

3-21-2013

# Emulation of Industrial Control Field Device Protocols

Robert M. Jaromin

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

Part of the [Controls and Control Theory Commons](#), and the [Digital Communications and Networking Commons](#)

---

## Recommended Citation

Jaromin, Robert M., "Emulation of Industrial Control Field Device Protocols" (2013). *Theses and Dissertations*. 877.  
<https://scholar.afit.edu/etd/877>

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](mailto:richard.mansfield@afit.edu).



**EMULATION OF INDUSTRIAL CONTROL FIELD DEVICE PROTOCOLS**

THESIS

Robert M. Jaromin, Captain, USAF

AFIT-ENG-13-M-27

**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-13-M-27

EMULATION OF INDUSTRIAL CONTROL FIELD DEVICE PROTOCOLS

THESIS

Presented to the Faculty  
Department of Computer Science and 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 Computer Science

Robert M. Jaromin, B.S.E.E.

Captain, USAF

March 2013

**DISTRIBUTION STATEMENT A.**  
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

EMULATION OF INDUSTRIAL CONTROL FIELD DEVICE PROTOCOLS

Robert M. Jaromin, B.S.E.E.  
Captain, USAF

Approved:



Barry E. Mullins, PhD (Chairman)

4 Mar 13

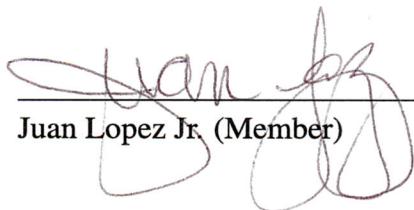
Date



Maj Jonathan W. Butts, PhD (Member)

4 Mar 13

Date



Juan Lopez Jr. (Member)

4 MAR 2013

Date

## Abstract

It has been shown that thousands of industrial control devices are exposed to the Internet, however, the extent and nature of attacks on such devices remains unknown. The first step to understanding security problems that face modern supervisory control and data acquisition (SCADA) and industrial controls networks is to understand the various attacks launched on Internet-connected field devices. This thesis describes the design and implementation of an industrial control emulator on a Gumstix single-board computer as a solution. This emulator acts as a decoy field device, or *honeypot*, intended to be probed and attacked via an Internet connection. Evaluation techniques are developed to assess the accuracy of the emulation implemented on the Gumstix and are compared against the implementation on a standard PC and the emulation target, a Koyo DirectLogic 405 programmable logic controller (PLC).

The results show that the both the Gumstix and PC emulator platforms are over 97% accurate for Nmap OS Fingerprint Scans, over 99% for standard web and industrial protocol queries, and 100% accurate for Metasploit exploit execution. The logging capabilities of both platforms are excellent, with 97.21% of all packets being logged by the Gumstix and 99.99% of all packets being logged by the PC when the results of all scenarios are combined. Though the timing-level accuracies for the PC platform are significantly faster than the target PLC, the results of the Gumstix platform significantly slower as defined by the 99% confidence intervals for the mean.

Based on these results, extensive knowledge of the specific implementations of the protocols or timing profiles of the target PLC are required to identify and fingerprint the Gumstix device as a honeypot. Furthermore, there are very few actively maintained SCADA honeypot systems available, and the number of ICS honeypots accessible from the Internet is also likely very few. As a result it is speculated that SCADA honeypots

are not actively being looked for by attackers. Based on this, the results suggest that a honeypot implemented on a Gumstix emulator is suitable for applications in SCADA attack-landscape research. Further research should be conducted to increase the timing performance of the emulator before being deployed in applications that are sensitive to emulation timing discrepancies, such as SCADA laboratory research. In these types of applications, the PC platform should be used.

*To my wife and our daughter. My hope, happiness, and future.*



## **Acknowledgments**

I would like to express my sincere gratitude to my advisor, Dr. Barry Mullins for his guidance and support. The encouragement, focus, and flexibility he provided enabled me to accomplish more than I thought possible throughout my graduate studies. I would also like to thank my committee, Maj Jonathan Butts and Mr. Juan Lopez for sharing their time, enthusiasm, and vision, and for providing the opportunities to grow professionally as well as academically.

I would also like to thank my wife for her endless encouragement through all the late nights, early mornings, and lonely weekends. Knowing my family was at home waiting for me made every minute in the lab bearable.

Robert M. Jaromin

## Table of Contents

	Page
Abstract . . . . .	iv
Dedication . . . . .	vi
Acknowledgments . . . . .	vii
Table of Contents . . . . .	viii
List of Figures . . . . .	xiii
List of Tables . . . . .	xv
I. Introduction . . . . .	1
1.1 Motivation . . . . .	1
1.2 Research Goals . . . . .	2
1.3 Thesis Layout . . . . .	3
II. Literature Review . . . . .	4
2.1 Introduction . . . . .	4
2.2 SCADA . . . . .	4
2.2.1 The Programmable Logic Controller . . . . .	8
2.2.2 ICS and SCADA Protocols . . . . .	9
2.2.2.1 Modbus Protocol . . . . .	9
2.2.2.2 DirectNet Protocol . . . . .	11
2.2.2.3 K-Sequence Protocol . . . . .	12
2.2.2.4 Host Automation Products (HAP) Protocol . . . . .	12
2.2.3 SCADA Communication Networks . . . . .	13
2.2.4 SCADA on the Internet . . . . .	14
2.2.5 SCADA Threats . . . . .	18
2.2.6 Cyber Attacks: The Modern State of the SCADA Landscape . . . . .	19
2.2.7 Researching SCADA Threats . . . . .	21
2.3 Emulators . . . . .	22
2.3.1 Emulated Honeypot Services . . . . .	24
2.3.2 Industrial Control Emulators . . . . .	24
2.3.3 Uses of ICS field device emulators . . . . .	25
2.3.3.1 SCADA Landscape Research Using Honeypots . . . . .	25

	Page
2.4 The Gumstix Platform . . . . .	28
2.5 Related Research . . . . .	30
2.6 Summary . . . . .	31
 III. Device Description . . . . .	 33
3.1 Design Considerations . . . . .	33
3.1.1 Accurate and Authentic . . . . .	33
3.1.2 Robust Operation . . . . .	35
3.1.3 Cost Effective . . . . .	36
3.1.4 Rapidly Configurable . . . . .	36
3.1.5 Scalability . . . . .	37
3.1.6 Record Activity . . . . .	37
3.2 Implementation of the PLC Emulator . . . . .	38
3.2.1 Custom Design Overview . . . . .	38
3.2.2 Process Layout and Implementation . . . . .	39
3.2.3 Emulator Configuration and Process Control Scripts . . . . .	40
3.2.3.1 iptables_rules.sh . . . . .	40
3.2.3.2 configure_emulator.sh . . . . .	40
3.2.3.3 start_emulator.sh . . . . .	42
3.2.3.4 start_emulator_no make.sh . . . . .	43
3.2.3.5 stop_emulator.sh . . . . .	43
3.2.4 Custom User-space Processes . . . . .	43
3.2.4.1 infilter.o . . . . .	43
3.2.4.2 outfilter.o . . . . .	45
3.2.4.3 ecom_svr.o . . . . .	46
3.2.4.4 Dependencies of ecom_svr.o (HAP protocol emulator) . . . . .	54
3.2.5 logging.o . . . . .	56
3.2.6 webserver.py . . . . .	58
3.2.7 modbus.py . . . . .	59
3.3 Incorporation of Design Considerations . . . . .	60
3.3.1 Accuracy . . . . .	60
3.3.2 Robustness . . . . .	61
3.3.3 Cost Considerations . . . . .	62
3.3.4 Configurability . . . . .	63
3.3.5 Scalability . . . . .	63
3.3.6 Logging Ability . . . . .	63
3.4 Summary . . . . .	64

	Page
IV. Experimental Methodology . . . . .	65
4.1 Goals . . . . .	65
4.2 Approach . . . . .	65
4.3 System Boundaries . . . . .	67
4.4 Services and Outcomes . . . . .	67
4.5 Parameters and Factors . . . . .	68
4.5.1 Workload Parameters and Factors . . . . .	69
4.5.1.1 Request Type . . . . .	69
4.5.1.2 Request Frequency . . . . .	72
4.5.2 System Parameters and Factors . . . . .	74
4.5.2.1 PLC Device . . . . .	74
4.6 Performance Metrics . . . . .	75
4.6.1 Packet Bytes . . . . .	75
4.6.2 Nmap OS Fingerprinting Fields . . . . .	76
4.6.3 Successful Unlock . . . . .	76
4.6.4 Response Time . . . . .	76
4.6.5 Logging Ability . . . . .	77
4.7 Experimental Design . . . . .	78
4.8 Evaluation Technique . . . . .	79
4.8.1 Accuracy . . . . .	79
4.8.1.1 Standard Web Workload Packet-Level Accuracy . . . . .	79
4.8.1.2 Standard HAP Workload Packet-Level Accuracy . . . . .	81
4.8.1.3 Nmap OS Scanning-Level Accuracy . . . . .	82
4.8.1.4 Metasploit Attack-Level Accuracy . . . . .	83
4.8.1.5 Timing of Standard Requests . . . . .	83
4.8.1.6 Timing of the Non-standard Requests . . . . .	84
4.8.1.7 Logging . . . . .	85
4.9 Experiment setup . . . . .	85
4.10 Validation of Custom Tools . . . . .	86
4.11 Summary . . . . .	87
V. Results and Analysis . . . . .	88
5.1 Validation of Custom Tools . . . . .	88
5.2 Packet Level Accuracy . . . . .	89
5.2.1 Gumstix Packet-Level Accuracy Results . . . . .	89
5.2.2 Logging of Standard Workloads on the Gumstix Platform . . . . .	92
5.2.3 PC Packet Level Accuracy Results . . . . .	93
5.2.4 Logging of Standard Workloads on the PC Platform . . . . .	94
5.3 Scanning Level Accuracy . . . . .	95
5.3.1 Gumstix Scanning Level Accuracy Results . . . . .	95

	Page
5.3.2 Logging Results of Non-Standard Workloads on the Gumstix Platform . . . . .	98
5.3.3 PC Scanning-Level Accuracy Results . . . . .	99
5.4 Attack Level Accuracy . . . . .	103
5.4.1 Gumstix Attack-Level Accuracy Results . . . . .	103
5.4.2 PC Attack Level Accuracy Results . . . . .	104
5.5 Timing Level Accuracy . . . . .	104
5.5.1 Gumstix Timing Level Accuracy Results for Standard Workloads . . . . .	105
5.5.2 PC Timing Accuracy Results for Standard Workloads . . . . .	111
5.5.3 Timing Accuracy Results for Web Non-Standard Workloads . . . . .	114
5.5.4 Timing Accuracy Results for HAP Non-Standard Workloads . . . . .	117
5.6 Summary . . . . .	119
VI. Conclusions . . . . .	121
6.1 Introduction . . . . .	121
6.2 Research Conclusions . . . . .	121
6.2.1 Accuracy of Standard Queries at the Packet Level . . . . .	121
6.2.2 Accuracy at the Scanning and Attack Levels . . . . .	121
6.2.3 Accuracy at the Timing Levels . . . . .	122
6.2.4 Logging Performance . . . . .	123
6.2.5 Performance Effects of Request Frequency . . . . .	123
6.2.6 Impact of the Gumstix Platform . . . . .	124
6.2.7 Final Assessment . . . . .	124
6.3 Significance of Research . . . . .	125
6.4 Future Work . . . . .	126
6.4.1 Enhance Timing Accuracy of Gumstix Emulator . . . . .	126
6.4.2 Timing Improvements through Efficient Logging . . . . .	126
6.4.3 Enhance the Nmap Scan Handler . . . . .	127
6.4.4 Enhanced Configurability . . . . .	128
6.4.5 Automated Capture of Ladder Logic for Arbitrary Programs . . . . .	130
6.4.6 Evaluation of Other SCADA Honey pots . . . . .	131
6.4.7 Implementation as Evidence Collection Tool . . . . .	131
6.4.8 Other Areas . . . . .	132
6.4.9 Hardware Emulation . . . . .	132
6.4.10 Implementation of a Large-Scale Network Using the Emulator . . . . .	133
Appendix A: Additional Applications for Industrial Control Emulators . . . . .	141
Appendix B: Features of Related Research . . . . .	145

	Page
Appendix C: Iptables Rules in <i>iptables_rules.sh</i> . . . . .	147
Appendix D: Web and HAP Functions Implemented . . . . .	150
Appendix E: Non-deterministic Fields . . . . .	152
Appendix F: Partial Factorial Design . . . . .	156
Appendix G: Custom Tool Description and Validation . . . . .	158
Appendix H: Additional Results for HAP standard Workload . . . . .	162
Appendix I: Configuring the Gumstix Emulator . . . . .	165

## List of Figures

Figure	Page
2.1 A Typical SCADA System . . . . .	6
2.2 Gumstix Module to Scale [Gum112] . . . . .	29
3.1 Routing of packets in the PLC emulator system. . . . .	44
3.2 Example packet flow for Nmap OS Probes . . . . .	46
3.3 Example packet flow through <i>outfilter.o</i> . . . . .	47
3.4 Example packet flow through <i>ecom_svr.o</i> . . . . .	48
3.5 NetEdit3 interface communicating using the HAP industrial protocol . . . . .	50
3.6 DirectSOFT 5 programmer interfacing with the HAP protocol emulator . . . . .	52
3.7 Status of emulator ladder logic a few moments later . . . . .	53
3.8 Metasploit screenshot . . . . .	54
3.9 Excerpt from <i>ccmdictionary.txt</i> . . . . .	56
3.10 Example packet flow through <i>logging.o</i> . . . . .	57
3.11 Example packet flow through <i>webserver.py</i> . . . . .	58
4.1 PLC control system . . . . .	68
4.2 Example OS Fingerprint scan results . . . . .	77
4.3 Method of byte-by-byte comparison of target PLC to emulator . . . . .	80
4.4 Experimental Setup. . . . .	86
5.1 Partial response packet number 4 for the Gumstix emulator and PLC . . . . .	91
5.2 Incorrect HTML byte in emulator . . . . .	91
5.3 Comparison of results between Gumstix and the target PLC Nmap scans . . . . .	96
5.4 Comparison of percentages of packets logged by the Gumstix emulator . . . . .	100
5.5 Comparison of percentages of packets logged by the PC emulator . . . . .	101
5.6 Comparison between Gumstix and PC logging functionality . . . . .	102

Figure	Page
5.7 Response times for the Gumstix emulator . . . . .	107
5.8 TCP connection with anomalous response time 8 SD above the mean . . . . .	108
5.9 Response Times versus observation number for web Standard workload . . . . .	109
5.10 Results comparing all three request frequency levels of the Gumstix . . . . .	110
5.11 Non-continuous jumps in response time for the Gumstix . . . . .	111
5.12 Mean response times for all devices during standard web workload . . . . .	115
5.13 Mean response times for the Nmap scan for all devices . . . . .	116
5.14 Results for Metasploit scenarios for all three devices. . . . .	118
5.15 Mean response times for the PC for the HAP non-standard workload . . . . .	119
5.16 Mean response times for the Gumstix for the HAP non-standard workload . . . . .	120
C.1 Example of homepage served by <i>webserver.py</i> . . . . .	153
G.1 Validation of workload generators. . . . .	160



## List of Tables

Table	Page
3.1 iptables rules contained in <i>iptables_rules.sh</i> . . . . .	41
3.2 Temporary settings configured in <i>configure_emulator.sh</i> . . . . .	42
4.1 Two services broken into four query types. . . . .	66
4.2 Levels of accuracy mapped to emulator services. . . . .	67
4.3 Selected nominal request frequency levels . . . . .	73
4.4 Summary of Factors and Levels. . . . .	74
4.5 Performance metrics and levels of accuracy . . . . .	78
4.6 Scenarios examined during the Web Standard Workload. . . . .	79
4.7 Scenarios examined during the HAP Standard Workload. . . . .	81
4.8 Scenarios examined during the Web Non-standard Workload. . . . .	82
4.9 Scenarios examined during the HAP Non-standard Workload. . . . .	84
5.1 Packet level accuracy and logging results for the Gumstix . . . . .	90
5.2 Packet level accuracy and logging results for the PC emulator . . . . .	94
5.3 Scanning-level accuracy and logging results for the Gumstix emulator . . . . .	97
5.4 Scanning-level accuracy and logging results for the PC emulator . . . . .	100
5.5 Attack results for the target PLC. . . . .	103
5.6 Attack-level accuracy and logging results for the Gumstix emulator. . . . .	103
5.7 Attack-level accuracy and logging results for the PC emulator. . . . .	104
5.8 Timing accuracy results for Gumstix emulator . . . . .	105
5.9 Timing Accuracy results for PC emulator . . . . .	112
5.10 Results of ANOVA test conducted on web standard workload . . . . .	113
5.11 Comparison between Gumstix and PC emulator for timing accuracy results. . .	114
5.12 Results of ANOVA test conducted on HAP standard workload . . . . .	114

Table	Page
5.13 Timing Accuracy results for Gumstix and PC emulators for the Nmap Workload	116
5.14 Timing accuracy results for the Metasploit workload . . . . .	117
D.1 Ethernet Header Fields . . . . .	152
D.2 IP Header Fields . . . . .	152
D.3 TCP Header Fields . . . . .	152
D.4 HTTP Data Fields . . . . .	153
E.5 Ethernet Header Fields . . . . .	154
E.6 IP Header Fields . . . . .	154
E.7 UDP Header Fields . . . . .	154
E.8 UDP Header Fields . . . . .	155

# EMULATION OF INDUSTRIAL CONTROL FIELD DEVICE PROTOCOLS

## I. Introduction

### 1.1 Motivation

In 2011, it was shown that a substantial number of Industrial Control Systems (ICS) devices were actively connected to the Internet despite claims to the contrary [Lev11]. The research provides strong evidence that ICS devices are exposed to the Internet, however, it is still largely unknown to what extent the identified industrial control devices and others are being attacked. The life expectancy, cost, and remote physical location of distributed control systems (DCS) and supervisory control and data acquisition (SCADA) devices make asset owners reluctant to replace them for the sole purpose of applying security [HoL10]. With the high number of attack opportunities that Internet exposure provides, and lack mandatory compliance with current security policies, there is little hope that these devices are safe. However, data is still needed describing the frequency and severity of attacks of Internet connected industrial controlled devices.

Furthermore, studying attacks of industrial systems requires researchers knowledgeable in industrial control systems and Information Technology (IT) infrastructure as well as data from past industrial attacks. Costs of even a modest industrial control system for purely educational use can quickly surpass \$100K, making research, education, and training a significant investment.

A need exists for a scalable and cost effective solution that is suitable to address research problems both in the laboratory and on the Internet. In addition, a need exists for a platform for training and education of SCADA and ICS attacks before progress can be made in understanding, and ultimately preventing attacks on industrial networks. Current

solutions exist that address one of these aspects, but none are adequate to address all needs in a cost effective manner.

One solution is the emulation of field devices such as programmable logic controllers (PLC) that can interact with other SCADA and DCS components including other field devices and control software. Emulation is different from simulation because simulators model the way a target system works, but do not actually need to behave the same, as in a flight simulator [Rot00]. Therefore, an emulated field device is able to perform the same actions as the target PLC, but is implemented completely in software. Emulated industrial control systems therefore combine the best of both real hardware devices for research and education, and software simulations for scalability and low cost through custom software implemented on dedicated hardware.

## **1.2 Research Goals**

The goal of this research is to develop a low-cost, portable, and configurable industrial control emulator for a popular type of PLC for application-level services. The design of the emulator is based on criteria that prioritize cost and performance in the implementation of a SCADA attack-landscape research honeypot. The target PLC is a low-cost PLC with Internet connectivity, the Koyo DirectLogic 405 with the ECOM-H4 Ethernet Module [Aut12b]. Though it is less common in critical infrastructure, this device is widely used at the plant and factory level in systems across many industry sectors [Dig12a].

This research aims to establish metrics to quantify the emulator's performance on two popular computing platforms, the Gumstix single board computer, and a standard laptop PC. The metrics are chosen to assess the accuracy of the emulation and its viability as a PLC honeypot for SCADA landscape research. These metrics are compared to baseline measurements taken for the emulation target (the Koyo PLC) to determine the quantitative accuracy of the emulation with regard to response packet bytes, Nmap OS Fingerprinting, Metasploit attack success, and response timing. Logging capability of the

emulator platforms is also measured to ensure suitability as honeypot and research tool to log all interactions.

The experiments conducted seek to answer questions about the accuracy and logging performance of the emulator on the Gumstix and PC platforms. The questions address packet, scanning, attack, and timing levels of accuracy, as well as the extent that queries and responses to and from the emulator are logged. The impact of request frequency and emulator platform (Gumstix vs PC) is also assessed.

These accuracy and logging metrics are designed to determine the suitability of the emulator as a honeypot and research tool. It is hypothesized that the industrial controls emulator implemented on both the Gumstix and PC platforms is accurate to both standard and non-standard queries, and is able to log all packets to a remote logging service. The metrics are designed to compare the target PLC and emulated systems to quantify the accuracy of the emulator. The results form the basis for a qualitative determination on emulator authenticity, as describe in Chapter 3.

### **1.3 Thesis Layout**

This chapter introduced the motivation and goals of the thesis research. Chapter 2 provides background information and fundamental concepts on SCADA systems and emulators, as well as recent work related to SCADA and ICS field device emulation. Chapter 3 details the design and implementation of the PLC emulator system. Chapter 4 outlines the methodology used to quantitatively asses the accuracy and functionality of the implementation. Chapter 5 provides discussion and analysis of the experimental results. Chapter 6 discusses conclusions from the results and provides details and suggestions for future work.

## **II. Literature Review**

### **2.1 Introduction**

This chapter provides background information needed for a basic understanding of SCADA systems, emulators, and honeypots. Section 2.2 provides relevant information on industrial control and SCADA, including relevant protocols, threats to their security and previous attacks on SCADA networks. Section 2.3 discusses the concept of the device emulation, examples of implementations, and applications. Research related to industrial control emulators and honeypots is discussed in Section 2.5

### **2.2 SCADA**

Supervisory Control and Data Acquisition (SCADA) systems are large, highly-distributed networks that monitor and control industrial processes and are highly integrated into critical national infrastructure systems. In fact, SCADA systems are the nervous system of the industrial infrastructure animal. Many national infrastructures such as oil fields, power grids, Uranium enrichment facilities, and utility delivery systems depend on SCADA networks and, on a smaller scale, distributed control systems (DCS). These systems provide real-time sensor data from the field to operators that may be miles or even thousands of miles away and are capable of remotely adjusting control system parameters. DCS and SCADA both fall under the broader classification of industrial control systems (ICS), and the three terms are often interchanged depending on the specific industrial application being discussed [SFK11]. Although the focus of the background research is on SCADA systems, all of the concepts developed, are directly applicable to DCS.

The main purpose of SCADA systems is to extend the operator's visibility and interaction to processes that would normally be impossible due to distance or access restraints [Boy09]. To increase the operator's span of control, SCADA systems are made

up of complex networks that include both hardware and software components. Figure 2.1 shows different sub-systems of a typical SCADA system, with the primary control center at the top. Inside the control center, are the human operator, the console with integrated human machine interface (HMI), and Master Terminal Unit (MTU), along with additional support equipment.

If the operator is the heart of the system, the MTU is the brains. The MTU is a computer whose primary role is to store and process data collected from distributed field devices, which is no small task considering there may be thousands of field devices scattered over thousands of square miles. In this context, anything local to the operator is considered to be physically located in the control center, and anything located remotely is considered to be in the field.

In addition to communicating with field devices, the MTU communicates with the operator via some type of HMI or console. For very simple systems, the entire HMI may simply be a switch board and lights. However, modern systems typically implement the HMI on a computer running customized software. The operator has complete view and control of the system through a graphical interface. He or she can poll data from different field units, modify the status of actuators or valves based on usage requirements, and in many cases, even re-program field devices. In addition to the primary control center, many large SCADA networks may also implement regional control centers. The primary control center delegates process supervision to regional control centers to more effectively control the system. This delegation both increases the span of control of the primary control center and reduces the distance that information must travel from remote field stations to the regional control center before being processed. Because of the distance between a primary and regional control center, the communication link may consist of a Wide Area Network (WAN), telephone network, or even satellite. Though the operator does have the ability to make changes to the SCADA system via the HMI, more than 99% of all messages from

the MTU are automatic [Boy09]. This is particularly true for very complex or very large systems where thousands of measurements and control signals are sent to and from the MTU every second.

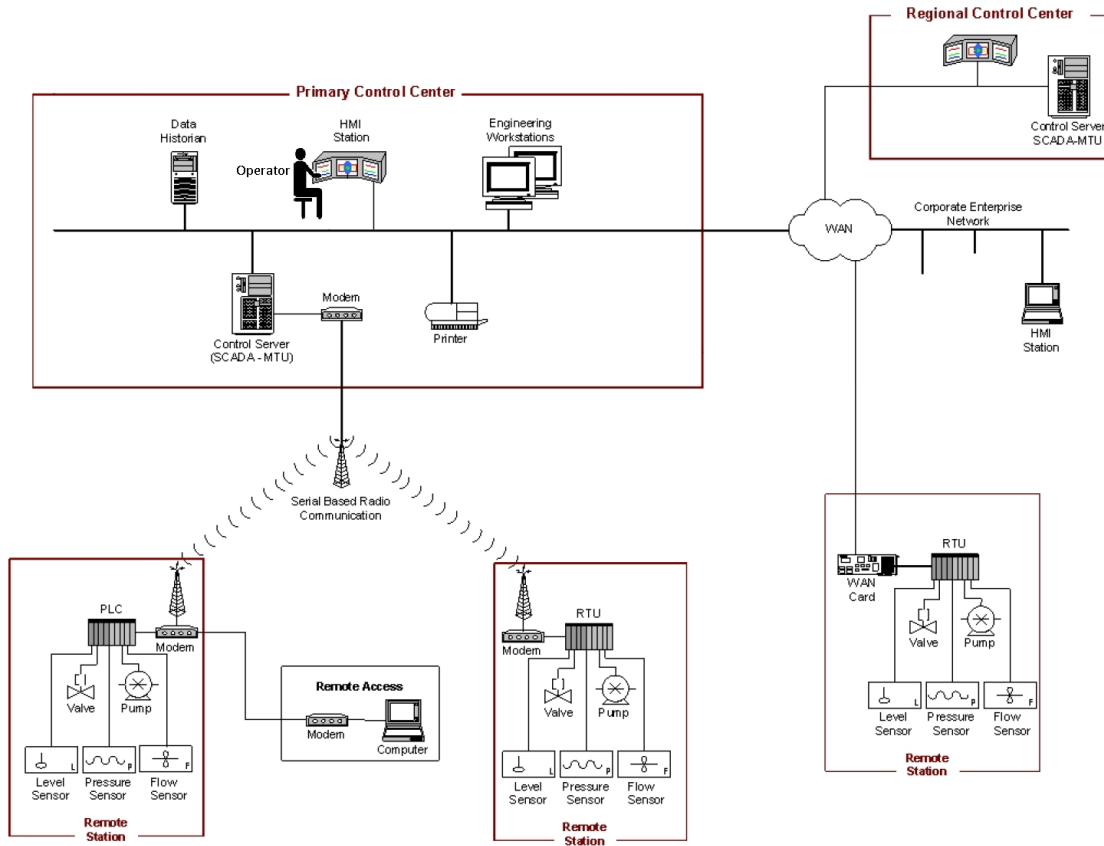


Figure 2.1: A Typical SCADA System(Adapted from [SFK11])

Remote development and maintenance is a significant consideration for SCADA systems. Despite the fact that many field devices in SCADA networks are not easily accessible or situated in harsh locations, their application software still needs to be upgraded and maintained. For this type of system development and troubleshooting, a special type of HMI called an engineering workstation is used and is typically located in the control center. This workstation has the same functionality of an HMI but includes



development tools that allow system designers to program remotely, collect diagnostic data not typically displayed to the operator, and modify the HMI software used on the operator's console. Since these workstations have access to the SCADA network in the same way as a standard operator's HMI, they are sometimes used as an operator's workstation when not used for development [Cap10]. When the MTU (or operator) queries a remote unit, the message travels from the control center to remote field locations and into specialized control units and measurement devices that directly interact with physical processes. There are many different types of field devices controlled by the MTU depending on the specific industry and the processes being controlled. The two primary hardware field devices with which the MTU communicates are remote telemetry units (RTU) (as opposed to remote *terminal* units) and programmable logic controllers (PLC).

While similar in concept, there are differences between RTUs and PLCs. Legacy RTUs, are limited to collecting sensor data and controlling actuators based solely on commands from the MTU, with very limited computational power to independently control processes. This type of RTU is only a slave device, responding to requests from the MTU, and does not initiate queries of its own.

PLCs, on the other hand, have complex software programs that allow them to not only transmit sensor data to the MTU, but also to control field processes independently. The ability to build automated control loops using PLCs independently from the MTUs makes them powerful. PLCs control complex processes locally based on user-programmed ladder logic and set points, maintaining only routine interaction with the MTU. In addition to reducing the amount of communication required between the MTU and PLCs, local control also promotes fault tolerance, a main objective of SCADA systems [Jae08]. In a SCADA system, PLCs are considered a slave device with respect to the MTU but are considered masters of the processes they independently control. While some PLCs can act as master devices to other PLCs, this functionality is more indicative of a small scale DCS rather

than a true SCADA system. The attractive cost, availability, and versatility, make PLCs the most universal field device. In addition to the PLC, MTUs also communicates with remote *terminal* units, another popular type of field device.

Modern remote terminal units (RTU) are very complex, flexible, extensible, and customizable with computational capacities far superior to even the most capable PLC. A single modern RTU for a custom application can support many different protocols simultaneously, provide long-term data logging, have expandable program memory capacity, and endure a wide range of operating environments [Mot07]. There is a price to be paid for such flexible and complicated systems, however. Lengthy setup time and considerable cost put RTUs at a disadvantage, not to mention complicated requirements scoping and lead times because they are often made-to-order systems.

Terminal RTUs frequently act as both the master and slave in SCADA networks. In a standard SCADA system, intelligent electronic devices (IED) are slave devices to a master RTU. IEDs include valves, breakers, sensors and protective relays which communicate on any of a dozen different protocols. Where money and lead-time are no object, modern RTUs may certainly be the answer, but this situation is practically non-existent in industry. PLCs are a better alternative for most applications.

### ***2.2.1 The Programmable Logic Controller.***

PLCs date back to the late 1960s, where their primary motivation was to replace the electromechanical relay-based control systems, which were prevalent in industrial systems [Boy09]. By adopting PLC technology, system engineers traded in relay systems built using hundreds of hard-wired devices with flexible, rapidly re-programmable, and physically smaller microprocessor-based devices. The main advantages of PLCs over RTUs are their relative low cost and modularity. PLCs are commercial off-the-shelf (COTS) devices, and are readily available and replaceable while most modern RTUs are custom- built devices.

PLCs are specialized embedded computers with many different manufacturers and implementations. PLCs are a type of low-level field device in SCADA and industrial networks that actually control the physical processes. For the purposes of this research, all PLCs are considered to have common core characteristics: (1) ability to complete electronic circuits and measure electronic signals, (2) are programmable in that, for some arbitrary input, they are able to respond with some predefined response, and (3) in the case of SCADA and industrial control systems, are networked with other devices.

### ***2.2.2 ICS and SCADA Protocols.***

Many different protocols exist for communicating between SCADA and ICS equipment. Some protocols are open, meaning that the standards and specifications are published, while others are proprietary, meaning the standards are unpublished. Open protocols can be well understood simply by reading the specifications. Proprietary protocols, however, can have many intricacies that can only be understood after observing many different types of protocol interactions. Even then, fully understanding a proprietary protocol may be impossible. Depending on the application, the proprietary specifications may be released to developers by the manufacturer under a non-disclosure or other written agreement.

#### ***2.2.2.1 Modbus Protocol.***

One of the most widely used ICS protocols is the Modbus protocol. In 1979 Modicon corporation released Modbus as the first industrial communications network protocol which was soon adopted as an industry standard [Plc12]. The ease of use without authentication or excessive overhead makes Modbus one of the most widely deployed industrial control communications protocols today. The protocol operates at the application layer of the open systems interconnection (OSI) model, meaning it operates independently of the underlying communication network. In the case of the Internet, the underlying communication network utilizes the transmission control protocol and Internet protocol (TCP/IP). The

Modbus protocol is limited to one-way data initiation, meaning that queries only originate from the MTU or other master device, to which the slave device responds [Kna11]. There are several variants of Modbus including Modbus RTU and Modbus ASCII which are very similar, and Modbus TCP. Any specific implementation of a SCADA network might use a combination of these or other protocols.

***Modbus RTU and Modbus ASCII Variants.*** Both the Modbus RTU and Modbus ASCII communication variants are transmitted via serial communications protocols, and are virtually identical. The nature of serial protocols such as RS-232 and RS-485 limits the end-to-end lengths between communicating devices. The current Modbus RTU Specification and Implementation Guide [Mod06] states that the maximum length for a cable depends on "baud rate, the cable (Gauge, Capacitance or Characteristic Impedance), the number of loads on the daisy chain, and the network configuration." Considering these factors, practical implementations of Modbus RTU/ASCII on RS-485 are limited to a maximum length of 1000m while RS-232 is limited to less than 20m [Mod06]. These factors severely limit direct serial connections for SCADA networks.

The useful range of the Modbus RTU/ASCII protocol can be significantly increased with modems to span the long distances using means such as radio, plain old telephone system (POTS), cell phone service, or even satellite. Modems operate at the data link and physical layers of the OSI model. Their job is to receive serial data from an MTU in the Modbus RTU or ASCII protocol, encapsulate and transmit the data to another modem across the selected medium. The second modem receives and decodes data back into a serial message at the remote location and transmits it to the PLC. The PLC then sends information back to the MTU the same way.

