 2014 research conducted by Roland Bodenheimer sought to determine Shodan's impact on ICS devices by measuring the change in network activity on an ICS honeypot after it was indexed by Shodan [5]. His research used TCP connections, TCP packet count, and number of unique IPs interacting with the device as metrics to quantify a change in network activity. His research also focused on detecting targeted attacks against Shodan indexed ICS devices by inspecting traffic at the honeypots against known ICS attack signatures.

Bodenheimer deployed a honeynet consisting of four Allen-Bradley PLCs using Internet-facing IP addresses. Two of the devices were configured with default authentication settings to mimic newly deployed devices, while the other two PLCs were configured to modify the service banner responses to Shodan requests. One of the modified banners replaced the string 'Server: GoAhead-Webs' with a random string to avoid detection, and the other modified banner had the original string replaced with 'Allen Bradley ControlLogix 1756' in an attempt to make it easier to detect using a targeted Shodan search query. All four devices were loaded with ladder logic and deployed for 55 days at the site of an ICS integrator.

The first of the honeypots was indexed by Shodan after 3 days with all four honeypots indexed by Shodan within 13 days. Bodenheimer showed that linear trending and 'goodness of fit' testing on network activity showed no evidence of any change to network activity after a device is indexed by Shodan. In fact, a comparison of mean averages for network activity metrics were all either below the 95% confidence interval, or could be attributed to an increase in automated network scans not specifically targeting the ICS honeypots. This indicated a lack of any statistical significance to the changes observed in network activity pre and post indexing.

Bodenheim used the Snort Intrusion Detection System (IDS) to check for targeted attacks matching Snort and DigitalBond Quickdraw SCADA signatures specific to ICS protocols. During the 55 day period, Snort identified only one high alert against one of the four honeypots. The alert was found to be an indiscriminate scan for PHP vulnerabilities, a service not used on the Allen-Bradley web server. While all four devices registered Snort alerts, it was found that none of the alerts were ICS specific, and results from the Snort IDS indicated that the activity was related to indiscriminate scans across public IP space, not targeted attacks associated with Shodan indexing.

Wilhoit saw scanning, generic attacks, and a small number of Modbus-specific IDS signature matches. Bodenheim saw no attacks on his PLCs even though his devices were exposed for the same duration as Wilhoit's first honeypots, attracting 39 attacks.

While no attacks to date have been directly attributed to Shodan searches, its existence does provide an anonymous reconnaissance platform that may facilitate the targeting of ICS devices for those actors with both a desire and capability to carry out attacks.

2.4 Related Work

A Rockwell Automations white paper [23] describes the recommended method to compare CPU performance between L6x and L7x model CPUs. The Rockwell Task Monitor utility measures CPU performance based on task execution times. The white paper states that Task Monitor is compatible with all RSLogix5000 controllers version 13 and above. The Task Monitor utility provides the means to request task execution times from a PLC and measure changes while the PLC is operating under different loads. This approach was used in two recent works to demonstrate CPU performance [10] [29].

Recent work performed by other researchers established methods used in this research to obtain PLC information and test PLCs for increases in CPU utilization or system process execution times. Stephen Dunlap in 2013 showed that analyzing timing characteristics of PLC ladder logic execution can be used to detect modifications to firmware or ladder logic

[10]. In his research, Dunlap developed a data collection mechanism to measure ladder logic execution times. He determined that a sample rate of 250ms over the course of 10,000 samples provided sufficient data to test for statistically significant changes to ladder logic execution times with a resolution of two microseconds.

Carl Schuett in 2013 expanded on Dunlap's method to measure ladder logic execution times as a part of performance analysis tools to measure process execution times [29]. Schuett added the capability to request process execution times for system processes running on the PLC CPU to detect statistically significant changes in performance. Schuett validated Dunlap's sample rate and size in his research, and found that system services must be sampled at a rate of 500ms to obtain valid results. Schuett uses the same one-way permutation test as Dunlap with an alpha value of 0.0001 to accept or reject the null hypothesis that there is no performance change between samples.

2.5 Conclusion

Shodan demonstrates that ICS devices are connected to the Internet, and much research has been conducted recently to show that the number of ICS devices Shodan indexes is growing. While this research focuses on numbers, no research has yet attempted to qualify Internet-facing ICS devices by function, or try to determine what Shodan's impacts are to critical infrastructure. This chapter focused on the systems comprising ICS networks and the current efforts to enumerate them using Shodan. In the following chapter, this research will outline the methodology used to distinguish ICS devices indexed by Shodan.

III. Methodology

THIS chapter describes the methodology used to distinguish Internet-facing ICS devices and conduct testing to ensure an ICS device under test is not impacted. Section 3.1 defines the problem of distinguishing ICS devices as it relates to this research. Section 3.2 describes the methodology used in this research. Section 3.3 describes the experimental environment. Section 3.4 describes in detail the procedures used to collect data. Section 3.5 outlines the analysis methods that will be used in the following chapter to interpret the data collected during experimentation.

3.1 Problem Definition

Shodan has been called “the scariest search engine on the planet” [13] due to the fact that Shodan indexes response messages for a set of TCP/IP and application layer protocols across the entire public IP address space. Since 2009, Shodan database queries have revealed ICS devices are being connected to the internet using public-facing IP addresses [19]. While the existence of ICS devices in the Shodan database has been examined by many different researchers, no research addresses the risk those devices pose to critical infrastructure. Many of the sensational news stories published regarding Shodan cite the discovery of a control system for a wine cellar or publicly-accessible webcams [13], however, no one has yet taken the steps necessary to determine the real threat to critical infrastructure posed by ICS devices cataloged in Shodan’s database.

The goal of this research is to develop a method to distinguish Internet-facing ICS devices indexed by Shodan based on PLC programming information.

Industry and governmental recommendations for secure ICS network design specify methods to prevent direct Internet connections [20]. Leverett’s work has shown that despite the recommendations, ICS devices are continuing to be connected directly to the Internet.

Related work quantifies Internet-facing ICS devices, but cannot answer the question “Why are these devices connected to the Internet?” This research attempts to answer that question by distinguishing Internet-facing ICS devices based on PLC programming information. This research defines PLC programming information as the Task, Program, Routine, and Tag names used by ICS engineers to program PLC code. Allen-Bradley PLCs using the CIP protocol return these names as strings in response to CIP attribute requests. Static analysis techniques are used to distinguish these strings as either process control terms or indeterminate. PLC code containing process control terms indicates the device is likely to be controlling a physical industrial process.

This research evaluates two primary objectives:

1. Evaluate the impact of collection methods on Allen-Bradley PLCs by measuring PLC code and system service execution times. This research defines impact as a measurable, statistically significant increase in task execution time caused by the collection methods developed in this chapter.
2. Collect PLC programming information from Internet-facing Allen-Bradley PLCs and distinguish each device based on matching PLC code to process control terms.

3.2 Approach

The goal of this research is to determine a method for distinguishing Internet-facing ICS devices that are controlling processes without impacting PLC task execution time. This section describes the general methodology used to evaluate PLC impacts during code collection, script development that automates the request process given a list of ICS devices, and the segregation of ICS devices by the presence of process control terms.

The proposed method of distinguishing Internet-facing ICS devices is shown in Figure 3.1. This section concludes with the explanation of performance analysis methods used in this research to determine PLC code collection impacts on PLC task execution times.

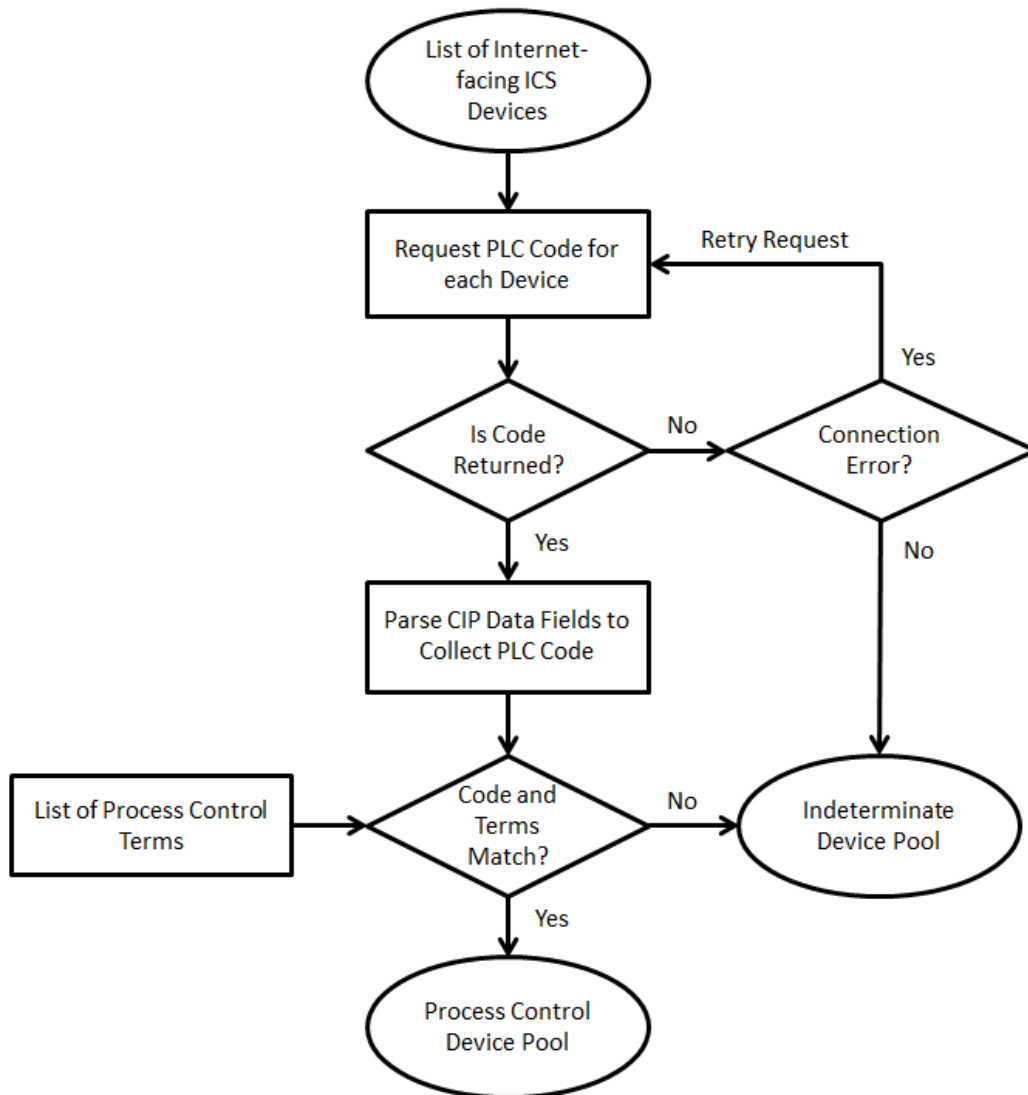


Figure 3.1: Process to Distinguish Internet-facing ICS Devices.

3.2.1 Evaluating Collection Impact on PLCs.

Availability is critical to the secure and safe operation of any ICS network. The methods used to collect data in this research must not degrade PLC performance to the point where task execution times are significantly impacted.

Exploratory testing demonstrates the CIP protocol can be reverse engineered and a python script can replicate the RSLogix5000 Upload process to receive PLC code from an

Internet-facing device. EtherNet/IP and CIP protocols are used to send attribute requests to the PLC, and responses are parsed and analyzed. The results of exploratory testing are discussed in detail in Chapter 4.

Collecting PLC code from Internet-facing ICS devices must not impact the device's operation. Impact is defined as a statistically significant increase in task execution time. Experimentation on ICS devices in a controlled environment provides the means to develop non-invasive data collection methods and measure performance metrics to determine impacts on ICS devices. Tests measure task execution times during requests for PLC code. During these experiments, user task execution times should not see a significant increase. Exploratory testing confirms that the Python scripts are able to collect PLC code from an ICS device. Testing is conducted measuring the task execution times to ensure ICS device performance is not negatively impacted. During this testing, four different PLC CPUs are tested, each with three firmware versions. During exploratory testing, firmware versions are cataloged to develop a better understanding of Allen-Bradley PLCs connected to the Internet. For each CPU, it is tested with firmware ranging from oldest to newest found on Internet-facing devices. Measuring PLC impacts during code collection determines if this method is feasible for employment on a set of Internet-facing PLCs indexed by Shodan.

3.2.2 PLC Code Collection Script Development.

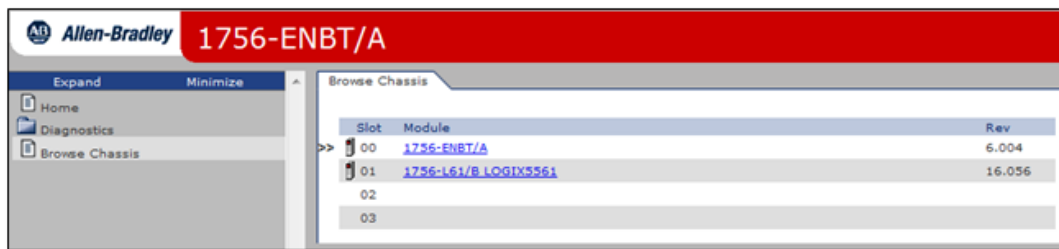
This research builds on the code developed by Dunlap that crafts EtherNet/IP and CIP packets requesting PLC code attributes [10][29]. Dunlap's script written for the Python interpreter version 2.7 obtains a Connection ID from the PLC used in connection-oriented commands. The connection manager provides routing for messages from the EtherNet/IP module, across the chassis backplane, to the CPU installed in its respective slot.

Code collection is established by replicating the process RSLogix 5000 uses to collect PLC code: register a session, establish a connection, send the appropriate series of Get Attribute List commands iterating over all global and program-specific instance values.

CompactLogix PLCs are designed so that the CPU always populates slot 0, however ControlLogix PLCs are more customizable and the CPU can be located in any slot on the backplane. Since the EtherNet/IP Forward Open request requires the CPU path, a method for determining the processor type and slot is required.

3.2.2.1 EtherNet/IP Connection Setup.

The 1756-ENBT module hosts a web server with Diagnostic Web Pages used primarily for troubleshooting network connections. One of the Diagnostic Web Pages lists all devices populated in the chassis by slot number shown in Figure 3.2. The collection process for a list of Internet-facing ICS devices returned by Shodan is automated by use of web-scraping techniques. These techniques make an HTTP GET request for the Diagnostic Web Page and search the web page source code for the processor type and slot. This information is then returned and is used in the EtherNet/IP Forward Open request message.



The screenshot shows a web browser window with the title "Allen-Bradley 1756-ENBT/A". The browser's address bar and navigation buttons are visible. The main content area displays a "Browse Chassis" page with a table listing modules in a chassis. The table has three columns: "Slot", "Module", and "Rev".

Slot	Module	Rev
00	1756-ENBT/A	6.004
01	1756-L61/B LOGIX5561	16.056
02		
03		

Figure 3.2: PLC Code Program-Specific Instance Request and Response.

3.2.2.2 CIP Attribute Requests.

Five functions are added to the python script in order to collect the PLC code shown in Figure 3.3. The first function creates a CIP message using the CIP service 0x4B Get Attribute List with a class value of 0x6B and an instance value of 0x00 to request all global instance values. The response message is parsed to strip the first four of every eight hex characters. These four hex characters represent a global instance value and the list

of instance values are passed to the next function. The second function takes in the list of global instance values and iterates over each value requesting a set of attributes. This function uses the CIP service 0x03 Get Attribute List to request the instance attributes for class 0x6B and a given instance value. The responses for each service request are parsed into attributes and returned as a list of hex values.

Program instance values and attributes are collected in a similar manner with a modification of the code to provide the request path pair for each global instance value and an associated program-specific instance. The third function in our script uses the CIP service 0x4B with class 0x68 and instance 0x00 to request a list of global program instance values. Note that the global instances returned from the first request cannot be used to obtain program-specific instances; RSLogix 5000 inserts this step requesting a new set of class 0x68 global instance values. This is the reason that no association is made between the global instance values returned by the first function and the global program instance values returned by this function. The fourth function iterates over each global program instance to obtain a set of program-specific instance values. This is the first message created using a request path pair. The first class and instance used is class 0x68, instance *0x(global program instance)*, class 0x6B, instance 0x00. As in other CIP requests, the message requests the global program instance values (*0x68,0x(global program instance)*)(*0x6B, 0x00*) associated with that specific program. The data field containing instance values is parsed, stored as a list of program-specific instance values, and passed to the fifth function. Finally, the CIP service 0x03 Get Attribute List request is sent for each program-specific instance value. This function also uses a request path pair of class 0x68, *0x(global program instance)*, 0x6B, *0x(program-specific instance)*. The script writes each response in a comma separated value (csv) coded file for further static analysis.

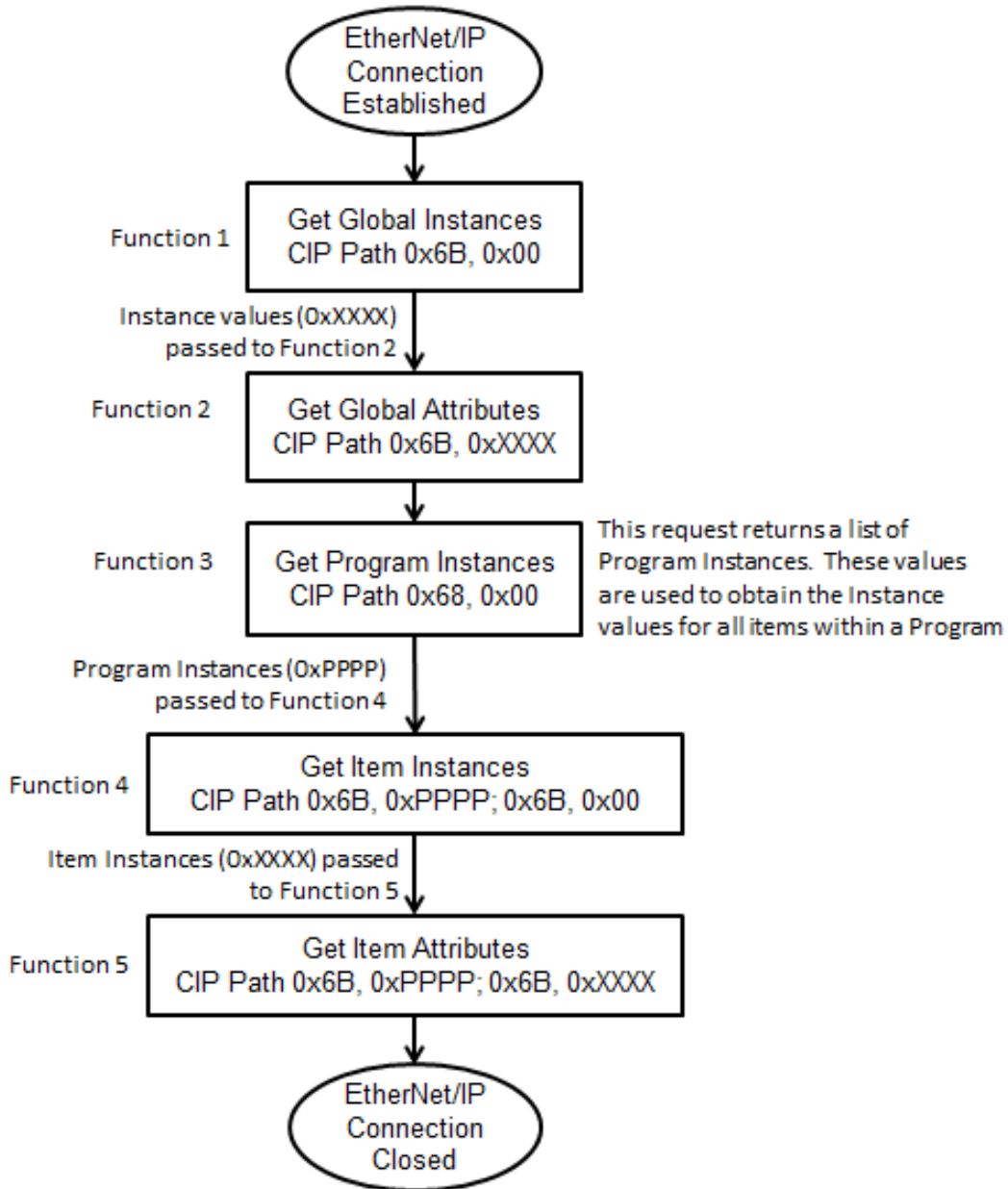


Figure 3.3: Flowchart depicting Global and Program-specific code collection.

3.2.3 Detecting Process Control Terms in PLC Programming Information.

The goal of static analysis on PLC code is to distinguish devices that are controlling an operation in an industrial process from those where their function cannot be determined. The tag names provide meaningful data for detecting process control terms. This research

uses a set of PLC project files developed by an ICS integrator for a metropolitan wastewater management system. Analyzing the wastewater project files, it is evident that engineers use process control terms to label the programs and tags used in PLC code so that it is readable and understandable. This permits static analysis of the PLC code obtained in this research for the presence of process control terms. A process control term in this research is defined as: a generally accepted terms widely used by ICS engineers to describe industrial processes. Examples of process control terms are pump, generator, start, HMI, and variations on each term (pump, pmp, pum1, pump01, etc.) Assuming that ICS engineers code similarly and by reviewing the wastewater PLC code, it is expected to find process control terms in the PLC code for ICS devices controlling industrial processes.

PLC code for each device is reviewed for the presence of process control terms. A list of the terms used is included in Appendix 1. This is the method that allows devices to be classified as Process Control or Indeterminate. The Process Control devices are selected based on the existence of process control terms in their PLC code. Devices that do not contain any process control terms are classified as indeterminate and separated from the Process Control set.

After selecting the group of Process Control devices, the list is presented to a group of Industrial Control Systems engineers for review. These industry experts are able to leverage their experience and deep understanding of process control systems to validate devices in the Process Control group and exclude any devices that contain process control terms, but do not meet their expectations for a device controlling an industrial process.

The limitation to this method of classifying devices are the process control terms themselves. It is possible that an engineer may use an odd naming convention for program names or tags. Also, devices with PLC code written in a non-english language will also avoid detection by this classification. In both cases, the device will be classified as indeterminate.

3.3 Environment

This section describes the testing environment constructed to measure performance impacts to Allen-Bradley 1756-L61/L71 and 1769-L23E/L32E PLCs during PLC code collection. Performance impact in this research is defined as a statistically significant increase to user task execution times.

3.3.1 Architecture and Hardware.

The experiment setup for this research consists of a workstation, hub, and Allen-Bradley PLC as shown in Figure 3.4 and Figure 3.5. The workstation depicted is a Dell Precision 690 with dual Intel Xeon 3GHz processors, 8GB RAM, Windows 7 Enterprise SP1 64-bit operating system. The workstation supports a Windows XP Virtual Machine with 1 processor and 4 cores, 3GB RAM, 60GB hard drive, and a bridged network connection to the test network. The workstation is connected to the PLC under test through a Linksys 5 port workgroup hub and CAT-5 Ethernet cable. The two ControlLogix PLC CPUs under test are the 1756-L61 and 1756-L71, both using the Allen-Bradley chassis 1756-A4 Series B, 1756-ENBT/A EtherNet/IP module v6.004, and Power Supply 1756-PA75 Series B. The two CompactLogix PLC CPUs under test are the Allen-Bradley 1769-L23E-QBFC1B Series A and Allen-Bradley 1769-L32E Series B. CompactLogix CPUs tests utilized the same L23E power supply and housing for both the L23E and L32E processor modules.

3.3.2 Software.

The workstation utilizes Wireshark Version 1.8.6 for packet capture on the test network and VMWare Workstation 10.0.1 build-1379776 to run the Windows XP Virtual Machine. The Virtual Machine has Rockwell Software installed that provides communication and a programming interface to the PLCs under test. The Rockwell Software used in this research is ControlFLASH v12.00.00 for firmware loading, RSLinx

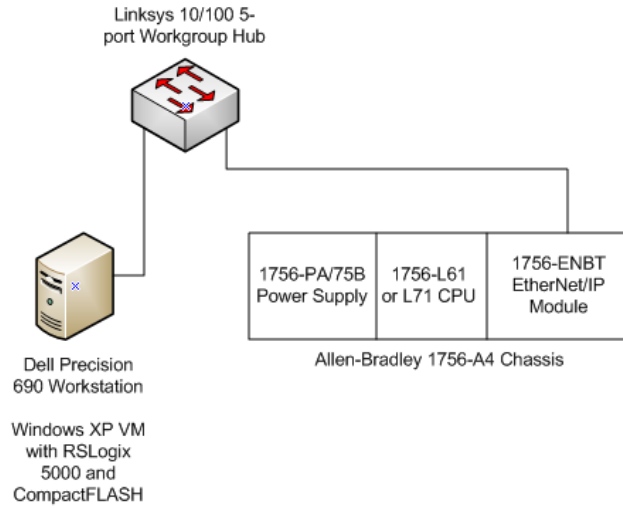


Figure 3.4: ControlLogix Experiment Architecture.

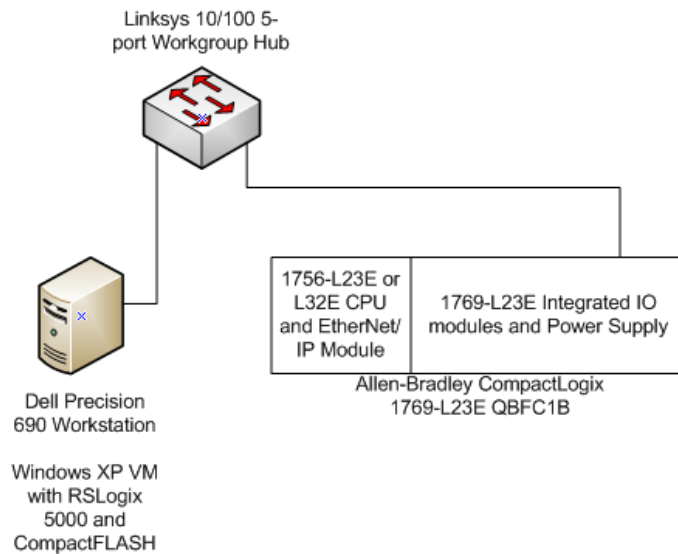


Figure 3.5: CompactLogix Experiment Architecture.

Classic Lite Revision 2.59.02 CPR 9 SR 5 for establishing a communications path to the PLC under test, RSLogix5000 V19.01.00 CPR 9 SR 3 for the programming interface and ability to upload and download project files to the CPU under test, and Logix5000

Task Monitor Version 3.0.3.0 to perform logging of user and system task execution times. Python scripts for this research use the Python 2.7.6 interpreter.

3.3.3 PLC Code collection tools.

Python scripts developed by Dunlap [10] for his research are modified to collect object information for Tasks, Programs, Routines, and Tags in the project files loaded into the CPU under test. Dunlap originally developed scripts that register a session and obtains a Connection ID. Through reverse engineering the RSLogix5000 upload process, Python is used to craft packets initiating Connected Send requests for PLC code.

3.3.4 Performance Analysis Tools.

CPU task execution time is measured using the user task and system service execution times with and without the PLC code collection script running. Comparative analysis is conducted between the two measurement sets to determine if a statistically significant change in execution times is detected. This research uses Python classes developed in Dunlap's [10] research and modified by Schuett [29] to take measurements of execution times and automate the project file download process. Measurements are taken every 500ms by means of a CIP request for execution times. The responses from these CIP requests provide user task and system service execution times in microseconds. A sample size of 120 measurements are taken for each level of CPU and firmware with and without Python PLC code collection scripts running.

3.3.5 Implementation.

Implementation of this method occurs after determining the impact code collection has on a device. Four different Allen-Bradley CPUs are tested for three firmware versions to determine impact. CPUs that are not impacted by the PLC code collection method are scanned using a Ubuntu Linux instance on the Amazon Web Services Elastic Compute Cloud (EC2) Virtual Server in the Cloud. Amazon EC2 provides an Ubuntu Server 14.04 LTS instance using 1 CPU, 0.613GB of RAM, 8 GB storage, and a public IP.

3.4 Experiment Design

This section details the design, implementation, and data collection used in this research to obtain PLC code from ICS devices indexed in the Shodan search engine. The data collection methods used in this research must not impact ICS device availability to be considered acceptable for use against Internet-facing ICS devices. These experiments show the impacts to a PLC in a test environment prior to data collection conducted on the Internet-facing devices.

3.4.1 Experiment Setup.

A total of 30 sets of measurements are collected on each CPU. Baseline sets are defined as a set of 120 task execution time measurements taken for a fixed CPU and firmware level. Treatment sets are defined as a set of 120 task execution time measurements taken for a fixed CPU and firmware level while a PLC code collection script is sending one EtherNet/IP message to the CPU for each measurement cycle. Three firmware versions are tested for each CPU. Five replications of baseline sets and five replications of treatment sets are recorded for each of the three firmware versions resulting in 30 total measurement sets. Table 3.1 lists the firmware versions used in the experiment.

Table 3.1: Firmware Versions used in performance analysis testing.

Experiment Firmware Versions				
Firmware	1756-L61	1756-L71	1769-L23	1769-L32
Low	16.56.47	20.11.59	17.7.63	16.23.15
Medium	19.11.56	20.12.79	18.12.57	17.12.64
High	20.11.59	20.13.81	19.11.16	20.13.81

Table 3.2 lists an example experiment setup before the entire list of 30 experiments is randomized to reduce bias. The first column represents the run order prior to

randomization. The second column lists the RSLogix5000 project file downloaded automatically to the PLC. The logic in each project file is identical, however each project file is matched to a specific firmware version. The fourth column lists the firmware version flashed to the PLC and the fifth column indicates if the run is Baseline or Treatment. When the workload value in the fifth column is zero, no code collection is performed during measurement. When the workload value is 1, a child process spawns running the code collection script in conjunction with measurements.

Each project file is a modification of the “Big” project file used in Dunlap and Schuett’s research. The project files are modified by adding global tags to each project file so that the PLC code collection scripts run for a period of time between 55 and 56 seconds.

3.4.2 PLC Execution Times.

Measurements are collected from the PLC using Dunlap and Schuett’s python script which records task and system process execution times every 500ms. The script was developed by reverse engineering RSLogix Task Monitor requests to the PLC for execution times and validated on the 1756-L61 CPU. This research builds upon Dunlap and Schuett’s work by reverse engineering the Task Monitor requests for the 1756-L71, 1769-L23, and 1769-L32 CPUs.

The requests are sent as CIP Command 0x6D unconnected send messages and routed to the PLC CPU via the CIP Connection Manager. The data fields returned are parsed as shown in Figure 3.6.

Task and system service execution times are recorded every 500ms for a total of 120 measurements, approximately 60 seconds. Each CPU is measured under three different firmware versions replicated five times, once with the PLC code collection script running and five times with no collection (baseline). This provides a total of 10 measurement sets per firmware version or 30 measurements per CPU.

Table 3.2: Listing of Experiment Runs for the 1756-L61 CPU.

Experimental Setup: L61			
Run	Project File Name	Firmware Version	Workload (B=0/T=1)
1	L61_20_11_59.ACD	20_11.bin	0
2	L61_20_11_59.ACD	20_11.bin	0
3	L61_20_11_59.ACD	20_11.bin	0
4	L61_20_11_59.ACD	20_11.bin	0
5	L61_20_11_59.ACD	20_11.bin	0
6	L61_20_11_59.ACD	20_11.bin	1
7	L61_20_11_59.ACD	20_11.bin	1
8	L61_20_11_59.ACD	20_11.bin	1
9	L61_20_11_59.ACD	20_11.bin	1
10	L61_20_11_59.ACD	20_11.bin	1
11	L61_19_11_16.ACD	19_11.bin	0
12	L61_19_11_16.ACD	19_11.bin	0
13	L61_19_11_16.ACD	19_11.bin	0
14	L61_19_11_16.ACD	19_11.bin	0
15	L61_19_11_16.ACD	19_11.bin	0
16	L61_19_11_16.ACD	19_11.bin	1
17	L61_19_11_16.ACD	19_11.bin	1
18	L61_19_11_16.ACD	19_11.bin	1
19	L61_19_11_16.ACD	19_11.bin	1
20	L61_19_11_16.ACD	19_11.bin	1
21	L61_16_23_16.ACD	16_56.bin	0
22	L61_16_23_16.ACD	16_56.bin	0
23	L61_16_23_16.ACD	16_56.bin	0
24	L61_16_23_16.ACD	16_56.bin	0
25	L61_16_23_16.ACD	16_56.bin	0
26	L61_16_23_16.ACD	16_56.bin	1
27	L61_16_23_16.ACD	16_56.bin	1
28	L61_16_23_16.ACD	16_56.bin	1
29	L61_16_23_16.ACD	16_56.bin	1
30	L61_16_23_16.ACD	16_56.bin	1

3.4.3 PLC Impact Experiments.

This experiment to measure impacts of code collection is conducted on a closed IP-based network shown in Figure 3.4 and Figure 3.5. Each CPU is measured independent of the other three CPUs.

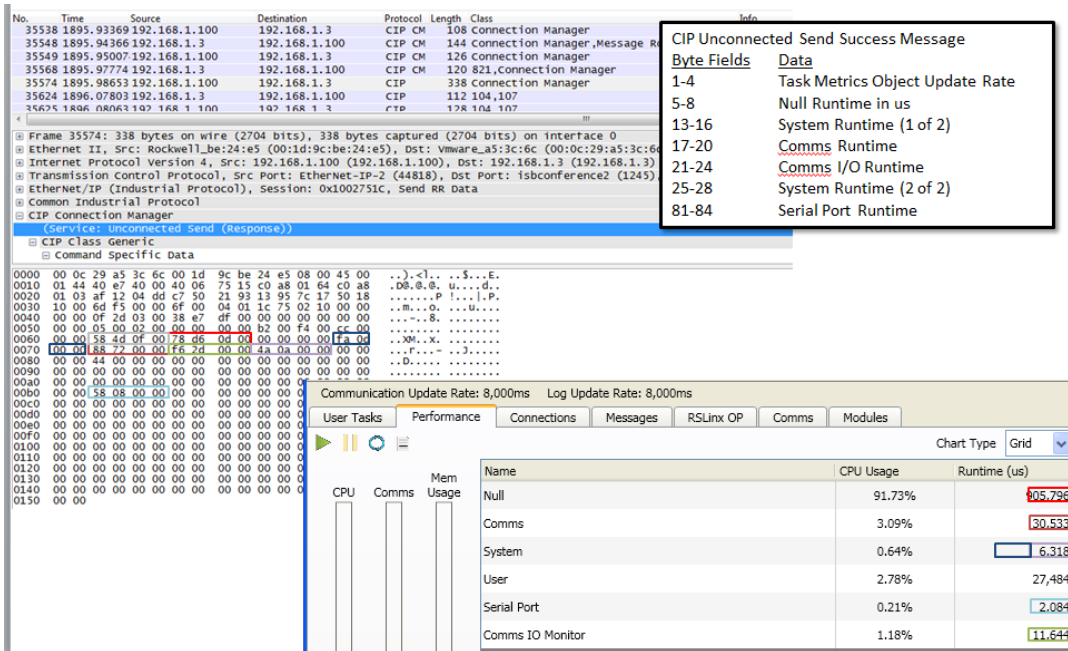


Figure 3.6: Task Monitor Graphical Interface shown with Wireshark CIP packet capture.

The procedure for downloading PLC code and taking execution time measurements is used as follows:

1. Configure the list of experimental runs as shown in Table 3.2 ensuring to randomize the list to reduce bias.
2. Begin the runexp.py script which automates project file download and task execution time measurement.
3. Use ControlFLASH software to load the appropriate Firmware to the PLC as identified by the runexp.py script.
4. Script loads the appropriate project file by replicating the RSLogix5000 download process.
5. Script places the PLC in Remote Run mode.