The need for dedicated modems to communicate across long distances makes the Modbus RTU and Modbus ASCII protocols less attractive than their Modbus TCP counterparts in many configurations. Additional considerations such as addressing limits,

serial repeaters, and other electrical requirements make direct Modbus RTU and Modbus ASCII less suitable for modern day, large-scale SCADA operations as compared to Modbus TCP.

***Modbus TCP Variant.*** Of the three Modbus protocol variants, Modbus TCP has become the most widely used. As the name implies, Modbus TCP is designed for use across TCP/IP networks, commonly referred to as *The Internet*. In Modbus TCP, the Modbus protocol fields are encapsulated in TCP packets in order to make remote communications between the MTU and the PLC easier, since they can be routed over modern information technology (IT) networks [Kna11].

Modbus TCP allows for an easier IT network topology to be designed and configured compared to serial networks. Furthermore, Modbus TCP is not constrained by the addressing or distance limits because Internet protocols and specifications already address those issues. There are very few reasons to build dedicated long-distance networks for modern SCADA and ICS implementations using serial protocols, when the infrastructure that comprises the Internet already exists. Many systems incorporate both serial and TCP versions of Modbus. In these systems, Modbus RTU and Modbus ASCII are used for short distance communication between devices at a remote field location, and Modbus TCP for the long-haul between the MTU in the command center and remote PLCs. Thus, even Modbus RTU and Modbus ASCII systems may encapsulate data in TCP/IP packets and send them across the Internet exposing the SCADA or ICS network.

#### ***2.2.2.2 DirectNet Protocol.***

The DirectNet protocol is an open-source protocol used by the DirectLogic family of PLCs manufactured by Koyo Electronics Industries. This is a simple protocol used to read and write memory locations on a PLC from another PLC, HMI, or engineering workstation. Common uses of DirectNet are to upload or download system data such as Timer/Counter status, I/O information, and variable memory information [Aut12a]. Like Modbus, the

protocol is used across serial connections such as RS-232 and RS-422 or encapsulated in IP packets for traversal across an IT infrastructure. Though DirectNet is not an industry-standard protocol, a popular variant known as the Communications Control Module (CCM) protocol is virtually identical to the DirectNet specification. It is supported by most General Electric PLCs and legacy Texas Instruments devices [Sof12, Gei10, Koy12].

#### ***2.2.2.3 K-Sequence Protocol.***

The K-Sequence protocol is a proprietary protocol used exclusively by Koyo and Automation Direct branded PLCs, including the DirectLogic family. This protocol can also be transmitted across a serial connection or encapsulated in an IP packet. The primary use of K-Sequence is to communicate between HMI software and PLCs including password protection of memory segments, and transferring ladder logic to and from the device [Aut11].

Because K-Sequence is a proprietary protocol, it is difficult to compare to the DirectNet protocol and where one protocol is used over the other. Many of the capabilities between the two protocols appear equivalent based on published features. One published difference between the capabilities of these protocols is that the K-Sequence protocol can perform direct write operations on individual memory bits, giving it the ability to control individual I/O points, while the DirectNet protocol is only writable at the byte (8 bit) resolution [Tho05]. The K-Sequence protocol is implemented on all Koyo/AutomationDirect PLCs, while DirectNet may not, making the K-Sequence protocol more universal.

#### ***2.2.2.4 Host Automation Products (HAP) Protocol.***

The HAP protocol is a proprietary protocol used for Ethernet communications between an HMI or engineering workstation and ECOM Ethernet Module which is an optional component for all Koyo/Automation Direct PLCs. HAP is also used for PLC to PLC communications. This protocol is encapsulated in a user datagram protocol

(UDP) packet, and is used to configure communications settings, upload firmware, and to encapsulate other protocols, such as DirectNet and K-Sequence. Though this protocol is proprietary, many free resources exist to help dissect it manually, including a software development kit (SDK) provided by the manufacturer [Hos12, Wig12]. The manufacturer can provide the SDK source code written in C to individuals directly for a legitimate use [Hos12].

### ***2.2.3 SCADA Communication Networks.***

Depending on the application, any variety of communication protocols may be utilized between an MTU and field PLCs. Many of them, like Modbus, K-Sequence, and DirectNet, offer different variants optimized for use over a particular medium. Modbus offers a TCP version intended to operate across a private IT network, but is often extended beyond the private network across SCADA control networks to the Internet.

Whether intentional or not, there are consequences for utilizing the Internet for SCADA communications. The most significant advantage for using the Internet for long distance communication is cost. Everyone with an Internet connection shares the cost to maintain the overall network, including the infrastructure. Furthermore, access to the Internet is easy even in remote locations with complex networks of satellites, telecom equipment, television cable, fiber optics, and radios connecting information systems globally.

There are also negative consequences for connecting devices to the Internet. For example, anyone can scan an Internet-connected device and potentially access it. Evildoers are able to identify, probe, and exploit unprotected computers in as little as fifteen minutes [Spi03a]. Attacks on Internet connected devices are commonplace, including attacks on Internet connected ICS devices, and will continue to be a problem well into the future [Con13].

Connecting devices to the Internet puts the devices and the entire system at risk. Risk is defined on a two-dimensional scale as a direct function of probability and consequence. With the integration of equipment running Internet protocols like Modbus TCP and HAP, SCADA system designers are accepting the risk of an attack as soon as they plug the PLC into the network. The higher the consequences of an attack, and the higher the probability, the more risk one accepts when connecting a device. Though the millions of people who connect personal devices to the Internet every day accept a similar risk, the SCADA risk is much greater because the consequences are much more severe.

Every SCADA device connected to the Internet is vulnerable to an attack, simply by being connected. Vulnerabilities [PeP09, Saw11, San11] and automated exploits [Wig12, Ras11, Rob12, Rap12] are released regularly. As more vulnerabilities are exposed, the list of potential ICS targets on the Internet increases, and automated exploits put sophisticated tools in the hands of all potential attackers, including low-skill *script kiddies*. The consequences of a successful SCADA attack using such automated tools are almost endless e.g., lack of drinkable water, electrical grids shutting down, oil pipelines burst, power generators destroyed, etc.

#### **2.2.4 SCADA on the Internet.**

In an attempt to discover just how many SCADA devices are connected to the Internet, and to what extent they are vulnerable, graduate student Eireann Leverett of the University of Cambridge developed a search tool in 2011 to match a list of vulnerable SCADA devices to publicly-available exploits, and visualize the vulnerable systems on a map [Lev11]. The methodology and results of the research are detailed in his Master's thesis. Leverett used the SHODAN search engine which allows users to find specific Internet connected devices like routers, servers, PLCs, HMIs, etc, based on simple search terms [Sho10]. The difference between SHODAN and a search engine like Google is that SHODAN allows users to search for *devices* as opposed to *websites* by indexing response messages (banners) from Internet



hardware. In combination with SHODAN, he used a well-known exploit engine to identify which devices could be attacked with open-source exploits. Leverett's SHODAN search turned up many types of SCADA devices including HMIs, engineering workstations, and various field devices by popular manufacturers including Allen Bradley, Powerlink and SoftPLC.

At the 2011 SCADA Security Scientific Symposium (S4) conference, Leverett announced that he discovered a total of 10,358 SCADA devices globally were directly connected to the Internet over a two year span [Pet11]. The United States had by far the most of any country with 3,920 Internet-connected devices [Lev11]. While not all of these devices were remotely exploitable, many of them were. The exact number of exploitable devices is unpublished. The fact that these SCADA devices were connected to the public-facing Internet at all puts them at risk, since new vulnerabilities are released regularly. Leverett's research proves that SCADA devices are, in actuality, being connected to the Internet, giving cause for concern. When presented with the results, many asset owners associated with the Internet-exposed SCADA systems were unaware that their devices were connected to the Internet at all.

To validate Leverett's work two security consultants and researchers from InfraCritical.com began project SHINE (SHodan INtelligence Extraction) in April, 2012 [Dhs12a, RaB12]. Also using the SHODAN search engine, an initial list of almost 500,000 of Internet-exposed control devices was compiled. Working with the ICS computer emergency response team (ICS-CERT), the researchers were able categorized each of the 500,000 suspect IP addresses by sector type, organization, and location, and determined approximately 7,200 control system devices across the United States were exposed on the Internet. This work undeniably confirms that SCADA and ICS devices are connected to the Internet, and heightens the need for better understanding of the SCADA attack landscape.

According to a 2009 Government Accountability Office (GAO) report, SCADA systems are often exposed via corporate IT networks [Gao09]. The integration of automated utility billing and the development of the smart grid are driving commercial business networks to interact directly and automatically with control networks.

Another reason SCADA and DCS devices are exposed to the Internet is by their default configuration. Security features on newly installed products are often turned-off by default for ease of installation [Doe02]. New models advertise their SCADA products as "drop-in replacements" with advanced features and capabilities such as remote maintenance, email service, and web service turned-on by default, whether needed or not [Doe02]. Unless SCADA system integrators are cognizant enough to notice or diligent enough to check, these features will remain on, leaving not just a single PLC but potentially the entire SCADA network vulnerable [Doe02].

Default settings pose a common problem among many Internet-connected devices; the manufacturers leave it to the users to protect themselves. Unlike most traditional IT, SCADA systems tend to be more fragile, especially to network scanners. Two popular network scanning utilities that are used to detect vulnerabilities on traditional IT networks are Nmap [Lyo12] and Nessus [Nes12].

Nmap is a multipurpose port scanning tool that scans computers for potential vulnerabilities such as open ports and services. It is also used to map complete network topologies. A view of open ports, services, and topologies on a system is vital to understanding and securing the network. Similarly, Nessus scans network computers, but is able to compare the vulnerabilities with publicly available exploits, giving the user a practical assessment of device security. These types of scanning tools are vital to traditional IT security administrators to control and assess their networks. The tools are also useful for scanning SCADA networks with the same benefits. However, network scanners like these have been shown to cause significant disruptions when used on SCADA networks.

For example, a 2005 report from Sandia National Laboratory describes several occurrences of real world losses due to network scans on SCADA networks [Dug05]. Actual examples of negative behavior in response to network scans include the erratic movement of a 9 foot robotic arm in an area shared with personnel, the malfunction of a microchip manufacturing process resulting in \$50,000 worth of lost electrical components, and a gas utility company that had to shut-down gas delivery to its entire customer base due to unresponsive SCADA equipment [Dug05]. SCADA system operators and maintainers are understandably hesitant to scan their systems for vulnerabilities when the systems are directly connected to physical processes. However, it is difficult to defend a network that cannot be adequately assessed for security vulnerabilities and misconfigurations.

Besides scanning sensitivity, other differences make SCADA systems more difficult to secure than traditional IT as well. The life expectancy of SCADA equipment is much longer than typical IT equipment at around 7-20 years as opposed to only a few years for most IT equipment [HoL10]. By nature of being highly distributed, many SCADA field devices are in areas that are physically isolated or difficult to reach. These factors also weigh heavily on SCADA managers, who concede it is far too expensive to replace equipment for the sole purpose of applying security [HoL10].

Aside from these differences, SCADA systems are beginning to look more like traditional IT networks in other ways. Modern SCADA networks implement SCADA protocols that ride on Internet protocols, while legacy SCADA networks are being retrofitted. Familiar user-level application interfaces, like web servers and email, are emerging on individual devices exposing new vectors into old systems. Like the Internet, the vast majority of our national critical infrastructure, much of which is controlled by SCADA, is privately owned and operated, with estimates as high as 90 percent held in the private sector [SFK11]. As SCADA networks start to look more like traditional IT networks, the "air-gap" (i.e., physical isolation of industrial networks from insecure

networks) between internal control and external corporate networks is being reduced. A need to protect SCADA networks like traditional IT networks exists in order to prevent them from being attacked.

### ***2.2.5 SCADA Threats.***

Because of both the similarities and differences from traditional IT, SCADA networks are vulnerable to attack. While the consequences of a successful attack on a corporate IT network can have devastating economic, personnel, and political consequences, traditional IT attacks do not have the immediate physical impact that attacks on infrastructure have. Stouffer et al. describes possible incidents can occur on ICS networks [SFK11]. In particular, any SCADA or ICS element could be the target of an attack, from the HMI all the way down to the field-level PLC devices, depending on what the desired outcome of the attack is. Stouffer's list of possible incidents includes:

- Blocked or delayed flow of information through ICS networks, which could disrupt ICS operation
- Unauthorized changes to instructions, commands, or alarm thresholds, which could damage, disable, or shut down equipment, create environmental impacts, and/or endanger human life
- Inaccurate information sent to system operators, either to disguise unauthorized changes, or to cause the operators to initiate inappropriate actions, which could have various negative effects
- ICS software or configuration settings modified, or ICS software infected with malware, which could have various negative effects
- Interference with the operation of safety systems, which could endanger human life

As a primary mitigation, Stouffer et al. emphasize the separation of industrial control networks from corporate networks to reduce the exposure of the control network. Gone, however, are the days where SCADA security can be based primarily on isolation; physical access to the network is no longer necessary. As discussed in [HoL10], SCADA networks are distributed by nature, and corporate networks often must be connected to control

networks to function properly. There is no better proof of Internet-connected SCADA systems than the recent empirical results from Leverett and Project SHINE, described in Section 2.2.4. With over 7,200 Internet-connected devices discovered in the U.S. alone, no one can dispute the Internet presence of SCADA.

Physical access *is* needed to infiltrate some systems. To emphasize this, Knapp adds another possible incident to Stouffer's list: malicious software (malware) infection [Kna11]. The threat of malware on SCADA systems represents a different concern that is unmitigated by system isolation as proposed in [SFK11]. The fact is, that even truly air-gapped systems may still be susceptible via other means. By carefully crafting malware to automatically propagate via removable media such as a USB drive, the malware is able to arrive at its target: the air-gapped SCADA network. A malware attack is particularly significant because, not only can malware initiate additional incidents, but it can also force significant downtime due to forensic analysis, cleaning and even replacement. Furthermore, malware attacks often go undetected by antivirus and other protection mechanisms due to their constantly changing signatures. Examples ranging from proof-of-concept SCADA malware [CNM08], to fully operational malware discovered *in the wild* are readily available. The most famous example of the latter is Stuxnet, which propagated onto an isolated network via a USB device, which is discussed further in Section 2.2.6. Both air-gapped and Internet-connected SCADA networks alike have vulnerabilities that have been exploited by attackers in the past, and will continue to be targets in the future.

### ***2.2.6 Cyber Attacks: The Modern State of the SCADA Landscape.***

One of the first publicly disclosed SCADA attacks was the 2000 attack on the sewage control system in Queensland, Australia. A disgruntled former employee who worked for the contractor that installed radio-controlled sewage equipment for the Maroochy Shire Council used stolen equipment and knowledge of the SCADA systems to release 264,000

gallons of raw sewage into local parks, rivers and the grounds of a Hyatt Regency hotel [AbW08]. The 46 attacks occurring over two months resulted in significant loss of wildlife, public health hazard, and an unbearable stench. This event demonstrated the consequences of an intentional, targeted attack by a person with specialized knowledge of SCADA systems, who in fact, was never even an employee of the organization he attacked [AbW08]. The SCADA network relied on wireless communication to transmit messages between control units and field devices, which is a concept similar to being connected to the Internet because potentially anyone with the appropriate equipment and knowledge can attack the SCADA system.

Many examples of SCADA malware attacks exist. Perhaps one of the most widely publicized has been the Stuxnet worm. Stuxnet is an example of a highly-targeted attack suspected to be directed towards nuclear enrichment control processes [Cla10, BMS11]. Taking advantage of several unpublished vulnerabilities or *zero-days* in Microsoft Windows, the self spreading mechanism of Stuxnet enabled it to move from an insecure system to the air-gapped industrial network via a USB connection. Once in place on the target network, Stuxnet gradually manipulated Siemens PLCs in a manner intended to ruin the Uranium enrichment process. Stuxnet then modified information sent to the HMI to make the operator believe the process was running correctly [FMC11]. The Stuxnet attack demonstrates the consequences of an air-gapping-only protection strategy.

In November 2011 the deputy assistant director of the FBI's Cyber Division announced that hackers had accessed and been in control of SCADA systems in three different cities, with one being a major US city [Hod11]. Though no actual attacks were mentioned, it was said the attackers could "dump raw sewage into the lake" or "shut down the power plant at the mall." This is a clear indication of the state of SCADA security.

### 2.2.7 *Researching SCADA Threats.*

To research current threats and vulnerabilities, SCADA research facilities have been developed within the public and private sector. Recognized as an issue of national concern, the National SCADA Testbed was developed as a collaboration effort between the Idaho National Engineering and Environmental Laboratory and Sandia National Laboratory in 2003 at a cost of \$114 million. As the Nation's premier SCADA research facility, it is used to research the identification and mitigation of existing system vulnerabilities and to develop advanced system architectures for more secure and robust SCADA systems on a full-scale industrial level [Ken09]. This system provides a suitable testbed for real-world SCADA security threats, but is hardly a design an educational institution can follow [Ine03].

On a smaller scale, the Mississippi State University has implemented their own educational SCADA network, but it is limited to five control devices and two master stations, which is practical for educational purposes, but inadequate for large industrial-scale research [MVD10, Vau09].

In addition to testbeds that use actual SCADA equipment, pure software-based simulations of SCADA networks as well as hybrid software-hardware configurations have been implemented, such as the University of Illinois' *Cyber Security Testbed*, or Royal Melbourne Institute of Technology's *SCADA Security Testbed* [DTO<sup>+</sup>06, QMJ<sup>+</sup>09]. These test station configurations are used primarily for SCADA research, and are inferior to commercial hardware configurations because the models are limited. However, their low cost and scalability make them attractive particularly for researching SCADA network protocols and interactions. These types of testbeds are typically built using network simulation frameworks such as the Real-time Immersive Network Simulation Environment for Network Security Exercises (RINSE) used in the University of Illinois's configuration, and Emulab [Emu12] and are useful for performing protocol simulations in complex

network situations. These testbeds lack fine adjustments needed to adequately simulate SCADA-specific aspects, such as firmware uploads or propagation of SCADA-specific malware, making extension to applications that require a standard user interface awkward at best.

Emulated industrial control systems combine the advantages of dedicated hardware and cost effective, scalable software simulations. Emulators achieve this by implementing industrial control system protocols and services on flexible programming platforms such as Linux.

### **2.3 Emulators**

An emulator is a "program that runs on one computer and thereby virtually recreates a different computer" [Sla03]. An emulation is virtual because it is not actually physically implemented, but functions like the original device. In many applications, the goal of emulation is to reproduce the functionality of the target device with high enough fidelity that emulation can replace it. The original device must be well understood at some level because its intent is to accurately represent functionality of the original target device. Emulation can be implemented at three different design levels, the application software level, system software level, or hardware level [HoW05]. These three levels are a starting point to define the difficulty in building an emulator.

Application level emulators reproduce the operation of individual programs, such as a document reader or editor. These types of emulators are difficult to implement because application software is often very complex and proprietary. Furthermore, it is difficult to properly define the actual behavior of the system due to size and complexity of applications and a total lack of specifications. Application level emulators can only reproduce the functionality of the specific application for which they are designed. This type of emulator is independent of the system software or hardware. It is also highly dependent on the



operating system and hardware configuration in its environment to function, just like the original application.

Emulating system software (i.e., operating system software) can be similarly challenging because, it also consists of proprietary programming code. However, there are usually much more complete specifications for system software compared to application software. These specifications exist to make it easy for software programmers to write applications based on the system software. Without them applications could not be written, making the system software useless. Accurate and complete emulation of system software enables execution of any arbitrary application. However, a system software emulation is still dependent on the hardware device on which it was implemented.

Unlike these kinds of software emulations, hardware emulations are easier to define and specify because the target hardware system must have been properly specified to be built in the first place [Rot00]. Software is meant to be interpreted automatically by hardware, and therefore the software *is* the specification. Hardware on the other hand is meant to be interpreted by humans (or software) to actually produce a physical device used as a building block for other hardware and software [Rot99]. Therefore, very detailed specifications are usually available again because programmers need to know how to access the hardware to write system and application software. After the hardware is accurately emulated, arbitrary system and application software can be used as-is to perform their respective operations.

In many modern implementations, especially embedded computing devices, all three emulation levels might be proprietary and its software, closed-source. In this case, the only "specification" available might be the actual operations and observable characteristics of the application software, restricting emulation only to application level implementations.

### ***2.3.1 Emulated Honeypot Services.***

There are many applications of emulators including the preservation of digital media [Sla03], the implementation of obsolete game platforms [Sal05] and to promote business efficiency as in the use of virtual machines like VMWare [Vmw13]. However, emulation is used in the cyber security context in development of computer decoys or *honeypots* used by security professionals to deter, delay, and gather information on network attacks.

Many honeypots emulate services and protocols designed to look and feel like the real thing, but are non-working. Emulation produces better results than re-purposing production machines because emulators can be customized to provide any set of services. Furthermore, robustness and security can be prioritized as design features in emulators, which may not be the case for a production device. These emulated honeypots usually provide only basic responses to a very limited set of commands invoked by attackers. Honeyd is an example of this type of emulated honeypot system. Created by Niels Provos, Honeyd is open-source honeypot software that allows networks of arbitrary devices and services to be implemented on an a single device [PrH07]. In addition to providing services for attackers to probe and compromise, logging is a critical function for all honeypots. Specifically, the logging of all network packets is critical to have a complete record of an attack. This record is used to dissect attack methodologies, capture custom tools, and gather information on the attackers themselves. The use of emulated honeypots is extended from traditional IT into industrial control systems in Section 2.3.3.1.

### ***2.3.2 Industrial Control Emulators.***

Implementation of industrial control emulators at the hardware level is most desirable. This allows emulation of the ICS hardware in software to run the industrial control device's firmware on any platform desired. Besides the legal and financial implications of reversing proprietary embedded platforms, there are significant technical challenges as well. Even on the same device, the use of many different architectures, platforms, and languages is likely,

making the task very challenging. Additionally, the components inside these devices are often unmarked, or are marked with custom serial numbers, revealing little about their primary function.

To further complicate the task, firmware can be packed or cryptographically encoded to only be decoded by the specific hardware [CKS<sup>+</sup>06]. From start to finish, the process of emulating the true hardware configuration of an industrial control device is possible, but it would be very expensive and time consuming. Furthermore, errors or failures in the target device's operating firmware would be in the emulator, by extension. From an accuracy standpoint this may be desirable, but not from a robustness standpoint. Thus, the best approach is to design and implement an application-software-level emulator on dedicated hardware that reproduces observable and measurable network services available on the target industrial control device.

### ***2.3.3 Uses of ICS field device emulators.***

With the development of an ICS field device emulator, extension of traditional IT technologies and methods into industrial settings is possible. The focus of this thesis is on applications for research of unknown SCADA attack factors for field devices exposed to the Internet, which is discussed in this subsection. Additional applications include laboratory research, evidence collection, cyber sensing, and SCADA education, and are each developed in Appendix A.1.

#### ***2.3.3.1 SCADA Landscape Research Using Emulated Honeypots .***

Traditional IT honeypots are often used to learn more about the specifics of attacks on computer network systems. Lance Spitzner, a honeypot pioneer and founder of The HoneyNet Project, defines a honeypot as "a security resource whose value lies in being probed, attacked, or compromised [Spi03a]." Industrial control emulators can be used in the same way to implement SCADA honeypots to learn more about SCADA attacks. Honeypots excel at surveying the landscape of a specific industry, commercial enterprise,

or single company for threat trends, attack tools, and techniques specifically targeting that area. Honeypots are especially effective in new research areas or industrial environments where there is little to no data available. Questions about the current environment can be answered such as: who is attacking, does exploit code exist for a particular vulnerability, or has injectable malware been developed.

This lack of data on attacks and trends describes the current state of the SCADA attack-landscape. It is suspected that many SCADA incidents go unreported. When they are reported, usually very little data is made available to effectively monitor trends, however, attack data is needed. SCADA vulnerabilities and exploits that target currently-deployed devices are available, as discussed in Section 2.2.6. However, it is unknown if industry wide attacks are actively occurring, and if so, what trends exist. Industrial control emulators designed as honeypots could provide answers to these questions.

Just as research honeypots provide data on new vulnerabilities and exploits for traditional IT systems, so too can SCADA honeypots achieve the same for industrial networks. SCADA honeypots can also collect information on the attackers and their tools, of special importance in SCADA applications where very little current data exists. In the past, security research was limited to the tools that hackers left behind, but SCADA honeypots can allow researchers to observe an entire attack from beginning to end because of the common characteristic of all honeypots to maintain extensive logs of interactions. Honeypots can be used to capture and identify automated attacks, as well as human-based attacks. They can also serve as an early warning indicator or to record new methods or approaches that have not been seen before, to understand the reasons behind an attack, or even to better understand elite hackers [Spi03b]. Though some of these applications might be beyond capabilities of emulated protocols and services, the concept of a SCADA honeypot to perform exploratory attack research is sound.

In these types of Internet landscape research applications, the main business of the network revolves around the honeypot and its logging capability. As such, it must be specially maintained to attract attention from hackers. Furthermore, it must look as authentic as possible, fail like the target devices, and allow the collection of useful research results in a conspicuous fashion. It is especially important to disguise the true function of a honeypot. For example, if attackers can fingerprint the system as a honeypot, they can feed it bad information over long periods in an effort to corrupt trend data.

Honeypots are of significant value on corporate networks as cyber sensors as well. They alert security professionals that someone has gained unauthorized access to a system or network when any interaction with the honeypot occurs. Alerts can simply be log entries sent by the honeypot to a remote logging server, indicating interaction with the honeypot. Because honeypots have no production value, there is no reason any legitimate traffic should be flowing to or emanating from the honeypot. All traffic flow is suspect, and as a result only *interesting* packets and details pertinent to an attack are logged. This implies that many fewer log entries must be analyzed compared to other IT protection mechanisms. As opposed to having to sort through gigabytes of logs and reports generated by IDSs, firewalls, and servers, megabytes may be generated daily. Less data means it can be analyzed in a more timely fashion, since almost all data generated by a honeypot is valuable.

As an emulator, a honeypot can be designed to operate on any platform. Because of its computing power, a standard PC is a strong candidate to implement a honeypot emulator. The capability of the PC and the efficiency of the emulator implementation may even make it possible for an entire network of many emulated honeypots (also called a *honeynet*) to be implemented on a single device, similar to Honeyd. Another approach is to use embedded systems with lower computing capacities to implement an emulated honeypot. A device that is small, inexpensive, and capable of running a complete emulator would

have many applications beyond use as a research tool, such as an educational tool, fly-away emergency response tool, and dedicated SCADA sensor. This research explores both approaches—emulation via a standard PC approach, and via an embedded platform, the Gumstix single-board computer.

## **2.4 The Gumstix Platform**

Gumstix is the brand name of a small single board computer that is capable of running various standard operating systems including Linux distributions. An example of its small form factor is seen in Figure 2.2. First developed in 2002, the Gumstix platform has become a popular embedded platform because it is inexpensive, small, and very powerful. They are available in two different processor architectures, ARM Cortex-A8 architecture on a Texas Instruments OMAP3503 applications processor, and the XScale architecture on a Marvell processor [Gum12a, Gum12b]. The Cortex-A8 is a 600MHz 32-bit processor based on the Advanced RISC Machine (ARM) architecture, where RISC stands for reduced instruction set computer. The OMAP3505 gives the Gumstix device the ability to run general purpose 32-bit operating systems such as Linux or Windows, which makes it different than other popular, less expensive embedded platforms such as the Arduino (based on the Atmel AVR architecture) or popular PIC architecture based devices, which are programmed for specific functions.

When coupled with an open-source, general purpose operating system like Linux, the Gumstix becomes a superior development platform because application software as well as kernel software can be reused, tailored, and optimized for a specific purpose in a familiar environment. The advantages of using the Gumstix over another embedded platform are specifically, the ability to run a full-featured operating system. This avoids many challenges with process scheduling, network control, and memory management.

Much of the software used on Gumstix computers was originally developed for the x86 platform. Since most of the differences between x86 and ARM are abstracted away, the

development on the Gumstix platform becomes Linux development rather than embedded development. Different Gumstix models are available and boast features like embedded wireless local area network (WLAN) (IEEE 802.11b/g standard), added Digital Signal Processors (DSPs), and extended operating ranges [Gum12a].

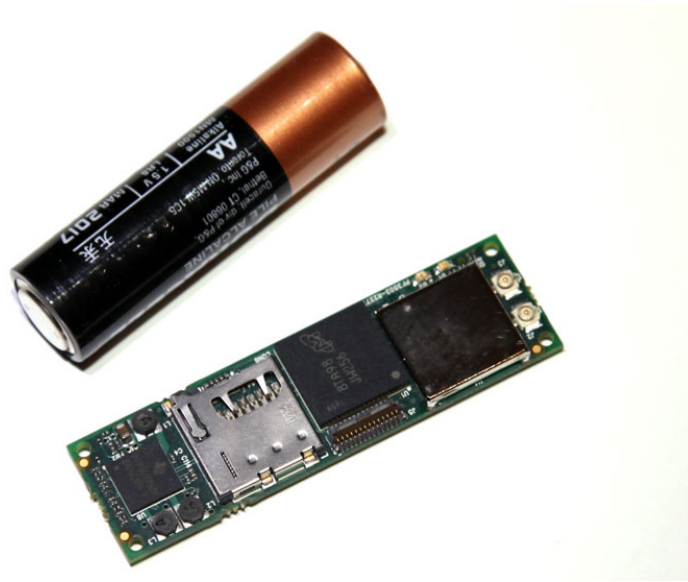


Figure 2.2: Gumstix Module to Scale [Gum112]

The Overo Earth COM, an ARM Cortex-A8 model, used in this research provides a 600MHz processor, with 512MB of RAM and 512MB of non-volatile on-board memory in addition to a microSD card slot for additional memory. These modules ship with the Angstrom Linux distribution 2.6.34 installed [Gum12a]. Another requirement for this research is the use of an Ethernet expansion board to provide use of two Ethernet interfaces; allowing collection of data over one *Internet-facing* connection and out-of-band logging capabilities over the other *LAN-facing* connection. Despite the fact that the hardware configuration of the Gumstix modules themselves is proprietary, the expansion modules are open-source, making troubleshooting and development easier. The Tobi-Duo expansion board provides two high-performance Ethernet controllers enabling simultaneous Ethernet

connections. This duo-connection configuration lends itself well to the implementation of an emulator for this research.

## 2.5 Related Research

Despite the fact that ICS field device emulators have the potential to fill a void in ICS network security, the concept remains relatively unimplemented. While many commercial solutions exist for traditional IT honeypots, there are no known commercial solutions that implement ICS honeypots in a complete form.

Berman's [Ber12] research entitled, "Emulating Industrial Controls System devices Using Gumstix Technology," motivated the further investigation and development of SCADA field device emulators on embedded devices for this Master's thesis. Berman's work demonstrated that it is possible to emulate basic functionality of the Modbus TCP protocol and out-of-band network logging on a low-interaction honeypot implemented on a Gumstix single board computer.

Among the open-source community, there is currently one known actively-maintained SCADA honeypot device developed by Digital Bond [Dig12b]. This project emulates key high-interaction and low-interaction components of Modbus TCP, FTP, Telnet, HTTP, and SNMP services for use on a Virtual Machine (VM). Because of the bulky VM, and steep system requirements, such a device cannot run on a Gumstix.

Another interesting open-source implementation available from Digital Bond is a *honeywall*. In this setup, an actual PLC is used to provide authentic interaction with the attacker while the honeywall, also implemented on a VM, sits between the attacker and PLC and is used to track and manage the attacker with various monitoring and capture tools. This configuration has value mostly as a research tool because the significant setup and overhead, in addition to the dedication of an actual PLC, makes implementation in an actual industrial environment impractical.



The first open-source SCADA honeynet project was developed by the Cisco Critical Infrastructures Assurance Group (CIAG) lead by Pothamsetty et al. [PoF05], but became defunct in 2005. The honeypot was based on Honeyd, the popular low-interaction honeypot developed by Niels Provos. Source code and limited documentation are still available at [PoF05], but the tool is no longer supported.

At least one implementation of the CIAG honeypot has been documented and the results published. According to a report for the European based Critical Utility Infrastructural Resilience (CRUTIAL) project, a low interaction honeypot based on the work by Pothamsetty et al. was deployed for a nine month period in 2008 [FAA<sup>+</sup>09]. Ports associated with published SCADA vulnerabilities were exposed to the Internet, as well as network ports commonly used for SCADA protocols including port 502 for Modbus TCP, and port 20000 for DNP3. TCP port 80 for HTTP and TCP port 21 for FTP were also exposed. Over the observation period, very little interaction took place across the exposed SCADA ports, with the majority of traffic occurring across the FTP port.

## **2.6 Summary**

SCADA and ICS are exposed via the Internet. ICS vulnerabilities and exploits are discovered regularly and despite their criticality, expense and convenience often overshadow security priorities. Because of the cost and complexity of ICS equipment, there is insufficient exploratory Internet and laboratory research being conducted to better understand and combat risks to ICS and SCADA networks.

Emulation is a way to virtually replicate functions of complex systems, including proprietary protocols and security measures. The development of an accurate ICS field device emulator for use as a honeypot is a viable solution to researching ICS attacks, particularly at the field device level where the actual control of processes occurs.

Despite the importance of ICS networks and their security, the extension of emulated ICS honeypots as both a research and security tool for ICS networks has been slow

to progress because of the proprietary and complicated nature of the devices, making emulation difficult. This thesis simplifies the implementation of an ICS field device emulator designed as a honeypot by duplicating the operation of key functionality on a single-board solution which requires minimal setup and minimal operational impact.

### **III. Device Description**

This chapter describes the design criteria and implementation of the PLC field device emulator destined for this thesis research. Section 3.1 outlines important design considerations for building an industrial control emulator for the purposes of SCADA attack environment research. Section 3.2 describes in detail the design of the emulator implemented for this research effort. Finally, Section 3.3 describes the how this specific implementation is consistent with the criteria established in Section 3.1.