6. Script pauses for verification that controller is in Remote Run. Execution times will not be recorded if PLC did not reach Remote Run state.
7. The PLC runs for 60 seconds before taking measurements to allow it to reach steady state.
8. PLC code collection requests are made once each 500 millisecond measurement cycle to record 120 measurements over a 60 second period. See Figure 3.7 for collection sequence.
9. For treatment measurements, PLC code collection begins after a two second delay. Code collection script sets up the EtherNet/IP connection with the PLC and then sends one CIP request to the PLC per 500 millisecond measurement cycle.
10. PLC code collection stops approximately two seconds before measurement collection ends.
11. Verify task execution time measurements are valid before advancing the runexp.py script to the next line specified in the configuration file.

This process as depicted in Figure 3.7 is replicated five times for a baseline measurement and five times for a treatment measurement covering each CPU and firmware version in the experiment for a total of 30 runs per CPU. The list of 30 runs is randomized using the Random.Org List Randomizer to reduce the possibility of bias being introduced.

3.4.4 Visual Inspection of PLC Code.

The final process in this method of distinguishing Internet-facing ICS devices is the classification of PLC code using process control terms. A visual inspection is conducted on the PLC code obtained from each device to identify well-known process control terms listed in Appendix 1. The presence of these terms in PLC code is assumed to indicate that the device is controlling an industrial process, and results in the device being classified as

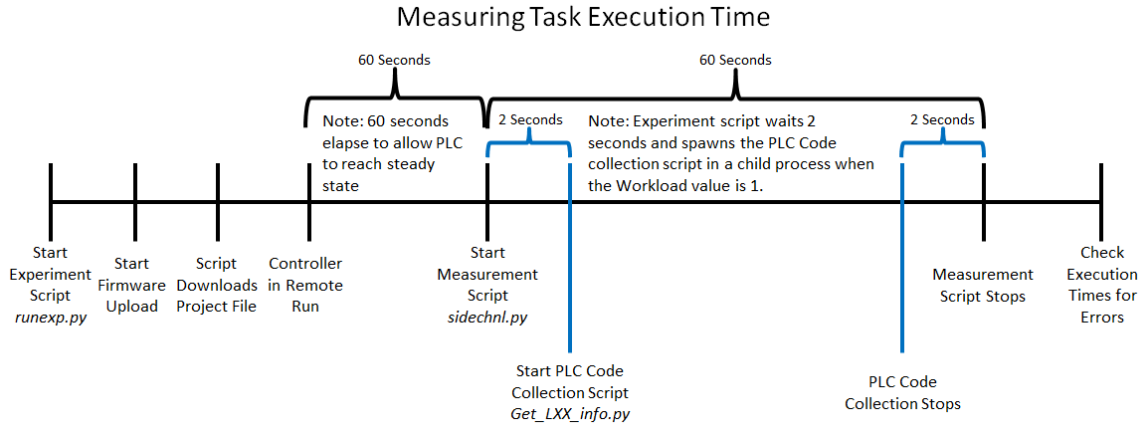


Figure 3.7: Measuring PLC Task Execution Times.

Process Control. When PLC code does not show the existence of process control terms, the device is classified as Indeterminate. During testing, the visual inspection of PLC code is performed to ensure PLC code collection scripts function properly and do not cause errors in the code or on the PLC.

3.5 Evaluation

Impacting a PLC controlling an industrial process has the potential to result in personal injury or have a devastating environmental or industrial impact. This research uses PLC performance analysis methods used in previous work by Schuett and Dunlap. Their work focused on detecting small changes in task execution times due to firmware or PLC code modifications. Measurements are collected over a sixty second run to obtain sufficient number of samples to perform statistical tests on the data. A total of 120 measurements are taken in 500ms intervals.

The measurements collected during experimentation are analyzed using the Monte-Carlo resampling technique. This is a nonparametric test that measures the effect of the treatment on a sample. In this research, the sample is the baseline task execution time and the treatment is the task execution times collected while the PLC code collection script is

running. The 9999 Monte-Carlo resamplings provide a p-value which is used to accept or reject a null hypothesis. The null hypothesis in this research is that there is no difference between the sample and treatment populations. The p-value returned represents the percent chance that the treatment value could have been obtained from the sample. If both the sample and treatment populations are the same, the p-value will be 1. If the sample and treatment populations are statistically different, the p-value will be below the chosen alpha value of 0.05 to provide a 95% confidence interval.

3.6 Conclusion

The methodology outlined in this chapter is used to identify Internet-facing ICS devices indexed by the Shodan search engine, obtain PLC code from those devices, and classify the device as Process Control or Indeterminate. Further, the experimental design measures the impact of PLC code collection on the ICS device in order to determine the feasibility of using this method to collect from ICS devices in production. The results of performance analysis and implementation of this methodology on Internet-facing ICS devices is discussed in the following chapter.

IV. Results and Implementation

THIS chapter describes the results obtained during exploratory testing, performance evaluation, and implementation of the research methodology. Exploratory testing details the reverse engineering of Allen-Bradley software and communications protocols. The performance evaluation section details the results of experiments designed to determine impact of PLC code collection on PLC task execution times. The implementation section describes the results of implementing the methodology on Internet-facing Allen-Bradley PLCs.

4.1 Exploratory Testing

This section describes the exploratory testing conducted to determine the CIP protocol requests that return PLC programming information.

4.1.1 Reverse Engineering Allen-Bradley PLC Code CIP Requests.

Programming information stored in the PLC is uploaded to a computer using a vendor-specific programming suite. In this research, a Windows XP Virtual Machine running RSLogix 5000 software is connected to an Allen-Bradley 1756-L61 PLC CPU using a 1756-ENBT EtherNet/IP module. Both devices are connected using Private IP space on a closed network. The environment for exploratory testing is shown in Figure 4.1.

Allen Bradley ControlLogix and CompactLogix PLCs use RSLogix 5000 for programming design and configuration. RSLogix 5000 uses an Upload process to load PLC code from a given PLC into the RSLogix 5000 interface. RSLogix 5000 uses CIP to request PLC code from the device, and by reverse engineering the Upload process, data packets are reconstructed which request PLC code from a device without using RSLogix 5000.

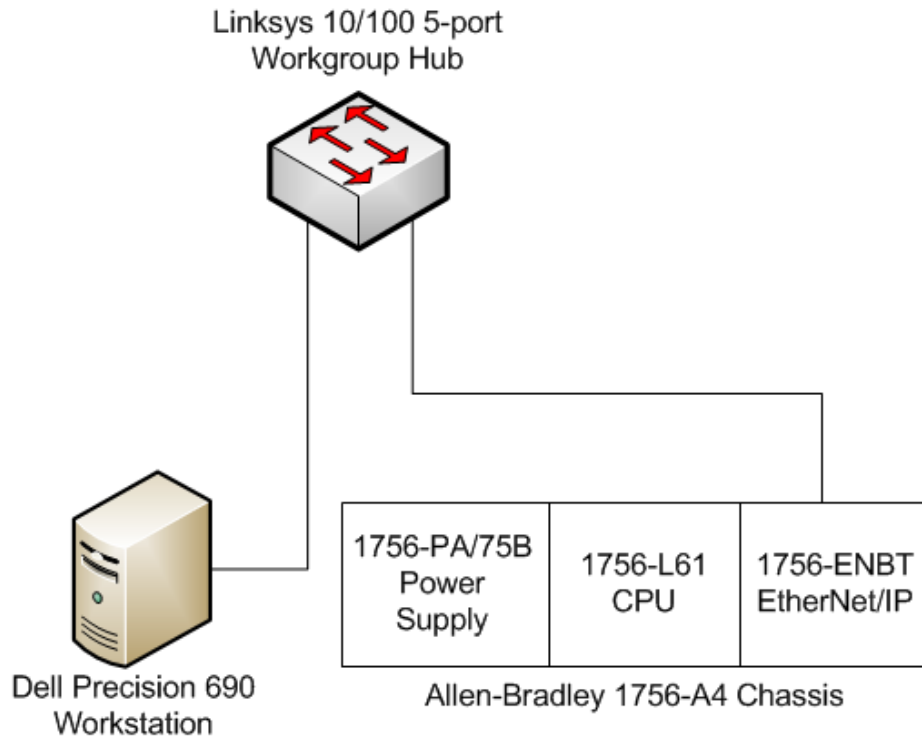


Figure 4.1: Hardware configuration for exploratory testing.

Wireshark is used to observe network traffic between the Virtual Machine hosting RSLogix 5000 software and an Allen-Bradley 1756-L61 CPU and 1756-ENBT EtherNet/IP module. Wireshark parses the files contained in each CIP data packet, which aids in static analysis.

RSLogix 5000 establishes an EtherNet/IP session with a PLC by executing the EtherNet/IP protocol Register Session command. Once this command is sent to the PLC, a session handle is returned and used in all subsequent communications between RSLogix 5000 and the PLC. The Register Session command is defined in Dunlap's work [10] and his code for registering an EtherNet/IP session and routing CIP messages to the PLC CPU is used in this research.

The CIP protocol is an object oriented protocol that uses classes and instances to describe data elements. PLC Tasks, Programs, Routines, and Tags are all referenced by a specific class and instance value. Allen-Bradley RSLogix 5000 PLCs use the class 0x6B to identify global data elements of PLC code such as Tasks, Programs, and Global Tags.

Observing RSLogix 5000 communications, the EtherNet/IP packet consists of the EtherNet/IP header, CIP header, and data. Figure 4.2 shows the hex representation of the EtherNet/IP packet used to request a list of nine attributes associated with the class and instance values listed in the request path.

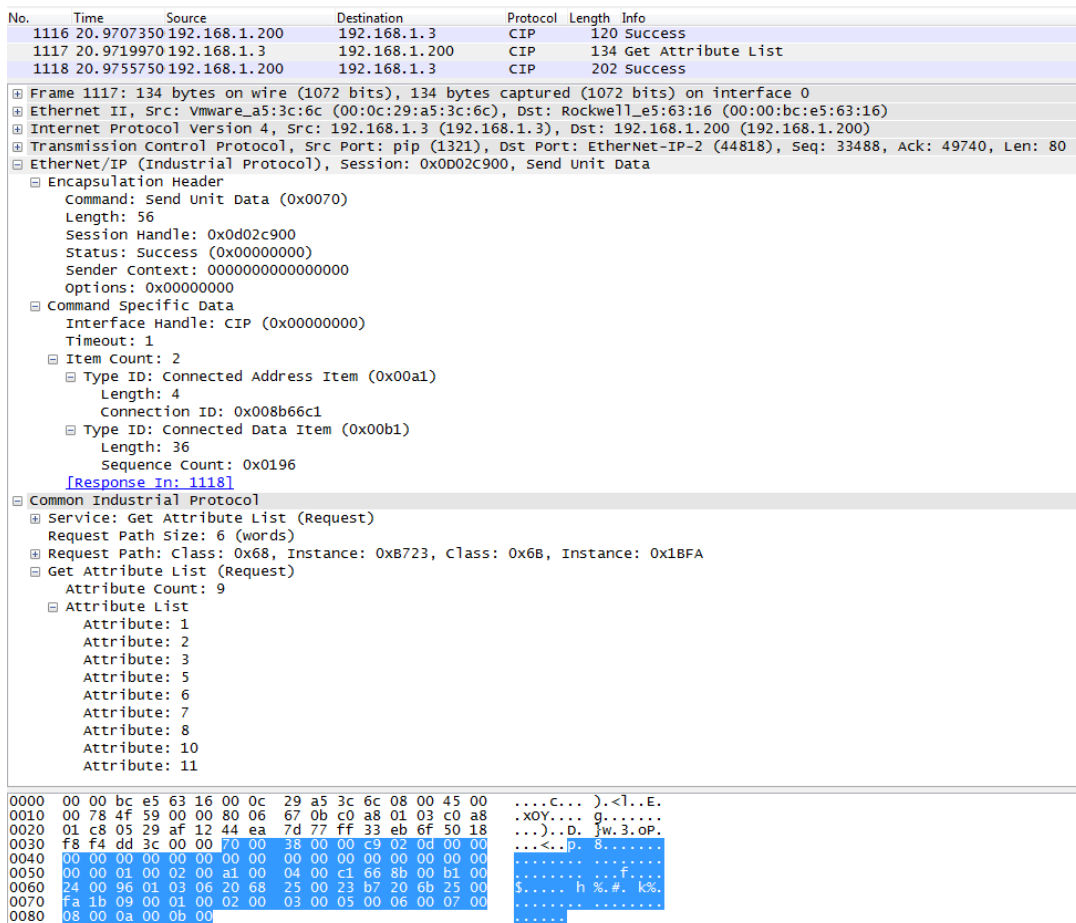


Figure 4.2: EtherNet/IP Packet.

RSLogix 5000 first sends a CIP request to the PLC with a request path of Class 0x6B, Instance 0x00 shown in Figure 4.3. The PLC responds with a CIP message listing the instance values of all global data elements. The response message data field is where an instance is represented by two bytes in little-endian formatting followed by an additional two Null bytes as padding. Similar request messages are crafted with Class 0x6B and a specific instance value which returns a data field for each instance containing a list of instance attributes.

No.	Time	Source	Destination	Protocol	Length	Status
1028	20.7005320	192.168.1.200	192.168.1.3	CIP	112 104	Success
1029	20.7017770	192.168.1.3	192.168.1.200	CIP	116 813,104	Unknown
1030	20.7050540	192.168.1.200	192.168.1.3	CIP	104 813,104	Success
1031	20.7065520	192.168.1.3	192.168.1.200	CIP	156 104	Get Attr

Frame 1028: 112 bytes on wire (896 bits), 112 bytes captured (896 bits) on interface 0 Ethernet II, Src: Rockwell_e5:63:16 (00:00:bc:e5:63:16), Dst: Vmware_a5:3c:6c (00:0c:29:a5:3c:6c) Internet Protocol Version 4, Src: 192.168.1.200 (192.168.1.200), Dst: 192.168.1.3 (192.168.1.3) Transmission Control Protocol, Src Port: EtherNet-IP-2 (44818), Dst Port: pip (1321), Seq: 45225, Ack: 1321 EtherNet/IP (Industrial Protocol), Session: 0x0D02C900, Send unit Data Common Industrial Protocol Service: Unknown Service (0x4b) (Response) Status: Success [Request Path Size: 2 (words)] [Request Path: class: 0x68, Instance: 0x00] [Path Segment: 0x20 (8-Bit Class Segment)] [Path Segment: 0x24 (8-Bit Instance Segment)] CIP Class Generic Command Specific Data Data: 2024000023b70000	
0000	00 0c 29 a5 3c 6c 00 00 bc e5 63 16 08 00 45 00 ..).<1.. ..C...E.
0010	00 62 e6 cc 40 00 40 06 cf ad c0 a8 01 c8 c0 a8 .b..@.@.
0020	01 03 af 12 05 29 ff 33 d9 cc 44 ea 71 21 50 18).3 ..D.q!P.
0030	10 00 5a f8 00 00 70 00 22 00 00 c9 02 0d 00 00 ...Z...p.
0040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0050	00 00 00 00 02 00 a1 00 04 00 0c 00 fe 80 b1 00
0060	0e 00 69 01 cb 00 00 20 24 00 00 23 b7 00 00 ...1..... 3..Z...

Figure 4.3: PLC Code Request for Global Instance Values.

At this point CIP requests transition from requesting instance values to requesting attributes assigned to each specific instance. The first messages observed returning attributes associated with PLC code are the pair of messages shown in Figure 4.4. Once the global instance values have been obtained, CIP attribute requests iterating over all global instance values return data fields for the nine attributes requested for each instance.

Attribute 1 contains American Standard Code for Information Interchange (ASCII) representations of each Task, Program, and Global Tag name. The Wireshark packet capture shown in the bottom right of Figure 4.4 decodes the ASCII text and *Task:task_bravo* is visible.

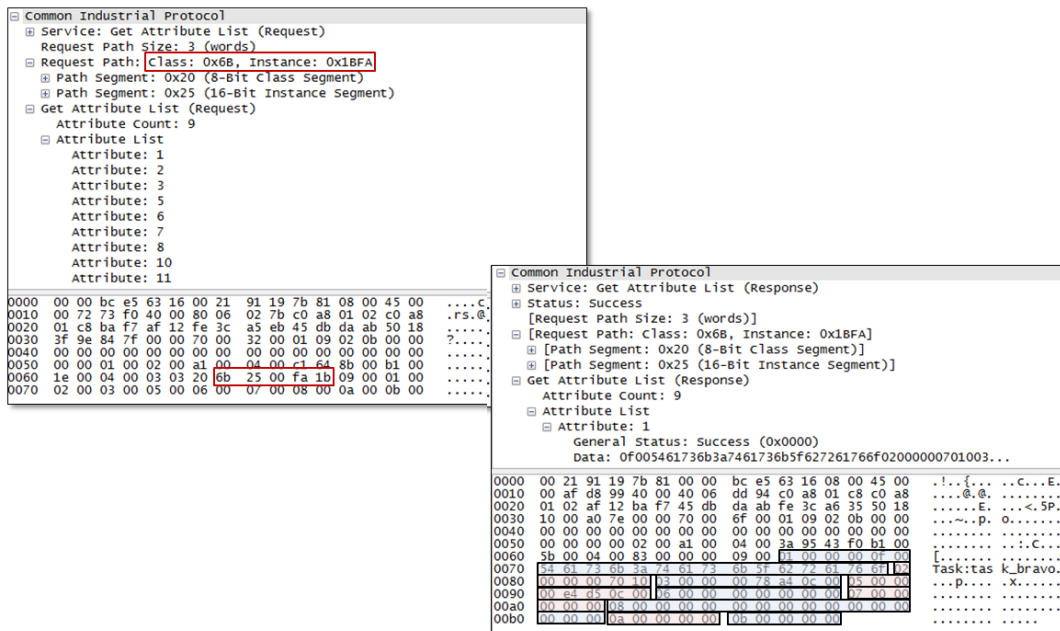


Figure 4.4: PLC Code Global Instance Request and Response.

RSLogix uses a similar process to obtain program-specific names for Routines and Program Tags as shown in Figure 4.5. RSLogix 5000 uses class and instance values for program-specific data. The first request path pair of Class 0x68 Instance 0x2420 indicate the request is for a program-specific item belonging to the global item (*Instance Value 0x2420*). The next request path pair of Class 0x6B Instance 0x00 indicates a request for all instance values belonging to the elements belonging to the parent Global instance value 0x2420.

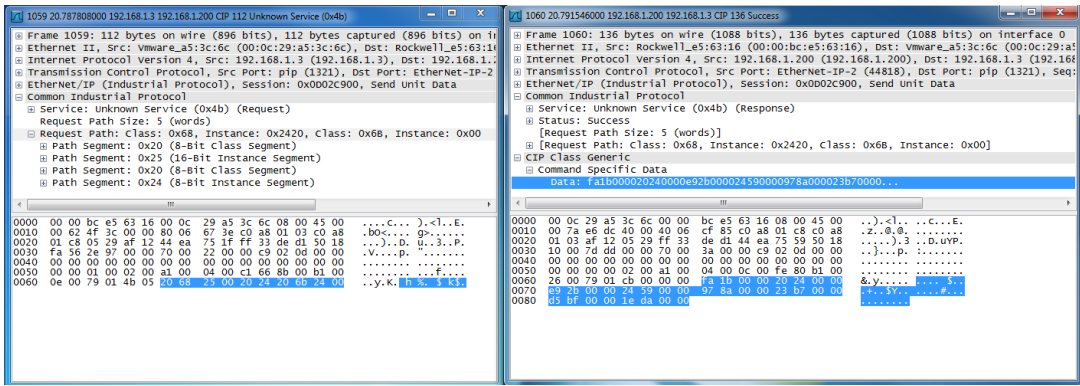


Figure 4.5: PLC Code Request for Program-Specific Instance Values.

RSLogix 5000 sends CIP requests iterating over all returned program-specific instance values to obtain the nine attributes for each program-specific Routine and Program Tag. Attribute 1 contains the ASCII name of each Routine and Tag returned as shown in Figure 4.6.

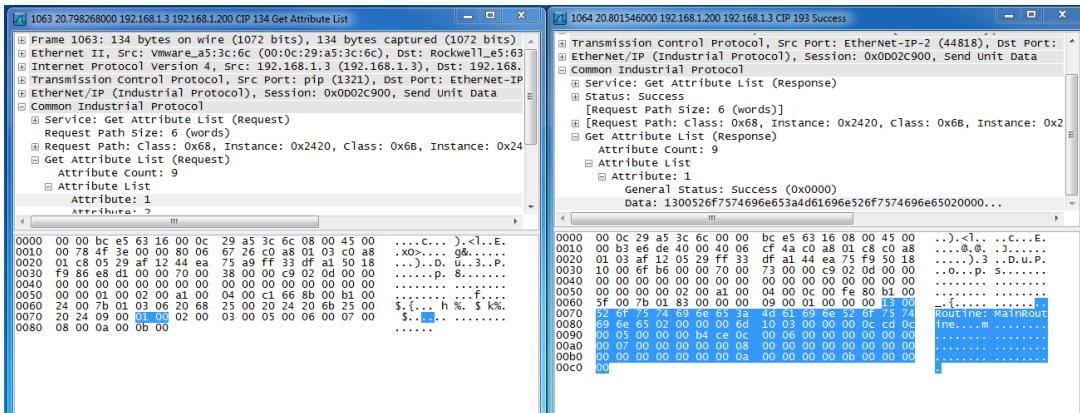


Figure 4.6: PLC Code Program-Specific Instance Request and Response.

Each CIP request is a request for nine attributes associated with a instance value. Attribute 1 is the ASCII string representing the name of the data element. Attribute 2 lists the data type of each data element. Observation of global and program-specific responses

identifies the values listed in Table 4.1. The fields for Attributes 3 through 9 are not readily identifiable and it is unknown what data is represented by the values in these fields.

Table 4.1: Data Type Values found in Attribute 2 CIP Instance Responses.

Selection of RSLogix 5000 Data Types	
Hex Value	Data Type
0x00C4	DINT Tag
0x00C1	Boolean Tag
0x106D	Routine
0x1070	Task
0x1068	Program
0x1069	Map (Global Tag)
0x107E	Cxn (Global Tag)

The Get Attribute List command discussed above returns a list of nine attributes given a global class and instance, or a program-specific class instance pair. It is important to note that during exploratory testing, there is no method developed that successfully mapped a program-specific instance back to a global instance. That is, a method for obtaining global and program specific PLC code is established, however, that method cannot determine which program-specific data elements are related to a global program. During static analysis of the remaining attributes, the values for the third and fourth attributes change each time a new project file is downloaded into the PLC. The fifth and sixth attributes only have values when referencing a global or program specific tag, and the remaining three attributes always contain a value of zero.

4.1.2 Conclusion.

This section describes reverse engineering that allows development of python scripts that replicate portions of the RSLogix 5000 Upload process. These scripts applied to Internet-facing ICS devices obtain PLC code used in this methodology to distinguish Internet-facing ICS devices. In order to implement these scripts on devices in production, performance analysis testing is conducted to determine if these code collection scripts impact PLC task execution times.

4.2 Performance Analysis Results

This section describes the results of experiments measuring impacts on PLC performance resulting from PLC code collection on an Allen-Bradley PLC. The data collected is defined in terms of statistical distribution, error conditions, outliers, group mean, standard deviation, minimum, and maximum values.

4.2.1 Data.

PLC execution measurements are read into the R software package for statistical analysis [10]. A total of 120 sets of measurements are recorded in microseconds with a resolution of two microseconds. Execution times are reported from the PLC every 500 milliseconds over a 60 second test period. The PLC code collection script includes a *time.sleep(0.5)* command in order to ensure only one request per measurement cycle is sent to the PLC. This maximizes the speed of code collection while limiting the number of CIP commands issued to the PLC CPU per cycle.

The results from execution time collection are not approximately normal as shown in the Q-Q plot in Figure 4.7. This plot consists of data taken during the first baseline run for the L61 CPU on version 16.56.47 firmware. The remaining runs measuring all CPUs and firmware versions have similar Q-Q plots, therefore assuming the data is approximately normal is not valid. The nonparametric Monte Carlo resampling test is used in this research to obtain p-values.

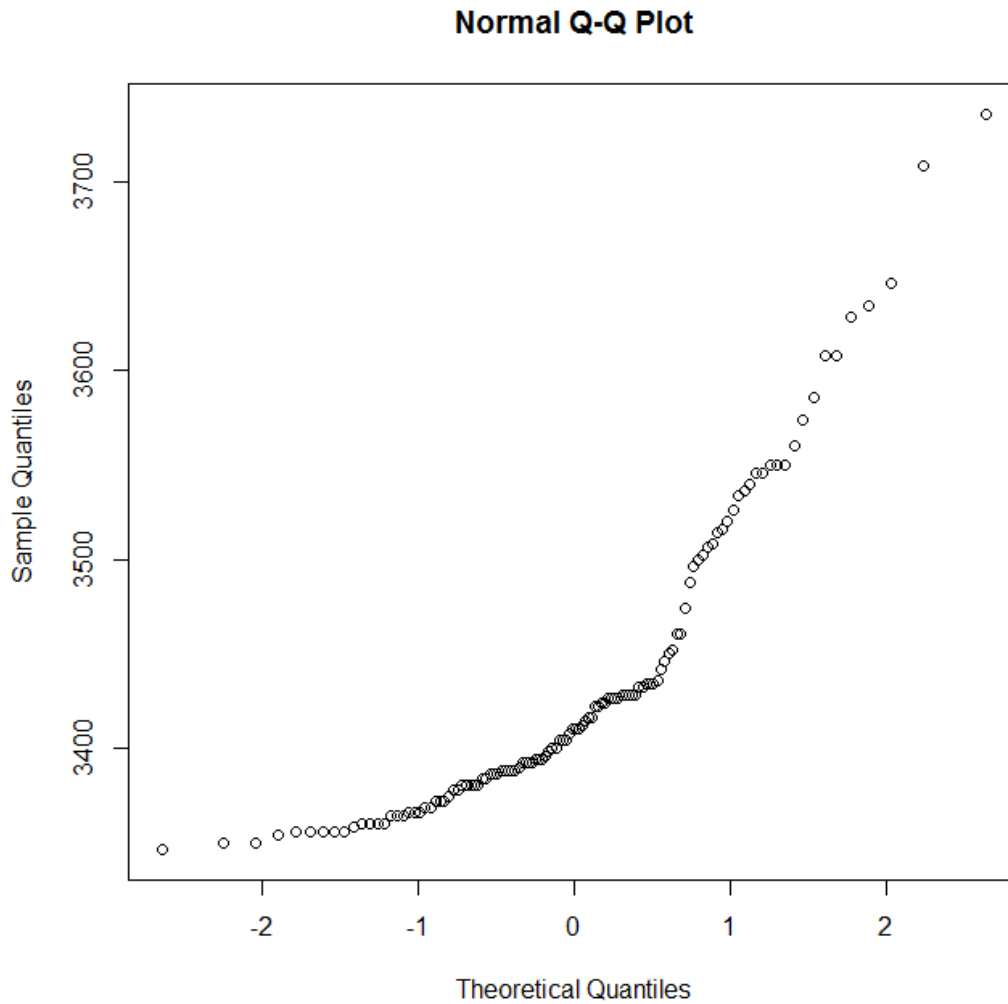


Figure 4.7: R Q-Q Plot of PLC Task Execution Times L61 v16.56.47.

Dunlap and Schuett identified outliers which may be caused by logical interrupts and excluded them from statistical analysis. The PLC execution times in their research are stable since the PLC is executing code without any external influences. Dunlap and Schuett noted that some execution times are well outside the normal variance and investigating due to software interrupts in the PLC scheduling algorithm causing artificial delays in code execution times. Outliers in this data set appear in a similar manner observed in Dunlap and Schuett's research; however, this research counts them as valid measurements.

A key difference between the work done by Dunlap and Schuett and this research is the measurement of a treatment on the sample data set. The effect that a CIP packet has on the CPU cannot be distinguished from the effect a software interrupt has on the CPU. Therefore, outliers are included in statistical analysis for this research.

Boxplots shown in Figure 4.8 depict the five Baseline runs (B1-B5) and five Treatment runs (T1-T5) for the L61 CPU on version 16.56.47 firmware. The boxplots are a graphical depiction of the data distribution. The solid lines in each box depict the median value and the box itself encompass the 25th to 75th percentiles of the data in each set of measurements. If each replication is stable and predictable, boxplots one through five would have similar means and quantiles, and boxplots six through ten would be similar to each other as well. Measurements show that there is significant unequal variance in the measurement of execution times among the replications of similar experiments. The distribution of execution times for run T4 appear to be much smaller than the distribution of execution times for run B1, even though run T4 had the treatment applied and run B1 is a baseline measurement. The boxplot also shows the outliers which are plotted above the top whisker line. While code collection cannot be eliminated as a contributor to outliers, it is evident that there are a number of outliers present in the baseline measurements B1 through B5.

Looking more closely at a run with a small variance, a scatter plot of run B2 shows the separation between PLC execution times with outliers. The scatter plot in Figure 4.9 shows the distribution of task execution time measurements with a solid line at the 3478 microsecond mark which corresponds to the top whisker shown in the Figure 4.8 boxplot. Measurements above the red line in the scatter plot represent the outliers shown in the Run B2 boxplot in Figure 4.8.

The differences in median and variance seen in the L61 version 16.56.47 boxplot and scatterplot are common among the data sets measuring the three firmware versions for all

L61 CPU and Firmware version 16.56.49

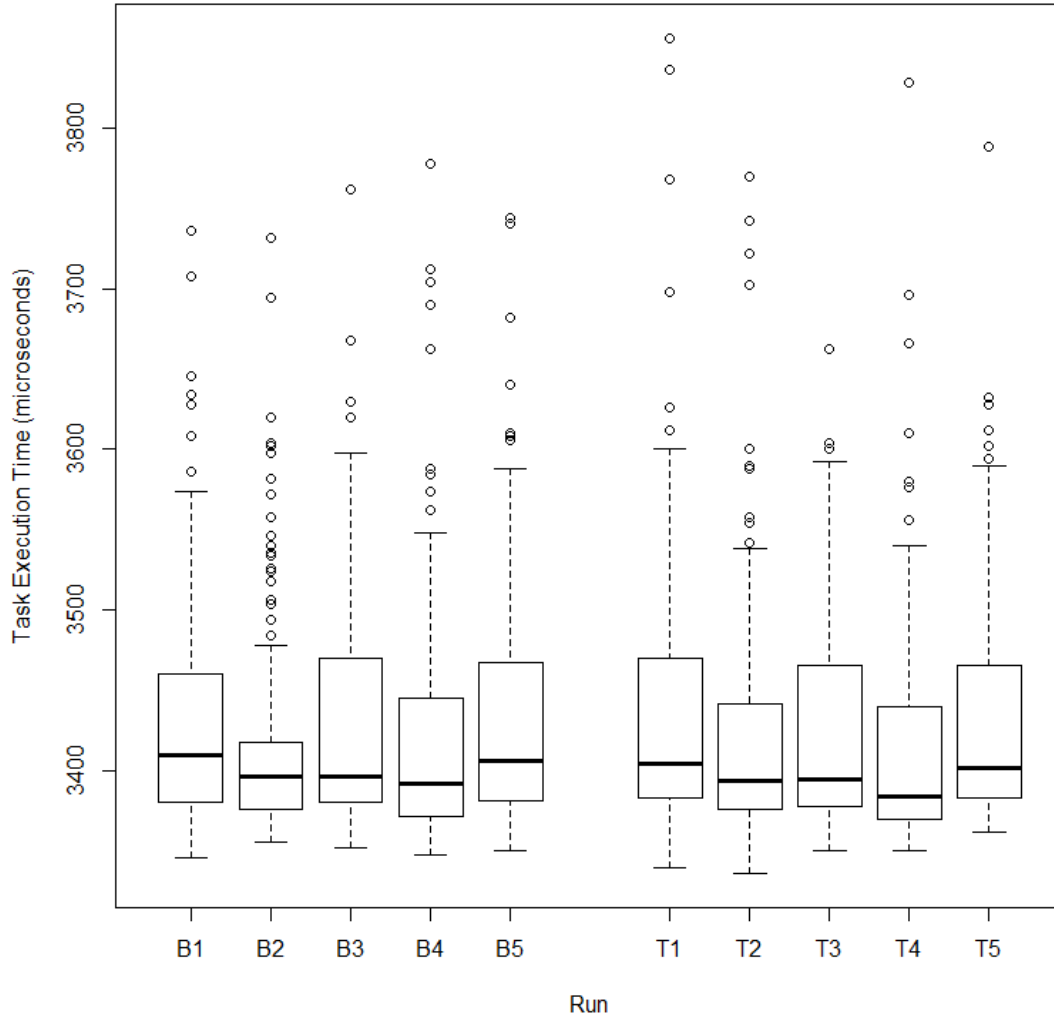


Figure 4.8: Boxplots of PLC Task Execution Times L61 v16.56.47.

four CPUs. The statistical differences determined by one way Monte Carlo resampling are discussed in the next section.

4.2.2 Non-parametric Statistical Analysis.

Statistical analysis for this data set uses the same one way permutation test used by Dunlap and Schuett in their research. The permutation test does not rely on an assumption that the data is approximately normal. The data collected in this research has been shown to

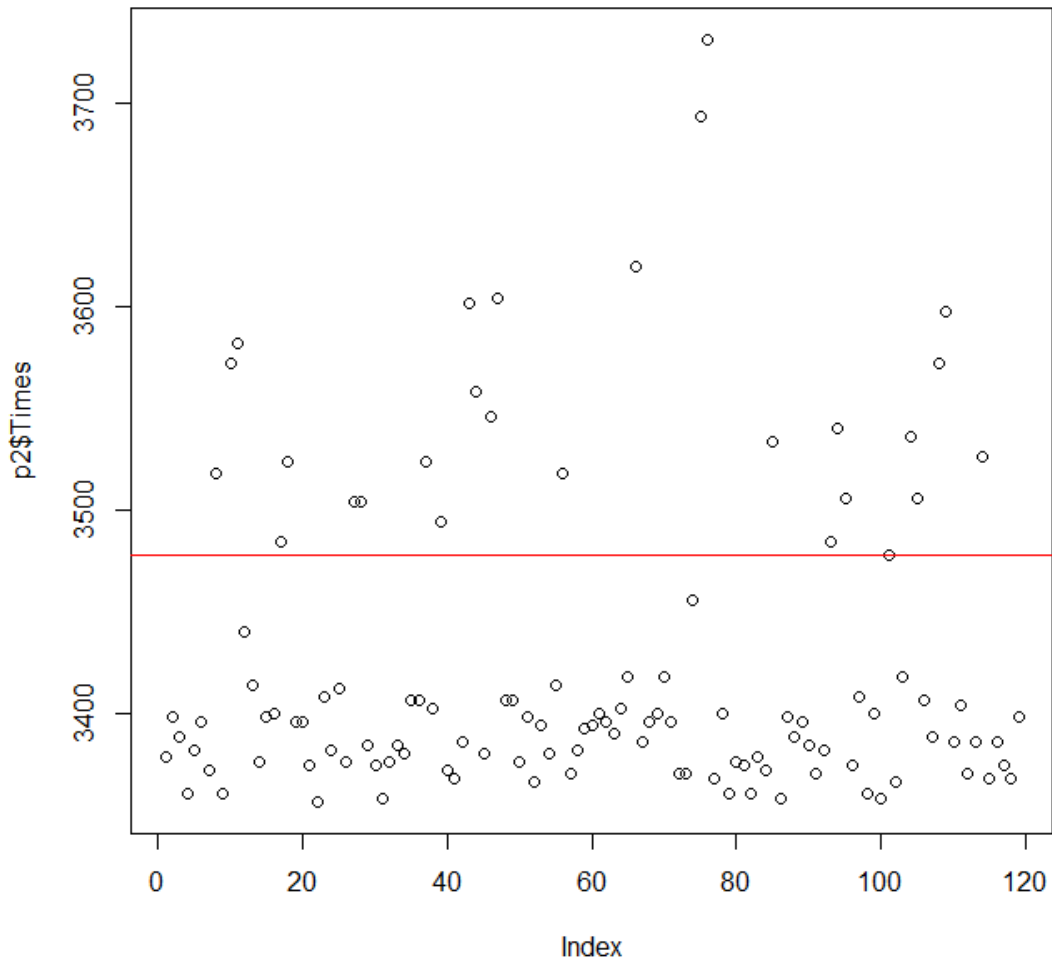


Figure 4.9: Run B2 Scatter Plot for PLC Task Execution Times L61 v16.56.47.

be skewed and not normally distributed. The one way permutation with 9999 Monte-Carlo resamplings is calculated using the Coin package in R.