#### **3.1 Design Considerations**

Despite the fact that industrial control systems are fragile, quickly antiquated, and often designed without security consideration, they are expensive, complex, and are networked with other devices providing a vector into otherwise secure IT networks. Through this network interface, most modern PLCs provide application-level services such as web configuration interfaces, file transfer services, remote programming services, and remote monitoring services, all which need to be replicated to create a valid emulation for any particular application.

Industrial control emulators have several design requirements in common with standard network emulators, but specific design requirements are defined by the industrial control honeypot application itself. In addition to the considerations of accuracy, scalability, cost, and configurability for a network emulator such as those described in [ZhN03], industrial control emulators must also consider authenticity, robustness, and record keeping.

##### ***3.1.1 Accurate and Authentic.***

Like network emulators, ICS emulations must be accurate to be authentic. Though related, these ideas are distinctly different. Accuracy is quantitative and can be observed,

measured, and emulated. Authenticity, however, is qualitative. Authenticity is the belief that the emulation is what it claims to be [Lee05]. It is a measure of how realistic the emulation is, compared to the target device. If the user looks at some emulated aspect and finds what is expected, authenticity is established through accurate emulation. In other words, based on the accuracy of the emulation, the user (human or machine) makes a determination on the device's authenticity and decides to proceed or not.

An emulation of a web-based interface, present on most modern PLCs, is accurate if the page has the same banner, form elements, graphics, etc. based on how the page looks in the web browser. If the web browser is the extent of the user's interaction, then posing as an authentic device is easier. However, if the user inspects individual packets, a device may reveal itself as inaccurate if the packets do not match the target device, and therefore may be discovered as inauthentic. To identify the target areas that require emulation, the concept of *levels of accuracy* is developed which describes observable characteristics a user may interrogate to establish device authenticity. To give the feeling of authenticity, an emulation must be accurate on as many levels as practical. The following are examples of levels of accuracy:

1. *Superficial accuracy* is established when a user believes the device is authentic based on what is visually observed, such as how the website or HMI looks.
2. *Packet-level accuracy* is established when a user believes the device is authentic based on the packets sent to and received from the device.
3. *Timing-level accuracy* is established when a user believes the device is authentic based on the timing of queries and responses to and from the device.
4. *Scanning tool accuracy* is established when a user believes the device is authentic based on the results of a scanning tool.
5. *Attack tool accuracy* is established when a user believes the device is authentic based on the success of an exploitation tool.

The more skeptical a user is about authenticity of the device, the deeper the interrogation. Therefore, the overall goal is to anticipate how deep a user will look for signs of authenticity, and make those portions of the emulator accurate.

The implementation of a perfectly accurate emulation is more difficult than simply presenting the same web page as the target device. For some applications, such as education and training applications, an emulation that is less accurate is acceptable because authenticity need not be established; the device can be presented as an acknowledged emulation with no ill effects to the training. However, as a landscape research honeypot, it is hard to predict which features users will interrogate to establish authenticity. As research is conducted, the emulation can evolve and become more accurate, and eventually look authentic to everything that interacts with it.

A highly accurate device should operate exactly as the target device, but it should also fail like the target device. Just because an emulator appears to fail the same way as the target device, does not mean the emulator actually has to fail. It still can log inputs despite no response from the emulator. Or in a honeypot or cyber sensing application a better solution is to have the emulator re-manifest itself on a different network location to allow for another attacker to find it and exploit it again, thinking that it is a different device. This approach makes it look as though the original device went offline, corresponding with device failure, but does not actually take down the honeypot, wasting potential data collection opportunities.

### ***3.1.2 Robust Operation.***

It is important that any software-based device, especially those destined for research environments be robust. A robust emulation is able to operate when any common computer error occurs, such as when out-of-bounds or malicious input is entered or the device runs out of memory, even when the target device would have failed (under conditions such as buffer overflow, DoS attack, bad firmware uploaded, etc.).

Industrial control devices are notoriously sensitive due to lack of input sanitization and testing as well as non-robust implementation of communications protocols. Even seemingly benign network mapping tools can cause serious problems on industrial control

devices as discussed in detail in Section 2.2.4. This makes industrial control emulators more attractive than using a real device for training and research applications. As briefly mentioned in Section 3.1.1, if the target device fails, an accurate emulator would *appear* to fail similarly, but would continue to operate, or at least fail gracefully.

A significant advantage over real industrial controls is that the emulator cannot be made unusable, commonly referred to as being *bricked*. Even if the emulator locks-up, there is no flash bootloader to accidentally overwrite, or electronically erasable programmable read-only memory (EEPROM) to accidentally erase because the emulator is implemented purely in software, ideally on a dedicated piece of hardware.

### ***3.1.3 Cost Effective.***

An industrial control emulator must also be cost effective. In fact, it is likely that most use cases for an industrial control emulator (including honeypot applications) consider cost a main driver for emulated versus using real hardware. Considering that single PLCs can easily exceed \$10,000, even using a high-end desktop PC as the emulation platform would seem inexpensive, but good emulators can be implemented for much less than that.

Aside from monetary cost, the power budget may also be a driving factor in developing an industrial control emulator. Hundreds of PLC systems emulating a network of many honeypots (honeynet) may not be possible simply due to power requirements.

### ***3.1.4 Rapidly Configurable.***

Configurability is also an important factor in the design of an industrial control emulator, but the extent is application dependent. For honeypot applications, there is value in rigid, hard-coded designs that emulate only a single type of device. These designs allow for configuration of only device-specific parameters, such as networking features, integrated protocol attributes, or firmware versions, by simply tweaking a configuration file that defines the emulator's operation. A single model or brand of PLC popular in a specific industry can be chosen as the emulation target to conduct research specific to that

industry. Despite the significant amount of effort required to implement a honeypot specific to a single industry, once the design work is complete, the honeypot can be in service for years, since a single PLC model might be used in industry for decades.

Nevertheless, a highly configurable device which enables the emulator to look like a Siemens, Allen-Bradley, or Koyo PLC is even more valuable as an ICS honeypot because a single device can be used to conduct research across any industry. Furthermore, an emulator that can quickly and easily be configured to act like any arbitrary device has significant potential in all ICS field device emulator applications. However, the amount of research and development required for this type of rapidly-, highly-configurable device is multiplied by the number of devices it can emulate, which may make it impractical from a design standpoint.

#### ***3.1.5 Scalability.***

An emulator is scalable if it can be easily ported to new platforms, and interoperate effectively when large numbers of emulators (and possibly real devices) are chained together in a honeynet. Scalability is an important consideration for any network-enabled device and must be considered during the requirements stage. Because network-oriented services that typically run on an industrial control system are designed to run on an embedded computing platform on the ICS device(e.g., web configuration servers, industrial control protocols like Modbus or HAP, FTP servers and email clients), the emulation of such services does not require a significant amount of memory or computational power. However, if the intent is to emulate many control devices on a single PC or limited-resourced Gumstix computer, scalability might become a significant challenge while trying to emulate device timing accurately and consistently.

#### ***3.1.6 Record Activity.***

Record keeping is the heart of an industrial control honeypot. Though most industrial control systems do not provide this feature at a field-device level, it is invaluable to conduct

attack research because complete and accurate records are required to fully dissect an attack. Though logging can take place either locally on the device, remote logging via some type of distributed service such as Syslog is superior because it can be used to instantly alert researchers that a device is being accessed, a feature also critical for cyber sensing applications.

The effectiveness of a logging mechanism is measured by the accuracy of the log entries, and its ability to keep up with the activity that is occurring on the honeypot. Considering that a typical device fingerprinting tool generates over 200,000 packets when performing a port and services scan on a device, the logging mechanism must ensure that it works efficiently but correctly in the sense that no entries are missed [Lyo09].

## **3.2 Implementation of the PLC Emulator**

This section describes the emulator designed and implemented for this research thesis. The tool is a PLC emulator designed primarily as a SCADA landscape research, and as a result, seeks to optimize cost, accuracy, scalability, efficient logging, and to a limited extent, configurability.

### ***3.2.1 Custom Design Overview.***

The PLC emulator is implemented on the 600MHz Gumstix Overo Earth single board computer running embedded Angstrom Linux distribution 3.0.0 with a Tobi Duo network expansion board to provide two 10/100 Ethernet connections. The Linux kernel is built with iptables and Netfilter NFQUEUE modules to provide necessary Linux firewalling and user-space network packet functionality. Iptables (pronounced "I-P-tables") is the standard Linux firewall that is available for all Linux distributions. The NFQUEUE kernel module allows packets to be pulled out of the Linux kernel and *queued* into user-space processes.

The target PLC is comprised of the Koyo DirectLogic 405 CPU, D4-08ND3S Digital Input module, D4-08TR Digital Relay module and the H4-ECOM-100 Ethernet Communications Module to provide the PLC with network connectivity. This hardware



configuration is chosen because of its low cost, networking functionality, and the work done by DigitalBond's Project Basecamp that demonstrated significant security vulnerabilities including an automated method to crack the device password [PeP09, Wig12].

To provide acceptable performance as well as reduced development time, the emulator has portions written in both C and interpreted Python modules. The emulator consists of seven simultaneously running user-space processes plus the Linux iptables firewall module to provide three network services to users. The three emulated services are (1) a web service on TCP port 80 that provides a configuration interface, (2) a Modbus TCP service on TCP port 502, and (3) a Host Automation Products protocol (HAP) service on UDP port 28784, consistent with the target PLC.

### ***3.2.2 Process Layout and Implementation.***

In addition to the seven custom-designed user-space processes, five Linux command-line scripts are used to configure, start, and stop these processes correctly. In addition to the user-space processes, the Linux iptables user-space program is critical to the operation of the emulator [Ayu10]. The iptables application, which is available as a Linux kernel module, allows implementation of a kernel firewall whose operation and properties are defined through a user-space interface [Eyc12]. In addition to iptables' ability to accept or drop arbitrary packets based on their source, destination, or content, iptables can pull both incoming and outgoing packets out of the kernel TCP/IP stack into user-space processes for deeper packet inspection or packet mangling through what is known as an *NFQUEUE target*. This allows for precise packet characteristic matching to identify arbitrary packets to allow, drop, or modify them in any way, even before or after the packet has been to the kernel TCP/IP stack for routing.

### 3.2.3 *Emulator Configuration and Process Control Scripts.*

There are a total of two configuration scripts that establish the correct operating environment for the implemented emulator, and three process control scripts that start and stop the emulator processes. This section defines these scripts.

#### 3.2.3.1 *iptables\_rules.sh.*

This configuration script establishes the necessary iptables rules that filter and queue the packets, which enter and leave the emulator. Iptables configuration rules build on each other, and therefore the order of rules is important. Table 3.1 shows the iptables rules described in the order they appear in *iptables\_rules.sh*.

In Table 3.1, the *queue* number (i.e., "queue number 3") refers to a designated userspace program to which iptables sends packets. These userspace programs are shown in Figure 3.1 and discussed later in this Section. This feature is enabled by including the NFQUEUE module into the kernel at compile time. Marking packets is another feature of iptables. Packets that meet rule criteria can be *marked* with an arbitrary number that remains with the packet as it traverses across the remaining iptables rules. Marks help ensure certain rules do not get run more than once. The actual rules in iptables syntax are available in Appendix C.

#### 3.2.3.2 *configure\_emulator.sh.*

This configuration script sets global OS network settings that establish the operation of the Linux kernel network stack. The values applied to these parameters in the *configure\_emulator.sh* script are temporary. That is, they return to their original default value after operating system is reset. Parameters configured and their values are shown in Table 3.2.

Table 3.1: Iptables rules in the order they appear in *iptables\_rules.sh* and their description.

<b>Rule 1:</b>	"For all incoming packets on eth0 that have not been marked with 0x02 send them to queue number 3"
<b>Rule 2:</b>	"For all incoming TCP packets on eth1, with TCP destination port 22 (SSH), accept"
<b>Rule 3:</b>	"For all incoming TCP packets on eth0 with destination port 80, send them to queue number 1"
<b>Rule 4:</b>	"For all incoming TCP packets on eth0 with destination port 1, send them to queue number 1"
<b>Rule 5:</b>	"For all incoming TCP packets on eth0 with source port 502 (Modbus), accept"
<b>Rule 6:</b>	"For all incoming TCP packets on eth0 with destination port 502 (Modbus), accept"
<b>Rule 7:</b>	"For all incoming TCP packets on eth0 that do not match a previous rule, reject with TCP reset"
<b>Rule 8:</b>	"For all incoming UDP packets on eth0 with destination port 28784 (HAP), accept"
<b>Rule 9:</b>	"For all incoming ICMP packets on eth0, send them to queue number 1"
<b>Rule 10:</b>	"For all incoming UDP packets on eth0 with destination port 1, send them to queue number 1"
<b>Rule 11:</b>	"For all incoming UDP packets that do not match a previous rule, reject with ICMP port unreachable"
<b>Rule 12:</b>	"For all outgoing TCP packets on eth0 with destination port 80, that have not been marked with 0x1, send them to queue number 2"
<b>Rule 13:</b>	"For all outgoing TCP packets on eth0 with destination port 80, that have been marked with 0x1, accept"
<b>Rule 14:</b>	"For all outgoing UDP packets on eth0 with destination port 28784 (HAP), that have not been marked with 0x1, send them to queue number 2"
<b>Rule 15:</b>	"For all outgoing UDP packets on eth0 with destination port 28784 (HAP), that have been marked with 0x1, accept"
<b>Rule 16:</b>	"For all outgoing packets on eth0, set the TTL to 255"
<b>Rule 17:</b>	"For all outgoing TCP packets on eth0, send them to queue number 3"

Table 3.2: Temporary settings configured in *configure\_emulator.sh* and their description.

Parameter	Modified via	Description	Value
mtu	"ifconfig" command	The Maximum Transmission Unit is used to control the maximum segment size (MSS) of the TCP segments. The MTU equals the MSS plus 40 bytes.	552
hw ether	"ifconfig" command	Defines the Ethernet address of the Internet facing interface to be within the vendor's prefix range.	00:e0:62:60:46:25
icmp_ratelimit	/proc/sys/net/ipv4/ icmp_ratelimit	Defines the wait time between ICMP replies from the interface.	0
tcp_sack	/proc/sys/net/ipv4/ tcp_sack	Controls the use of TCP Selective Acknowledgments in TCP connections. Configuration is either enabled (1) or disabled (0).	0
tcp_window_scaling	/proc/sys/net/ipv4/ tcp_window_scaling	Controls the use of TCP window scaling on TCP connections. Configuration is either enabled (1) or disabled (0).	0
tcp_timestamps	/proc/sys/net/ipv4/ tcp_timestamps	Controls the use of TCP timestamps on packets from the kernel stack. configuration is either enabled (1) or disabled(0).	0

### 3.2.3.3 *start\_emulator.sh*.

*Start\_emulator.sh* is the single starting point for the emulator from which all other scripts and processes are started. This script first executes the two configuration scripts (*iptables\_rules.sh* and *configure\_emulator.sh*), builds and runs the C programs, and then

starts the Python modules. This process control script is used when changes are made to program header files (*.h* files) to alter certain parameters (such as the *device ID*, used to identify log entries for a specific honeypot in a the remote log).

#### **3.2.3.4 *start\_emulator\_no\_make.sh.***

This process control script is identical to the *start\_emulator.sh* script, but does not build the C programs. This saves time during test execution because building executables natively from C source on the Gumstix can take about a minute.

#### **3.2.3.5 *stop\_emulator.sh.***

This process control script stops all processes and flushes the iptables rules. It does not, however return the parameters changed by *configure\_emulator.sh* back to their original values.

### **3.2.4 *Custom User-space Processes.***

Of the seven custom processes, six involve network packet manipulation for recognizing different types of queries and responding appropriately. As a result, these processes revolve tightly around the iptables firewall (the seventh process is used to generate simulated input/output data to/from the device to act like a real PLC, discussed in a Section 3.2.4.4). Figure 3.1 depicts the entire routing process of packets in the PLC emulator system through the iptables firewall and Linux kernel. This diagram gives a full view of all processes to clarify the complex path a single query-response packet interaction takes through the emulator system.

#### **3.2.4.1 *infilter.o.***

The purpose of the *infilter.o* process is to identify Nmap OS fingerprinting packets, and deal with them according to a complex set of rules. This separate process is necessary because the level of filtering and mangling required to identify and handle fingerprinting packets reaches far beyond the functionality that iptables is able to provide.

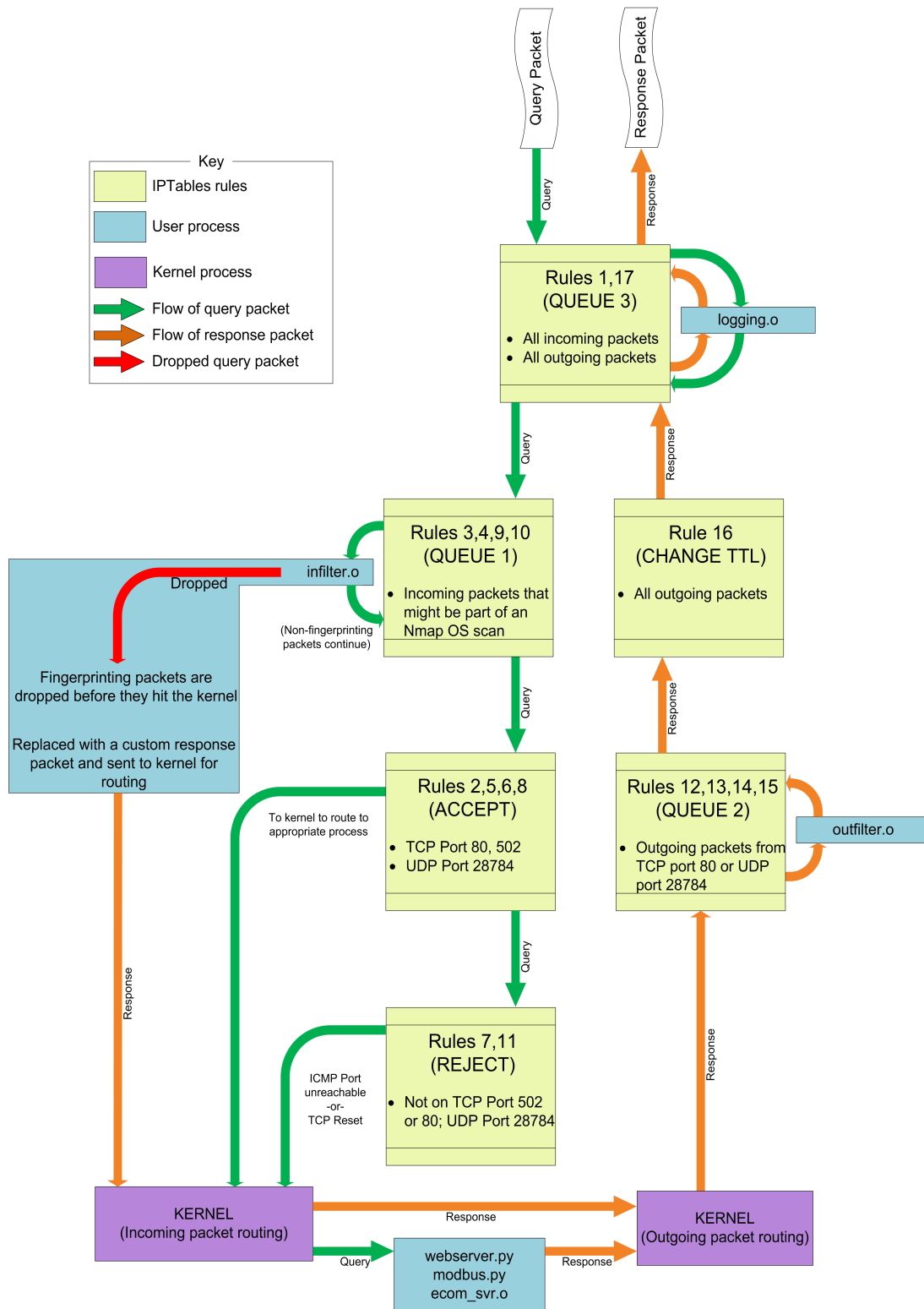


Figure 3.1: Routing of packets in the PLC emulator system. If the packet matches the listed iptables rules, the packet will be diverted to the side path, if the packet does not match the rule, it continues to the next iptables rule until a match is made, or until it reaches the end of the iptables chain, in which case the packet will be rejected.

The *infilter.o* process is written in C and receives packets from iptables NFQUEUE target queue number 1 (according to iptables rules 3, 4, 9, and 10) before they are sent to the Linux kernel TCP/IP stack for routing. Its functionality is depicted in Figure 3.2. If the received packets are identified as OS fingerprinting packets, they are dropped to prevent them from continuing deeper into the Linux kernel stack. Based on the contents of the dropped fingerprinting packet, *infilter.o* crafts a new packet from scratch to send back to the scanning PC. This new packet contains appropriate Ethernet, IP, and TCP or UDP headers to ensure it is correctly routed back to the scanning PC, but incorporates scan response data from the actual target PLC to fool the Nmap scanner into thinking the emulator is the target PLC. This functionality is sometimes called packet *mangling*. If the query packets are not dropped by *infilter.o*, two response packets would be sent to the scanner, one from the Linux kernel stack, and one from *infilter.o*, which is highly undesirable. If the packet is not identified as an Nmap OS probe scan packet, the packet is accepted by *infilter.o*, and continues on its way to the next iptables rule.

#### **3.2.4.2 *outfilter.o*.**

The purpose of the *outfilter.o* process is to alter parameters in the IP packet header that are not configurable otherwise. Though iptables has the capacity to change certain OSI layer 2 fields such as the IP TTL field, many fields such as the IP ID cannot be changed by iptables. Therefore, *outfilter.o* is required for modification of arbitrary header fields in outgoing packets. The functionality of *outfilter.o* is depicted in Figure 3.3.

The *outfilter.o* process is written in C and receives packets from iptables NFQUEUE target queue number 2 (according to iptables rules 12 and 14) before they enter the Linux kernel TCP/IP stack routing mechanism. This process increases the accuracy of the emulation by changing header fields to match those of the target PLC. To match functionality of the target PLC, the *outfilter.o* process only changes the IP ID of the packets to zero. The iptables rules match output packets destined for the querying computer

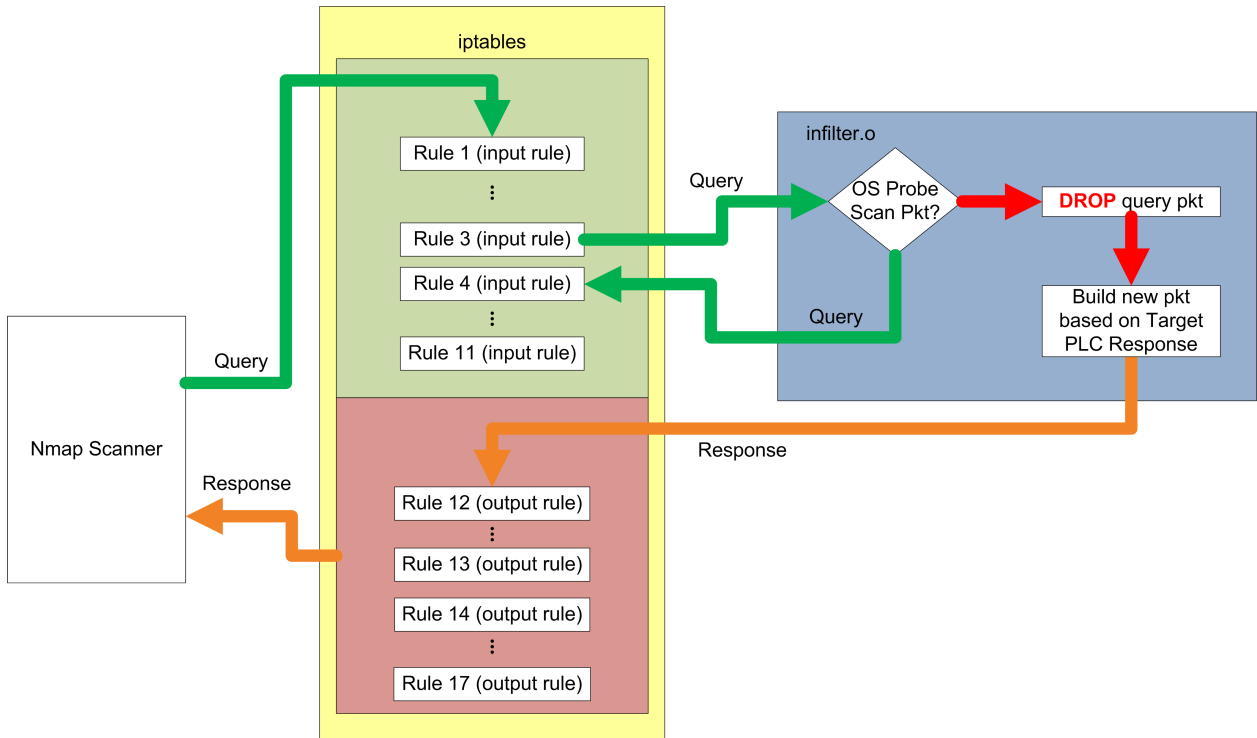


Figure 3.2: Example packet flow for Nmap OS probes for both query packets (green) and response packets (orange). The upper (green) portion of iptables contains input rules that match query packets, while the lower (red) portion contains output rules that match response packets.

originating from the *webserver.py* or *ecom\_svr.o* processes where responses are created from queries. If the output packets match rule 12 or 14, the IP ID is changed to zero and the packets are re-injected back into iptables starting with the next rule. The functionality of this processes is scalable to change any byte of any outgoing packet (not just the IP ID) to match the output of any target PLC.

### 3.2.4.3 *ecom\_svr.o*.

The purpose of the *ecom\_svr.o* process is to receive HAP protocol queries from a user on UDP port 28784, and generate responses to those queries in the same way as the target PLC. The functionality of *ecom\_svr.o* is depicted in Figure 3.4. This is the core of



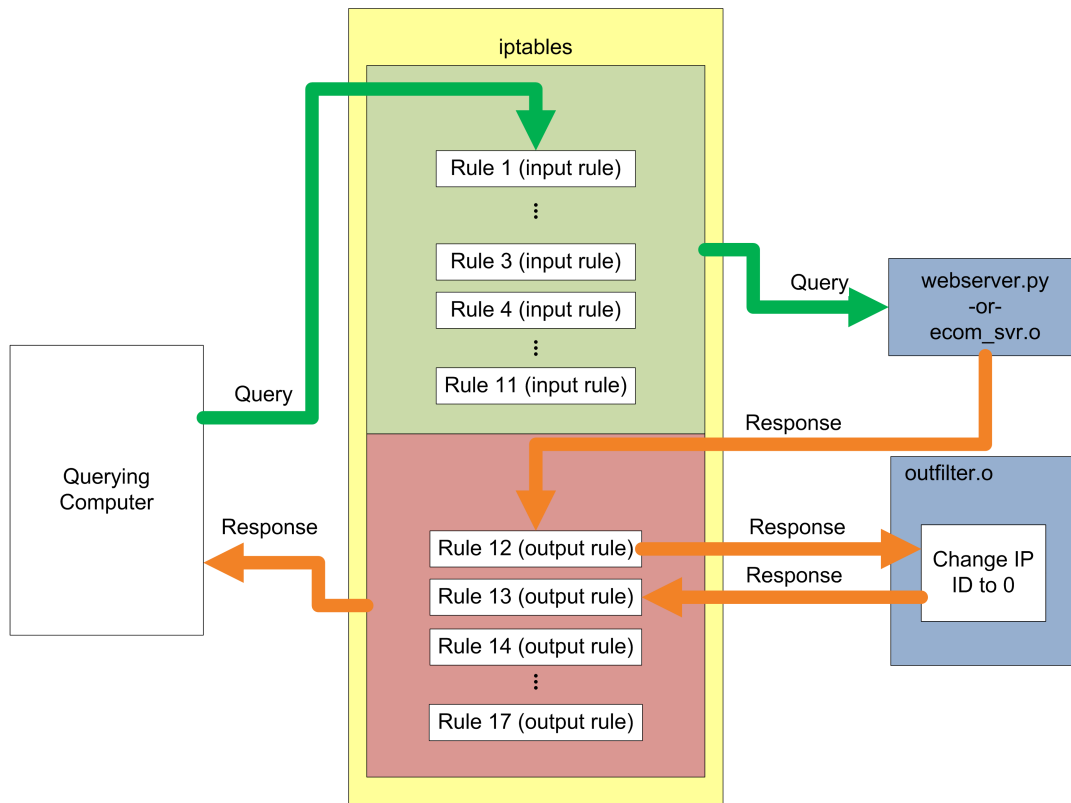


Figure 3.3: Example packet flow for a packet that has its IP ID changed by *outfilter.o*.

the industrial protocol interface and defines all of the protocol functionality for the HAP industrial protocol. Process development requirements for deep protocol dissection, reverse engineering, and low-level memory manipulations make this process the most complex, and warrant further description. Though many standard functions of the protocol are implemented, it is only a subset of the total capability of the HAP protocol. A complete list of the functions that are implemented is located in Appendix E.

The *ecom\_svr.o* process is written in C using the Libpcap Application Programming Interface (API) to receive packets, and Libnet API to send packets. An initial implementation of this processes was developed in Python using the Scapy network packet module, but early testing on the Gumstix platform showed that this configuration was too

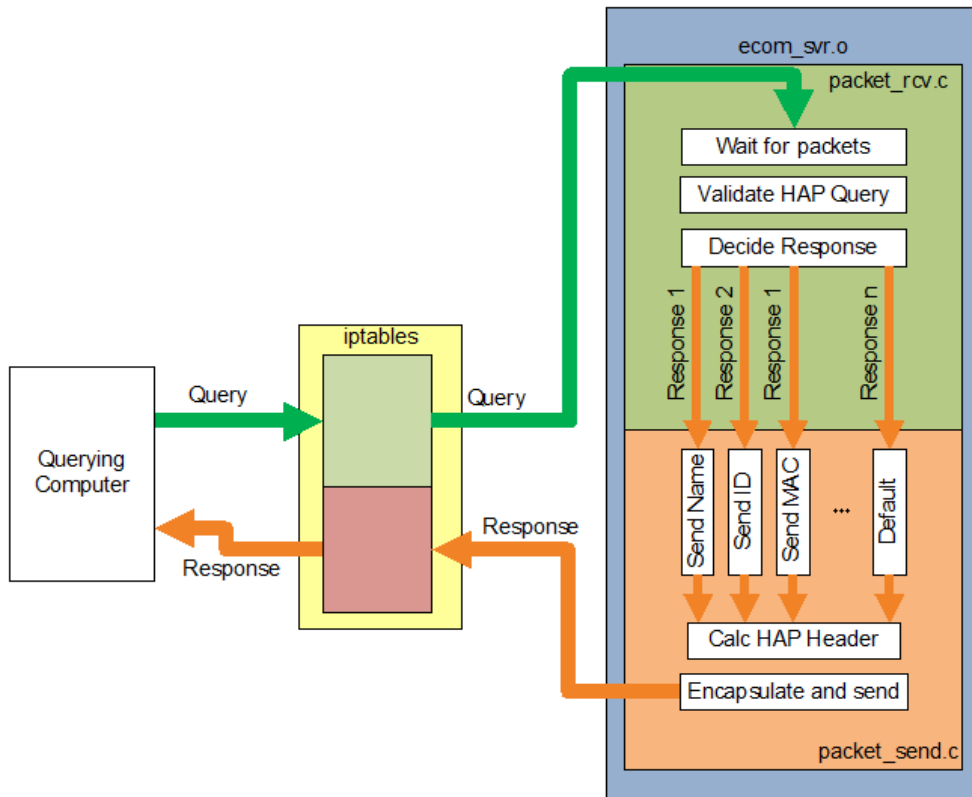


Figure 3.4: Example packet flow for a query and response from *ecom\_svr.o*. Query packets (green) are received by the process from the iptables firewall rules. Appropriate responses (orange) are formed by the process and sent back into the iptables firewall to be sent to the querying computer.

slow, causing the querying software on the user workstation to timeout. The early decision to use the Libpcap and Libnet libraries over standard Linux UDP sockets is arbitrary, and the *ecom\_svr.o* process could easily be modified to utilize Linux UDP sockets if the Libpcap or Libnet APIs are not available.

To aid in the development of the HAP emulator, the source for an API from Host Engineering must be obtained [Hos12]. The API (written in C) is intended for developing applications that communicate with HAP systems, such as the target PLC. This functionality of this software had to be reversed to make it communicate with HAP software

interfaces. The most important piece of information obtained from the API source is the *HAP.h* header file that included data structure definitions used throughout the target PLC.

The acquisition of the API source code enables rapid development of the HAP emulator, but does not trivialize its implementation. The HAP protocol is used on all Koyo PLCs to encapsulate several other industrial protocols (described in Section 2.2.2) over traditional IT infrastructure, giving it a great deal of flexibility, but also adds to the importance of a thorough understanding of the protocol. Deep protocol dissection using packet captures of real HAP traffic are tedious but critical to troubleshooting and understanding the protocol well enough to create an accurate emulator.

The *ecom\_svr.o* process is compiled from two separate source files that divide the core network functions. The *packet\_rcv.c* source file uses the Libpcap API to receive query packets (that have arrived via iptables rule 8), validate them, and decide what the appropriate response to the query should be. Once *packet\_rcv.c* decides a response it passes execution to functions in the *packet\_send.c* source file. The *packet\_send.c* functions then create responses from scratch including custom HAP checksums and sequence numbers. The *packet\_send.c* process then uses the Libnet API to encapsulate the HAP protocol data into UDP/IP response packets, and send them out of UDP port 28784 where they eventually arrive at the iptables output rules, and back out to the querying computer.

***User Interfaces to the ecom\_svr.o(HAP protocol emulator).*** To access the functionality of the HAP protocol there are three tools considered. Two are manufacturer's tools, NetEdit3 and DirectSOFT5 Programming, and the third is a non-standard tool to exploit vulnerabilities in the protocol, Metasploit. Despite the fact that these three interfaces to the HAP protocol emulator are the focus of this research, any software that follows the HAP industrial protocol specification should be able to access the emulator.