The one way permutation test for differences in mean task execution times provides a p-value which represents the percent chance that a random variable representing the treatment set could come from the baseline set. When comparing the first run of the baseline set to the first run of the treatment set, a p-value below the threshold value means there is a statistically significant difference between the two sets. The threshold value selected for this research is 0.05.

4.2.3 Analysis.

The p-values obtained for each CPU/firmware combination show statistically significant differences among three of the four CPUs. Since the one way permutation function tests for differences in the mean sample, some of the differences are due to faster, not slower, task execution times. In those cases, the difference is noted, however it is not considered an impact in this research.

The following subsections list tables containing p-values for each of the four CPUs tested. The baseline runs are listed on the table's vertical axis and are labeled B1 through B5. The treatment runs are on the table's horizontal axis and labeled T1 through T5. The p-values listed in each Table are obtained from the one way permutation test comparing the treatment runs to the baseline runs. P-values are listed in bold when they are below the threshold value due to an increase in task execution time during the treatment run. P-values are underlined when the permutation test measures a statistically significant difference in means, however the treatment run executed faster than the baseline run.

Where statistically significant differences are shown, boxplots are included to illustrate the variance in median, variance, quantile, and outlier values.

4.2.3.1 L61 CPU.

Table 4.2 lists the p-values obtained for the L61 CPU. Treatment 2, 3, and 4 all show statistically slower runs when compared to Baseline runs 2, 4, and 5. Examining the boxplot for this run in Figure 4.10, it is clear that Treatment 2, 3, and 4 fit within the range of execution times returned in the Baseline runs. In fact, the run with the largest execution times for this CPU and firmware is Baseline 3. Statistically faster runs are detected comparing Treatment 1 with Baseline 1 and 3.

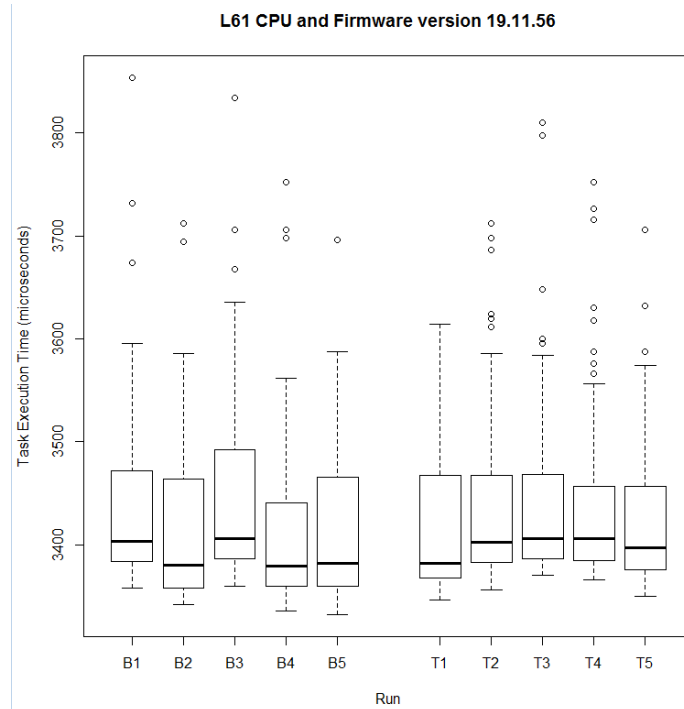


Figure 4.10: Boxplot for L61 v19.11.56.

4.2.3.2 L71 CPU.

Table 4.3 lists the p-values obtained for the L71 CPU. None of the p-values indicate a statistically significant increase in task execution times.

4.2.3.3 L23E CPU.

Table 4.4 lists the p-values obtained for the L23E CPU. Eight of the 25 p-values returned for firmware version 17.7.63 show statistically longer task execution times. Examining the boxplot in Figure 4.11 Baseline 5 has the third fastest task execution times and lowest variance across all ten sets while Treatment 5 has one of the highest median and 3rd Quadrant values among all ten sets. When these two sets are excluded, only one of the remaining 20 p-values are below the selected threshold with slower task execution times.

Table 4.2: L61 Baseline vs. Treatment p-values (Bold indicates slower execution time).

1756-L61 P-values					
Firmware version 16.56.47					
	T1	T2	T3	T4	T5
B1	0.710071007	0.385838584	0.220822082	0.087008701	0.860286029
B2	0.172017202	0.873787379	0.920192019	0.485248525	0.359835984
B3	0.468046805	0.627062706	0.412841284	0.182018202	0.828682868
B4	0.269426943	0.97839784	0.768376838	0.406840684	0.507150715
B5	0.922092209	0.253625363	0.130213021	<u>0.04850485</u>	0.619261926
Firmware version 19.11.56					
	T1	T2	T3	T4	T5
B1	<u>0.01438924</u>	0.872059554	0.738535269	0.847818191	0.291268318
B2	0.753083092	0.012227441	0.00246279	0.004025775	0.093014581
B3	<u>0.003532936</u>	0.54701249	0.918845547	0.80667756	0.130273047
B4	0.825904893	0.017721752	0.003962985	0.006263625	0.120199879
B5	0.927038441	0.021955656	0.004996289	0.007835988	0.143674488
Firmware version 20.11.59					
	T1	T2	T3	T4	T5
B1	0.608040308	0.694409633	0.59214477	0.704535777	0.066645462
B2	0.552817863	0.776946749	0.677119928	0.790750211	0.092981131
B3	0.671688605	0.653498384	0.555890695	0.661875626	0.064103369
B4	0.650188876	0.648850737	0.546769661	0.655479734	0.057078883
B5	0.484296053	0.859180663	0.7587753	0.876351546	0.116058067

Table 4.3: L71 Baseline vs. Treatment p-values (Bold indicates slower execution time).

1756-L71 P-values					
Firmware version 20.11.59					
	T1	T2	T3	T4	T5
B1	0.628762876	0.918891889	0.756875688	0.97379738	0.899289929
B2	0.98219822	0.765276528	0.848384838	0.606960696	0.570357036
B3	0.403840384	0.683468347	0.473547355	0.735473547	0.832683268
B4	0.411741174	0.696869687	0.498049805	0.746874687	0.842984298
B5	0.415041504	0.280928093	0.251525153	0.139113911	0.154515452
Firmware version 20.12.79					
	T1	T2	T3	T4	T5
B1	0.354835484	0.815881588	0.611161116	0.114611461	0.919191919
B2	0.819481948	0.656265627	0.177617762	0.391339134	0.441844184
B3	0.733573357	0.785178518	0.289428943	0.364836484	0.567256726
B4	0.848284828	0.654065407	0.182118212	0.436443644	0.443944394
B5	0.260826083	0.613461346	0.847084708	0.079607961	0.881588159
Firmware version 20.13.81					
	T1	T2	T3	T4	T5
B1	0.511351135	0.162716272	0.320632063	0.716271627	0.164116412
B2	0.218821882	0.659865987	0.320332033	0.102710271	0.617661766
B3	0.853685369	0.314931493	0.608160816	0.877087709	0.307430743
B4	0.303430343	0.820982098	0.453545355	0.150215022	0.801880188
B5	0.850185019	0.509750975	0.891889189	0.565756576	0.518451845

4.2.3.4 L32E CPU.

The L32E p-values in Table 4.5 show statistically significant differences for all three firmware versions tested. Firmware 16.23.15 has 11 of 25 p-values below the selected

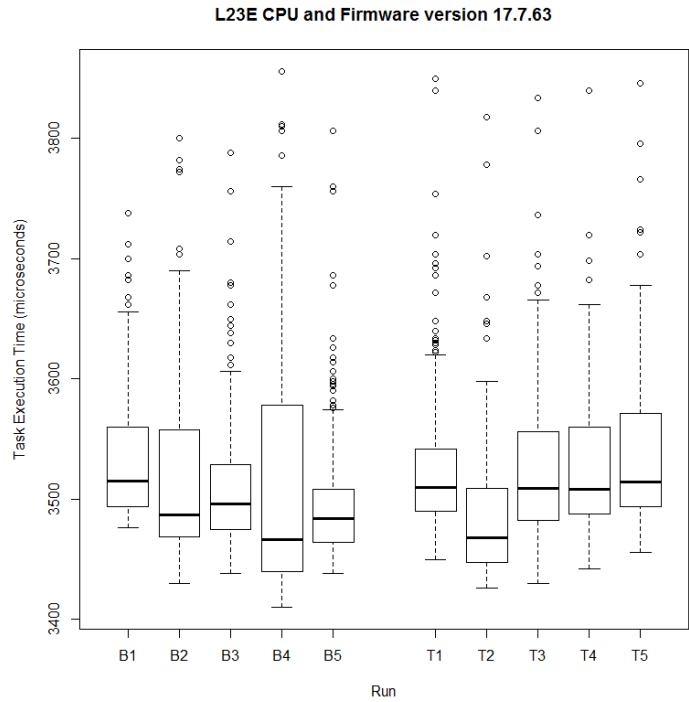


Figure 4.11: Boxplot for L23E v17.07.63.

threshold, however 10 of the 11 have faster mean task execution times in the Treatment set than in the Baseline set. The only p-value that indicates a statistically slower task execution time in the Treatment set is the p-value comparing the fastest Baseline set (Baseline 2) to the slowest Treatment set (Treatment 3).

Firmware 17.12.64 has eight of the 25 p-values below the selected threshold. All eight below the threshold p-values are common to the Baseline 2, 4, and 5 sets and the Treatment 1, 2, and 5 sets. The boxplot in Figure 4.12 shows that Baseline 2, 4, and 5 have lower median values than the remaining two Baseline sets. Similarly, Treatment 1, 2, and 5 have the highest three median values. These treatment sets have statistically slower task execution times when compared to the fastest Baseline sets, however, the range of values in each of the Treatment sets are not outside what is observed in the Baseline sets.

Table 4.4: L23E Baseline vs. Treatment p-values (Bold indicates slower execution time).

1769-L23E P-values					
Firmware version 17.7.63					
	T1	T2	T3	T4	T5
B1	0.886988699	<u>0</u>	0.455445545	0.471947195	0.525352535
B2	0.062206221	<u>0.02640264</u>	0.169716972	0.133813381	0.01190119
B3	0.03780378	<u>0.01360136</u>	0.126212621	0.089108911	0.00420042
B4	0.155215522	<u>0.0340034</u>	0.323032303	0.281228123	0.04710471
B5	0.00180018	0.183718372	0.00730073	0.00360036	0
Firmware version 18.12.57					
	T1	T2	T3	T4	T5
B1	0.804880488	0.440844084	0.127912791	0.294729473	0.420442044
B2	0.709270927	0.525052505	0.164716472	0.371537154	0.503150315
B3	0.126712671	0.550755076	0.810181018	0.773877388	0.634963496
B4	0.308030803	0.95049505	0.439443944	0.805680568	0.97779778
B5	0.04280428	0.219321932	0.736573657	0.381338134	0.298629863
Firmware version 19.11.16					
	T1	T2	T3	T4	T5
B1	0.649164916	0.869386939	0.514851485	0.929592959	0.660566057
B2	0.799279928	0.734273427	0.442844284	0.924792479	0.567356736
B3	0.695069507	0.837583758	0.506550655	0.96649665	0.638363836
B4	0.619761976	0.907790779	0.546054605	0.894489449	0.705470547
B5	0.749074907	0.760376038	0.447344734	0.96539654	0.589458946

The boxplot for firmware version 20.13.81 shown in Figure 4.13 shows two common groupings of task execution times. Baseline 1, 2, and 3 are comparable with Treatment 4

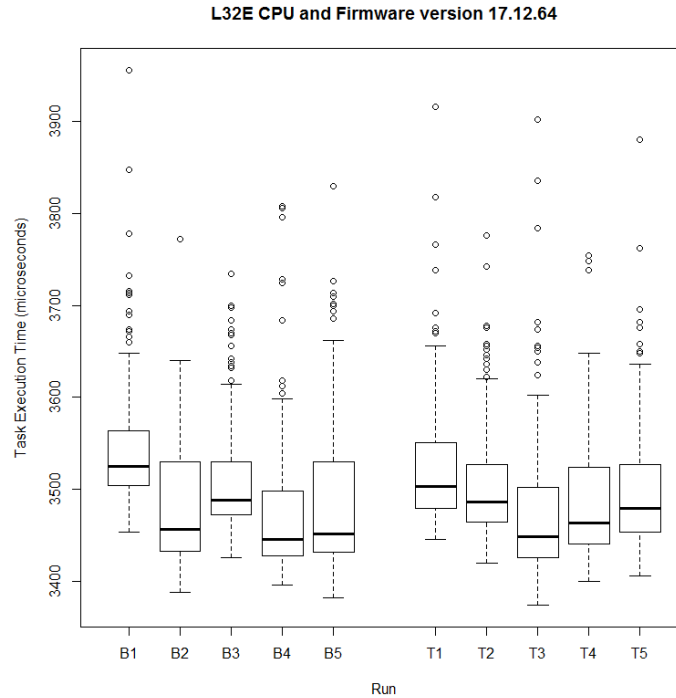


Figure 4.12: Boxplot for L32E v17.12.64.

and 5 while Baseline 4 and 5 are comparable with Treatment 1, 2 and 3. When analyzed in these two subgroups, all calculated p-values are above the selected threshold.

4.2.4 Discussion.

Three of the four CPUs tested exhibited statistically different outcomes comparing a given Baseline run to a Treatment run. The experiments are performed with five replications per Baseline and five replications per Treatment run. The boxplots of these runs show differences within the five replications that are similar to the differences observed when Baseline and Treatment runs are compared.

The PLC code collection script makes one CIP request to the PLC per measurement cycle. This ensures the script places the lowest load possible on the CPU. The p-values above demonstrate statistically significant differences among individual Baseline

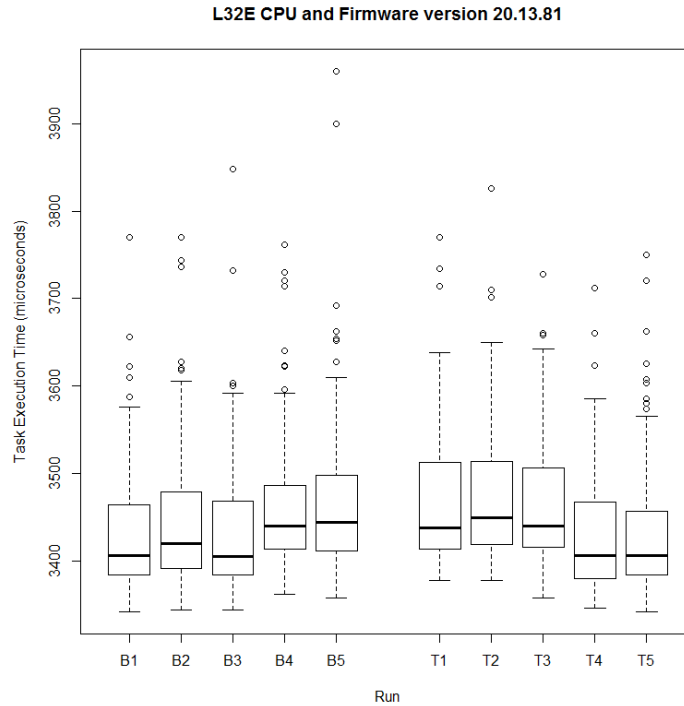


Figure 4.13: Boxplot for L32E v20.13.81.

or Treatment runs, however, none of the experiments demonstrate statistically significant differences across more than nine of 25 calculated p-values.

During all 120 experiments, the PLC code collection script had a 100 percent success rate collecting all PLC code from the PLCs under test. Major and minor PLC faults are not detected or logged during data collection. Fault detection during code collection relies on comparing the results of code collection against the PLC code downloaded into the PLC.

For each CPU and firmware version tested, the minimum and maximum values measured for all five Baseline experiments approximates the minimum and maximum values measured for all five Treatment experiments. The data fails to indicate an overall statistically slower task execution time during code collection, therefore the code collection scripts can be implemented on Internet-facing ICS devices.

Table 4.5: L32E Baseline vs. Treatment p-values (Bold indicates slower execution time).

1769-L32E P-values					
Firmware version 16.23.15					
	T1	T2	T3	T4	T5
B1	0.894689469	<u>0.00020002</u>	0.204620462	0.711871187	<u>0</u>
B2	0.171417142	<u>0.0130013</u>	0.01790179	0.093909391	<u>0.00970097</u>
B3	0.792979298	<u>0.00010001</u>	0.188318832	0.627462746	<u>0.00050005</u>
B4	0.718171817	<u>0</u>	0.433043304	0.874587459	<u>0</u>
B5	0.512251225	<u>0</u>	0.654365437	0.630563056	<u>0</u>
Firmware version 17.12.64					
	T1	T2	T3	T4	T5
B1	<u>0.0180018</u>	<u>0</u>	<u>0</u>	<u>0</u>	<u>0</u>
B2	0	0.0040004	0.757975798	0.498649865	0.02470247
B3	0.157715772	0.789378938	<u>0.00120012</u>	<u>0.01370137</u>	0.383338334
B4	0	0.00370037	0.98819882	0.355835584	0.02140214
B5	0.00010001	0.03090309	0.493449345	0.885888589	0.126512651
Firmware version 20.13.81					
	T1	T2	T3	T4	T5
B1	0.00050005	0	0.00050005	0.98659866	0.96969697
B2	0.02930293	0.01820182	0.04570457	0.185318532	0.193719372
B3	0.00170017	0.00040004	0.00280028	0.733073307	0.748974897
B4	0.398539854	0.315131513	0.561256126	<u>0.00460046</u>	<u>0.00570057</u>
B5	0.99469947	0.930493049	0.778077808	<u>0.00080008</u>	<u>0.00080008</u>

4.3 Implementation Results

This section describes the results of implementing the methodology for distinguishing ICS devices on a set of Internet-facing Allen-Bradley PLCs obtained from a Shodan search query.

The process shown in Figure 3.1 is used to obtain a list of Internet-facing Allen-Bradley PLCs, determine CPU and CPU slot for each device, collect PLC code, check for errors, and distinguish PLCs by groups of Process Control and Indeterminate devices.

4.3.1 Identifying Internet-facing ICS devices.

Collecting PLC code from Internet-facing ICS devices begins with developing a list of devices to interrogate. The search query used, “GoAhead index.html close”, is the same used by Bodenheimer in his research [5]. This query returns a list of Allen-Bradley CompactLogix and ControlLogix PLCs that have been previously indexed by Shodan. Shodan returns a list shown in Figure 4.14 that is parsed to obtain a list of 540 target IP addresses.

Shodan responds to the query “GoAhead index.html close” with a list of indexed Allen-Bradley CompactLogix and ControlLogix PLCs. Shodan lists IP addresses for each device indexed along with the indexed response message matching the search query. Shodan also allows registered users to access the Details view shown in Figure 4.15. The Details view shows the service and response received for each instance Shodan indexed the device. The figure shows the last two instances where the device responded to a Shodan request on HTTP port 80, and also shows a response on EtherNet/IP port 44818. During the course of this research, it was discovered that Shodan is indexing more ports and protocols than those listed in its documentation. Port 44818 is not advertised as a port that Shodan indexes, however, when entering the query “port:44818” into Shodan, a list is returned of every Internet-facing device that responded to a Shodan request for port 44818. At the time of this research, Shodan has indexed 3608 devices on port 44818.

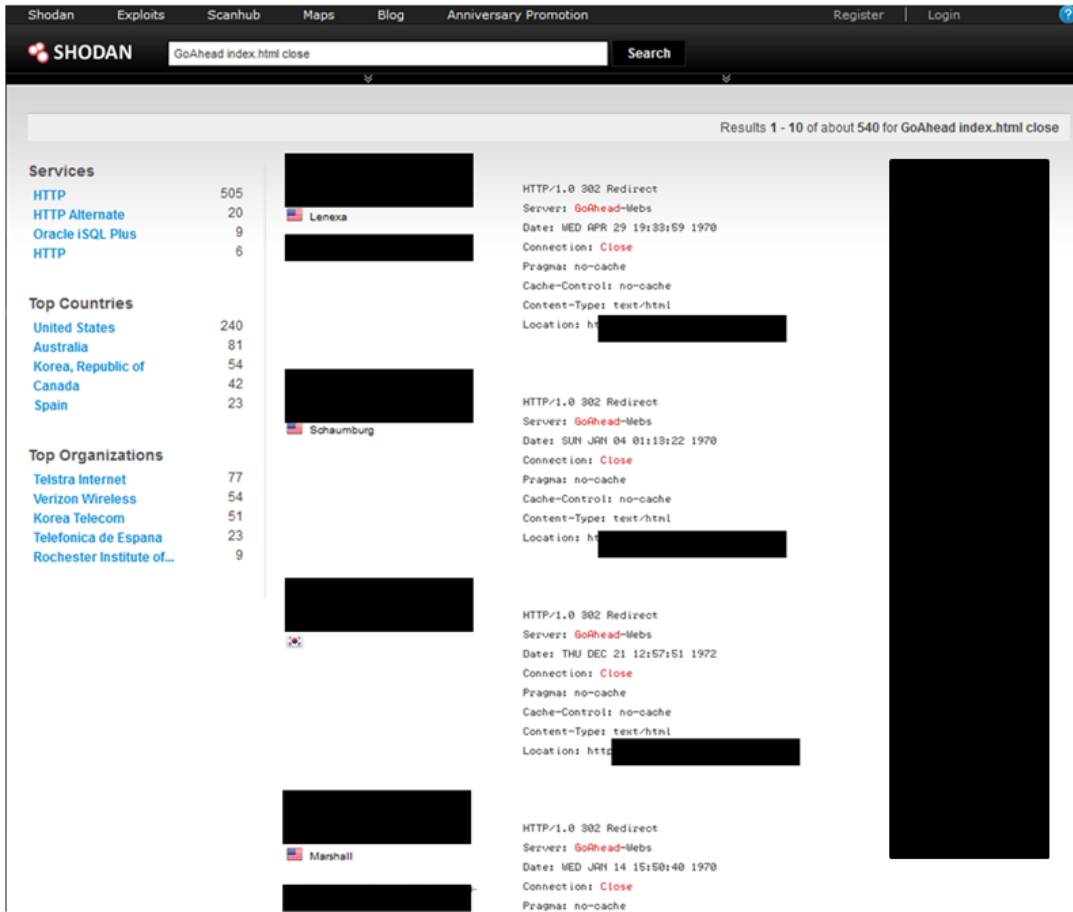


Figure 4.14: Process to Distinguish Internet-facing ICS Devices (Black boxes contain redacted information).

Shodan also provides a fee-based service where search results are exported to an XML file. The XML file returns a set of metadata on each device including location based on IP registration, last update, port and response message indexed, and IP address. The XML file returned by Shodan for the query “GoAhead index.html close” listed 493 unique IP addresses for indexed devices. The next step is to extract a list of IP addresses using a simple python script that parses XML and writes IP addresses to a text file used to collect Diagnostic Web Page data.

Host Profile:

Summary IP: [REDACTED]
 Location: 🇺🇸 Schaumburg, United States
 Latitude/ Longitude: 42.0334, -88.0834

HTTP HTTP/1.0 302 Redirect
 Server: GoAhead-Webs
 Date: SUN JAN 04 01:13:22 1970
 Connection: Close
 Pragma: no-cache
 Cache-Control: no-cache
 Content-Type: text/html
 Location: [REDACTED]

HTTP HTTP/1.0 302 Redirect
 Server: GoAhead-Webs
 Date: THU SEP 17 10:36:28 1970
 Connection: Close
 Pragma: no-cache
 Cache-Control: no-cache
 Content-Type: text/html
 Location: [REDACTED]

EtherNetIP Product name: 1769-L35E Ethernet Port
 Vendor ID: Rockwell Automation/Allen-Bradley
 Serial number: 438695758, Device type: Communications Adapter

Figure 4.15: Registered User Details view for an ICS device cataloged by Shodan (IP addresses redacted).

4.3.2 Collecting Diagnostic Web Page Data.

A simple python script collects data from Diagnostic Web Pages on indexed PLCs. Module type and firmware revision for CompactLogix devices is collected from the Home page and the Browse Chassis page lists CPU, slot, and firmware version for ControlLogix

devices. This information is parsed by the python script and written to a csv file used to create four CPU-specific configuration files used to obtain PLC code. 200 of the 493 IPs returned by Shodan are Internet-facing Allen-Bradley PLCs, with 167 being PLCs using CPU families tested in this research: 1756-L6x, 1756-L7x, 1769-L23E, 1769-L32E/35E.

4.3.3 Collecting PLC Code.

The python scripts developed for code collection are CPU-specific because each CPU returns data formatted in a different manner. PLC code collection scripts take in a configuration file listing IP addresses for CompactLogix PLCs and IP address, processor slot pairs for ControlLogix PLCs. Configuration files are taken from Diagnostic Web Page information and randomized using the Random.Org List Randomizer. Each code collection script sends EtherNet/IP messages to a device and logs response messages and PLC code into csv files. The script writes the raw EtherNet/IP protocol response message, then parses the data field and writes the ASCII text string found in Attribute 1.

The scripts log errors when no data is received from PLC code requests, and in three cases responses are received with no data. These three devices are assumed to have no ladder logic loaded into the PLC. Pilot testing shows that responses from these three devices match responses received from a PLC without PLC code. These three devices are not considered in the pool of error conditions.

During PLC code collection, 10 errors are recorded. One device did not respond to any requests on port 80 or port 44818 and is assumed to be no longer connected to the Internet. Eight devices resulted in a connection timeout error and one device resulted in a connection refused error.

4.3.4 Distinguishing ICS Devices.

The names returned in Attribute 1 are reviewed by two researchers familiar with process control systems and PLC programming. The panel visually inspects the PLC code written for each device for process control terms. The presence of these process control

terms is assumed to indicate the device is controlling an industrial process, and the device is classified Process Control by the panel. The list of terms the panel identified in the Process Control group are listed in Appendix 1.

After the panel review, the PLC code is delivered to an ICS Engineer for expert review. The engineer was given instructions to review the PLC code and try to determine if it is controlling an industrial process. The engineer was also asked to attempt to determine the industrial sector that the device supports.

4.3.5 Analysis.

The panel's results selected 91 devices as Process Control and 63 devices as Indeterminate. This segregates the population of identified devices with 38% of Internet-facing ICS devices classified as Indeterminate.

Expert analysis shows that it is possible to identify the specific sector associated with a device using PLC code. The ICS engineer independently reviewing code selected 91 devices as Process Control, with a 100% match for each Process Control device selected by the panel. The ICS engineer also classified the sector or industrial category of 65 of the 91 Process Control devices used. Figure 4.16 shows the results of categorizing devices by sector. The results indicate that this method distinguishes ICS devices not only by function, but by industrial sector, including critical infrastructure. CI sectors represented include Water and Wastewater, Energy, Food and Agriculture, Transportation, and Commercial Facilities.

4.4 Conclusion

Statistically significant differences are calculated when comparing individual runs, however the range of values obtained from Treatment runs are within the observed values obtained measuring Baseline runs. This demonstrates that making one CIP request per

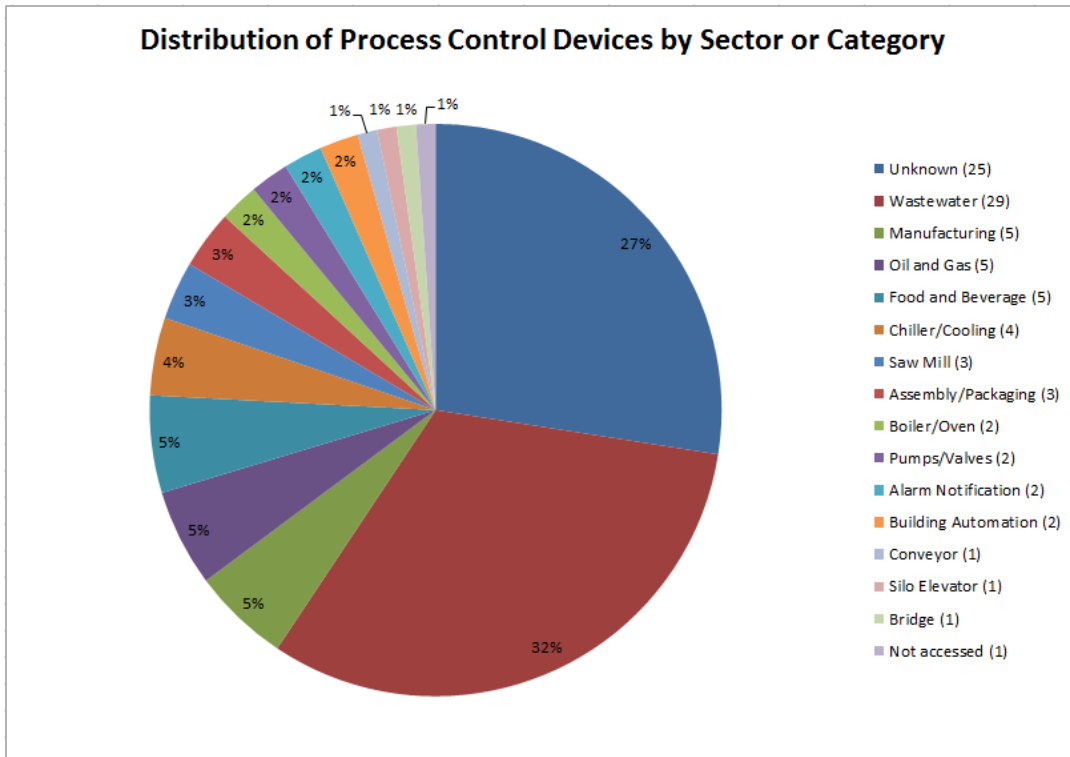


Figure 4.16: Sector/Industrial Category Distribution of Process Control Devices.

measurement cycle will not negatively impact task execution times on the PLC CPUs tested in this research.

V. Conclusions

5.1 Conclusions

This overall findings of this research shows that PLC code collected from an Internet-facing ICS device does not significantly impact PLC task execution times, and the PLC code returned from the device is useful in distinguishing the device as to its function in a process control system. The initial detection rate of Process Control devices using a panel of ICS researchers yielded a 54% selection of Process Control PLCs.

Subsequent analysis by an industry expert categorized the Process Control devices by the sector or industrial category the device supports. This is an important and unexpected capability provided by this methodology.

This research is exploratory in nature and the methods used for matching PLC code with process control terms are very elementary. This indicates that this methodology can be readily adapted to identify Internet-facing ICS systems.

5.2 Impact

This research demonstrates a significant impact to answering the question “So what?” when trying to understand the risk associated with Internet-facing ICS devices. The classification by panel reduced the pool of ICS devices by 46%. Further review by industry experts classified the devices by the industrial sector supported.

This research demonstrates the ability to identify Internet-facing ICS devices controlling physical industrial processes, and then classify those devices by the industrial sector they support. Current research has focused on the quantity of Internet-facing ICS devices. This research accomplishes its goal of distinguishing ICS devices based on PLC code, and further demonstrates a method to distinguish Internet-facing ICS devices by

sector. This methodology establishes the ability to gain a deeper understanding of the risk to critical infrastructure being incurred at the device level.

5.3 Recommendations

None of this research would be possible if PLC code could not be collected from an Internet-facing device. The most important act ICS administrators can take is to follow NIST recommendations and remove any Internet connections from their ICS networks. Following NIST recommendations is currently voluntary, however, this should not discourage ICS administrators from adhering to NIST recommendations.

Where Internet connections are determined to be a requirement by ICS administrators, the employment of network security appliances such as firewalls is essential. One device of the 164 interrogated returned a Connection Refused error message. It is likely that port 44818 is open on this device, however, a firewall is configured to block inbound traffic on that port. This stopped the PLC code collection script from successfully getting any CIP requests to the PLC.

Finally, the lack of authentication in the EtherNet/IP or CIP protocols allows the scripts used in this research to send CIP commands to any Internet-facing PLC and have the PLC execute those commands. While legacy equipment and procedures is at the core of ICS security issues, an ICS application layer protocol with security features like authentication built-in would dramatically increase the security of any ICS device which is able to be discovered on an Internet connection.

5.4 Future Work

5.4.1 Machine Learning and Process Control Term Matching.

The exploratory nature of this research leaves room for improvement. Visually inspecting code and qualitatively distinguishing Internet-facing ICS devices can be improved by exploring the use of machine learning techniques to distinguish these devices.

Future research can use machine learning to develop algorithms that use process control terms to match PLC code and select devices based on different metric such as assigning weight to the values of select process control terms or establishing a threshold value for a minimum number of matched terms.

5.4.2 Distinguishing Internet-facing ICS Devices by Sector.

This research demonstrated the ability to use PLC code to distinguish Internet-facing ICS devices by function and use industry experts to classify devices by sector. Further research can make use of the methodology in this research and build sector-specific process control term lists and explore the ability to identify other critical infrastructure sectors.

Industry experts use years of experience to make qualitative decisions supporting the classification of these devices. Using current analytic methods such as machine learning may provide automated methods of replicating those qualitative decisions.

Inferences cannot be made from this data alone. The ICS engineer called on 25 years experience to categorize device by sector. The engineer is also limited by geographical region and types of sectors supported. Further study is required to identify additional factors in sector distribution. The data shows a large proportion of wastewater systems, however the causal relationship cannot be determined. Allen-Bradley PLCs may be more common in wastewater systems, or confirmation bias on the part of the ICS engineer may have inflated the wastewater categorization count.

5.4.3 Determining Methodology Portability.

This researched focused solely on Allen-Bradley PLCs, however there are other manufacturers of ICS devices represented in the Shodan search engine. Future research can expand the scope of PLCs tested to determine if the results of this research's success rate translates for other manufacturer's devices.

5.5 Summary

Shodan demonstrated that despite popular belief, ICS devices are connected to the Internet. Researchers and security professionals began enumerating ICS devices found in Shodan and media reports started generating concern regarding possibilities of critical infrastructure attacks on these Internet-facing devices. This research distinguishes the function of Internet-facing ICS devices in regards to Internet-facing ICS devices. Using a novel method to collect PLC programming information to distinguish PLCs, this research is able to distinguish Internet-facing ICS devices by function and industrial sector.