*NetEdit3* . This tool is used to view and set parameters related to the communication and operation of the device, and is available free from the target PLC manufacturer [Hos12].

A screenshot of the NetEdit3 tool displaying three devices, two of them emulated, is shown in Figure 3.5. The real (target) PLC has Ethernet address that ends in 23. The *ecom\_svr.o* process that runs the HAP protocol emulator is intended to respond to queries in the same manner as the target PLC. The NetEdit3 tool communicates with devices on UDP port 28784 through both broadcast and addressed IP traffic depending on the parameter being accessed. This tool is chosen as a primary focus because of the straightforward interface it provides to the target PLC and emulator.

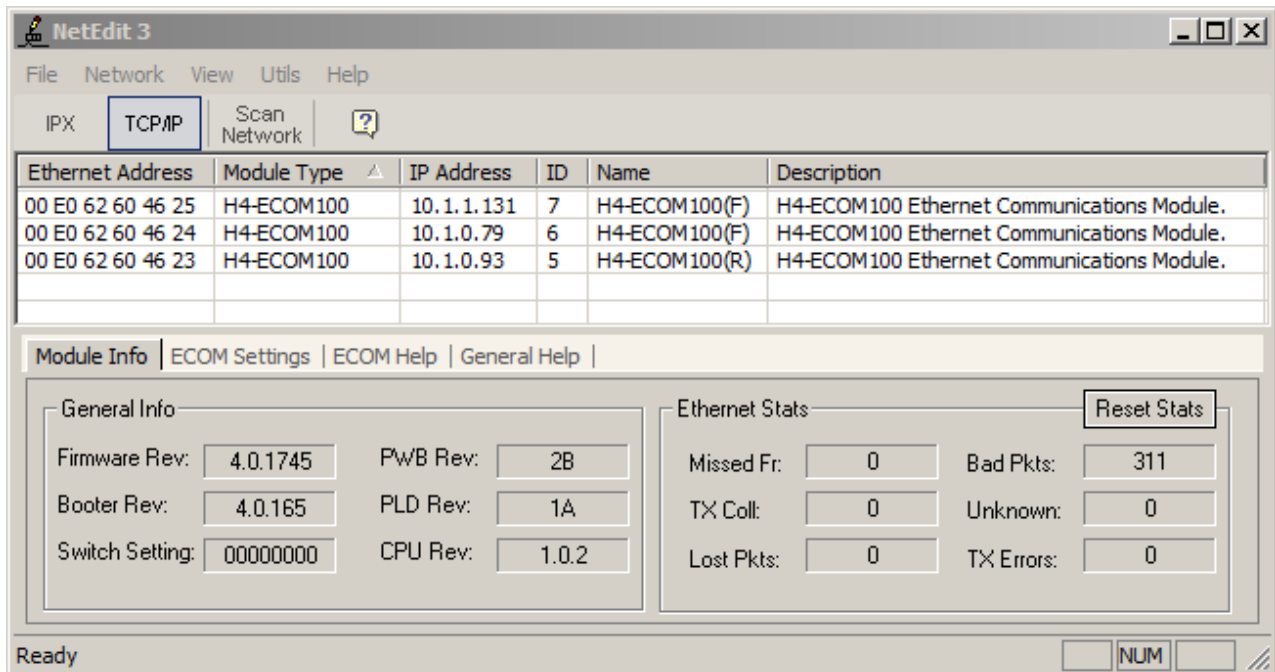


Figure 3.5: NetEdit3 interface communicating using the HAP industrial protocol. Shown are three devices, one real PLC (ID 5) and two emulated PLCs (ID 6, 7). The emulated PLCs are also denoted with an 'F' in the *Name* column for easy identification ('F' for fake, 'R' for real).

*DirectSOFT5.* The primary programming interface to Koyo brand PLCs, including the target PLC, is the DirectSOFT5 package. The manufacturer provides the DirectSOFT5 software free for limited use from its website [Aut12c]. This software allows direct access

to the device memory (which may be password protected) enabling a user to read and write arbitrary memory segments, view status of PLC switches and coils, and download and upload ladder logic programs. There are hundreds of different commands that can be sent to the device using this software, but only a few of the most critical are implemented in the emulator. To support the superficial level of accuracy, the emulator can be locked and unlocked with a 7-digit password and allows the ladder logic program to be downloaded from the device, exactly like the target PLC. The emulator also supports viewing a dynamic representation of the ladder logic that displays the constantly changing state of the PLC.

The dynamic representation feature of the emulator makes it look as though the emulator is actually controlling a process as the switches, coils, and timers in the ladder logic view change. Figure 3.6 and Figure 3.7 show screenshots of DirectSOFT5 being used to view the changing status of the control elements on the emulator. The fields highlighted are coils and switches that are currently active. The dynamic functionality is controlled by the *output\_simulator.py* process discussed in Section 3.2.4.4, and is readily configurable.

*Metasploit.* The final interface to HAP industrial protocol is the Metasploit exploitation tool [Met12]. This is a popular, open-source, extensible tool used for exploiting vulnerabilities in computer systems. It is used as a penetration testing tool to identify and correct vulnerabilities in IT systems. The use of Metasploit is extending beyond traditional IT however, and security researchers are actively developing Metasploit modules to exploit vulnerabilities in SCADA devices as well. A screenshot of the Metasploit module successfully obtaining the password from the emulator device is shown in Figure 3.8. The target Koyo PLC is one such device that has a Metasploit exploit available that uses brute force to crack the HAP protocol password that would normally be entered through the DirectSOFT5 interface giving full, unrestricted access to the device [Wig12]. The application of the device password is described in Section 3.2.4.3. The password vulnerabilities, listed with ICS-CERT in January 2012 include weak

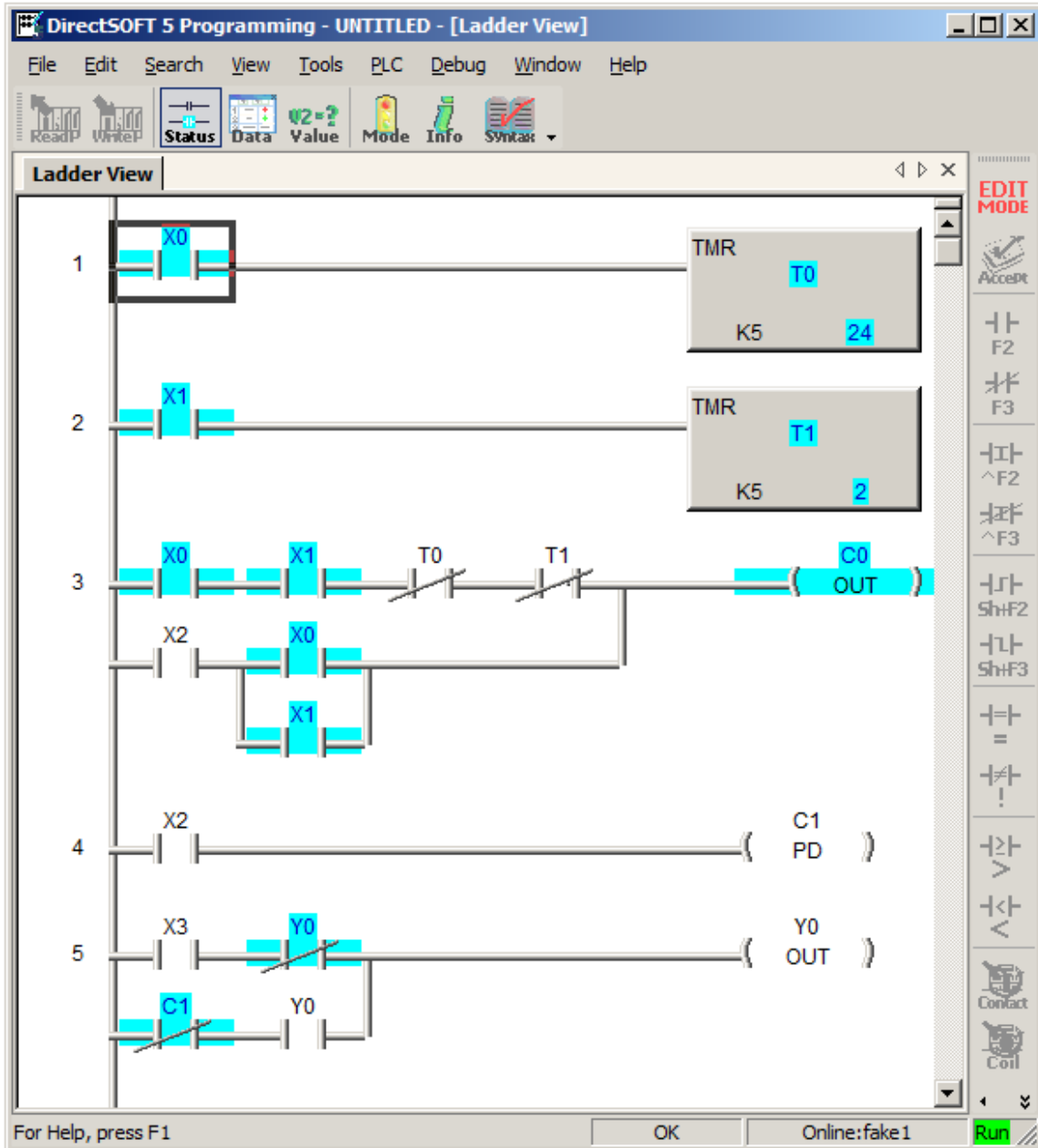


Figure 3.6: DirectSOFT 5 programmer interfacing with the HAP protocol emulator. When the Status button is depressed, the emulator displays a dynamic view of the current state of the PLC ladder logic.

authentication and unlimited attempts with no lockout period [Dhs12]. The vulnerabilities are corrected with firmware version 4.0.1776 dated 26 March 2012. Firmware versions 4.0.1735 and 4.0.1745 are confirmed to have this vulnerability. The vulnerability of the

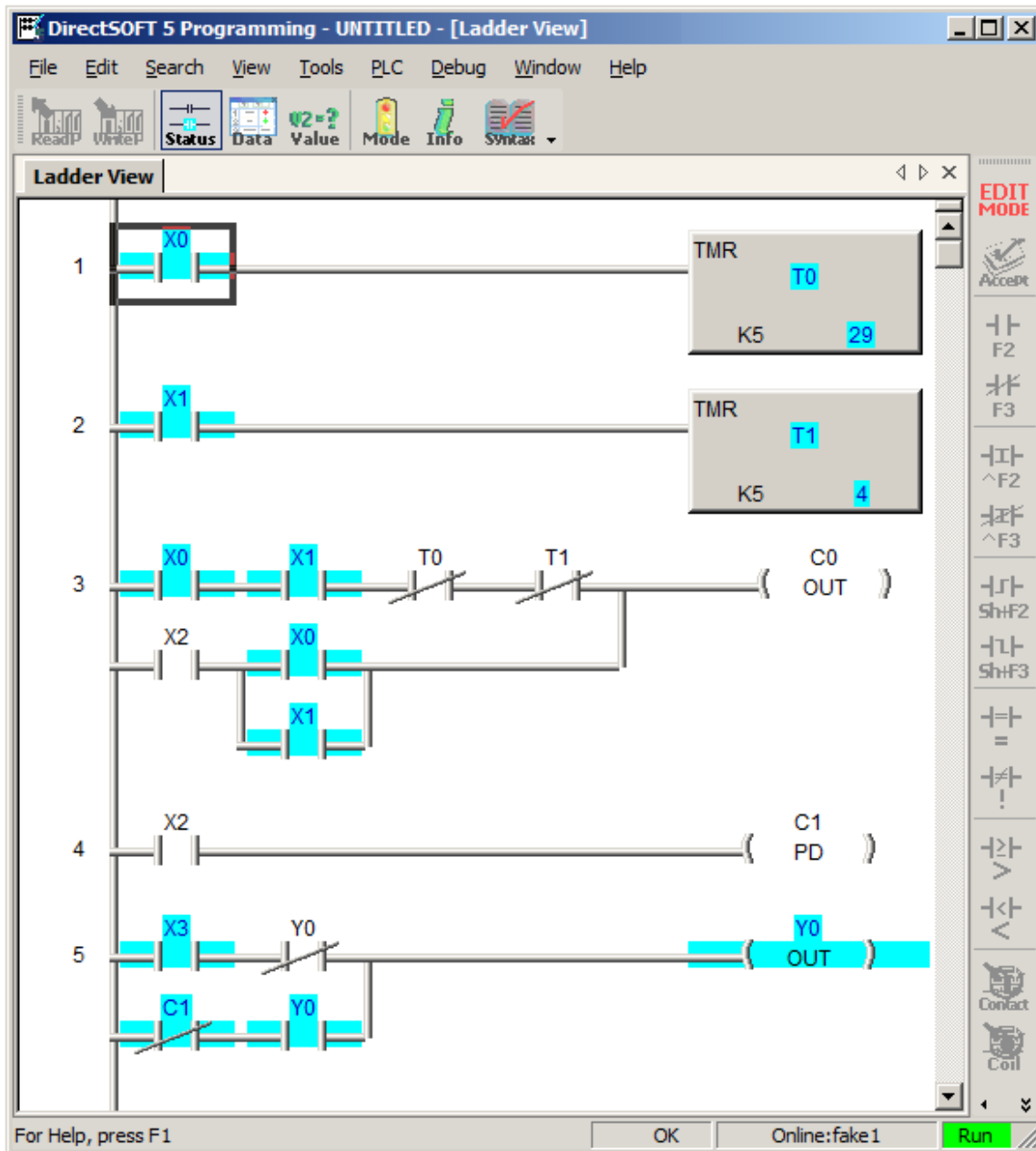


Figure 3.7: Status of emulator ladder logic a few moments later. Observe the alternate state (highlighting shows an electrically closed connection).

target PLC with firmware version 4.0.1735 is implemented on the emulator to ensure that the emulated device fails in the same way as the target PLC. Though this software is not

the most current, it is feasible for outdated, insecure software to be loaded on a PLC, as discussed in Chapter 2.

```

Module options (auxiliary/scanner/scada/koyo_login):

  Name          Current Setting  Required  Description
  ----          -
  PREFIX        A                yes       The prefix to use for the password (default: A)
  RECV_TIMEOUT  3                no        Time (in seconds) to wait between packets
  RHOSTS        10.1.0.79       yes       The target address range or CIDR identifier
  RPORT         28784           yes       The target port
  THREADS       1                yes       The number of concurrent threads

msf auxiliary(koyo_login) > exploit

[*] 10.1.0.79:28784 - KOYO - Checking the controller for locked memory...
[*] 10.1.0.79:28784 - KOYO - Controller locked; commencing bruteforce...
[+] 10.1.0.79:28784 - KOYO - Found passcode: A0000322...
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(koyo_login) >

```

Figure 3.8: Metasploit exploit software brute-forcing the password on the emulator. Metasploit found the password to be A0000322, and will unlock the device.

#### 3.2.4.4 Dependencies of *ecom\_svr.o* (HAP protocol emulator).

In addition to the executable, there are several configuration files and a helper executable, *output\_simulator.py* used to make the emulator more accurate, configurable, and scalable. By removing as many configuration options as possible from the executables and into dedicated files, settings can be changed without the need to re-compile the emulator binary files.

***config.txt.*** This configuration file is used to keep track of stateful parameters such as the device name, ID, firmware version, and password. In total, there are 437 different parameters that can be manipulated to give the emulator the desired characteristics. The HAP protocol emulator reads and writes all parameters to *config.txt*. Therefore, many *config.txt* files can be made ahead of time with different system configurations and selected at a later time to give the emulator any personality just by replacing the *config.txt* file.



To provide consistency, *config.txt* is also used by the *webserver.py* process. When asked to serve or change data, *webserver.py* also consults *config.txt* to ensure changes made with one interface are immediately readable by the other interface.

***ccmdictionary.txt* and *kseqdictionary.txt*.** These files are dictionaries of query-response fields that are used often by the emulator based on two different protocols, DirectNet (aka CCM), and K-Sequence. DirectNet is an open industrial protocol used by Koyo devices to read and write memory values on the target PLC discussed in Section 2.2.2. On this particular target PLC, the DirectNet protocol (encapsulated in the HAP protocol) is used to exchange CPU identification information to NetEdit3 and DirectSOFT5, as well as provide the dynamic status representation described in Section 3.2.4.3, Figure 3.6, and Figure 3.7. K-Sequence is a proprietary protocol used by Koyo devices to read and write memory values on the target PLC. On the target PLC, the K-Sequence protocol (also encapsulated in the HAP protocol) is used to exchange large amounts of data quickly, such as the ladder logic program. The dictionary files can be thought of as a way to emulate different memory locations on the PLC. That is, when a query is received, the emulator reads or writes to the file as if it were memory on the target PLC.

Both the *ccmdictionary.txt* and *kseqdictionary.txt* files contain a "query = response" format that *ecom\_svr.o* uses to decide which message to send in response to a query for the protocol. For example, when *ecom\_svr.o* receives a query that contains a CCM message of "1e01010132", it responds with the CCM payload of "00000b" as dictated by the entry in *ccmdictionary.txt*. An excerpt from *ccmdictionary.txt* is provided in Figure 3.9, showing this "query = response" format. Though not shown, *kseqdictionary.txt* works exactly the same, but the queries and responses are much longer.

***output\_simulator.py*.** The small *output\_simulator.py* helper process is written in Python and changes several response values in *ccmdictionary.txt* in specific ways to simulate changing memory values. Since *ccmdictionary.txt* is used to emulate memory

```
[CCM_READ]
1e01010132 = 00000b
1e01010133 = 000000
1e01010333 = 000007
1e01810133 = 000000
1e04010031 = 000000000000
1e01777736 = 60000
1e010f0036 = 00005a
1e06ee0f31 = 0000000000000000
1e01840132 = 000000
```

Figure 3.9: Excerpt from *ccmdictionary.txt*

locations, the responses modified by *output\_simulator.py* correspond to the locations in memory on the PLC that keep track of the current status of its coils, switches, and counters. The way that *output\_simulator.py* modifies *ccmdictionary.txt* is not arbitrary. The values specifically reflect valid operation according the ladder logic that is stored on the emulator. That is, the simulated values follow the logic to represent real functionality of some real control process. By altering this process, the emulator can appear to be controlling any type of system desirable for the particular emulator application.

Even though the values of the current state of the coils, switches, and counters are stored in *ccmdictionary.txt*, the actual ladder queries the emulator for its ladder logic program, *ecom\_svr.o* simply retrieves the values from *kseqdictionary.txt*. Therefore, to send back a different ladder logic program to DirectSOFT5, values for a different ladder logic program must be recorded, and placed into *kseqdictionary.txt*. Different methods for accomplishing this are left for future work and discussed in Section 6.4

### 3.2.5 *logging.o*.

The purpose of the *logging.o* process is to send a copy of all packets arriving and departing from the Internet facing interface of the emulator to a remote Syslog server via the LAN facing interface on the emulator. Iptables has a built-in logging capability, but only

logs the packet headers, and not the complete packet. This is not suitable for maintaining a complete record of activity on the emulator required for honeypot applications, therefore, *logging.o* is required to log all bytes of every packet coming from, or going to the emulator. The functionality of *logging.o* is depicted in Figure 3.10.

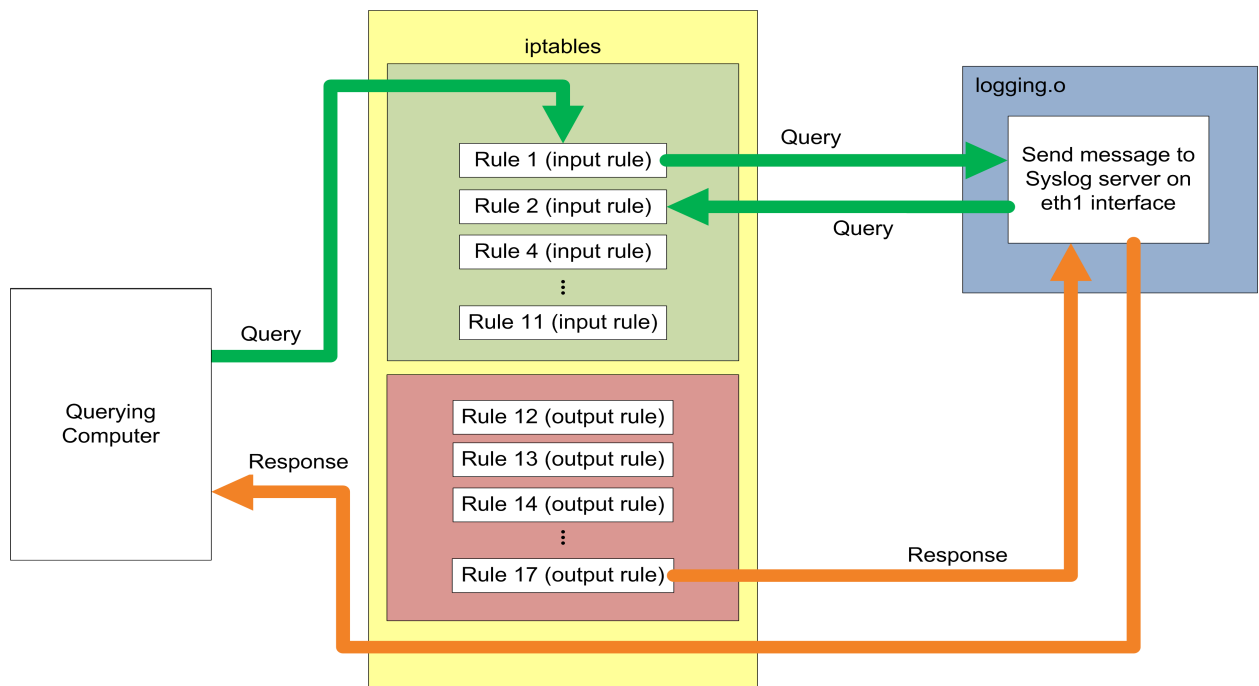


Figure 3.10: Example packet flow packet logging both query packets (green) and response packets (orange) in *logging.o*.

The *logging.o* process is written in C and receives packets from the iptables queue number 3 at two different times in a query-response interaction (according to iptables rules 1 and 17). According to iptables rule 1, packets are sent to queue 3 before they are sent to the Linux kernel TCP/IP stack for process routing. Similarly, according to iptables rule 17, packets are sent to queue 3 after they leave the Linux kernel TCP/IP stack for routing back to the querying computer, but before they are transmitted on the wire. After the packets are logged by *logging.o*, the packets are re-injected back into iptables starting with the

next rule, or in the case of rule 17, are sent to the kernel for transmission to the querying computer.

### 3.2.6 *webserver.py*.

The purpose of the *webserver.py* process is to emulate the HTTP web server functionality of the target PLC. The webserver interface on the device allows for reading and writing of many of the same industrial control settings and communication parameters as the HAP protocol interface, and more. The webserver interface is valuable to a user because no special tools (such as NetEdit3) are required to set or view the parameters; any web browser can be used to access the website. As a universal portal to the target device, the emulated webserver service implemented by *webserver.py* is critical to the accuracy of the PLC emulator. The functionality of the *webserver.py* process is depicted in Figure 3.11.

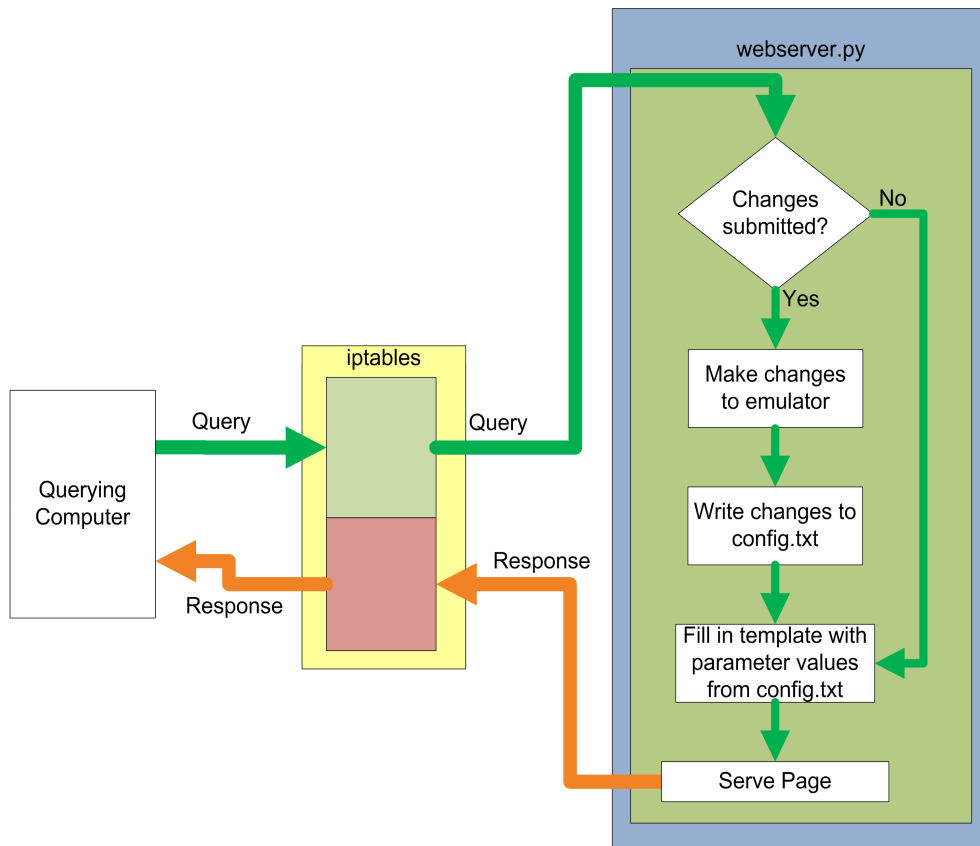


Figure 3.11: Example query-response flow for a web request from *webserver.py*.

Python is a natural choice for implementation of the web server because of its extensive collection of modules that make implementation of a generic web server easy. However, to accurately emulate the target PLC's web server, extensive modifications are made to the base HTTP web server. One example of a required modification is the number and format of the HTTP error messages. On the target PLC no error messages are sent to the browser—the webserver simply times out (not robust). The *webserver.py* process suppresses these messages and overrides other imported classes to create a custom webserver environment. The ease of changing these types of operations, as well as the existing Python runtime environment, make a Python webserver a good choice for industrial control emulators over other web servers like Apache. A screenshot of the main webpage served by the emulator is shown in Appendix E.

The *webserver.py* process uses template .html pages to serve static pages to the web browser. This strategy allows a single template for each webpage whose fields are filled in just before the page is sent back to the web browser.

Similar to *ecom\_svr.o*, only select functionality of the webserver is implemented into the emulator. A complete list of all functions implemented, including those implemented by *webserver.py* is located in Appendix D. Though not all available functions are operational, every form and entry field on the website is stateful. For example, although the email functionality of the target PLC is not implemented in the emulator, the email fields such as the email server address, sever port, and timeout values can be saved and retrieved later.

### **3.2.7 *modbus.py*.**

The purpose of this Python process is to emulate generic Modbus protocol functionality in the industrial control emulator. It is a direct integration from a previous research effort conducted by Dustin Bermen in 2012 [Ber12]. His Master's thesis titled "Emulating Industrial Controls System devices Using Gumstix Technology" contains the implementation details of the *modbus.py* process.

### **3.3 Incorporation of Design Considerations**

The PLC emulator is designed primarily as a tool for studying the SCADA/ICS attack-landscape as a ICS honeypot. An emulator that accurately implements communication protocols and services has applications in laboratory research and educational environments as well (discussed in Appendix A.1), but this design remains focused on implementation of a SCADA/ICS landscape research honeypot. As a result, all design considerations discussed in Section 3.1 play a significant role in the implementation.

#### ***3.3.1 Accuracy.***

To develop an accurate emulation that responds to service queries in the same manner as the target PLC, it is first necessary to decide how many layers of accuracy are necessary to feel authentic. The Modbus service is not considered because this research focuses on higher-level application protocols. Therefore, only the web and HAP protocols are evaluated for accuracy based on the tools that are frequently used with the target PLC. Five levels of accuracy emulated– (1) superficial accuracy, (2) packet accuracy, (3) timing accuracy, (4) scanning tool accuracy, and (5) attack tool accuracy. In other words, the emulator feels authentic when a user:

1. looks at the website or uses a manufacturer tool to view HAP data,
2. looks at packet bytes associated with the standard types of web or HAP queries,
3. measures the timing associated with queries and responses,
4. conducts an OS fingerprint scan on the device to see that the results tell the same story as the webpage, and
5. executes a publicly-available exploit for a known vulnerability on the device.

The two emulated services, web and HAP, are subdivided into standard, and non-standard types of interactions. The emulator then provides accurate responses for these four different query request types to feel authentic: (1) standard web request type (browser), (2) non-standard web request type (Nmap scan), (3) standard HAP request type (NetEdit3), and (4) non-standard HAP request type (Metasploit module), defined as follows:

1. *Standard Web* queries are interactions with the web server which is listening on TCP port 80. The queries represent normal, within-bounds requests from an Internet Explorer web browser.
2. *Non-standard Web* queries are OS fingerprinting scans using the Nmap tool that attempt identification of the device based on the response to non-standard TCP, IP, and ICMP packets sent to the device. By default, Nmap chooses the lowest open TCP port on the device to perform its tests on, and therefore will always pick TCP port 80 unless told otherwise.
3. *Standard HAP* queries are interactions with the HAP industrial protocol server which is listening on UDP port 28784. These queries represent normal, within-bounds requests from the free manufacturer tool, NetEdit3.
4. *Non-Standard HAP* queries are attacks on the HAP protocol security measures using the *koyo\_login* Metasploit module to exploit vulnerabilities found by Project Basecamp.

The emulator is designed to mimic responses to each type of request as accurately as possible when compared to the target PLC.

The HAP service provides two additional levels of superficial accuracy beyond the Metasploit attack. After logging into the device using the password from the Metasploit attack, the DirectSOFT5 software downloads firmware from the emulator (first additional level) through the *ecom\_svr.o* process. When the user attempts to view the current status of coils and switches, the emulator provides a dynamic view of the ladder logic as its values fluctuate as depicted in Figure 3.6 and Figure 3.7 (second additional level). Being superficial levels of accuracy, they are subjective and difficult to quantify and therefore are not tested in this research.

Among all the emulator design considerations, accuracy is the most difficult to measure and quantify. Chapter 4, describes the method used in this research for quantifying the accuracy of the emulator implementation.

### **3.3.2 Robustness.**

Robustness is also an important consideration for the design. Since the device is implemented completely in software, there is minimal risk to *bricking* the device to the

point of un-usability. The target PLC has several DoS vulnerabilities that render the webserver unusable for a period of several minutes as described in the January 2012 ICS-CERT notification [Dhs12]. These vulnerabilities include webserver buffer overflow and resource exhaustion. The device is protected against the known and reproducible vulnerabilities of the target PLC which including the resource exhaustion vulnerabilities. However, not enough information is known about the buffer overflow conditions to reproduce them on the target PLC and therefore, emulate its functionality. Because the emulator is implemented on a different architecture (presumably in a different manner), it is very unlikely that the emulator suffers the same vulnerability. While this is good news for emulator robustness, it means that the emulator does not accurately represent the buffer overflow vulnerability.

In support of accuracy and robustness, the emulator *appears* to fail under the conditions of the resource exhaustion vulnerability, but does not actually fail. To make a more accurate emulation, the emulator is configurable via the *webserver.py* Python script to block for several minutes, just like the target PLC. With additional research and improvements, the device can be made to emulate the buffer overflow vulnerability in a similar, configurable way.

### **3.3.3 Cost Considerations.**

Existing methods of implementing a SCADA/ICS honeypot involve using real PLCs, (discussed in Section 2.5) which dramatically increases cost. The design implemented for this thesis research lowers these costs significantly though the use of the Gumstix platform. The cost and features of the Gumstix device are described in Section 2.4. Because the emulator software is easily copied to other Gumstix or Linux computers, a single design can be used to emulate an entire network, compounding the cost savings as the number of devices increases.



The Gumstix platform is a cost-effective solution with respect power budgets as well [Gum12c]. Because of the low power consumption of an embedded platforms, the Gumstix computer can even run on batteries, making the device portable and rapidly deployable. Additional cost effective solutions are suggested in the Section 6.1.

#### **3.3.4 Configurability.**

The more configurable the emulator, the more portable to other applications it will be. However, this implementation only emulates the DirectLogic 405 PLC. From a single-device standpoint, the emulator is very configurable. Every stateful web or HAP field can be modified and customized for the particular research or educational application *du jour*.

#### **3.3.5 Scalability.**

Because the emulator is designed primarily as a honeypot and can be potentially integrated into a honeynet, both scalability and code portability are important considerations. The choice to use Linux is a result of selecting to uses the Gumstix single-board computer. Using Linux gives the device the ability to run on many hardware configurations due to it being highly customizable. Furthermore, the use of Linux allows the integration of the very powerful iptables firewall.

#### **3.3.6 LoggingAbility.**

Logging is a critical function for an ICS field device emulator applied as a honeypot. This implementation uses the second Ethernet port on the Gumstix device to send out-of-band logging to a remote Syslog server, an open-source Linux based program for large-scale enterprise logging. By using a robust, scalable logging service, potentially hundreds of emulators creating log entries as part of a large SCADA/ICS honeynet could be managed with a few entries in a Syslog configuration file.

Because of its importance to SCADA/ICS attack-landscape research, the logging ability of this emulator implementation is directly measured and assessed. Chapter

4, describes the method used for quantifying the logging ability of the emulator implementation, and the results of the experiments are analyzed in Chapter 5.

### **3.4 Summary**

The design of an industrial control emulator is complex. Depending on the intended application, many design elements including accuracy, robustness, cost, configurability, scalability and record keeping must be considered. The emulator implementation for this research effort incorporates all of these considerations for applications in exploratory SCADA/ICS attack-landscape research into design of the PLC emulator based on the emulation target, the Koyo DirectLogic 405 PLC.