Appendix A: 1: Process Control Terms

Recycle_WW
Sludge_Transfer_Pumps
Flow_Totalizer_Cont
Well_Data_S6CP
Routine:B_Level_Controller
Routine:B_Mixer_Alarms
Routine:B_Pump_Alarms
Control
FCV_1010_High_Flow_SP
FCV_1010_Low_Flow_SP
Program_Control
Bell_Sounds_Before_P15_Moves
E_STOP_1ST_CONV
HMI_JOG_MAIN_SLAT_PB
Map:Controller
P17_Conveyor_Move_Complete
P3_READY_TO_XFER_TO_P5_PE5
PAUSE_MAIN_SLAT
RESPONSE_FROM_SMTP_SERVER
SAFETY_RESET
STACKER_DOOR_E_STOP
START_SWITCH_P1
VFD1_OUTPUT_VOLTAGE
VFD2_INPUT_VOLTAGE
Routine:VFD1_STATUS
Routine:_2_MAIN_CNTRL
Routine:_40_STRT_STOP
Routine:_61_STACKER
FIFO_CONTROL
PV_NOM_SPEED_DOWN_PB
PV_NOM_SPEED_UP_PB
PV_RECIPE_MOVE_UP_OS
VFD1_OUTPUT_WATTS
Auxilliary_Shut_Down
Engine_Pumps_Shut_Down
Remote_Reset_RedLion
Routine:AFR_Control_PID
Routine:Engine_Start_Stop_Run
Start_Fuel_Timer1
Cond_VFD_2_Bypass

Condensing_SetPt_Calcs
Evap_16_Fan_Current_Relay
Evap_27_Fan_VFD_Run_Confirm
Floor_Heat_Control_Temp
Glycol_Tank_Temp
LT_Comp1_Current_Switch
LT_Comp1_Low_Pressure_Control
MT_Comp3_Crankcase_Heater
MOV1302_Open_Status
MOV301A_Closed_Status
P300_VFD_Frequency_Command
P300_VFD_Run_Command
SK205_ModbusTCP_Error_Code
Shipping_Oil_To_Sinclair_Active
Start_P301_Pump
CHILLER_SETPOINT
CONTAINER_CUT_SYSTEM_GALLONS
EMPTY_Water_Separator
Detergent_Tank_Prep_Timer
Flow_Meter_Control
MMI_SYRUP_TANK_START_FILL
Routine:FILLER
Routine:PID_FILLER
Cond_B1_VFD1_Bypass
Cond_B1_VFD_Run_Fans_1_3_5_7
AS_Lower_Final_Pallet_Stop
AS_Rbt_OK_to_Place
LD_Door_3_Open
LD_Door_Lock_2
OF_Enable_Pallet_Conveyor
Routine:H1_Conveyor
Routine:_SelectBufferLev
Alm_N1J_MotorStarterFlt
TPAControl_
CrPs_Flow_Low
CrPs_Flow_Scaled_AI
_PumpControl_Lag_Start_ONS
Routine:Motor_Operated_Valves
Routine:SBR_PumpControl
Z_DO_PMP0001_CommandStart
Z_ValueSlowScanFromScada
Routine:Booster_Processing
Routine:MTR_PUMPS_Valencia_Send

Routine:Valencia_AO_PID_Out
Program:HartProgram
CribPump1
ChlorineLeakAlm1
Override
Routine:CribPumpControl
CribAutoToggle
A400_3_Jog
C111_AtSpeed
C111_Active
V100_2_AtSpeed
V101_Start
V102_2_Stop
Routine:PowerUpMain
Routine:PanelView
PV_PB_V106_2_Rev
PV_PB_V106_2_Fwd
COMP01_START_MODE
EC06_FAN2_OVERLOAD
EC06_FAN2_VFD_SPEED
Freezer1980_AU2C_2D_ENABLE_SW
P1_COND5_PUMP_PUMP_OFF
Infeed_Speed_Ratio
Packer_Run
PackerJogging
ServoDriveEnabled
VFD19_BottleTurner:I
VFD19_BottleTurner:O
Routine:Glue_Control
Routine:TrayControl
Skid_B_Backwash_Step4
pv_Aeration_Level_Setpoint3_Max
pv_Turbidity_Alarm_SP
Routine:Anoxic_Tank_Mixing_System
Routine:Aeration_Tank_Denite_Pump
Routine:SV_Solenoid_Control
Program:Conveyor_Control
Program:Traffic_Control
Program:Physical_IO
Routine:Car_Positioning
Routine:Deliver_Loads
Routine:Laser_Positioning
Auto_Car_Forward

A_TurretDriveJogSpeed_WRT
B_TorqueRequired
HydraulicPumpKeepOnONS
RecipeTransferCancel
RollExtractorDownMessageBit
Routine:Cantilvered_Arm_Logic
Routine:Knives_Bed_Control
Center_Seal_Top_Pressure
FoldedFilm1_Brake_CV_Auto
FoldedFilm1_Brake_PID_OUT
Temp_PortSeal_when_TempOK
Routine:A01_PortSealSequence
DistanceinPRTuntoFinssesTun_HMI
Map:PF400_SprayBoothNorthwestRecircFan_VFD
Routine:Process_Message
Routine:VFD_Comms
ChilledWaterPump
CausticWastePump
Com_CDMA_Initiate
Com_Disable
Routine:RSP_PID_Routine
Enable_Full_Speed_Cmd
CC_COMMUNITOR_RUN
CC_GEN_RUN
CRWNCT_PMP1_RUN_DI
Flouride_Tank_Scale
Pumps_Start
Vent_Flouride
Routine:_010A_Tower_Control
Routine:_600B_Flouride_DT_Pump
Bridge_Opening
Ok_To_Open_Bridge
Routine:Watchdog
Bioxie_Tank_Filling_Latch
Map:WASTEWATER_PUMP_1
PLC_IO_Status_HMI_1
WASTEWATER_PUMP_1:0
WASTEWATER_PUMP_2:I
Routine:TRANSFER_SWITCH
Routine:FeedLine
Routine:_DefoamerPump
FCV2MSG
Program:FlowControlValve

PUMP10_STOP
OUTLET_VALVE_OPEN_COMMAND
Routine:WASTE_TRANSFER
Program:E_STOP_ZONE_1
Routine:Interlock_Override
Routine:MotorControl_Z1
Routine:LakeWater_Pump_Speed
Routine:Scale_Analog_Outputs
Routine:Lead_pump_select
Routine:Wet_well_alarms
Routine:Aborting
Routine:ANIONIC_POLYMER_CTRL
BACKWASH_AVIVOEN
Routine:_713_Robot_1_Send_Data
Routine:_711_Robot_1_Cycle_Start
A21_Regulator_Position_Value
Routine:SCADA_Mapping
Batch_Control_Load_Valve_Open_Request
Routine:Rail>Loading
Routine:Modbus_Comms
Routine:M17_Heavies_Picking_Conveyor
Routine:Start_Up
Routine:System_Pause
Routine:B1_LOAD_XFR_ROUTINE
Routine:Filter_102_Auto_BW_mode_startup
Routine:F101_Backwash_pump
AutoSetSlowDown
Program:PIDExecution
Routine:F1_Backwash_Abort
Backwash1_Control
Routine:T2_BACKFLUSH
Routine:A15_Load_Control
M203_OK_TO_RUN
OverHead_One_Shot

Appendix B: 2: List of Boxplots

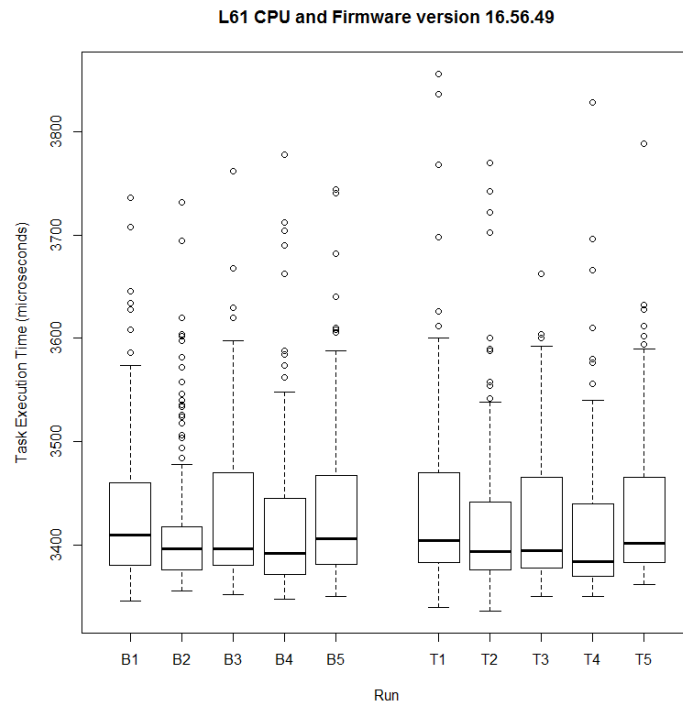


Figure B.1: Boxplot for L61 v16.56.47.

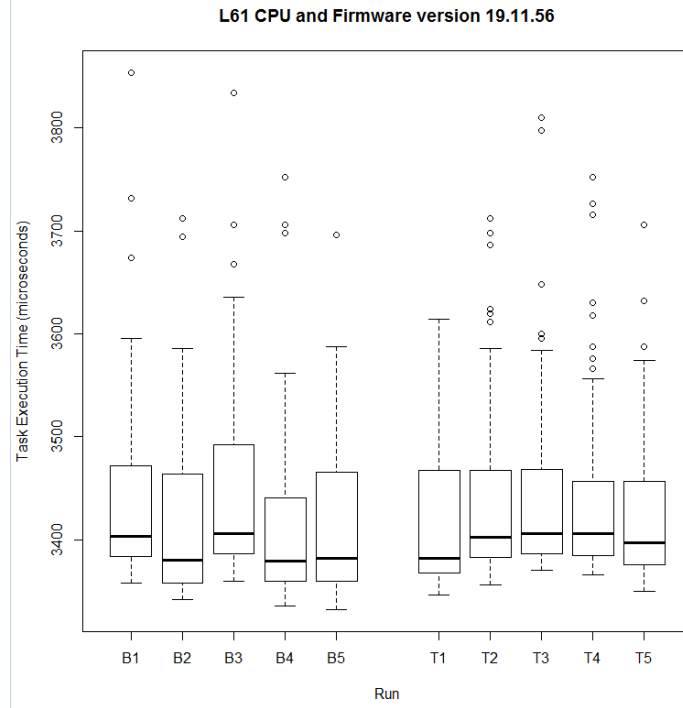


Figure B.2: Boxplot for L61 v19.11.56.

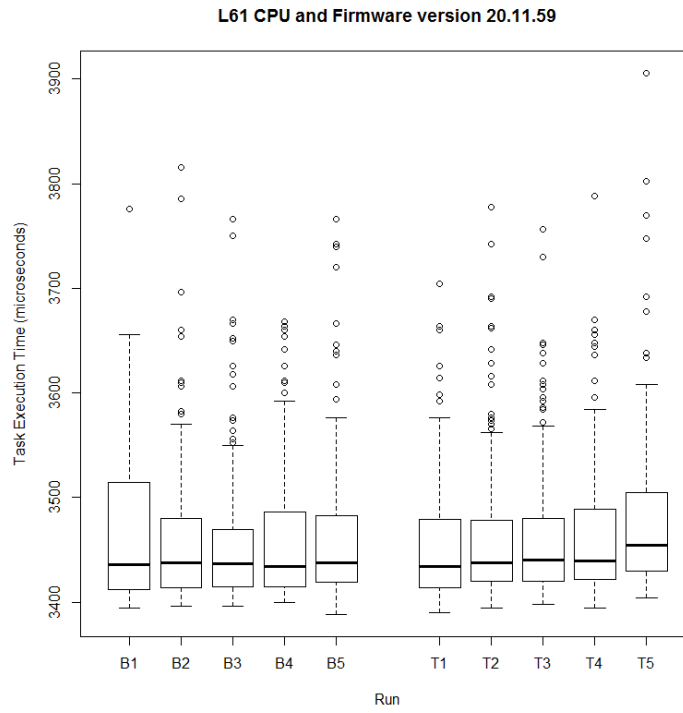


Figure B.3: Boxplot for L61 v20.11.59.

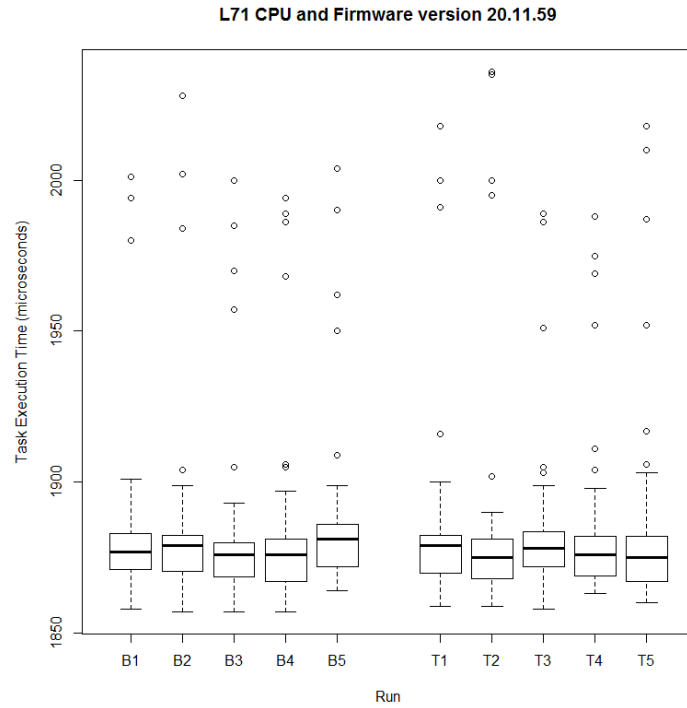


Figure B.4: Boxplot for L71 v20.11.59.

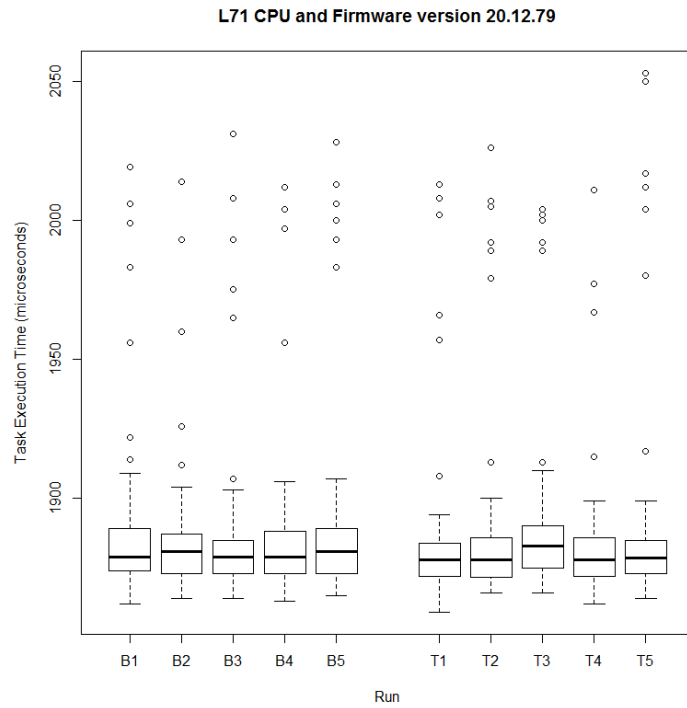


Figure B.5: Boxplot for L71 v20.12.79.

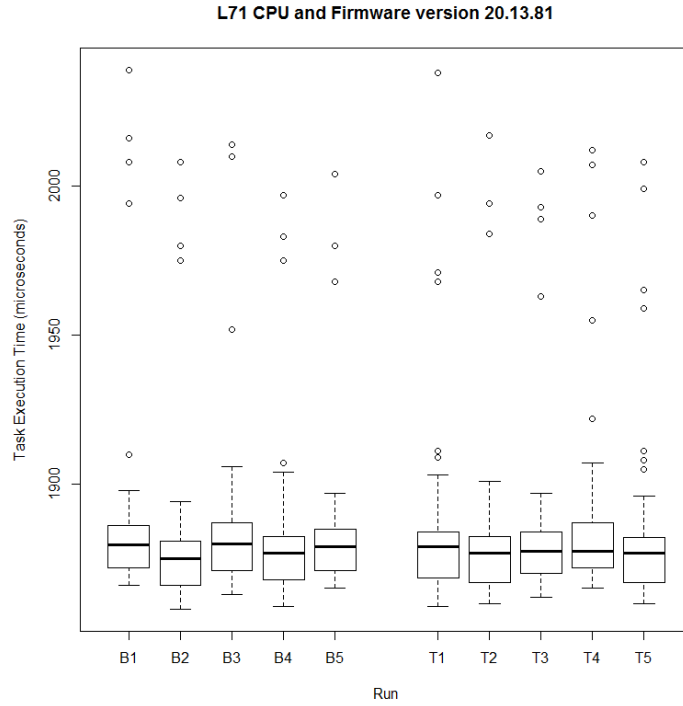


Figure B.6: Boxplot for L71 v20.13.81.

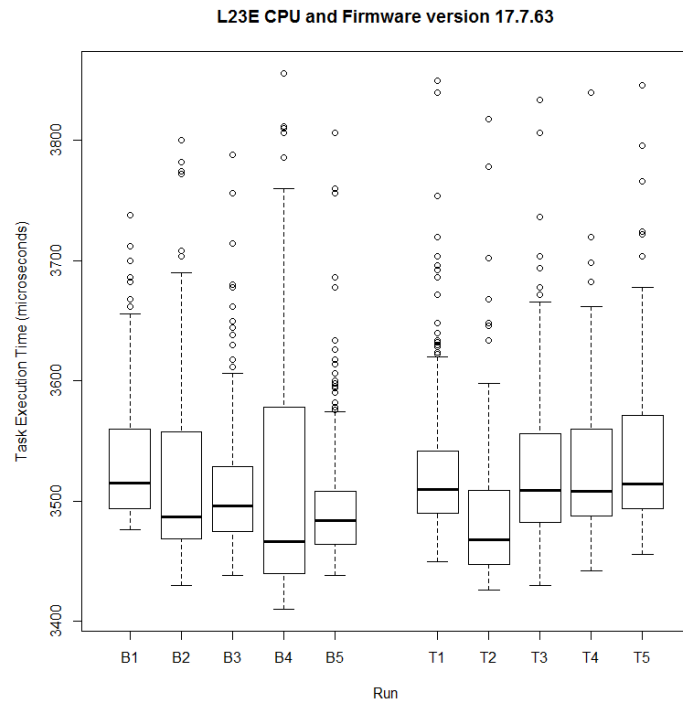


Figure B.7: Boxplot for L23E v17.7.63.