## IV. Experimental Methodology

### 4.1 Goals

As discussed in Chapter 3, accuracy and authenticity are important design considerations for an emulator designed for exploratory Internet research as a honeypot. The goal of this experiment is to quantify accuracy and logging capability of the emulator to assess its usefulness as an attack-landscape research device. Six questions about the accuracy and logging performance of the emulator on the Gumstix and PC platforms are addressed:

1. To what extent are the responses from standard queries accurate at the packet level for the emulator compared to the target PLC?
2. To what extent are the responses from non-standard queries from scanning and attack tools accurate for the emulator compared to the target PLC?
3. To what extent are the timing of responses for standard and non-standard queries accurate for the emulator compared to the PLC?
4. To what extent are the queries and responses to and from the emulator logged?
5. To what extent does request frequency affect the accuracy and timing performance of the emulator?
6. How significant is the impact of running the emulator on the Gumstix platform versus the PC platform for all five of the above categories?

### 4.2 Approach

This is accomplished by methodically dividing the goal into testable and quantifiable experiments. Chapter 3 previously defined accuracy at five levels: superficial, packet, scanning, attack, and timing. Therefore to quantify accuracy, the experiments include scenarios designed to test each of these levels independently.

The emulator is divided into two services, web and HAP. Both are subdivided by the types of interactions the emulator is designed to handle: (1) web standard, (2) web non-standard, (3) hap standard, (4) hap non-standard. These reflect interactions with the

emulator via web browser, Nmap, NetEdit3, and Metasploit respectively. The services and query types are summarized in Table 4.1.

Table 4.1: Two services broken into four query types.

<b>Service</b>	<b>Query Type</b>	<b>Tool</b>
Web	Standard	Browser
	Non-Standard	Nmap
HAP	Standard	NetEdit3
	Non-Standard	Metasploit

The approach to test accuracy of the emulator on both the Gumstix and PC platforms is to send identical queries to each of the Gumstix, PC, and target PLC configurations in different experimental scenarios to cover all aspects of accuracy and logging. A comparison of the responses to these queries is made between the target PLC and both emulator platforms. Additionally, the completeness of the number of log entries is assessed to determine the ability of the emulator to successfully log all query-response interactions. To ensure complete coverage for accuracy testing, Table 4.2 maps the five levels of accuracy to the services in Table 4.1. In this experiment, packet-level accuracy is considered to be a more rigorous determination on authenticity than superficial (or visual) accuracy, and thus superficial accuracy is not measured.

It is hypothesized that the industrial controls emulator implemented on both the Gumstix and PC platforms is (1) accurate to both standard and non-standard queries at the five levels when compared to the target PLC, and (2) able to log all packets to a remote logging service. A comparison of target PLC and emulated system quantifies the accuracy of the emulator and forms the basis for a qualitative determination on authenticity,

Table 4.2: Levels of accuracy mapped to emulator services.

		Levels of Accuracy					
		Superficial	Packet	Timing	Scanning	Attack	
Request Types	Web	Standard(browser)	X	X	X		
		Non-Standard(Nmap)			X	X	
	HAP	Standard(NetEdit3)	X	X	X		
		Non-Standard(Metasploit)			X		X

as describe in Chapter 3. That is, the more accurate the emulator, the more authentic interactions with it will feel to the user.

### 4.3 System Boundaries

The system under test (SUT), shown in Figure 4.1, is the PLC Control System consisting of the industrial network interface connecting the PLC device to the Internet, the Logging Interface where the emulator connects to the logging server, and the PLC device itself. The PLC device component is the component under test (CUT) and is controlled by the PLC device Type parameter. Depending on the experimental scenario, the CUT is at one of three levels, (1) the target PLC to define baseline metrics, (2) the emulator running on the Gumstix platform, or (3) the emulator running on the PC platform.

### 4.4 Services and Outcomes

There are four services the system provides as discussed in Chapter 3, three Internet services and a logging service. The Internet services include (1) a web service that enables configuration of the device through an interactive interface, (2) a HAP protocol service that allows direct access to module memory including programmable logic (ladder logic) of the device as well as configuration, and (3) a generic Modbus service used to control functionality of the device. The logging service (4) provides a record of incoming and

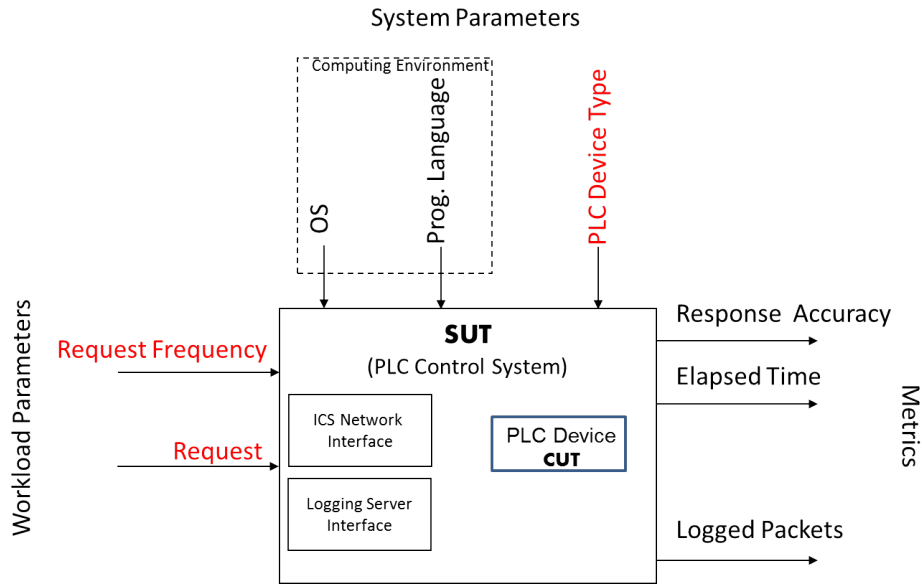


Figure 4.1: PLC control system

outgoing packets (sent to and from the three Internet services) in emulator experiments. Only the web, HAP, and logging services are considered because the scope of this research concentrates on higher application-level protocols. The process that runs the Modbus service was developed as an independent research effort [Ber12]. The validity of this software module is not tested beyond general functionality before integration into the ICS field device emulator.

The logging service is provided while the PLC device component is at either of the two emulator levels, the Gumstix platform or the PC platform. There is no logging on the target PLC device, but logging is a critical function of an ICS field device emulator designed for honeypot applications, as described in Section 3.3.6.

#### 4.5 Parameters and Factors

Two types of parameters are considered in this experiment, Workload Parameters and System Parameters. Both types of parameters affect the performance of the SUT, as measured by the *metrics*. Varied parameters are called *factors*. *Levels* are the values or

conditions at which each factor is held during an experiment. In this section, experimental elements are introduced and justified as they apply to this SUT. A summary of chosen factors and their levels is listed in Table 4.4.

#### **4.5.1 Workload Parameters and Factors.**

The workload is Internet traffic most likely to be sent to a real PLC by users in cyber sensing, laboratory, and educational applications. Since any traffic to a honeypot is suspicious as discussed in Chapter 2, any traffic and therefore any workload sent to the system is treated as malicious. There are two workload parameters based on how a user would submit workload requests to the SUT, *Request Type* and *Request Frequency*.

##### **4.5.1.1 Request Type.**

Depending on the user, different requests are made to the device. The Request Type parameter defines four different workloads corresponding to the four different query types sent to the system to represent different requests made by users to a standard PLC. These four request types are selected because they are the most likely ways a user would interact with the target PLC.

Request type is a factor in this experiment, and the levels are divided into four categories according to Table 4.1: (1) web standard workload type, (2) web non-standard workload type, (3) industrial standard workload type, and (4) industrial non-standard workload type using the following definitions.

**Web Standard Workload Requests.** The web standard workload request represents a typical workload any user could submit to the webserver. This type of request would be used in all emulator applications including cyber sensing, laboratory research, and education.

The specific workload request chosen is to display the device homepage. A screenshot of the homepage served from the emulator is available in Appendix E. On the homepage (*main.html*), the current configuration of the device is displayed with hyperlinks to other

pages where the configuration can be changed. This workload is chosen for the experiment because the *main.html* file is sufficiently large as to force TCP segmentation. The maximum segment size (MSS) of a TCP segment dictates the maximum amount of data that can be sent in a single TCP packet. The MSS on the target PLC is 512 bytes while the size of *main.html* is 1,932 bytes, this forces delivery of the page into four deterministic TCP packets. Additionally, the serving complexity of *main.html* is higher than other pages because of large numbers of memory read operations that pull information from the emulated device's configuration file to send in response to the query, as described in Chapter 3.

As an alternative to a web browser such as Internet Explorer or Firefox, an automated tool is created to generate web standard workload requests to obtain consistent results between trials and configurations. This tool, written in C, uses raw TCP sockets to open a connection to the CUT's webserver, send the HTTP GET request, accept data, and handle connection tear down. The tool is multi-threaded to allow an unlimited number open sockets to the webserver at a time. This enables many simultaneous connections without waiting for the previous GET request to have been handled by the CUT. Additional information about this custom workload generator is available in Appendix G.

***Web Non-standard Workload Requests.*** Beyond standard users, those interested in exploiting the device are more likely to challenge it with different tests of accuracy to determine its authenticity. A common approach is using a network scanner such as Nmap and observing what is returned. If the results are consistent with what are expected, authenticity is further established.

Using Nmap for the web non-standard workload request is a good choice because it is popular, open-source, and free. Additionally, the documentation available is excellent, which is a major enabler for the Nmap probe handling process (*infilter.o*) of the emulator described in Chapter 3. Because of the default configuration of Nmap, the first open port



found on a device is the port selected for OS fingerprinting probes. On the target PLC, this is TCP port 80, the webserver. The basic command issued to Nmap to conduct a single web non-standard workload request is:

```
nmap -sS -sU -p 1-65535 -T3 - -O -v - --max-parallelization 1
```

***Industrial Standard Workload Requests.*** Like the web standard workload request, the industrial standard workload request represents a typical request a user submits to the HAP protocol server on UDP port 28784. As described in Chapter 3, NetEdit3 can be used to submit standard workloads to the CUT on this interface. This type of request would be used in all emulator applications including SCADA attack-landscape research.

The specific request chosen is the request of the *Module Description* via the HAP protocol. This workload is chosen for the experiment because the CUT responds with the most bytes of any other HAP response (267 data bytes).

As an alternative to the proprietary NetEdit3 tool, an automated tool is also created to generate standard industrial workload requests to obtain consistent results between trials and configurations. This tool, written in C, uses raw UDP sockets to send the query and receive for the response. The functionality is similar to the custom workload generator developed for the standard web workload, and additional information is available in Appendix G.

***Industrial Non-standard Workload Requests.*** Again, non-standard requests are sent by those interested in exploiting the device. After a device has been identified, either through information found on the website or via an Nmap scan, publicly-available exploits may be attempted to verify authenticity of a device. A common tool used to execute exploits is the Metasploit exploitation framework. An exploitable vulnerability gives more authenticity to an emulated device by failing similarly to the target PLC

Metasploit is free, easy to use, and is ubiquitous among highly-skilled and lower-skilled hackers alike (because it is free and easy to use). Written in Ruby, the framework is

easily extended making the addition of new exploits by anyone straightforward, which is how the exploit for the target PLC was created [Wig12]. The module used is the *koyo\_login.rb* module. The module successfully exploiting the target PLC is seen in Figure 3.8. The exploit is executed through the Metasploit console:

```
$use auxiliary/scanner/scada/koyo_login  
$set rhosts 10.1.0.79  
$exploit
```

#### ***4.5.1.2 Request Frequency.***

The request frequency workload parameter defines how quickly workload requests are sent to the SUT. This metric is chosen because the rate at which the requests are sent by the workload generator is easily reproduced by attackers trying to determine the true identity of the device. The accuracy of the log generated by the SUT (if CUT is the emulator) may be affected as the time in between requests decreases. Additionally, the elapsed time and response rate metrics may be affected depending on the ability of the SUT to process responses as fast as or faster than the request frequency. These are both experimental questions outlined in Section 4.1.

Request Frequency is a factor in this experiment. All three devices are tested at two levels, *slow* and *PLC Break*. Depending on the level of the *Workload Request Type* factor, the standard and accelerated frequencies are different values. The levels chosen are the result of pilot studies conducted on the system for each of the target PLC, Gumstix emulator, and PC emulator platforms. These nominal levels are summarized in Table 4.3.

The *Slow* level is defined as the fastest rate that all three devices can handle successfully, without dropping requests. The *PLC Break* level is defined as the fastest rate that the target PLC can handle without failing to respond to queries. In addition to these levels, a level only performed on the emulators is called the *Emulator Break* level. To

Table 4.3: Selected nominal request frequency levels for all four types of frequency requests. These levels are a result of pilot studies.

	Slow(query/sec)	PLC Break(query/sec)	Emulator Break (query/sec)	
			PC	Gumstix
Standard Web	10	27.49	203.198	10.263
Standard HAP	100	382.203	2023.976	259.238
Non-Standard Web	T3†	T5‡	n/a	n/a
Non-Standard HAP	0.637	2.066	1328.643	2273.404

†timing: Standard T3 Nmap Speeds plus options: -max-parallelism 1 -max-scan-delay 5 -max-retries 2

‡timing: Insane T5 Nmap Speeds plus options: -max-scan-delay 1 -max-retries 2

NOTE: this is not the frequency at which the PLC breaks, simply a faster Nmap scanning profile.

determine these values, the emulators are driven to their own break frequencies to establish the maximum request frequency they can handle without failing to respond to queries.

As seen in Table 4.3, the Non-standard web request (Nmap scan) does not list specific frequency values, only that the default Nmap timing profile (T3) plus options, and the *insane* timing profile (T5) plus options are used. Because of the nature of the Nmap scan, the true timing and parameters are trivial, and beyond the scope of this research. Default-plus-option timing profiles are used to simplify the experiment since there are at least 10 variables that affect the frequency of probe packets sent by Nmap. Additionally, each Nmap scan takes several minutes, and attempting to find the exact point that the scans begin to fail is difficult and not productive to this research.

For the non-standard HAP workload, the *slow* level is the standard brute force password attack as executed using the default Metasploit speed. In Metasploit, there are no options to change the rate at which queries are made. Therefore, the tool *tcpreplay* [Syn12] is used to record and play back the Metasploit attack packets at arbitrary speeds to facilitate the *PLC Break* and *Emulator Break* request frequency levels. Again, pilot studies are used

to determine the actual break levels for each CUT. The selected levels shown in Table 4.3 are those used in the actual experiments.

#### 4.5.2 System Parameters and Factors.

Only the PLC device parameter is a factor in this experiment. Other parameters that are not being considered but acknowledged as potentially significant are the device architecture (Gumstix: ARM, PC: x86) and operating system (Gumstix: Angstrom 3.3.0, PC: Ubuntu 2.6.35)

##### 4.5.2.1 PLC Device.

The PLC Device parameter defines the device that is plugged into the system. The PLC Device parameter specifies which component under test (CUT) is in the system. The PLC Device parameter is a factor being considered in this experiment. The PLC Device parameter has three levels (1) target PLC, (2) emulated PLC on Gumstix platform, and (3) emulated PLC on PC platform.

Table 4.4: Summary of Factors and Levels.

<b>Factors</b>	<b>Levels</b>
Request Frequency <sup>†</sup>	Slow
	PLC Break
	Emulator Break
Request Type	Web Standard
	HAP Standard
	Web Non-standard
	HAP Non-standard
PLC Device	Target PLC
	Emulated on Gumstix
	Emulated on PC

<sup>†</sup>As assigned by Table 4.3 according to request types.

## 4.6 Performance Metrics

There are no standard objective metrics for measuring the effectiveness of an emulator to imitate its target device and therefore metrics are developed to quantify emulator accuracy. A summary of the performance metrics measured for each request type and corresponding levels of accuracy is shown in Table 4.5.

### 4.6.1 Packet Bytes.

The packet bytes metric measures the byte values of a response from the CUT to a query. Packet level accuracy is determined by how many packet bytes of an emulator response match the *expected* response bytes from the target PLC. In the SUT, packet bytes are measured for two request types, (1) the web standard workload, and (2) the industrial standard workload. Therefore, the metric is only measured during scenarios 1, 2, 2a, 5, 6, 6a, 9, 10, 10a, 13, 14, 14a, 15, 16, 16a, 17, 18, 21, 22 according to Appendix F.

The expected response of the emulator is defined as the response provided by the PLC Control System when the target PLC device is in the system as the CUT. A baseline response for the target PLC is measured for all other variable factors, and the response under those same conditions is measured with the emulators in the system as the CUT to determine the packet byte accuracy.

All deterministic bytes from the emulator response must be identical to the equivalent bytes in a response from the target PLC for the emulator response to be 100% accurate. Deterministic bytes in the Internet headers include parameters whose accuracy is considered not relevant, including fields such as the query source port number, initial sequence number, etc. The complete list of bytes considered non-deterministic in this experiment is available in E. If all deterministic bytes are not identical, then the accuracy is less than 100% for that query.

#### ***4.6.2 Nmap OS Fingerprinting Fields.***

The Nmap OS Fingerprinting Fields (hereafter referred to as Nmap fields) metric measures selected values found in the output of an Nmap OS Fingerprint scan. Scan level accuracy is determined by how many Nmap field values of a response from the emulator match the *expected* field values from the target PLC for same scan. This metric is only measured during the experiments that submit web non-standard queries to the CUT (scenarios 3, 4, 11, 12, 19, 20 according to Appendix F).

The fields considered are the Port Status, OS Guesses, Scan Details, TCP sequence Prediction (which is also reported as the SP field in the Scan Details section), IP ID Sequence Generation. The results of an Nmap scan with these fields highlighted is seen in Figure 4.2.

#### ***4.6.3 Successful Unlock.***

The successful unlock metric measures whether the Metasploit exploit is successful, or not. This metric is binary, either the device is successfully unlocked by the attack or it is not. Attack level accuracy is determined by the outcome of the Successful Unlock metric. This metric is only measured during the experiments that submit HAP non-standard queries to the CUT (scenarios 7, 8, 8a, 15, 16, 16a, 23, 24 according to Appendix F).

#### ***4.6.4 Response Time.***

The response time metric measures the time from when the first packet of the workload is sent from the workload generator to the CUT, to when the final packet of the response is sent from the CUT back to the workload generator. Timing accuracy is obtained by comparing the measured baseline response times of the target PLC to the measured performance of the emulator CUTs. This metric is measured during all experiments. This is a common metric in network analysis and is of particular interest for the emulator system because it is easy to measure by researchers, students, and attackers alike. The measurement of elapsed time or latency has been shown to be a fingerprintable

```

Not shown: 131067 closed ports
PORT      STATE      SERVICE
80/tcp    open       http
502/tcp    open       asa-appl-prot
28784/udp open|filtered unknown
MAC Address: 00:E0:62:60:46:23 (Host Engineering) OS Guess
Device type: specialized
Running: Koyo embedded
OS details: Koyo DirectLogic PLC
TCP/IP fingerprint:
OS:SCAN(V=5.21%D=12/15%OT=80%CT=1%CU=1%PV=Y%DS=1%DC=D%G=Y Scan Details
OS:062%P=x86_64-unknown-linux-gnu)SEQ(SP=2F%GCD=1%ISR=9A%11=4%11=K1%15
OS:=U)OPS(O1=M200%O2=M200%O3=M200%O4=M200%O5=M200%O6=M109)WIN(W1=800%W2=800
OS:%W3=800%W4=800%W5=800%W6=800)ECN(R=Y%DF=Y%T=FF%W=800%O=M200%CC=N%Q=)T1(R
OS:=Y%DF=Y%T=FF%S=O%A=S+%F=AS%RD=0%Q=)T2(R=N)T3(R=Y%DF=Y%T=FF%W=800%S=O%A=S
OS:+%F=AS%O=M109%RD=0%Q=)T4(R=Y%DF=Y%T=FF%W=800%S=A+%A=S%F=AR%O=%RD=0%Q=)T5
OS:(R=Y%DF=Y%T=FF%W=800%S=A%A=S+%F=AR%O=%RD=0%Q=)T6(R=Y%DF=Y%T=FF%W=800%S=A
OS:%A=S%F=AR%O=%RD=0%Q=)T7(R=Y%DF=Y%T=FF%W=800%S=A%A=S+%F=AR%O=%RD=0%Q=)U1(
OS:R=Y%DF=Y%T=FF%IPL=38%UN=D046%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE(R=Y%DF
OS:I=S%T=FF%CD=S)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=47
IP ID Sequence Generation: All zeros IPID Sequence Generation
Read data files from: /usr/share/nmap
OS detection performed. Please report any incorrect results at http://
Nmap done: 1 IP address (1 host up) scanned in 125.85 seconds
Raw packets sent: 131091 (4.721MB) | Rcvd: 131085 (6.292MB)

```

Figure 4.2: Example OS Fingerprint scan results. The values of the fields highlighted are measured by the Nmap OS Fingerprinting fields metric.

characteristic in other computer service emulator systems such as the popular open-source program Honeyd [FYC06].

#### 4.6.5 Logging Ability.

The logging metric counts the number of Syslog packets sent from the emulator to the remote Syslog server. Correct operation of the logging service requires that a Syslog entry be created on the LAN-facing interface for every packet *to* and *from* the emulator seen on the Internet-facing interface. Because capturing all emulator interactions is one of the most important functions, the logging metric is a critical measurement. This metric is measured during all experiments that submit queries to an emulator CUT(scenarios 1-16a according to Appendix F). Since the emulator logging is out-of-band (meaning log traffic

travels across a different communication channel than the workload), no timing rate is associated with the logging service. As long as all interactions are sent to the log allowing time for send and receive buffers to clear, it is not important how long it takes for them to get there. The method used to test this is described in Section 4.8.1.7.

Table 4.5: Performance metrics measured for each request type and corresponding levels of accuracy.

		Levels of Accuracy				Logging Ability	
		Packet	Timing	Scanning	Attack		
Request Types	Web	Standard(browser)	X	X		X	
		Non-Standard(Nmap)		X	X		X
	HAP	Standard(NetEdit3)	X	X			X
		Non-Standard(Nmap)		X		X	X
		Packet Bytes	Response Time	Nmap Fields	Successful Unlock	Logging	
<b>Performance Metrics Measured</b>							

#### 4.7 Experimental Design

A total of thirty query-response scenarios are divided into the four workload components. Based on an expectation of relatively low variability in all three metrics, this experimental setup is expected to result in 99% confidence intervals to determine performance differences. A partial factorial design is used for this experiment because the target PLC is only submitted to two request frequency levels, Slow and PLC Break. A complete list of all scenarios and factors is listed in Appendix F.



## 4.8 Evaluation Technique

### 4.8.1 Accuracy.

#### 4.8.1.1 Standard Web Workload Packet-Level Accuracy.

The selected web standard workload queries are GET requests sent to the CUT to retrieve the homepage, *main.html*. The response to these queries is measured by the Packet Bytes metric. Scenarios examined during the web standard workload accuracy trials are listed in Table 4.6. The baseline with the PLC Device at the Target PLC level is recorded in scenarios 17 and 18 according to Appendix F. For every query, exactly 2,372 bytes across 8 packets are sent by the CUT including the TCP connection setup, data exchange, and TCP connection teardown. Of these 2,372 bytes, 2,330 bytes are deterministic.

Table 4.6: Scenarios examined during the Web Standard Workload.

<b>Scenario</b>	<b>Request Frequency (queries/sec)</b>	<b>Platform</b>
1	10	PC
2	27.124	PC
2a	203.198	PC
9	10	Gum
10	27.124	Gum
10a	10.263	Gum
17	10	PLC
18	27.124	PLC

With the PLC Device factor at the Emulated PLC levels (scenarios 1, 2, 2a, and 9, 10, 10a according to Appendix F) the results are again recorded. A binary comparison (match or do not match) is then done between the deterministic bytes of these responses to the baseline measurements taken for the Target PLC. A quantitative assessment is then

made on packet-level accuracy for web standard queries. A depiction of the byte-by-byte comparison between target and emulated PLC for all 8 packets of a response is shown in Figure 4.3.

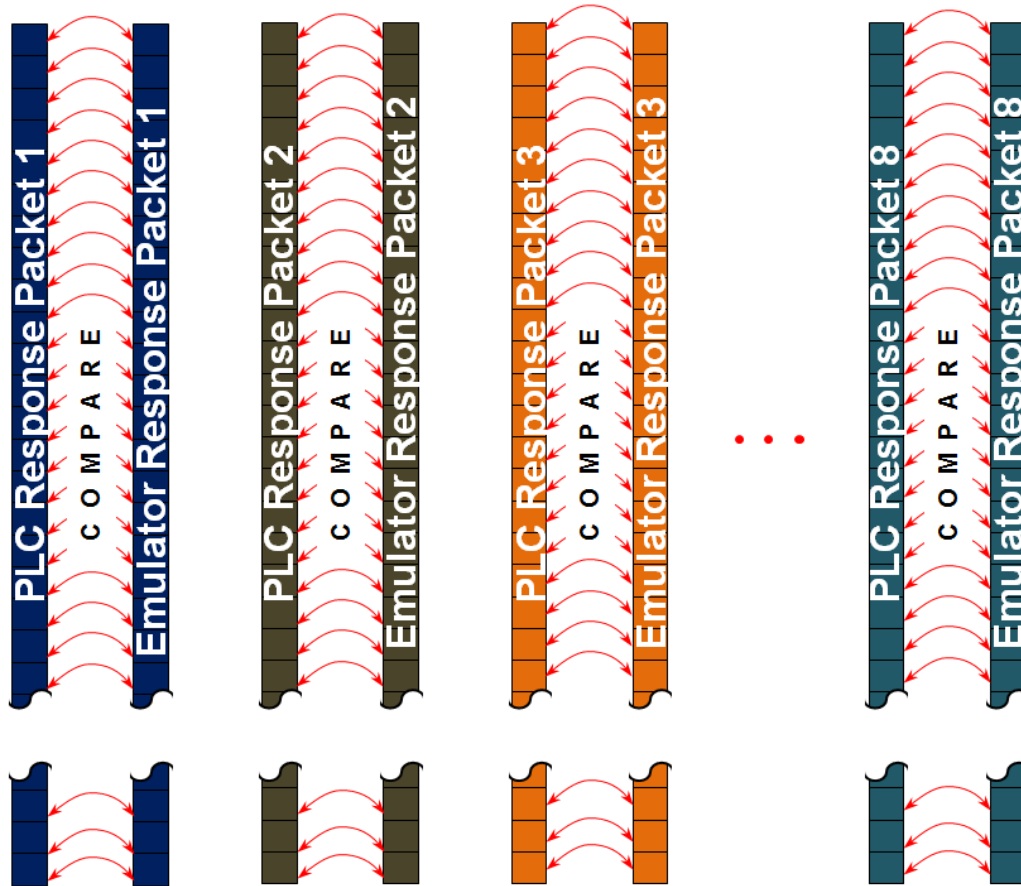


Figure 4.3: Example of byte-by-byte comparison of target PLC responses to emulator responses for a homepage request query. The CUT responds with exactly 2,372 bytes across 8 packets. 2,330 deterministic bytes are compared.

Because the response is always exactly the same for the same query, theoretically only one trial is necessary to perform an assessment of packet-level accuracy. However, the Response Time metric requires more trials. Therefore, with the ease of conducting trials using the automated workload generator, a large number of trials is conducted, significantly reducing the size of the confidence intervals for the Response Time metric, as well as

provide confidence in packet-level accuracy. A total of 1000 trials of the web standard workload are sent to each CUT. Based on an expectation of relatively low variability in packet byte accuracy and timing accuracy, this experimental setup, a 99% confidence interval (CI) is chosen to determine performance differences.

**4.8.1.2 Standard HAP Workload Packet-Level Accuracy.**

The HAP standard workload for the emulators is evaluated in exactly the same way as the web standard workload. Scenarios examined during the HAP standard workload are listed in Table 4.7. The baseline, with the PLC Device factor at the Target PLC level, is done in scenarios 21 and 22. For every query, exactly 309 bytes in 1 UDP response packet are sent by the CUT. Of these 309 bytes, 302 bytes are deterministic. With the PLC Device at the Emulated PLC levels (scenarios 5, 6, 6a, and 13, 14, 14a according to Table 4.7) the results are recorded and a binary comparison is again made to the baseline for 1000 trials.

Table 4.7: Scenarios examined during the HAP Standard Workload.

<b>Scenario</b>	<b>Request Frequency (queries/sec)</b>	<b>Platform</b>
5	100	PC
6	382.203	PC
6a	2023.976	PC
13	100	Gum
14	382.203	Gum
14a	259.238	Gum
21	100	PLC
22	382.203	PLC

#### 4.8.1.3 Nmap OS Scanning-Level Accuracy.

The web non-standard workload response is evaluated similarly to the packet-level accuracies just described, but Nmap fields are measured instead of packet bytes. Scenarios examined during the web non-standard workload accuracy trials are listed in Table 4.8. There are 113 fields in an Nmap OS Fingerprint scan across the five categories seen previously in Figure 4.2. The baseline, with the PLC Device at the target PLC level, is done in scenarios 19 and 20. With the PLC Device at the Emulated PLC levels (scenarios 3, 4, and 11, 12) the scan results are recorded and a binary comparison is done to the baseline. All Nmap fields from the emulators should match the baseline.

Table 4.8: Scenarios examined during the Web Non-standard Workload.

<b>Scenario</b>	<b>Request Frequency</b>	<b>Platform</b>
3	T3(+options)	PC
4	T5(+options)	PC
11	T3(+options)	Gum
12	T5(+options)	Gum
19	T3(+options)	PLC
20	T5(+options)	PLC

Five scan trials are done to ensure validity of test results. The lengthy duration of around 11 minutes for each scan and the complexity of the data output make it very difficult to compare the results of many trials, which must be done manually. Based on an expectation of relatively low variability in accuracy, this experimental setup is expected to result in 99% CI to determine performance differences in accuracy based on the five trials.

#### ***4.8.1.4 Metasploit Attack-Level Accuracy.***

The HAP standard workload response is evaluated simply by running the exploit under each test configuration, and verifying that the exploit is successful. Scenarios examined during the HAP non-standard workload accuracy trials are listed in Table 4.9. The workload input values show the *tcpreplay* playback multiplier needed to generate the nominal request frequency for the indicated scenario. Successes is determined using the DirectSOFT5 software; if the software asks for a password before downloading the ladder logic, then the device is locked, otherwise if the software begins downloading the ladder logic without asking for a password, the device is unlocked and the exploit was successful. For each experimental scenario, the passwords of all devices are set to the same arbitrary value.

Similar to the Nmap OS Scans described in Section 4.8.1.3, five trials are conducted. Based on an expectation of relatively low variability in attack-level accuracy, this experimental setup is expected to result in 99% CI to determine performance differences in accuracy based on the five trials.

#### ***4.8.1.5 Timing of Standard Requests.***

Timing for standard requests is evaluated from packet capture timestamps, and validated by outputs generated by the workload generators. The start reference for each trial begins with the initial query packet's timestamp and ends with the final packet of the response.

For the web standard workload, the clock for an arbitrary query starts with the first packet of the TCP handshake, and concludes with the final TCP ACK packet from the CUT. The HAP standard workload is connectionless UDP with only one packet for every query and one packet for every response. The clock starts with the timestamp of the query, and ends with the timestamp of the response.

This data is collected at the same time as the packet byte level accuracy metrics. Therefore, 1000 trials are conducted to provide a 99% CI for the response time for standard

Table 4.9: Scenarios examined during the HAP Non-standard Workload.

Scenario	Request Frequency (queries/sec)	Workload Input Value	Platform
7	0.637	0.071	PC
8	2.066	0.2304	PC
8a	1328.643	500	PC
15	0.637	0.081	Gum
16	2.066	0.266	Gum
16a	2273.404	480	Gum
23	0.637	1	PLC
24	2.066	3.2555	PLC

requests. Based on low expected variability in the timing data, 99% CI is a reasonable expectation based on the selected number of trials.

#### ***4.8.1.6 Timing of the Non-standard Requests.***

Timing for the HAP non-standard (Metasploit) requests is evaluated the same way as described in Section 4.8.1.5 for HAP standard workloads.

Timing for web non-standard requests is evaluated differently. Because the web non-standard (Nmap) queries are collections of many packets and responses, time is measured as the difference between the first packet sent by the workload generator and the last packet sent by either the workload generator or the CUT as recorded in the packet capture of the scan.

Because of the significant amount of time needed to conduct each test and low expected variability, a minimal number of five trials is conducted to obtain confidence intervals of 99% to compare response times of the respective non-standard workload scenarios.

#### **4.8.1.7 Logging.**

The logging metric records the number of log entries sent to the remote logging server on the LAN-facing network interface via a packet capture program running on the remote logging server. The number of packets sent is equal to the number of log entries generated. Simultaneously, a packet capture program is running on the workload generator recording all traffic going to and from the emulator. Every packet captured on the workload generator going to or from the CUT should have a packet generated on the logging server. The number of packets captured by each packet capture program is compared to make a quantitative assessment on the ability of the CUT to generate log entries.

Performing rigorous statistical analysis on logging is difficult because the data is pooled. When the number of log packets sent does not equal the number of query and response packets sent, there is no easy way of determining which specific packet was not logged. Therefore the analysis of the logging metric is limited to discussing the ratio of log entries made to expected entries.