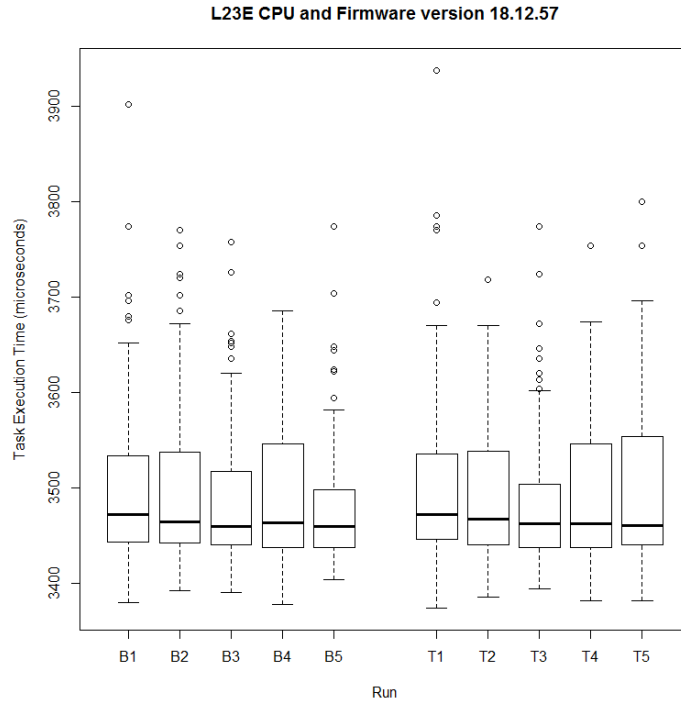


Figure B.8: Boxplot for L23E v18.12.57.

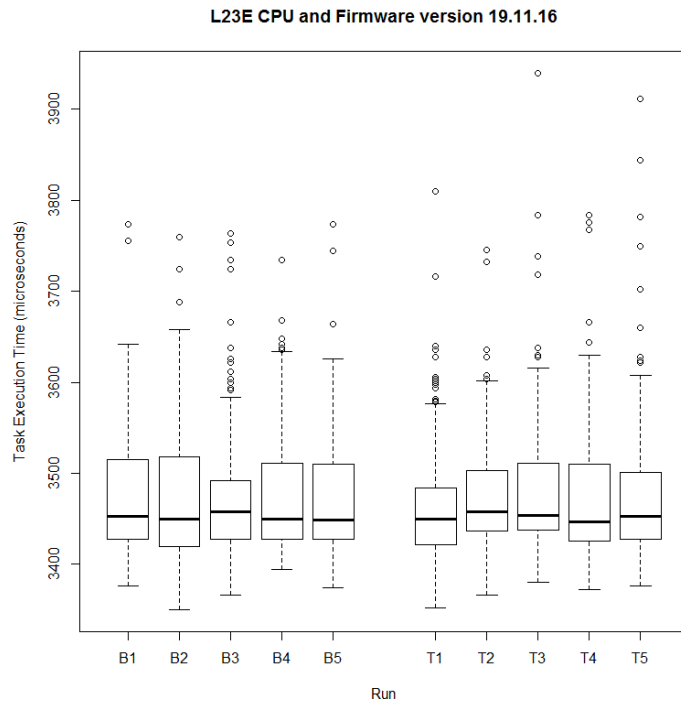


Figure B.9: Boxplot for L23E v19.11.16.

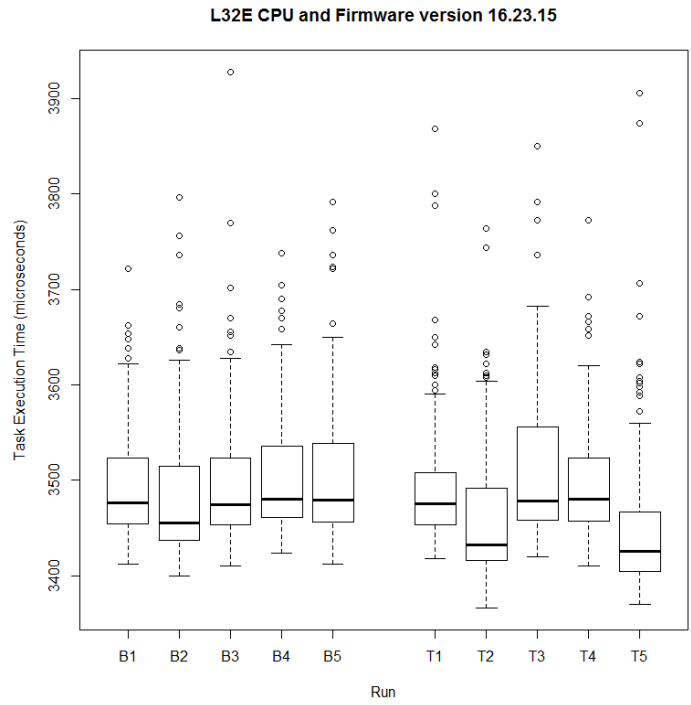


Figure B.10: Boxplot for L32E v16.23.15.

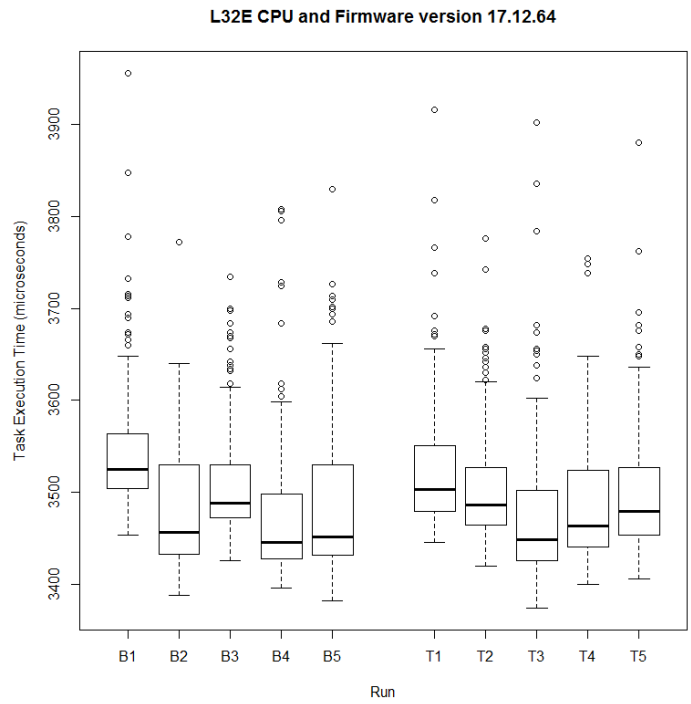


Figure B.11: Boxplot for L32E v17.12.64.

L32E CPU and Firmware version 20.13.81

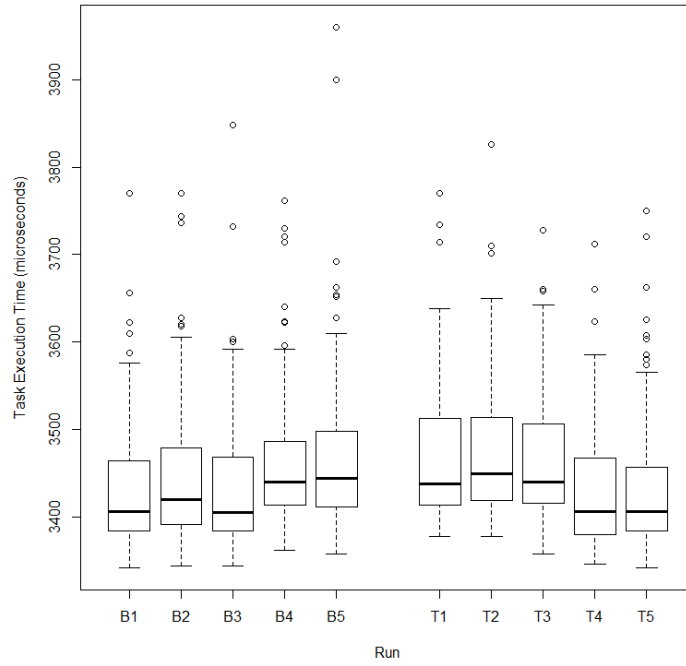


Figure B.12: Boxplot for L32E v20.13.81.

Bibliography

- [1] *The Role of Authenticated Communications for Electric Power Distribution*. Technical report, Pacific Northwest National Laboratory. URL http://www.science.upm.ro/~traian/web_curs/Scada/docum/SCADA_archit.pdf.
- [2] *EtherNet/IP - CIP on Ethernet Technology*. Technical report, ODVA, 2008. URL http://www.odva.org/Portals/0/Library/Publications_Numbered/UB00138R3_CIP_Adv_Tech_Series_EtherNetIP.pdf.
- [3] *The Shmoon Attacks*. Technical report, Symantec, 2012. URL <http://www.symantec.com/connect/blogs/shmoon-attacks>.
- [4] *SHODAN - Computer Search Engine*. Technical report, SHODAN, 2014. URL <http://http://www.shodanhq.com/help/filters>.
- [5] Bodenheim, Roland C. *Impact of the Shodan Computer Search Engine on Internet-facing Industrial Control System Devices*. Master's thesis, Air Force Institute of Technology, 2014.
- [6] Brooks, Paul. *EtherNet/IP: Industrial Protocol White Paper*. Technical report, Rockwell Automation, 2001. URL http://literature.rockwellautomation.com/idc/groups/literature/documents/wp/enet-wp001_-en-p.pdf.
- [7] Clements, Sam. "Is Shodan Really The World's Most Dangerous Search Engine?", 2013. URL http://www.vice.com/en_uk/read/shodan-exposes-the-dark-side-of-the-net.
- [8] Department of Homeland Security. "A Guide to Critical Infrastructure and Key Resources Protection at the State, Regional, Local, Tribal and Territorial Level", 2008.
- [9] Department of Homeland Security. "National Infrastructure Protection Plan", 2009.
- [10] Dunlap, Stephen J. *Timing-based Side Channel Analysis for Anomaly Detection in the Industrial Control System Environment*. Master's thesis, Air Force Institute of Technology, 2013.
- [11] Falliere, Nicolas. *Exploring Stuxnets PLC Infection Process*. Technical report, Symantec, 2010. URL <http://www.symantec.com/connect/blogs/exploring-stuxnet-s-plc-infection-process>.
- [12] Fidelis Cybersecurity Solutions. *Fidelis Threat Advisory #1012 Gathering in the Middle East, Operation STTEAM*. Technical report, General Dynamics, 2014. URL http://www.fidelissecurity.com/webfm_send/377.

- [13] Goldman, David. “Shodan: The scariest search engine on the Internet”, 2013. URL <http://money.cnn.com/2013/04/08/technology/security/shodan/>.
- [14] Higgins, Kelly Jackson. “Utilities Facing Brute-Force Attack Threat”, 2012. URL <http://www.darkreading.com/attacks-breaches/utilities-facing-brute-force-attack-thre/232600345>.
- [15] Higgins, Kelly Jackson. *‘Project SHINE’ Illuminates Sad State Of SCADA/ICS Security On The Net*. Technical report, Dark Reading, 2013. URL <http://www.darkreading.com/vulnerability/project-shine-illuminates-sad-state-of-s/240162739>.
- [16] Higgins, Kelly Jackson. “Oil & Gas Firms Targeted In Web Server Hacks”, 2014. URL <http://www.darkreading.com/attacks-breaches/oil-gas-firms-targeted-in-web-server-ha/240166515>.
- [17] Hill, Kashmir. “The Terrifying Search Engine That Finds Internet-Connected Cameras, Traffic Lights, Medical Devices, Baby Monitors and Power Plants”, 2013. URL <http://www.forbes.com/sites/kashmirhill/2013/09/04/shodan-terrifying-search-engine/>.
- [18] Kalapatapu, Rao. “SCADA Protocols and Communication Trends”. *ISA2004*, 2004.
- [19] Leverett, Eireann P. *Quantitatively Assessing and Visualizing Industrial System Attack Surfaces*. Master’s thesis, University of Cambridge, 2011.
- [20] National Institute of Standards and Technology. “Guide to Industrial Control Systems (ICS) Security Special Publication 800-82”, 2011.
- [21] Peterson, Dale G. “S4 Video: Denial of Surface ICS on the Internet”, 2012. URL <https://www.digitalbond.com/blog/2012/02/09/s4-video-denial-of-surface-ics-on-the-internet/>.
- [22] Radvanovsky, Bob. “Shining a Light on a Big Problem”, 2013. URL <http://www.tofinosecurity.com/blog/project-shine-1000000-internet-connected-scada-and-ics-systems-and-counting>.
- [23] Rockwell Automation. “Comparing performance of L7x vs L6x using Logix5000 Task Monitor tool”, 2011. URL <https://www.rockwellautomation.com/resources/downloads/rockwellautomation/pdf/solutions/integrated-architecture/LogixTaskMonitorInstruction.pdf>.
- [24] Rockwell Automation. “Optimize Productivity with RSLogix 5000 Design and Configuration Software”, 2011. URL http://literature.rockwellautomation.com/idc/groups/literature/documents/pp/9324-pp001_-en-p.pdf.
- [25] Rockwell Automation. “ControlLogix System. Rockwell Automation Publication 1756-SG001R-EN-P”, 2012. URL http://literature.rockwellautomation.com/idc/groups/literature/documents/sg/1756-sg001_-en-p.pdf.

- [26] Rockwell Automation. “Logix5000 Controllers IEC 61131-3 Compliance”, 2012.
- [27] Rockwell Automation. “CompactLogix System. Rockwell Automation Publication 1769-SG001P-EN-P”, 2014. URL http://literature.rockwellautomation.com/idc/groups/literature/documents/sg/1769-sg001_-en-p.pdf.
- [28] Schearer, Michael. “SHODAN for Penetration Testers”, 2010. URL <https://www.defcon.org/images/defcon-18/dc-18-presentations/Schearer/DEFCON-18-Schearer-SHODAN.pdf>.
- [29] Schuett, Carl D. *Programmable Logic Controller Modification Attacks for use in Detection Analysis*. Master’s thesis, Air Force Institute of Technology, 2014.
- [30] Schwartz, Moses D., John Mulder, and William D. Trent, Jason & Atkins. *Control System Devices: Architectures and Supply Channels Overview*. Technical report, Sandia National Laboratories, 2010.
- [31] Voss, Katherine. *The CIP Advantage: Proven and Future-Proof, CIP Provides a Unified Architecture for Delivering Standards-Based Open Networking Technologies Throughout the Enterprise*. Technical report, ODVA, 2008. URL http://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00072R0_CIP_Advantage_Technical_Paper.pdf.
- [32] Weiss, Gus W. “The Farewell Dossier”, 2007. URL <https://www.cia.gov/library/center-for-the-study-of-intelligence/csi-publications/csi-studies/studies/96unclass/farewell.htm>.
- [33] Wilhoit, Kyle. “The SCADA That Didnt Cry Wolf. Whos Really Attacking Your ICS Equipment? (Part 2)”, 2013. URL <http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-the-scada-that-didnt-cry-wolf.pdf>.
- [34] Wilhoit, Kyle. “Whos Really Attacking Your ICS Equipment?”, 2013. URL <http://www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/white-papers/wp-whos-really-attacking-your-ics-equipment.pdf>.
- [35] Zetter, Kim. “10K Reasons to Worry About Critical Infrastructure”, 2012. URL <http://www.wired.com/threatlevel/2012/01/10000-control-systems-online/>.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

The public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. **PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.**

1. REPORT DATE (DD-MM-YYYY) 19-06-2014		2. REPORT TYPE Master's Thesis		3. DATES COVERED (From — To) Jan 2013-Jun 2014	
4. TITLE AND SUBTITLE Distinguishing Internet-facing ICS devices using PLC programming information				5a. CONTRACT NUMBER	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
				5d. PROJECT NUMBER	
				5e. TASK NUMBER	
6. AUTHOR(S) Williams, Paul M., Major, USA				5f. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765			8. PERFORMING ORGANIZATION REPORT NUMBER AFIT-ENG-T-14-J-41		
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) Intentionally Left Blank			10. SPONSOR/MONITOR'S ACRONYM(S)		
			11. SPONSOR/MONITOR'S REPORT NUMBER(S)		
12. DISTRIBUTION / AVAILABILITY STATEMENT DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
13. SUPPLEMENTARY NOTES This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
14. ABSTRACT The Shodan search engine reveals Industrial Control System (ICS) devices around the globe are directly connected to the Internet. After Shodan's inception in 2009, multiple news reports have focused on the increased threat to infrastructure posed by Shodan. While no attacks to date have been directly attributed to Shodan searches, its existence provides an anonymous reconnaissance platform that facilitates ICS targeting for those actors with both a desire and capability to carry out attacks. Recent research has demonstrated that simple search queries return thousands of ICS devices indexed by Shodan, and the number of newly indexed ICS devices is growing. This research discusses the method used to distinguish the Internet-facing ICS devices indexed by the Shodan search engine. PLC code is obtained by sending specifically crafted CIP request messages to the devices, capitalizing on the fact that authentication is not built in to the CIP application layer protocol. This data allows categorization of Internet-facing devices by comparing PLC code attributes. The results of this research show PLC code can be collected from Internet-facing ICS devices with no significant impact to task execution times. Also, this research demonstrates a method to distinguish Internet-facing ICS devices by function and by Critical Infrastructure sector. This capability develops an understanding of the function and purpose of ICS devices that are being connected to the Internet.					
15. SUBJECT TERMS SCADA ICS SHODAN Critical Infrastructure Allen-Bradley Distinguishing Internet-facing PLC					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Maj Jonathan W. Butts (ENG)
U	U	U	UU	114	19b. TELEPHONE NUMBER (include area code) (937) 255-3636 x4332 Jonathan.Butts@afit.edu