### **4.9 Experiment setup**

To conduct this experiment, the operational system is implemented and measured according to the diagram in Figure 4.4. A Dell Latitude D630 with 2GB RAM, Intel Core2 Duo CPU, 2.00GHz Processor, running Linux Ubuntu 2.6.35 is used as the CUT that runs the emulator. This computer adds an additional Ethernet port via an SMC Networks USB to Ethernet stick model number SMC2290USB/ETH. A second computer is running Linux Ubuntu 3.2.0 used as both the workload generator and the response recorder. The computer is also a Dell Latitude D630 loaded with VMWare Player 4.0.2 with a Windows XP SP3 virtual machine to run the NetEdit3 and DirectSOFT5 HAP tools. The computer is also loaded with Nmap version 5.12 to generate the web Non-standard workload and receive the response, and Metasploit version 4.4 to generate the HAP non-standard workload and receive the response. The two custom workload generators for web standard and HAP

standard are called *htmlget\_mt* and *hap\_mt* respectively. Wireshark version 1.8.1 packet sniffer is used to measure the response accuracy and timing metrics. Instructions for setting up the emulator are found in Appendix I.

A third computer running Ubuntu Linux version 11.10 with Syslog server version 1.5.0 records the out-of-band logging from the emulated honeypot device, and is only used during measurements that involve the emulator CUT. This computer is also running Wireshark version 1.8.1 to record the number of logging packets sent to the logging server from the CUT.

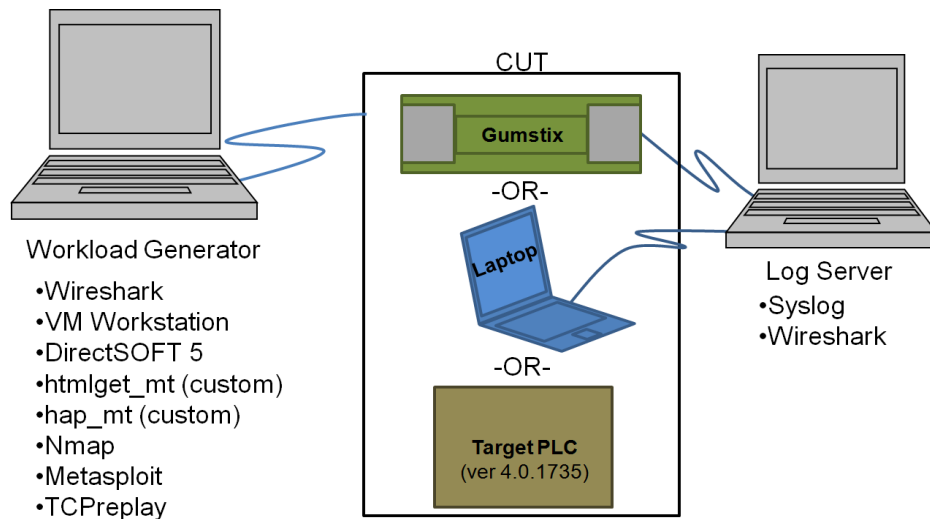


Figure 4.4: Experimental Setup.

The scope of this experiment is limited to a quantitative assessment of the emulator to the Koyo DirectLogic DL405 PLC with firmware version 4.0.1735 for the four types of workloads described, and the emulator's logging capabilities.

#### 4.10 Validation of Custom Tools

Because two custom workload generators are used to send and collect data from the CUT, a tool validation is done to ensure that the tools represent valid data representative of



what is sent by real tools. The workload generated by the custom web workload generator, *htmlget\_mt*, is compared to the workload sent by Internet Explorer version 9.0.8112 to ensure similar queries are generated. Similarly, the custom HAP workload generator, *hap\_mt*, is validated against the NetEdit3 manufacturer tool.

In addition to workload tools, several custom packet capture parsing tools are developed to automate packet and timing accuracy measurements described in Section 4.8 for the 1000 trials for standard workloads. Further details about the tools and their validation are found in Chapter 5 and Appendix G.

#### **4.11 Summary**

This chapter describes the methodology used to quantify accuracy and logging capability of the emulator. Four request types are used to test accuracy or the emulator on four levels: packet, scanning, attack, and timing. These request types are conducted at different request frequencies to determine the effects on accuracy and logging capability. All scenarios are conducted on the target PLC to establish baseline values for the selected metrics. The scenarios are then conducted on the PC and Gumstix measuring the same metrics, which are compared to the baseline and each other.

It is hypothesized that the Gumstix and PC emulators are 100% accurate at the packet, scanning, and attack levels for all request types and request frequencies. It is also hypothesized that the response times of the Gumstix and PC emulators for all request types and request frequencies are faster or equivalent to the target PLC. Finally, it is hypothesized that all packets going to and from the Gumstix and PC emulators for all request types and request frequencies are successfully logged.

## V. Results and Analysis

This chapter discusses the results of the experiments described in Chapter 4. The results show that the emulator on both the Gumstix and PC platforms are over 95% accurate for web non-standard workloads (Nmap OS Scans) and over 99% accurate for all other types of workloads. The logging capabilities of both platforms are also excellent, with 99.99% of all packets being logged by the PC and 97.21% of all packets being logged by the Gumstix when the results of all scenarios are combined. Though the timing-level accuracies for the PC platform are significantly faster than the target PLC, the results of the Gumstix platform significantly slower as defined by the 99% confidence intervals for the mean.

### 5.1 Validation of Custom Tools

The use of custom tools to generate the workload and parse the experimental data are necessary because of the large number of scenarios and trials. Two workload generators are used to produce the web and HAP standard workloads, *htmlget\_mt* and *hap\_mt*, respectively whose function is described in Chapter 4. Validation is accomplished by measuring the query and response using the custom web and HAP workload generators and comparing them to browser and NetEdit3 interactions, respectively. The details of the workload generators and their validation are in Appendix G.

In addition to custom workload generation tools, data parsers are necessary to parse the packet captures (.pcap format) collected for each experiment. Three custom parsers are developed: *web\_time\_parse.py* and *udp\_time\_parse.py*, for reading the packet timestamps, and *pcap\_accuracy\_parse.py* to compare responses of two .pcap files to similar workloads to determine differences in response bytes. The details of how the timing and accuracy parsers work as well as their validation are in Appendix G.

## 5.2 Packet Level Accuracy

Packet level accuracy only considers responses from the web standard queries and HAP standard queries of both the Gumstix and PC platforms. Each scenario is repeated 1000 times to ensure consistency in packet level accuracy. The metric used to measure the packet-level accuracy is the *percentage of packet bytes correct* which is a derived metric based on the packet bytes metric. The percentage of packet bytes correct is the percentage of packet byte values measured with the emulator in the system, compared to the baseline packet byte values measured with the target PLC in the system. Both the Gumstix and PC platforms are evaluated using this metric.

### 5.2.1 Gumstix Packet-Level Accuracy Results .

The results for packet-level accuracy as measured by the *percent packets bytes correct* metric of the Gumstix and percentage of packets logged are shown in Table 5.1. The results show that almost every byte of response is accurate on the Gumstix emulator for all 1000 of the standard web queries. There are a total of 2,432 bytes sent from actual and emulated PLC servers for a single standard web query for the *index.html* page seen in Appendix E. Excluding non-deterministic header fields also defined in Appendix E, there are 2,330 bytes across eight packets that should be identical. Of these total bytes, the Gumstix emulator differs by 5 bytes for a success rate of 99.79%. Four of these byte differences occur in the TCP header for three packets of the response and are a result of how the target PLC's TCP/IP stack is implemented. These results are confirmed for all successful trials at all three request frequency levels. For packet-level accuracy, there are no confidence intervals because there is no variance among the trials conducted.

The differences between the Gumstix emulator and target PLC responses are in the handling of the TCP Push flag and TCP congestion window and are not correctable outside of the Linux kernel in the current emulator implementation. Figure 5.1 shows an example of one of these differences manifested in packet 4 of the standard web response, indicated

Table 5.1: Packet level accuracy and logging results for the Gumstix for standard web and HAP workloads.

Scenario	Service	Request Frequency	Successful trials	% packet bytes correct	% packets logged
9	Web	slow	1000/1000	99.79%	100.00%
10	Web	PLC Break	319/1000	99.79%	100.00%
10a	Web	Gum Break	1000/1000	99.79%	100.00%
13	HAP	slow	1000/1000	100.00%	100.00%
14	HAP	PLC Break	636/1000	100.00%	76.96%†
14a	HAP	Gum Break	1000/1000	100.00%	100.00%

†This percentage is revised to 98.98% after accounting for the 364 queries not seen by the emulator.

by the arrow between packets. This example shows the difference in byte 0x2F, the TCP Flags field. Value 0x10 in the Gumstix response indicates that the TCP Acknowledgment (ACK) flag is set, while the PLC response shows value 0x18, indicating that the TCP ACK and TCP Push (PSH) flags are both set. The presence of the PSH flag tells the receiver that the sender has no more data to send right now, and that it can *push* what is in its receive buffer to the application. This feature is not controllable in user-space, and its use varies in TCP/IP stack implementations [Ste93]. Making these corrections with the kernel is a topic for future work, described in Chapter 6.

Of the five byte differences between the emulator and the target PLC, the fifth different byte is the 0x37<sup>th</sup> byte of response packet 6, which is the second byte of HTML data payload in this packet. As seen in Figure 5.2, the emulator value is 0x3a, an ASCII colon (':') character, and the PLC value is 0x0a, an ASCII line feed character. Because this difference occurs with a whitespace character, it is likely due to a copy-and-paste situation where a change to whitespace characters occurs when the byte that is pasted is not what is copied. Another possibility is that a mistake was made when preparing the *.html* document

### Gumstix response packet 4 for standard web query

```

0000 00 1c 23 1a 37 21 00 e0 62 60 46 25 08 00 45 00 | ..#.7!.. b`F%..E.
0010 02 28 00 00 40 00 ff 06 64 ec 0a 01 00 4f 0a 01 | .(..@... d....O..
0020 00 93 00 50 13 88 8d 8d 9e 27 5f c0 0d 90 50 10 | ...P.... .'_...P.
0030 08 00 c6 4a 00 00 30 20 45 74 68 65 72 6e 65 74 | ...J..0 Ethernet
0040 20 43 6f 6d 6d 75 6e 69 63 61 74 69 6f 6e 73 20 | Communi cations
0050 4d 6f 64 75 6c 65 2e 3c 2f 41 3e 3c 2f 42 3e 0d | Module.< /A></B>.
0060 0a 3c 54 52 3e 3c 54 44 20 41 4c 49 47 4e 3d 72 | .<TR><TD ALIGN=r
0070 69 67 68 74 3e 45 74 68 65 72 6e 65 74 20 41 64 | ight>Eth ernet Ad
0080 64 72 65 73 73 3a 20 3c 54 44 3e 3c 42 3e 30 30 | dress: < TD><B>00
0090 20 45 30 20 36 32 20 36 30 20 34 36 20 32 35 3c | E0 62 6 0 46 25<
00a0 2f 42 3e 0d 0a 3c 54 52 3e 3c 54 44 20 41 4c 49 | /B>..<TR ><TD ALI

```

### PLC response packet 4 for same query

```

0000 00 1c 23 1a 37 21 00 e0 62 60 46 23 08 00 45 00 | ..#.7!.. b`F#..E.
0010 02 28 00 00 40 00 ff 06 64 de 0a 01 00 5d 0a 01 | .(..@... d....]..
0020 00 93 00 50 13 88 20 3b 43 ea e5 89 9b de 50 18 | ...P.. ; C....P.
0030 08 00 81 aa 00 00 30 20 45 74 68 65 72 6e 65 74 | .....0 Ethernet
0040 20 43 6f 6d 6d 75 6e 69 63 61 74 69 6f 6e 73 20 | Communi cations
0050 4d 6f 64 75 6c 65 2e 3c 2f 41 3e 3c 2f 42 3e 0d | Module.< /A></B>.
0060 0a 3c 54 52 3e 3c 54 44 20 41 4c 49 47 4e 3d 72 | .<TR><TD ALIGN=r
0070 69 67 68 74 3e 45 74 68 65 72 6e 65 74 20 41 64 | ight>Eth ernet Ad
0080 64 72 65 73 73 3a 20 3c 54 44 3e 3c 42 3e 30 30 | dress: < TD><B>00
0090 20 45 30 20 36 32 20 36 30 20 34 36 20 32 33 3c | E0 62 6 0 46 23<
00a0 2f 42 3e 0d 0a 3c 54 52 3e 3c 54 44 20 41 4c 49 | /B>..<TR ><TD ALI

```

Figure 5.1: Partial response packet number 4 for the Gumstix emulator and PLC. The solid rectangles identify non-deterministic bytes identified in Appendix D. The arrow points to the TCP Flags byte in both responses. This is a fingerprintable difference.

to be served by the Python webserver. Whatever the cause, this difference is easily fixed once it has been discovered by changing the *index.html* document served by the Gumstix emulator.

```

Gumstix: 0030| 08 00 26 35 00 00 0d 3a 3c 54 52 3e 3c 54 44 20 ..&5...: <TR><TD
PLC: 0030| 08 00 d9 ce 00 00 0d 0a 3c 54 52 3e 3c 54 44 20 ..... <TR><TD

```

Figure 5.2: Incorrect HTML byte in emulator, likely due to copy and paste error during implementation.

The accuracy of the responses for the HAP standard requests is better than that of web standard requests, seen in Table 5.1. This is because of the simplicity of UDP which encapsulates the HAP protocol. There are a total of 309 bytes across one packet in a HAP standard workload response. After excluding non-deterministic header fields defined in Appendix E, 305 are deterministic, and exactly 305 are accurately emulated for each query. Thus, the Gumstix emulator is 100% accurate for these types of queries. Again, there are no confidence intervals because there is no variance among the trials conducted.

Also in Table 5.1 are results for the three request frequencies (detailed in Section 4.5.1.2) for both types of queries. Each scenario in Table 5.1 shows that the Gumstix has no loss in packet-level accuracy as the frequencies are varied for either the web or HAP standard workloads.

At the PLC Break request frequency level for both the web and HAP standard workloads, only a fraction of the 1000 queries made to the Gumstix emulator are successfully completed. Only the 319 and 636 respective completions are considered in the percentage of packet bytes. Since the timing is the only parameter varied in this scenario, these data suggests that the request frequency does not have an impact on packet byte accuracy of the Gumstix emulator, since there is exactly zero difference in accuracy between the different request frequencies for every successful trial.

### ***5.2.2 Logging of Standard Workloads on the Gumstix Platform.***

The *percentage of packets logged* is a derived metric based on the logging metric that compares the number of packets sent to and received from the workload generator by the emulator to the number of packets sent out-of-band to the logging server by the emulator. The percentage of packets logged metric considers all the packets from all trials, and the results of all 1000 trials for each timing level are pooled. Pooling is allowed because they are all samples of same population, taken at different times.

For 1000 successful standard web queries, 8,000 packets are sent to the emulator, and 8,000 packets are received from it, for a total of 16,000 packets. The percentage of logging packets sent to the logging server compared to the 16,000 total packets is the percent packets of packets logged metric.

Table 5.1 shows that for every packet sent and received, 100% are logged for all scenarios except the HAP workload at the PLC Break frequency, which is at 76.96% packets logged. However, a closer look at this scenario shows a higher measure for the percent of packets logged. Of the 1000 packets sent by the workload generator, only 636 are responded to by the emulator. After accounting for the 364 queries not seen (1000 queries – 636 responses), only 1272 packets should have been logged by the emulator bringing the total percentage of packets logged to 98.98% as expressed by

$$\frac{1259 \text{ logged}}{1636 - 364 \text{ received by emulator}} = \frac{1259}{1272} = 98.98\% \text{ packets logged} \quad (5.1)$$

Because all results are pooled it is difficult to conclude that, across all three request frequencies the logging performance of the Gumstix emulator to standard web and HAP workloads are statistically the same or different. While pooling makes it easier to determine overall results, it prevents a confidence interval from being calculated since each log entry is not tied directly to an individual query or response packet. However, the 100% success rate for all web workload frequencies is strong evidence that no difference exists between the Gumstix logging capability for these scenarios. The 1% difference among the standard HAP logging percentages is also very favorable, and would likely have little to no impact on real world applications of this emulator.

### ***5.2.3 PC Packet Level Accuracy Results.***

Table 5.2 shows the packet-level accuracy results the PC emulator platform. These results are very similar to those for the Gumstix emulator platform presented in Section 5.2.1. The percentages of packet bytes correct are the same as the Gumstix. The differences

between the PC emulator and target PLC byte values are exactly the same as the Gumstix emulator, which is expected. The differences between the Gumstix and PC results are reflected in the PLC Break request frequency scenarios, where the PC shows no degradation in performance, in successful trials or packets logged.

Table 5.2: Packet level accuracy and logging results for the PC emulator.

Scenario	Service	Request Frequency	Successful trials	%packet bytes correct	%packets logged
1	Web	slow	1000/1000	99.79%	100.00%
2	Web	PLC Break	1000/1000	99.79%	100.00%
2a	Web	PC Break	1000/1000	99.79%	100.00%
5	HAP	slow	1000/1000	100.00%	100.00%
6	HAP	PLC Break	1000/1000	100.00%	100.00%
6a	HAP	PC Break	1000/1000	100.00%	100.00%

Despite the different Linux platforms and therefore the likely different implementations of Linux kernel TCP/IP stacks, the same differences in the deterministic fields as compared to the target PLC platform occur in the PC as they do in the Gumstix platform. The result is identical values for the percentages of packet bytes correct metrics for both the standard web and standard HAP workloads.

The different request frequencies seem to have no effect on the accuracy of the PC emulator, with both web and HAP percentages of bytes correct metrics consistent across all three request frequency levels at 99.79% and 100% for the standard web and HAP workloads, respectively.

#### ***5.2.4 Logging of Standard Workloads on the PC Platform.***

The logging performance of the PC platform as measured by the percentage of packets logged metric shows that the PC logs all packets successfully. The ability of the PC to log



standard workloads is only different under the PLC break conditions when compared to the Gumstix platform. Though pooling of the trials prevents a statistical analysis, the results suggest that the logging capability between the Gumstix and PC platforms for standard web and HAP workloads is equivalent.

Despite the fact that the accuracy rate of 99.7% is less than 100% for the web standard workload, it is concluded that these results are suitable for cyber sensor and honeypot applications because attackers are not actively looking for ICS honeypots because there are very few known implementations, and none using this target PLC.

### **5.3 Scanning Level Accuracy**

#### ***5.3.1 Gumstix Scanning-Level Accuracy Results.***

The *percentage of Nmap Fields correct* is a derived metric based on the Nmap OS Fingerprinting Fields metric, similar to the percentage of bytes correct metric. Instead of comparing bytes, the percentage of Nmap Fields correct metric compares the results of the emulator Nmap fields metric to the baseline Nmap fields metric for the target PLC.

There are 113 unique fields across four sections. Like the packet bytes metric, there are non-deterministic fields. These fields vary across repeated scans of the target PLC. The non-deterministic fields unique to Nmap for OS detection scans include the Sequence Prediction Index (SP) (also reported as the TCP sequence prediction), TCP Initial Sequence Number Counter Rate (ISR), and Unused ICMP port unreachable field nonzero (UN) fields. A screenshot of the non-deterministic fields and their values is seen in Figure 5.3. The SP and ISR fields report statistical data of the TCP initial sequence numbers (ISN) of response packets returned by the scanned device. The TCP ISN is intended to be pseudorandom (i.e., non-deterministic) to protect against TCP hijacking and other attacks.

The scope of this research is to accurately reproduce the most important, deterministic fields, which excluded accurate reproduction of the target PLC's sequence number

## Gumstix Emulator Nmap OS Scan

```
Not shown: 131067 closed ports
PORT      STATE SERVICE
80/tcp    open  http
502/tcp    open  asa-appl-proto
28784/udp open|filtered unknown
MAC Address: 00:E0:62:60:46:24 (Host Engineering)
Device type: specialized
Running: Koyo embedded
OS details: Koyo DirectLogic PLC
TCP/IP fingerprint:
OS:SCAN (V=5.21%D=12/13%OT=80%CT=1%CU=1%PV=Y%DS=1%PC=D%G=Y%M=00E062%TM=50CA4
OS:A87%P=x86_64-unknown-linux-gnu)SEQ(SP=5D%GCD=1%ISR=C5%YI=0%CT=7%II=RI%TS
OS:(U)OPS (O1=M200%O2=M200%O3=M200%O4=M200%O5=M200%O6=M109)WIN (W1=800%W2=800
OS:%W3=800%W4=800%W5=800%W6=800)ECN (R=Y%DF=Y%T=FF%W=800%O=M200%CC=N%Q=)T1 (R
OS:Y%DF=Y%T=FF%S=O%A=S+%F=AS%RD=0%Q=)T2 (R=N)T3 (R=Y%DF=Y%T=FF%W=800%S=O%A=S
OS:+%F=AS%O=M109%RD=0%Q=)T4 (R=Y%DF=Y%T=FF%W=800%S=A+A=S%F=AR%O=%RD=0%Q=)T5
OS:(R=Y%DF=Y%T=FF%W=800%S=A%A=S+%F=AR%O=%RD=0%Q=)T6 (R=Y%DF=Y%T=FF%W=800%S=A
OS:%A=S%F=AR%O=%RD=0%Q=)T7 (R=Y%DF=Y%T=FF%W=800%S=A%A=S+%F=AR%O=%RD=0%Q=)U1 (
OS:R=Y%DF=Y%T=FF%IPL=38%UN=B390%GRIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE (R=Y%DF
OS:I=S%T=FF%CD=S)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=93 (Good luck!)
IP ID Sequence Generation: All zeros
Read data files from: /usr/share/nmap
OS detection performed. Please report any incorrect results at http://
Nmap done: 1 IP address (1 host up) scanned in 3657.08 seconds
Raw packets sent: 159858 (5.634MB) | Rcvd: 131086 (6.293MB)
```

Highlighted Fields: MAC=24, SP=5D, ISR=C5, UN=B390

## Target PLC Nmap OS Scan

```
Not shown: 131067 closed ports
PORT      STATE SERVICE
80/tcp    open  http
502/tcp    open  asa-appl-proto
28784/udp open|filtered unknown
MAC Address: 00:E0:62:60:46:23 (Host Engineering)
Device type: specialized
Running: Koyo embedded
OS details: Koyo DirectLogic PLC
TCP/IP fingerprint:
OS:SCAN (V=5.21%D=12/13%OT=80%CT=1%CU=1%PV=Y%DS=1%DC=D%G=Y%M=00E062%TM=50CCF
OS:062%P=x86_64-unknown-linux-gnu)SEQ(SP=2F%GCD=1%ISR=9A%YI=2%CI=2%II=RI%TS
OS:(U)OPS (O1=M200%O2=M200%O3=M200%O4=M200%O5=M200%O6=M109)WIN (W1=800%W2=800
OS:%W3=800%W4=800%W5=800%W6=800)ECN (R=Y%DF=Y%T=FF%W=800%O=M200%CC=N%Q=)T1 (R
OS:Y%DF=Y%T=FF%S=O%A=S+%F=AS%RD=0%Q=)T2 (R=N)T3 (R=Y%DF=Y%T=FF%W=800%S=O%A=S
OS:+%F=AS%O=M109%RD=0%Q=)T4 (R=Y%DF=Y%T=FF%W=800%S=A+A=S%F=AR%O=%RD=0%Q=)T5
OS:(R=Y%DF=Y%T=FF%W=800%S=A%A=S+%F=AR%O=%RD=0%Q=)T6 (R=Y%DF=Y%T=FF%W=800%S=A
OS:%A=S%F=AR%O=%RD=0%Q=)T7 (R=Y%DF=Y%T=FF%W=800%S=A%A=S+%F=AR%O=%RD=0%Q=)U1 (
OS:R=Y%DF=Y%T=FF%IPL=38%UN=D046%GRIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G)IE (R=Y%DF
OS:I=S%T=FF%CD=S)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=47 (Good luck!)
IP ID Sequence Generation: All zeros
Read data files from: /usr/share/nmap
OS detection performed. Please report any incorrect results at http://
Nmap done: 1 IP address (1 host up) scanned in 125.85 seconds
Raw packets sent: 131091 (4.721MB) | Rcvd: 131085 (6.292MB)
```

Highlighted Fields: MAC=23, SP=2F, ISR=9A, UN=D046

Figure 5.3: Comparison between the results of the Nmap Scans of the Gumstix emulator and the target PLC with the non-deterministic fields highlighted.

generation algorithm. Relative to the other purely deterministic fields, these three Nmap fields are much harder to fingerprint because a statistical analysis must be done to determine differences between the emulator and target PLC. The design of the emulator uses recycled ISNs from hard-coded configurations in the *infilter.o* executable, and further investigation of comparison of emulator to target PLC sequence number generation is left for future work. Because no effort has been done to disprove that the ISN numbers are statistically different, and because this is an acknowledged difference between the Gumstix (and PC) emulation and the target PLC, these three fields are classified as important, and incorrect matches in these fields count against the accuracy.

The UN field represents data that is incorrectly placed in the last four bytes of the ICMP header, in contradiction to ICMP request for comments (RFC) 792 [Lyo09, Pos81]. When these bytes are set, Nmap places their value in the UN in the OS Scan details section.

This field also seems random based on the observed responses from the target PLC, but without statistical proof that Gumstix (and PC) emulation of the UN field is the same as the target PLC, incorrect matches in this field count against the accuracy. Thus, the emulator is 97.25% accurate (106 correct out of 109 deterministic fields) at the scanning-level overall when accounting for deterministic fields.

Table 5.3 shows that scanning-level accuracy of the Gumstix emulator platform as measured by the percentage of Nmap Fields correct metric is 97.25% at both slow and fast request frequency levels. The slow and fast frequency levels are defined in Section 4.3.

Table 5.3: Scanning-level accuracy and logging results for the Gumstix emulator. The percentages include include the non-deterministic fields.

Scenario	Request Frequency	Successful trials	%Nmap fields correct	%packets logged
11	slow	5/5	97.25%	95.64%
12	fast	5/5	97.25%	98.75%

Despite the fact that the Gumstix emulation differs by 3 fields, the results are still excellent, and the chances of the emulator being fingerprinted by the Nmap OS fields are small. This is primarily due to the fact that the vast majority of fields are consistent with the device, and if a public fingerprint of the device was available in the Nmap OS database, there is a very high probability that the emulator would match that fingerprint. Additionally, to fingerprint the device based on these erroneous fields, a user would have to know the underlying distribution of the sequence numbers for the target device *a priori*, which is not likely. The user then must accurately measure the distribution of the emulator's sequence numbers, also not likely.

The accuracy rate among all deterministic fields is 97.25%. It is concluded that the results for scanning-level accuracy are suitable for the emulator's application in SCADA attack-landscape research as a honeypot based on this analysis.

The results for both slow and fast request frequencies shown in Table 5.3 indicate that the Gumstix has no change in accuracy as the request frequencies are varied. Because this is the only parameter varied in this scenario, and the number of trials conducted, the data suggests that the request frequency does not have an impact on Nmap field accuracy of the Gumstix emulator.

### ***5.3.2 Logging Results of Non-Standard Workloads on the Gumstix Platform .***

To determine the ability of the Gumstix emulator to log non-standard workloads, the percentage of packets logged metric is again used. Table 5.3 shows 95.64% success for all packets for the five trials at the slow request frequency and 98.75% success for the five trials at the high frequency. The results of all five trials for each timing level are pooled, generating over 1.3 million packets for each timing level. They can be pooled because they are all samples of same population, taken at different times.

It is difficult to do the same type of analysis performed with Equation 5.1 because of the complexity of Nmap probes. The calculation performed in Equation 5.1 depends on the fact 636 packets are *known* to have been received by the emulator because responses for 636 queries were sent. With Nmap scans, there is no easy way of knowing which probe packets are received by the emulator, because some Nmap probe packets illicit no response (i.e., an open UDP port sends no response when probed).

A comparison in Figure 5.4 of the percentages of packets logged at the different request frequency timing levels shows that the Gumstix logs a higher percentage of packets when scanned at the faster, more efficient (T5) timing profile, compared to being scanned at the slower, less efficient, default (T3) timing profile, and it is concluded that request

frequency does have an affect on the Gumstix ability to log non-standard web workload (Nmap scan).

This is an interesting and counterintuitive observation which may be due to a number of confounding factors related to the complexity of Nmap OS scans. These T5 timing profile configuration features include differences in handling of parallelization, probe retries, and timeouts designed to make the T5 profile (also known as the "insane" timing profile) faster and more efficient. It is evident by Figure 5.4 that in this scenario, these efficiencies enable better performance for the Gumstix emulator for the T5 timing profile. As a result, fewer probes are actually sent to the emulator, which means fewer packets needed to be logged. For example, the five trials for the T3 timing profile generates 1,377,140 packets (logging 95.64% according to Table 5.3), while the T5 timing profile generates only 1,330,925 packets (logging 98.75% according to Table 5.3).

### ***5.3.3 PC Scanning-Level Accuracy Results.***

Table 5.4 shows the scanning-level accuracy results for the PC emulator platform. These results are similar to those for the Gumstix emulator platform presented in Section 5.3.

Similar to the Gumstix, the different request frequencies have no affect on the scanning-level accuracy of the PC emulator, with the percentage of correct Nmap fields consistent across both request frequency levels, both at 97.25%. These values are the same as the Gumstix because the differences between the PC emulator and target PLC Nmap fields are exactly the same as the Gumstix emulator, as expected. Figure 5.5 shows that there is no apparent difference in the percentages of packets logged for the PC emulator between the fast and slow web non-standard workload frequencies, however every packet is not logged in either scenario.

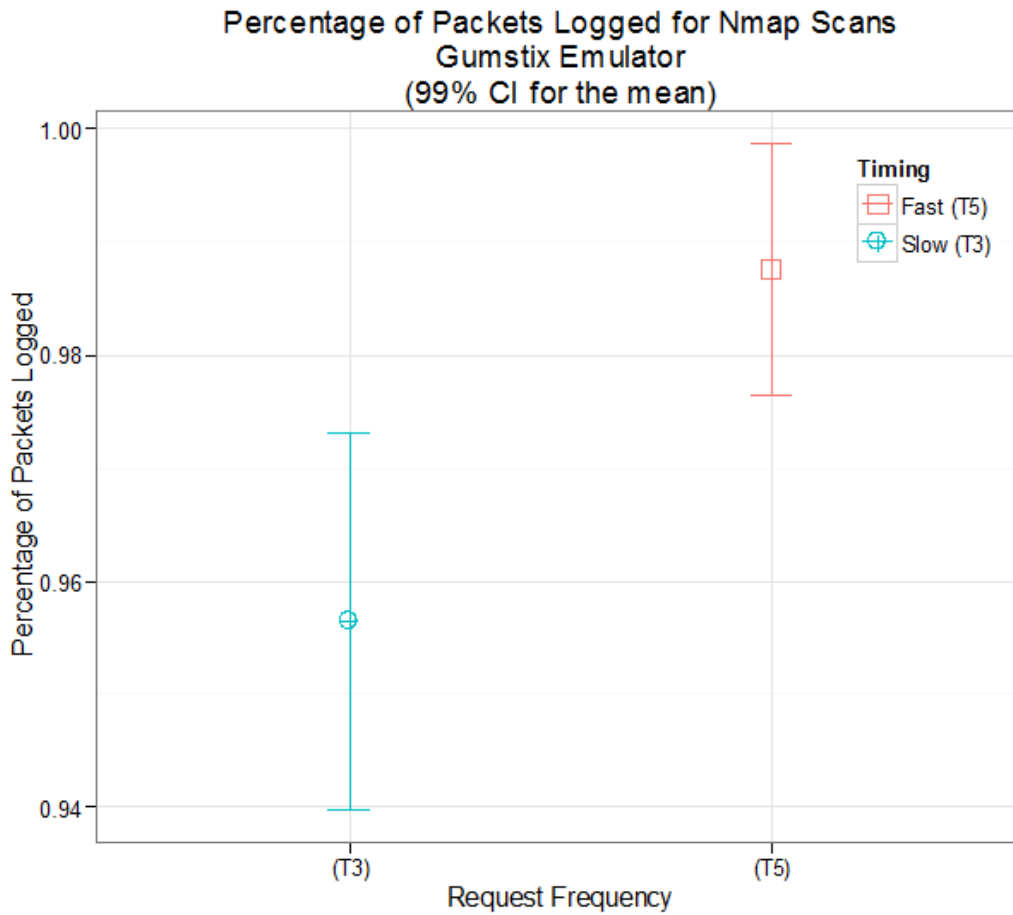


Figure 5.4: Comparison of percentage of packets logged by the Gumstix emulator at different request frequency levels showing 99% CI for the mean.

Table 5.4: Scanning-level accuracy and logging results for the PC emulator. The percentages include include the non-deterministic fields.

Scenario	Request Frequency	Successful trials	%Nmap fields correct	%packets logged
3	slow	5/5	97.25%	99.97%
4	PLC Break	5/5	97.25%	100.00%

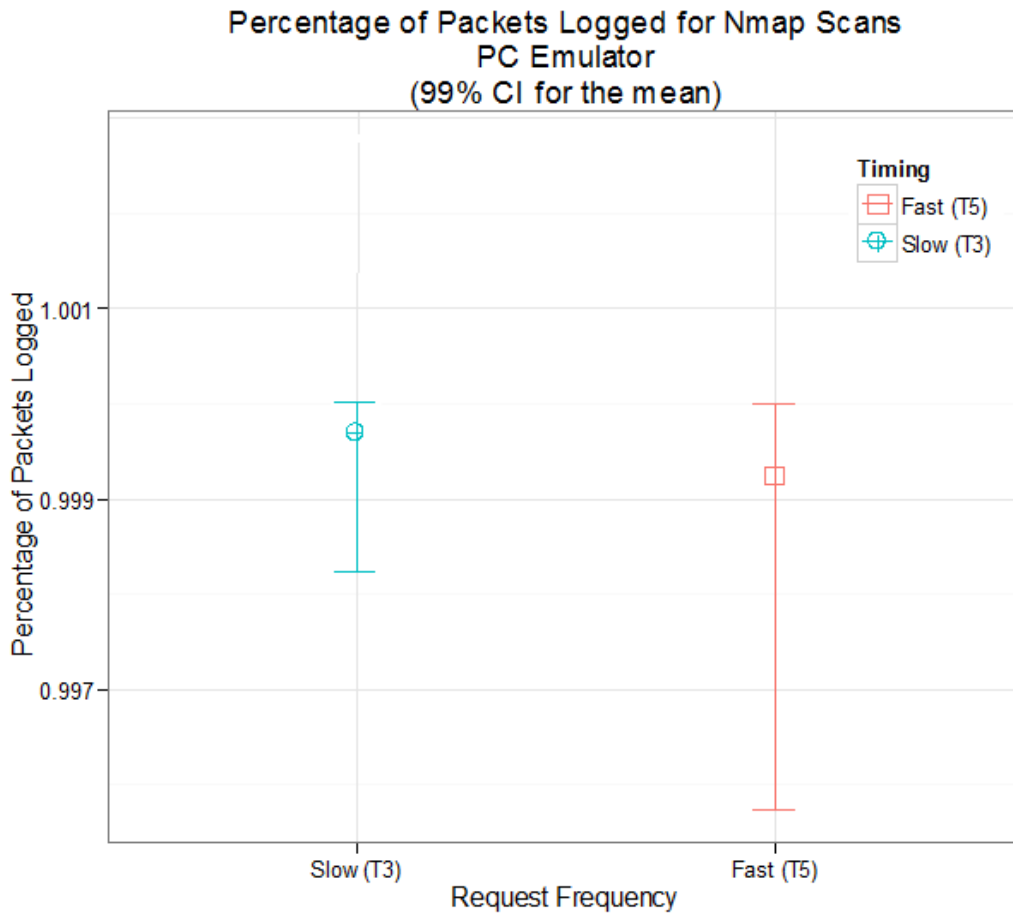


Figure 5.5: Comparison of percentage of packets logged by the PC emulator at different request frequency levels showing 99% CI for the mean.

With such high numbers of packets, it is difficult to assess exactly why packets at respective timing levels for both the Gumstix and PC did not get logged. Because the number of packets needing to be logged is so high (around 270,000 for each scan), it is likely that when the send buffer on the LAN-facing interface of the Gumstix becomes full, queued packets are simply dumped.

Is it also a possibility that, like experimental scenario 14 described in Section 5.2.1, a number of packets are never seen by the emulator, and therefore are not logged. However, due to the large number of packets, the same type of analysis performed in Section 5.2.1

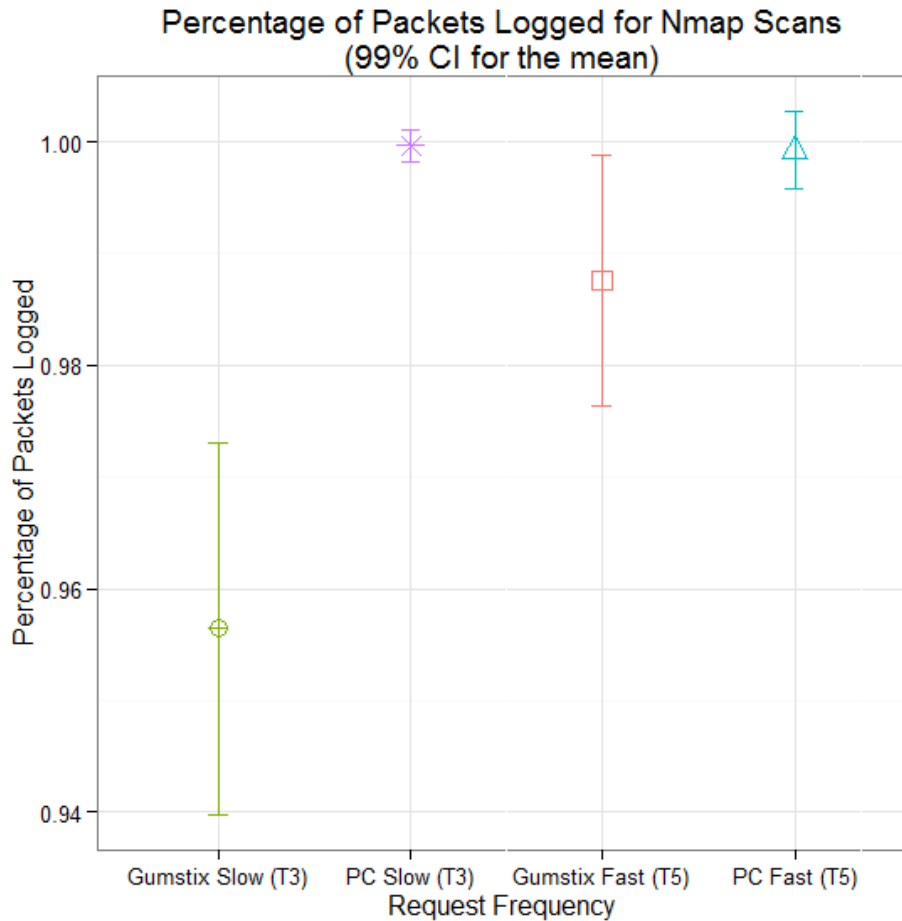


Figure 5.6: Comparison between Gumstix and PC logging functionality for web non-standard workload at both request frequencies.

is not possible. Figure 5.6 shows the results of a comparison of all percentages of packets logged. No statistical difference between the Gumstix and PC ability to log Nmap scan workloads at the T5 request frequency is evident. However the PC’s logging capability is significantly better (at a 99% CI) for the T3 Nmap scan workload compared to the Gumstix’s logging ability at the same level. The higher percentage of packets logged of the T5 workload versus the T3 workload for the Gumstix are explained in Section 5.3.2.



## 5.4 Attack Level Accuracy

The *successful unlocks* metric measures if the Metasploit exploit successfully unlocked the CUT. The results for the baseline target PLC in Table 5.5 show all attacks are successful at unlocking the target PLC for both the slow and PLC Break request frequency levels. Thus, the baseline for the emulators is that all 5 trials at the prescribed request frequencies complete successfully.

Table 5.5: Attack results for the target PLC.

Scenario	Request Frequency	Workload Input value	Trials	successful unlocks	%packets logged
23	slow	1	5	5	100.00%
24	PLC Break	3.2555	5	5	100.00%

### 5.4.1 Gumstix Attack-Level Accuracy Results.

The results for attack-level accuracy as measured by the successful unlocks metric of the Gumstix and percentage of packets logged are shown in Table 5.6. All attacks are successful at all timing levels.

Table 5.6: Attack-level accuracy and logging results for the Gumstix emulator.

Scenario	Request Frequency	Workload Input value	Trials	successful unlocks	%packets logged
15	slow	0.081	5	5	100.00%
16	PLC Break	0.266	5	5	100.00%
16a	Gum Break	480	5	5	100.00%

#### 5.4.2 PC Attack Level Accuracy Results.

The results for attack-level accuracy as measured by the successful unlocks metric of the PC and percentage of packets logged are shown in Table 5.7. All attacks are successful at all timing levels.

Table 5.7: Attack-level accuracy and logging results for the PC emulator.

Scenario	Request Frequency	Workload Input value	Trials	successful unlocks	%packets logged
7	slow	0.071	5	5	100.00%
8	PLC Break	0.2304	5	5	100.00%
8a	PC Break	3000	5	5	100.00%

It is concluded that the Gumstix and PC are both 100% accurate for all request frequency levels for attack-level accuracy. Request frequency also appears to have no effect on logging for either the Gumstix or PC since 100% of packets are logged under all request frequency levels.

### 5.5 Timing Level Accuracy

The *Deviation from the Baseline* is a derived metric based on the *Response Time* metric used to compare the results for the emulator responses to the baseline target PLC responses. The data shown are from the scenarios already presented, but are discussed separately because they represent a different level of accuracy.

The emulator results are compared to the baseline two different ways, additive and multiplicative. Both values are shown to allow for intuitive interpretation of the results. Negative signs in Table 5.8 through Table 5.14 indicate that the mean response time for emulator responses is longer than the baseline (target PLC) mean response time.

### 5.5.1 Gumstix Timing Level Accuracy Results for Standard Workloads.

The timing results for the web and HAP standard workloads are shown in Table 5.8. Only successful trials are considered when calculating the mean. For all scenarios, the Gumstix responded slower than the target PLC. The most notable differences occur at the PLC Break request frequency levels for both the web and HAP standard workloads. The purpose of this experimental scenario is to determine how the Gumstix emulator performs at the maximum capability of the target PLC.

Table 5.8: Timing accuracy results for Gumstix emulator for the web and HAP standard workloads.

Scenario	Service	Request Frequency	Success Trials	Response (s)		PLC Baseline (s)	Deviation from baseline	
				Mean	S.D.			
9	Web	slow	1000	0.0842	3.70e-3	0.0376	-0.046s	2.2 × Slower
10	Web	PLC Break	319	2.3503	6.44	0.093	-2.256s	25 × Slower
10a	Web	Gum Break	1000	0.1831	3.16e-2		-2.439s	1.9 × Slower
13	HAP	slow	1000	0.0034	4.39e-4	0.0026	-0.0007s	1.3 × Slower
14	HAP	PLC Break	636	0.2003	1.74e-4	0.0020	-0.193s	98.6 × Slower
14a	HAP	Gum Break	1000	0.1522	1.40e-1		-0.350s	74.9 × Slower

The results show the Gumstix emulator is up to 25 and 98.6 times slower in response to web and HAP respectively compared to the PLC. Figure 5.7 plots the 319 successful web standard requests by their response time. The dashed lines on the graph represent 1, 2, 4, and 8 standard deviations in response times above the mean. The distribution of response times shows that the majority of queries are completed with a response time around 0.678s. There are only 47 of 319 observations above the mean return time of 2.3503s, meaning that the results are drastically skewed due to the very high relative response times of only a few data points.

Additional investigation into these extreme data points reveals that they are valid and represent actual response times. The cause is long TCP timeout values that go into effect after the TCP three-way handshake has been completed. Figure 5.8 shows the capture of the TCP stream for the connection that is over eight standard deviations from the mean in Figure 5.7. The stream shows that after the initial handshake, the GET request times out three times before the emulator finally responds. Because the TCP retransmission delay doubles with each timeout, the workload generator waits 45.123s before getting an ACK from the emulator. Therefore, it is concluded that because these extreme response times represent valid functionality of TCP, the data points are valid, and must be considered in the analysis.

Another interesting result of the scenarios in Table 5.8 is that the mean response times for the data at the device's respective break request frequency (PLC Break or Gumstix Break) are longer than those of the respective slow request frequencies. The target PLC is slower to respond, on average, to consecutive requests at its maximum request frequency when compared to the target PLC's response times for the slow request frequency. The same observation is true for the Gumstix. This implies that as the request frequency increases to and beyond the device's maximum response rate, the trials are no longer independent. That is, the trials begin to interact with each other, which is not the original intention of this experimental scenario. To verify this phenomenon, Figure 5.9 shows a plot of response times in the order that they are observed. The Gumstix and the target PLC are shown both at the PLC break request frequency level. The upper graph shows the response time for the Gumstix linearly increasing to a plateau at 0.678s, and the lower graph shows the response times of the target PLC gradually increasing, then gradually decreasing over 1000 trials. On both graphs the mean and delay between queries are shown by dashed lines. In both cases, the delay between queries is just below the very first response from the device, indicating that queries are arriving at the devices faster than they can respond. The

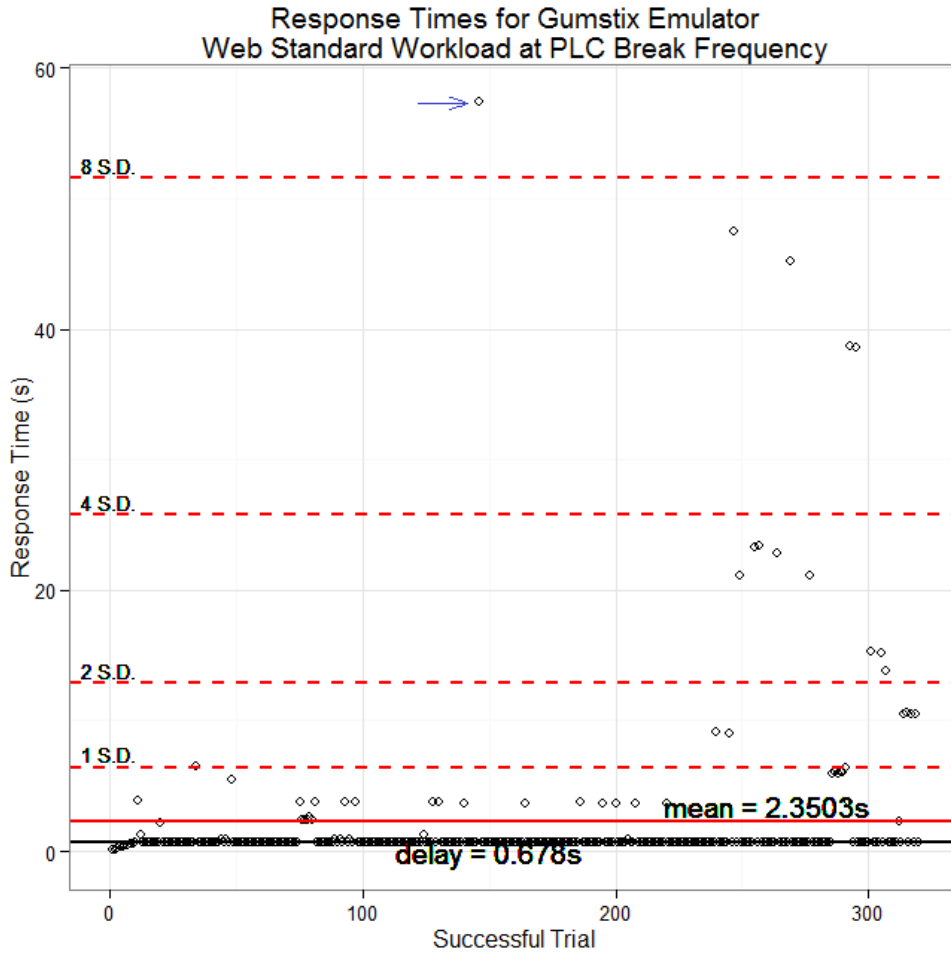


Figure 5.7: Response times for the Gumstix emulator at the PLC break frequency showing the data points that are 1, 2, 4, and 8 standard deviations from the mean.

result is a steady increase in response times. While the Gumstix plateaus, the target PLC exhibits a cyclic pattern. These phenomena are likely due to the handling of the send and receive buffers on the respective devices.

Figure 5.10 shows the results of the three request frequency levels for the Gumstix at 99% CI for the mean for the web standard workload. The graph shows there is a statistical difference between the means of all three response speeds for the three different request frequency levels. Therefore, it is concluded that request frequency does affect the timing accuracy of the Gumstix emulator for the web standard workload.

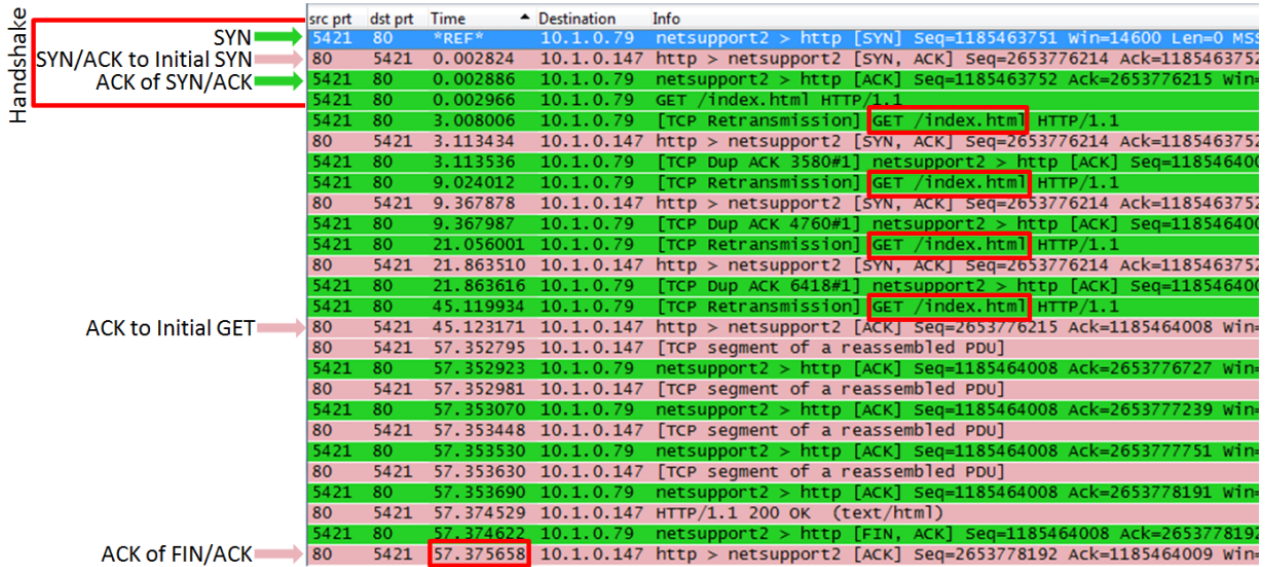


Figure 5.8: TCP connection with anomalous response time 8 SD above the mean. Highlights show workload generator repeatedly trying to send the GET request and timing out.

The Gumstix timing accuracy results for the HAP standard workload are also in Table 5.8 and follow a similar pattern as the web mean response times: all mean response times for the Gumstix are slower than the target PLC.

Observing the results of the response times for Gumstix at the PLC Break request frequency level, another interesting trend is apparent in Figure 5.11. Drastic jumps in response time of around one-quarter second each are seen. The dashed lines represent where all but 3 of the 364 dropped queries for this workload occurred. Again, this repetitive behavior at the Gumstix boundary conditions is likely due to receiving buffer limitations. As the receive buffer gradually becomes full, response times lengthen as queries spend more time waiting in the input buffer. When the receive buffer is full, additional incoming queries are dropped. It is possible that quarter-second increases occur when the receive buffer is being processed, or possibly when the log packets are sent from the second LAN-facing interface, but it is impossible to tell exactly what causes these regular jump without

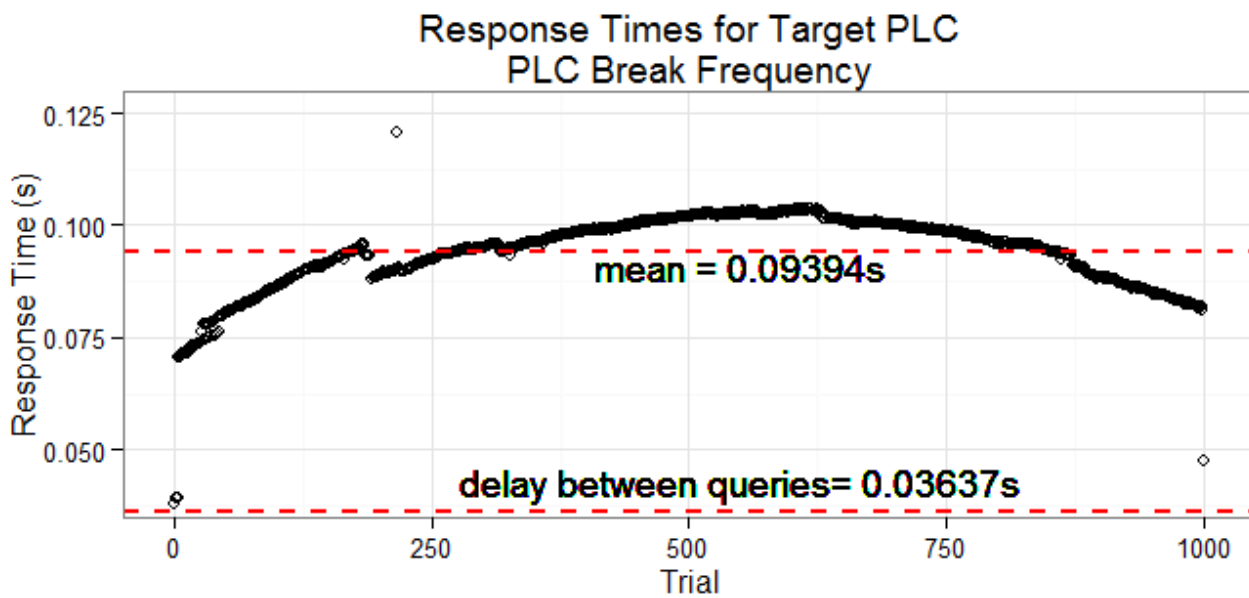
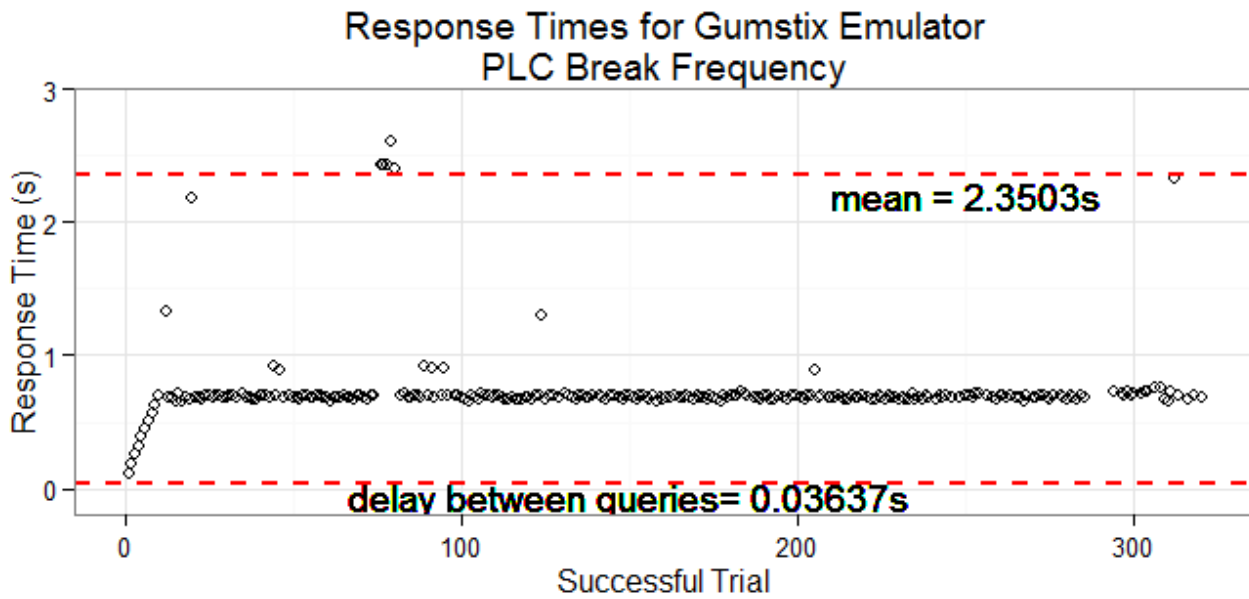


Figure 5.9: Response times plotted against observation number for web Standard workload at PLC break frequency. These graphs demonstrate that as the request frequencies approach (lower graph) or exceed (upper graph) their fastest response time, trials are no longer independent. Not all observations are shown for the Gumstix responses.

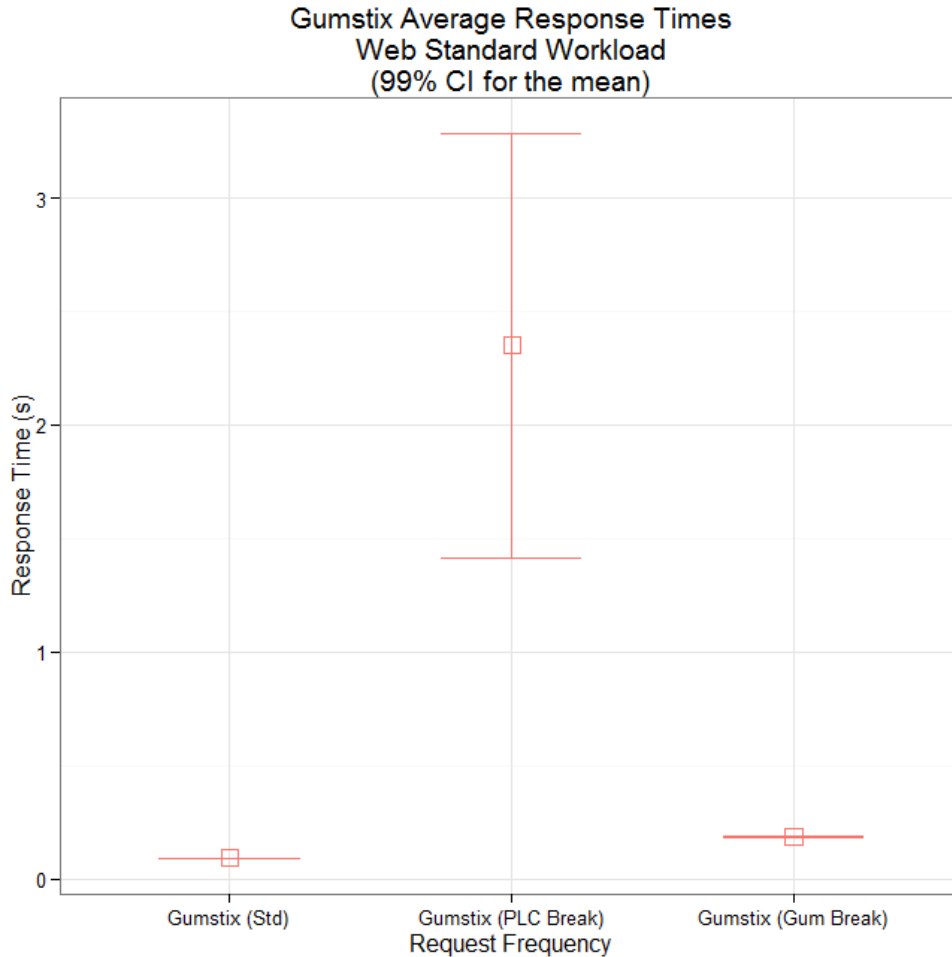


Figure 5.10: Results comparing all three request frequency levels of the Gumstix for the web standard workload.

diving deep into the Gumstix Ethernet hardware and Linux drivers, which is beyond scope of this research.

Because neither PC or PLC platforms exhibit this behavior at this request frequency for the HAP standard workload (graphs available in Appendix H), the response time measurement may be a way to fingerprint the Gumstix hardware, independently of the emulator application. Further investigation of fingerprinting of the Gumstix hardware via timing is beyond scope of this research, and is left for future work.



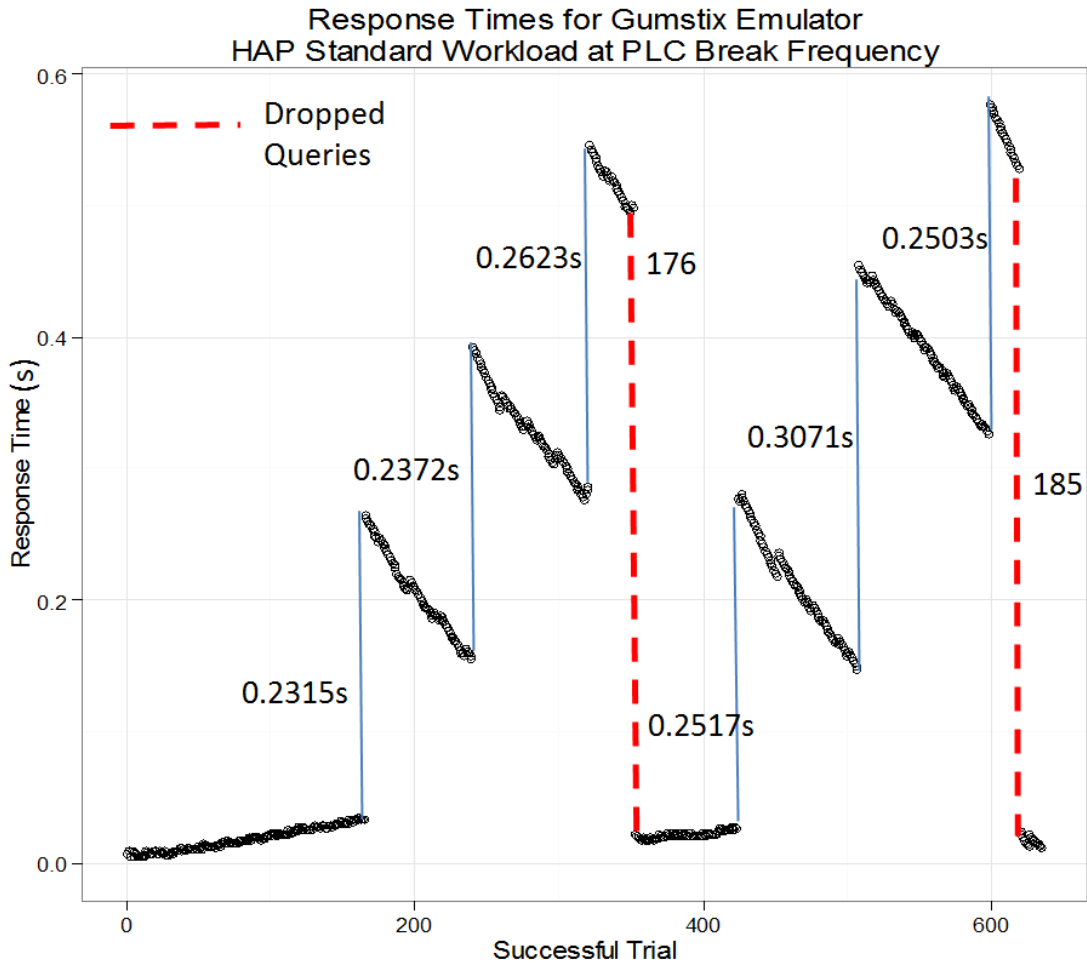


Figure 5.11: Non-continuous jumps in response time of a quarter second (nominal) for the Gumstix. The dashed lines represent where most of the queries are dropped.

The results for the HAP standard workload on the Gumstix also show there is a statistical difference between the means of all three response speeds for the three different request frequency levels at the 99% CI. The plot of this data is similar to Figure 5.10 and can be found in Appendix H. Therefore it is concluded that request frequency does affect the timing accuracy of the Gumstix emulator.

### 5.5.2 PC Timing Accuracy Results for Standard Workloads.

Results for the PC timing-level accuracy for standard workloads are shown in Table 5.9. All mean response times are faster for the PC as compared to the mean response times

for the target PLC. The results show that, compared to the baseline for the web standard workload, the PC is 13.4 times faster at responding to the slow request frequency workload, and 38.4 times faster at responding to the maximum target PLC request, when compared to the target PLC. The PC is 6.8 and 5.6 times faster at responding to the HAP standard workloads when compared to the target PLC for the respective request frequency levels.

An analysis similar to the Gumstix is performed to determine if the means of the results at different request frequency levels are statistically different. For the web standard workload on the PC, there is no statistical difference between the slow and PLC Break request frequencies, but there is a difference from these two levels to PC Break request frequency, all at the 99% CI for the mean. For the HAP standard workload on the PC, all three means show a statistical difference at the 99% CI. The resulting graphs can be found in Appendix H. Therefore, it is concluded that request frequency does have a significant effect on PC response times for web and HAP standard workloads.

Table 5.9: Timing Accuracy results for PC emulator for the web and HAP standard workloads.

Scenario	Service	Request Frequency	Success Trials	Response (s)		PLC Baseline (s)	Deviation from baseline	
				Mean	S.D.			
1	Web	slow	1000	0.002817	6.26e-4	0.037685	0.0349	13.4 × Faster
2	Web	PLC Break	1000	0.002449	6.58e-4	0.09395	0.0915	38.4 × Faster
2a	Web	PC Break	1000	0.003034	6.11e-4		0.0885	39.6 × Faster
5	HAP	slow	1000	0.000397	2.04e-4	0.002699	0.0023	6.8 × Faster
6	HAP	PLC Break	1000	0.000362	4.55e-5	0.002032	0.0017	5.6 × Faster
6a	HAP	PC Break	1000	0.000309	4.56e-5		0.0014	6.6 × Faster

Table 5.11 compares mean response times of the Gumstix and PC for all web and HAP standard workload scenarios. The data show that the PC has a faster mean response time for all scenarios.

A comparison between all three devices and their mean response times for the web standard workload is shown in Figure 5.12 with the 99% confidence intervals for the mean. Two groups for the standard and PLC Break request frequency levels are shown.

To aid visual comparison, two scales are also shown, one for the Gumstix at PLC Break request frequency level scenario, and one for the other five scenarios. The confidence levels for five of the plots are not visible with the shown scale which is accounted for by the high number of samples taken and the low variability of the samples.

The Gumstix at PLC Break request frequency level shown in Figure 5.12 does have a visible confidence level (note the scale). The results show that there is a statistically significant differences between all three devices at both the slow (std) and PLC Break request frequency levels.

To verify these results, an analysis of variance (ANOVA) test is conducted for all three devices, grouped by request frequency level. The results are shown in Table 5.10 with resulting p-values of 2.2e-16. The p-value here is the probability of seeing results at least as extreme as these assuming that the three means are equal. The ANOVA confirms that the three means are different. Therefore, it is concluded that for the standard web workload, the PC has significantly faster return times compared to the Gumstix.

Table 5.10: Results of ANOVA test conducted on web standard workload for all three devices.

	Df	Sum Sq	Mean Sq	F Value	Pr(>F)
Factor(levels)	2	4.0875	2.0437	45301	<2.2e-16
Residuals	2997	0.0141	0.0000		

Table 5.11: Comparison between Gumstix and PC emulator for timing accuracy results.

Service	Request Frequency	Gumstix		PC		Deviation			
		Scenario	Response Time (s)		Scenario	Response Time (s)		(s)	Times
			Mean	S.D.		Mean	S.D.		
Web	slow	9	0.0842	8.42e-02	1	0.002817	6.26e-4	-0.0813s	29.9 ×
Web	PLC Break	10	2.3503	2.35e+00	2	0.002449	6.58e-4	-2.3478s	959.7 ×
Web	Device Break	10a	0.1831	1.83e-01	2a	0.003034	6.11e-4	-0.1800s	60.3 ×
HAP	slow	13	0.0034	3.40e-03	5	0.000397	2.04e-4	-0.0030s	8.6 ×
HAP	PLC Break	14	0.2003	2.00e-01	6	0.000362	4.55e-5	-0.1999s	553.3 ×
HAP	Device Break	14a	0.1522	1.52e-01	6a	0.000309	4.56e-5	-0.1518s	492.6 ×

A similar analysis is performed for the HAP standard workload. The shapes of the graph is almost identical to the results in Figure 5.9. The results of the ANOVA done on these scenarios are shown in Table 5.12. These values are extremely low (2.2e-16) and confirm that none of the three means are the same. This graph can be found in Appendix H. The conclusion is also that the PC response times are significantly faster than the Gumstix for the HAP standard workload.

Table 5.12: Results of ANOVA test conducted on HAP standard workload for all three devices.

	Df	Sum Sq	Mean Sq	F Value	Pr(>F)
Factor(levels)	2	0.0049516	0.00247579	31126	<2.2e-16
Residuals	2317	0.0002384	0.00000008		

### 5.5.3 Timing Accuracy Results for Web Non-Standard Workloads .

The results for the mean completion times for the Nmap scans (web non-standard workload) for all three devices are shown in Table 5.13. Like the previous results for the

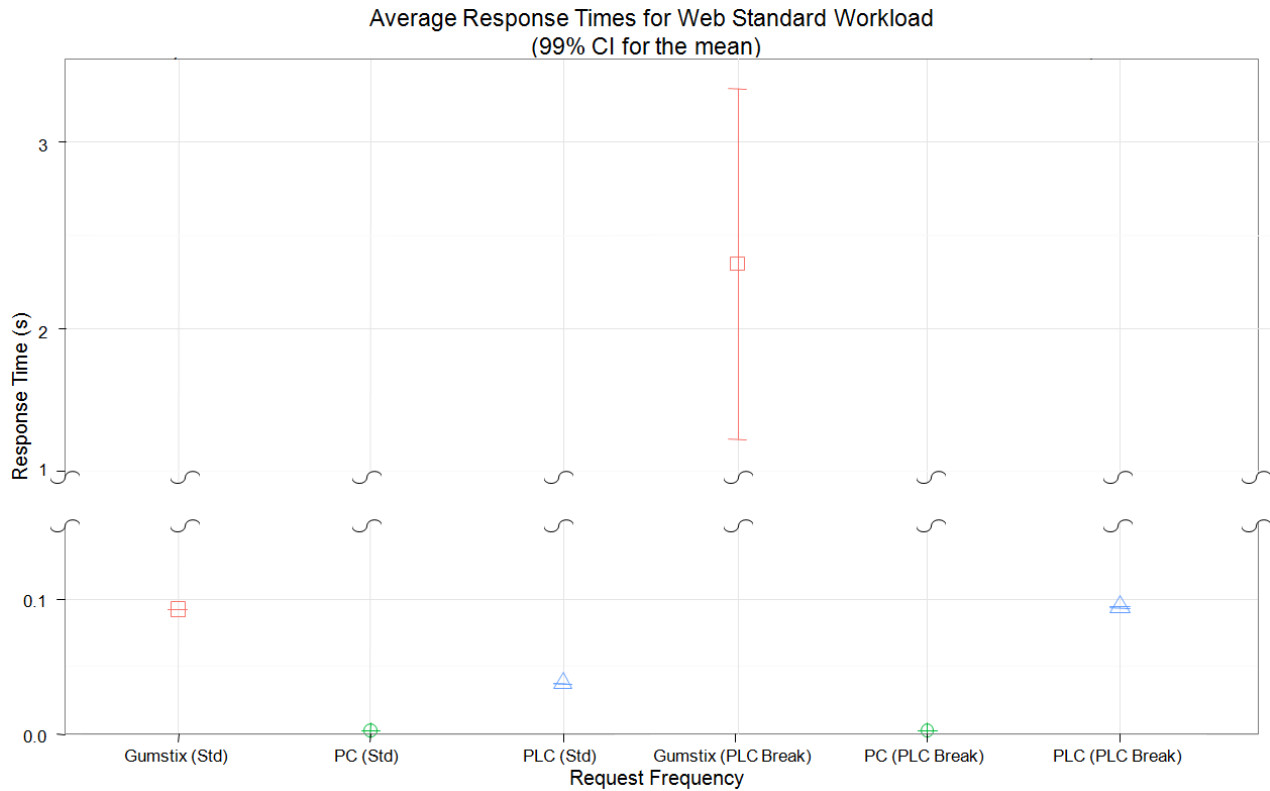


Figure 5.12: Mean response times for Gumstix, PC, and PLC for the web standard workload (Standard and PLC Break request frequencies).

standard workloads, the mean response time for the Gumstix is slower than that of the target PLC by multiples of 4.36 and 2.59 relative to the request frequency levels. The PC performs faster than the target by 1.48 and 1.55 times, respectively. When compared to the PC, the Gumstix performs 16.43 times slower for the slow request frequency level and 10.13 times slower for the PLC Break request frequency level. Figure 5.13 confirms that there is a significant difference between all three devices for their respective response times with 99% CI for the mean. While there is no significant difference between the fast (T5) and slow (T3) frequency request levels for either the PC or target PLC, there is a statistically significant difference between the two levels for the Gumstix. The results show that the PC response time is significantly faster than the target PLC to the web non-standard workload, and the Gumstix is significantly slower.

Table 5.13: Timing Accuracy results for Gumstix and PC emulators for the Web Non-Standard (Nmap) Workload.

Scenario	Device	Request Frequency	Average Completion time (s)	PLC Baseline	deviation from baseline	
11	Gumstix	slow	693.068	129.376s	563.6920	4.36 × Slower
12	Gumstix	PLC Break	385.942	107.464s	278.4780	2.59 × Slower
3	PC	slow	42.19	129.376s	-87.1860	1.48 × Faster
4	PC	PLC Break	37.992	107.464s	-69.4720	1.55 × Faster

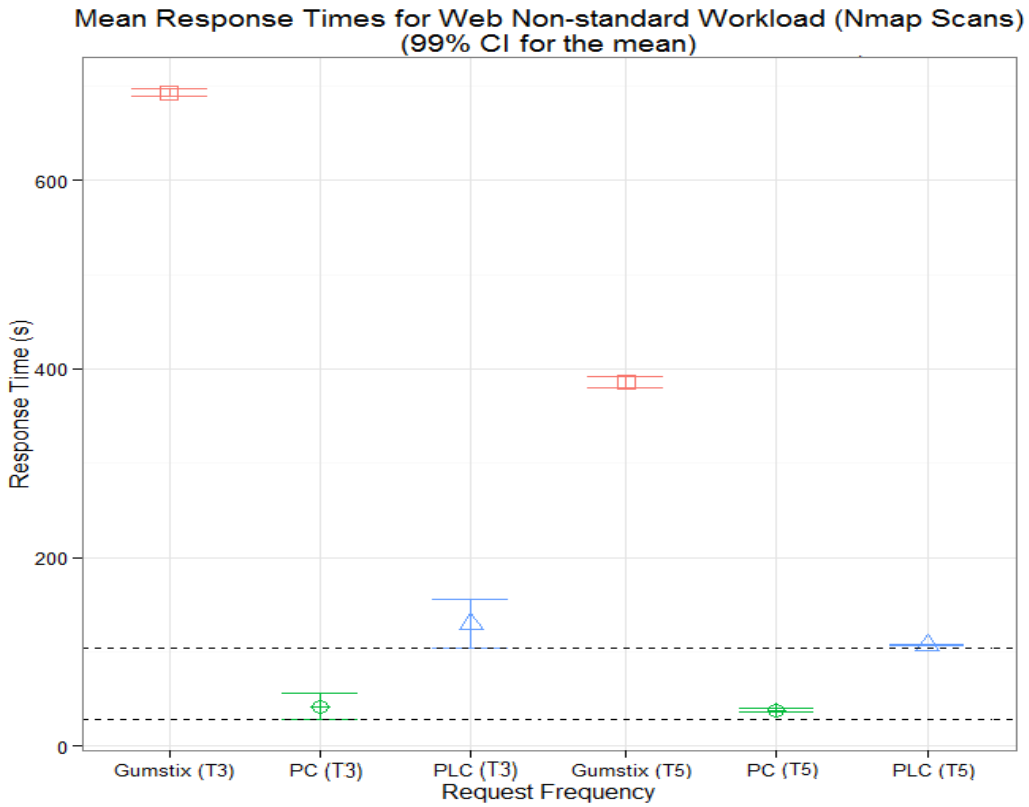


Figure 5.13: Mean response times for the Nmap scan for all devices at the Fast (T5) and Slow (T3) request frequencies.

#### 5.5.4 Timing Accuracy Results for HAP Non-Standard Workloads .

The results for the mean completion times for the Metasploit attack (HAP non-standard workload) for all three devices are shown in Table 5.14. During pilot studies to determine request frequency levels, the Metasploit workload frequency is artificially increased using *tcpreplay* to create multiple request frequency levels. The PLC break request frequency level is determined by recording the final request frequency at which the Metasploit exploit would no longer work on the target PLC.

Table 5.14: Timing accuracy results for the Gumstix and PC for the HAP non-standard (Metasploit) workload.

Scenario	Query	Request Frequency	Mean Response Time (s)	PLC Baseline	deviation from baseline	
15	Gumstix	slow	0.009335	0.014877	-0.0055	1.59 × faster
16	Gumstix	PLC Break	0.0086	0.016386	-0.0078	1.91 × faster
16a	Gumstix	PC Break	0.38192			
7	PC	slow	0.000927	0.014877	-0.0140	16.05 × faster
8	PC	PLC Break	0.000571	0.016386	-0.0158	28.70 × faster
8a	PC	PC Break	0.01428			

Figure 5.14 shows a graphical comparison of all three devices at the PLC break and slow (std) request frequency levels. Both the Gumstix and PC emulators mean response times are faster than the mean response time of the target PLC. Figure 5.14 also shows that the request frequency does not have a statistically significant effect on the target PLC, but it does have a significant effect on the response times of the Gumstix and the PC.

Figure 5.15 and Figure 5.16 respectively show the results for the mean response times for both PC and the Gumstix emulators at all three tested request frequency levels. The

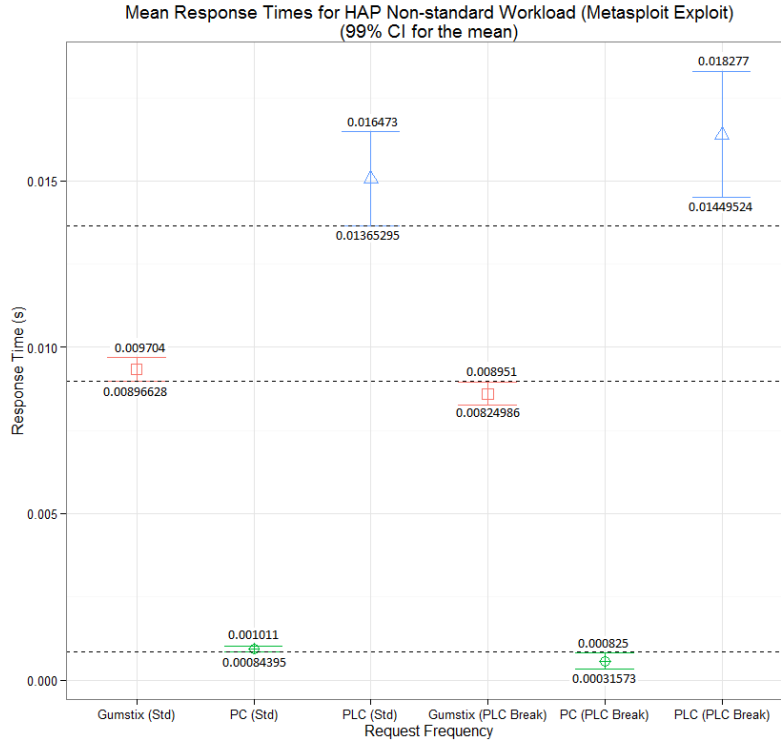


Figure 5.14: Results for Metasploit scenarios comparing mean response time of all three devices.

results show a very small statistical significance differential in the means for both emulators at the 99% CI. At the 99% CI, values are extremely close, with a difference on the order of  $10^{-4}$ . To provide clear analysis, the decision to drop the level from 99% CI is made because 95% CI is still acceptable.

Figure 5.16 shows two graphs of the same data, but at different scales. The lower graph is provided because it is unclear in the upper graph if the CIs of the means overlap or not. Based on these results, it is concluded that request frequency does have a significant effect on the mean response times for both the PC and Gumstix emulators for the HAP non-standard workload.



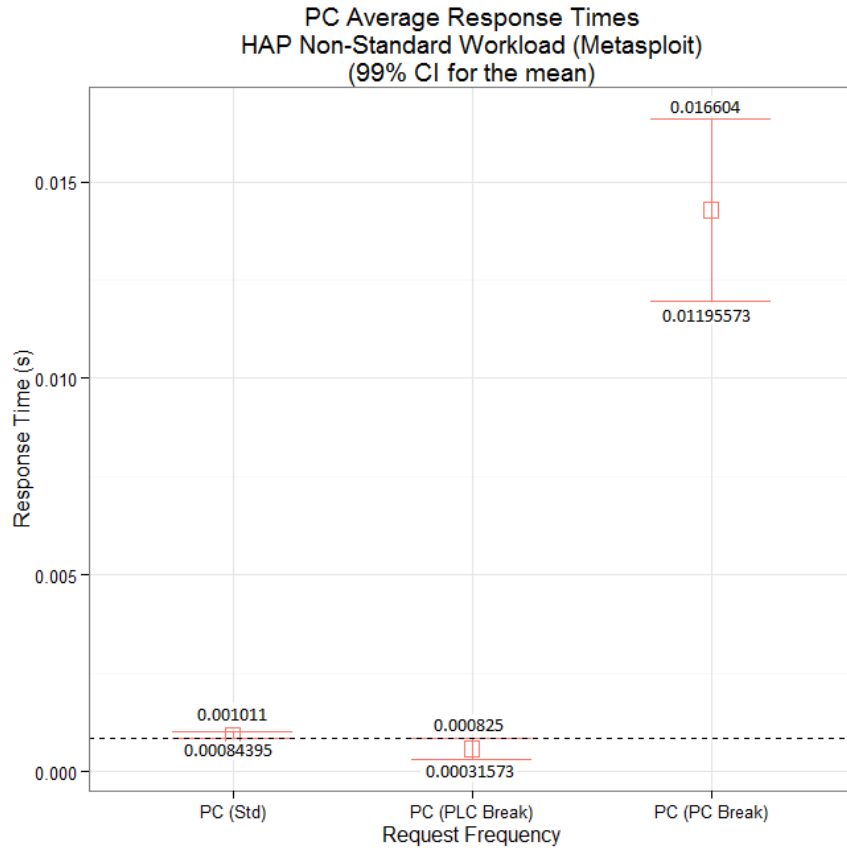


Figure 5.15: Mean response times for the PC for the HAP non-standard (Metasploit) workload at all three request frequency levels.

## 5.6 Summary

This chapter presents the results and analysis for the scenarios conducted on the emulator at different request frequencies for four different workload types. Results and analysis includes metrics for packet bytes, Nmap fields, successful Metasploit execution, and response time. Finally, logging results and analysis are also presented.

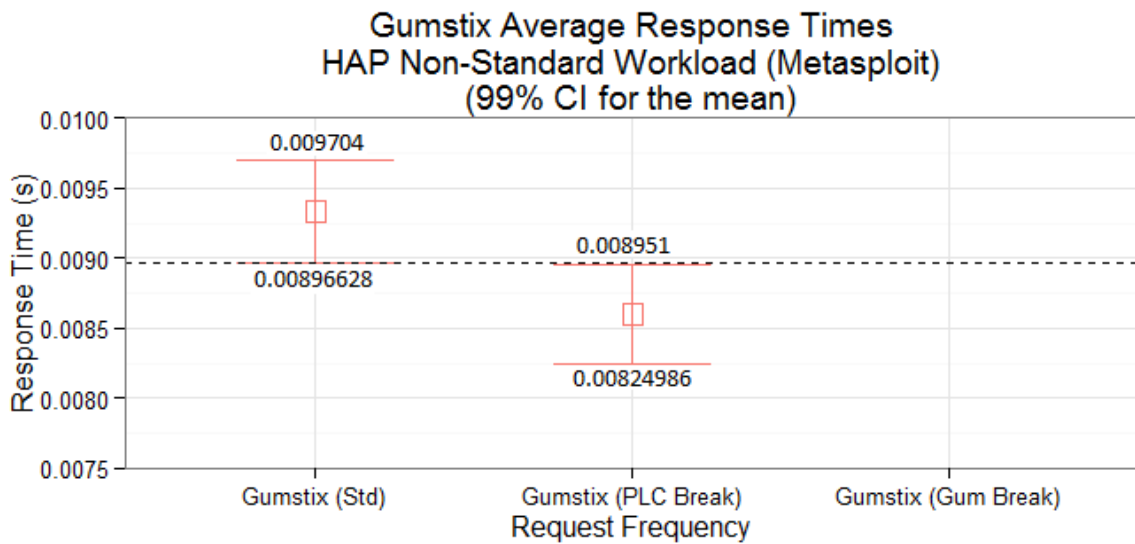
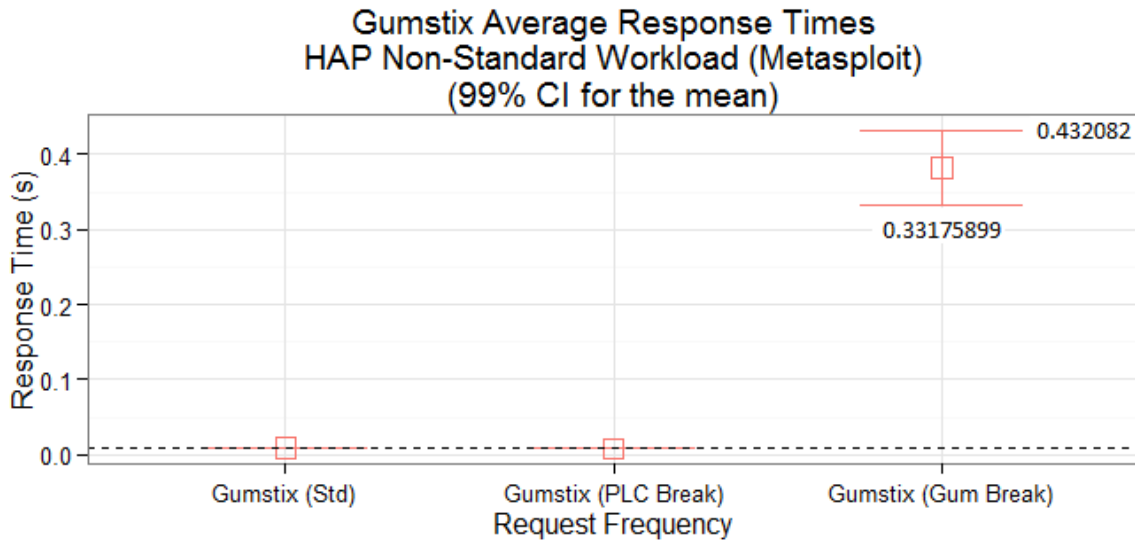


Figure 5.16: Mean response times for the Gumstix for the HAP non-standard (Metasploit) workload at all three request frequency levels.

## VI. Conclusions

### 6.1 Introduction

This chapter summarizes the overall conclusions of the research. Section 6.2 presents conclusions based on the results and analysis from Chapter 5. Section 6.3 discusses the significance of this research. Section 6.4 discusses recommendations and approaches for future research.

### 6.2 Research Conclusions

#### *6.2.1 Accuracy of Standard Queries at the Packet Level.*

At the packet level, the Gumstix and PC are both very accurate with both achieving 99.79% and 100% bytes of accuracy for the standard web and HAP workloads, respectively. For packet-level accuracy, confidence intervals are zero due to no variance among the trials conducted. Therefore, the performance of both platforms is equivalent with regard to packet-level accuracy. Accuracies less than 100% that are consistent and predictable enable fingerprinting of the emulated device. These rates however, are not expected to affect SCADA landscape research applications. The number of ICS honeypots accessible on the Internet is likely very few, and therefore there is no reason that an attacker would suspect an emulator to be anything but authentic. These results are detailed in Section 5.2.

#### *6.2.2 Accuracy at the Scanning and Attack Levels .*

At the scanning-level, both the Gumstix and PC platforms are equivalently accurate with each of them being 97.25% accurate at all request frequency levels. The fields that are not accurate are associated with the difference between the emulator's and target PLC's initial sequence number generator algorithm. To fingerprint the emulator based on these erroneous fields, significant additional information and analysis is needed. Considering the

fact that the purpose of tools like Nmap is to make such analysis unnecessary, it is not likely any user would second guess the Nmap results, and conduct such analysis.

At the attack-level, both platforms performed perfectly in all request scenarios, with each tested scenario resulting in 100% of attacks executed being successful. These results lead to the conclusion that the scanning-level and attack-level accuracies of both emulator platforms are adequate for all applications, including as a SCADA landscape research tool. These results are detailed in Sections 5.3 and 5.4.

### ***6.2.3 Accuracy at the Timing Levels .***

There are significant departures from the baseline timing measurements taken for both emulator platforms. For nearly all of the standard and non-standard scenarios, the Gumstix platform underperformed the target PLC with regard to mean response times, being as much as 98.6 times slower for some workloads when compared to the target PLC. The PC emulator platform however, over-performed the target PLC for all scenarios with regard to mean response times, being as much as 959.7 times faster for some workloads when compared to the target PLC. While it is trivial to increase the delay of the PC to more closely match the timing profile of target PLC, it is more difficult decrease the delay of the Gumstix.

A qualitative assessment of the emulator is based on scenarios that are likely to occur during standard use as an emulated honeypot. At request frequencies that would be typical in an actual attack (slow request frequencies), the Gumstix timing performance for the Web standard and HAP standard workloads are only 0.046s and 0.0007s slower when compared to the target PLC. An increase of 0.046s is very unlikely to be noticed by a user interacting with the emulator across the Internet, even if the emulator is implemented on the same node as an actual PLC. Therefore, both device platforms are suitable as honeypot emulators based on to timing performance to perform attack landscape research.

For applications that require a high level of timing accuracy, such as laboratory research, the PC platform is preferred over the Gumstix platform. With enhancements and optimizations for speed, it is speculated that the Gumstix platform could be brought within the timing range of the target PLC. These results are detailed in Section 5.5.

#### ***6.2.4 Logging Performance .***

Both platforms did exceptionally well logging all incoming and outgoing packets, with a success rate of 97.21% for the Gumstix and 99.99% for the PC. The most significant deficiencies of logging packets for both platforms occurred during the web non-standard (Nmap scan) workloads. However, because there are other methods to log (as opposed to logging every single packet), and it is not practical to log Nmap scans anyway (the size of the logs reaches 34 MB for a single OS scan), these results could be even better.

Based on the ability of both emulators to successfully log over 97% at such high data throughputs, it is likely that a subset of all incoming and outgoing packets will be logged at the same or better rates since logging every packet of an Nmap OS Fingerprint scan is not desired or practical. A better approach for these scans is to identify an active scan, and log vital packets only. This is much more desirable, and is left for future work. Therefore, it is concluded that the logging performance of both emulator platforms is considered adequate for all applications. These results are detailed in Sections 5.2, 5.3, and 5.4.

#### ***6.2.5 Performance Effects of Request Frequency.***

The effects of varying the request frequency during test scenarios differ among the various workloads. While the PC is able to perform with no significant loss of accuracy or logging performance at the PLC Break request frequency level in all scenarios, the Gumstix does suffer loss in both accuracy and performance. These results are likely linked with the Timing Accuracy results, and are only solved by decreasing the response time of the Gumstix emulator. As the request frequency approaches the Gumstix's fastest response rate, the response time increases and some queries stop receiving replies. It is likely the

Gumstix's receive buffer is being filled faster than it can process the responses until it reaches a plateau at which point queries start being dropped, and the Gumstix is never able to recover. It may be possible to increase the receive buffer of the Gumstix to reduce dropped packets, but the response times will grow significantly based on the results in Section 5.5.1. The only viable solution is to increase the Gumstix's top response speed, which also increases the emulator's timing accuracy.

The PLC break request frequency level is specifically designed to test the emulators at the target PLC's maximum limits. These rates are not indicative of standard or even common conditions a real PLC would ever see in operation unless it is under a DoS or other attack, or being fingerprinted. At standard request frequencies, Gumstix response times could easily be due to link delay, as opposed to device delay. These results are detailed in Chapter 5.

#### ***6.2.6 Impact of the Gumstix Platform.***

The results show that there is an impact to timing accuracy when the emulator operates on the Gumstix platform, and a marginal impact to logging capability. It is speculated that there are very few SCADA honeypots currently on the Internet. Therefore it is not likely that the minor timing differences between the target PLC and the Gumstix honeypot emulator would be attributed to anything besides latency. In other applications that warrant precise timing accuracy, a PC could be used to perform the emulation. These results are detailed in Chapter 5.

#### ***6.2.7 Final Assessment.***

Based on the results, extensive knowledge of the specific implementations of the protocols or timing profiles of the target PLC are required to identify and fingerprint the Gumstix device as a honeypot. Furthermore, there are very few actively maintained SCADA honeypot systems available and it is speculated that SCADA honeypots are not actively being looked for by attackers. These facts coupled with the experimental data suggest that

a honeypot implemented on a Gumstix emulator is suitable for applications in SCADA attack-landscape research. Further research should be conducted to increase the timing performance of the emulator before being deployed in applications that are sensitive to emulation timing discrepancies, such as SCADA laboratory research. In these types of applications, the PC platform should be used.

### **6.3 Significance of Research**

It has been shown that there are thousands of industrial control devices, including field level devices such as PLCs exposed on the Internet, but the extent that these devices are being exploited is still largely unknown. Furthermore, costs of even a modest industrial control systems comprised of several devices escalates very quickly, making research, education, and training of ICS attacks and protection methods significant investment.

The accuracy and logging performance of the PLC emulator implemented on the Gumstix single-board computer for this effort are quantified and compared against the emulator implemented on a standard PC and the actual PLC that the emulation seeks to imitate. The research suggests that the Gumstix platform provides highly accurate responses in all categories except response timing.

Therefore, the Gumstix and PC platforms are adequate for applications such as SCADA attack-landscape research and cyber sensing, two functions that can be implemented with SCADA honeypots. Even if the device is fingerprinted as a honeypot, valuable data will be gathered from the attack that will provide information on how to better hide honeypots in the future. For other applications that require a high degree of timing accuracy, such as SCADA laboratory research, it is suggested that the PC be used as the emulation platform.

Further development of the Gumstix emulator device is suggested to optimize timing performance. With these enhancements to the emulation software, the Gumstix platform

emulator device is speculated to show significant timing improvements, enabling use of the Gumstix platform for all emulation applications including laboratory research.

## **6.4 Future Work**

This section discusses various avenues for future research. Sections 6.4.1 through 6.4.7 discuss enhancements to this implementation of a SCADA/ICS field device emulator honeypot or design ideas that should be considered in the implementation of a new implementation. Sections 6.4.8 through 6.4.10 discuss other areas and ideas related to SCADA/ICS field device emulators and honeypots, but not necessarily this implementation.

### ***6.4.1 Enhance Timing Accuracy of Gumstix Emulator.***

In the current implementation, timing for the Gumstix Emulator is significantly slower than the target PLC. It is speculated that a significant amount of time is wasted when packets are pulled out of the Linux kernel into the user-space queuing programs through the iptables functionality. Preliminary results suggest that there are big gains to be made if avoiding the use of NFQUEUE targets is possible. This means eliminating the logging queue (*logging.o*), input filter queue (*infilter.o*), and output filter queue (*outfilter.o*). Elimination of the logging queue is discussed in Section 6.4.2, and elimination of the filter queues is possible with the implementation of a user-space TCP/IP stack, discussed in Section 6.4.3.

### ***6.4.2 Timing Improvements through Efficient Logging.***

In the current implementation, every packet that goes into or out of the emulator is sent to the Syslog server. There are many thousands of packets that are sent to and from the emulator during routine use, and as the results in Chapter 5 show, some packet loss occurs during high packet workloads such as Nmap OS fingerprint scans. Furthermore, a log file exceeds 30MB for a single scan. Not every packet from an OS scan has useful information, but some might. Rather than recording every packet, a better approach would



be to log only packets that are interesting. The C Syslog API that is currently used in *logging.o* makes it trivial to send log entries from any C program, and a similar API exists for Python. Therefore, it may be a better approach to selectively submit log entries based on the logic within the processes. There is a potential double-impact to response times (1) not all packets are logged, therefore processing time is diverted from the main processes to send logs, and (2) every packet does not need to come out of the kernel to be logged. These efficiencies could help increase the timing-level accuracy of the Gumstix in its current form.

There is a different approach to log every packet, if that functionality is desired. The current logging mechanism uses an NFQUEUE target to pull every single packet from the kernel into user-space to process, then sends it back into the kernel. This is very inefficient. A different, better approach would be to use the Libpcap C API to sniff the wire and log every packet. Every packet is still pulled out of the kernel, but it is never re-injected back into the kernel, potentially saving time.

#### **6.4.3 Enhance the Nmap Scan Handler.**

Currently, the device is limited to handling default configurations of Nmap OS scans. By adding additional logic to the *infilter.o* process, any number of configurations should be able to be handled by the emulator to successfully fool Nmap. For example, probes are sent to the first non-open port found by Nmap, which by default, is TCP port 1. Portions of the emulator scanner may fail if the default configuration of Nmap is modified, degrading scanning-level accuracy. Cases inside of *infilter.o* can be created to handle such non-default situations.

As an alternative to modifying the *infilter.o* process, the Honeyd emulator should be studied for its personality engine. Rather than using iptables and a Netfilter NFQUEUE target to handle Nmap OS probes, Honeyd implements an entirely separate user-space TCP/IP stack can be implemented to process the probes. In this way, every single feature of the stack is completely controllable. The Honeyd emulator can even accept Nmap

fingerprints to impersonate any device easily. However, there may be performance issues with implementing a user-space TCP/IP stack. These features could be integrated into the PLC emulator in place of the current *infilter.o* process and perform a performance comparison.

Either of these methods would likely still result in values for the ISN that are not statistically equivalent to the target PLC. Therefore, an automated method for capturing and duplicating ISN's should be implemented. This can be done in a number of ways, even trivially by recording several thousand ISN's from the target device and playing them back. Because of the potential for transient variables this technique may still not yield a distribution similar to the target PLC.

#### **6.4.4 Enhanced Configurability.**

Configuration can be thought of at two levels, device-level configuration and service level-configuration. Configuration in this implementation is defined only at the service level, meaning that each of the services offered by the emulator are configurable, but the device is hard programmed as a Koyo DirectLogic PLC. Service level configuration occurs via the *config.txt*, *ccmdictionary.txt* (see Chapter 3), and *kseqdictionary.txt* files to define the ways the emulator responds, but the definitions only apply to the Koyo DirectLogic PLC.

An emulator device such as this is defined by its scanning personality, and the services it offers to users. Device- level configuration is difficult because it requires accuracy in both the scanning definition and services definitions. Currently, both the scanning personality (implemented in *infilter.o*) and services (implemented in *ecom\_emulator.o*, *webserver.py*, and *modbus.py*) are hardcoded.

Future work would be to implement an industrial control emulator that is configurable at the device level. An emulator configurable at the device level would have the ability to look like whatever make or model PLC that is desired. The current method of manually

hard coding the personality and services makes this goal very time consuming. A process for automating the creation of definitions for any arbitrary device would be a significant step towards patenting and commercializing the industrial control emulator.

A potential way to accomplish this is to devise a method of recording queries and responses from target PLCs under normal use, similar to the procedures used in this research for CCM and K-Sequence protocols. Dictionaries would then be built for the device that match queries to responses, and when queried the emulator would simply respond with some variation of the dictionary response. With a sufficient number of recordings of queries and responses, it may be possible to reproduce the most common functionality of a device. An automated software tool could enumerate all possible entries to a device's website or manufacturers' tools and record the responses. These recordings would become potentially huge, but post recording analysis could be done to create heuristics that define many-to-one and one-to-many query-response relations, potentially reducing the size of the dictionary.

A variation of this technique would be to first implement base protocols, and use recordings to adjust them according to the definitions of the specific device. Baseline industrial protocols would be implemented in software, much like the HAP protocol is implemented for this research effort, or the Modbus protocol for a previous research effort [Bur12]. Once the baselines are established, a method for identifying and implementing specific implementations of the protocol between different manufacturers must be done. For example, there is a Modbus specification, but all manufacturers implement its features differently. To design a device configurable PLC, these differences in implementations must be defined. In this way, when the emulator is configured to be a Siemens PLC, the emulator would use the Siemens Modbus implementation for all of its Modbus communications. An automated method for identifying protocol deviations or implementations, and incorporating them into the emulator definitions would

be the alternative to hard coded process definitions. This could be done by creating a software device that enumerates all possible protocol variations, records their results, and incorporates the results into the emulator definitions for that device. This is very similar to how Nmap accomplishes its OS Fingerprint scans.

This is not trivial because of CRCs in headers, wide ranges of valid input values (like in web based configurations), and other protocol features specifically designed to prevent replay attacks. Furthermore, diversity of control devices and their complex operation suggests a fully automated system may not be possible; some features will still need to be coded manually.

#### ***6.4.5 Automated Capture of Ladder Logic for Arbitrary Programs.***

The current method used to emulate the ladder logic program uploads from the emulator to DirectSOFT5 is to record the queries and responses for an upload from the target PLC, and place those corresponding query and response values into *kseqdictionary.txt*. This method is labor intensive and restricts the operation to a static ladder logic program that cannot be modified by a user. Since the K-Sequence protocol is proprietary, reversing the ladder logic functionality must be done in order to add support for any arbitrary upload or download program.

Based on this thesis, reversing this portion of the protocol may be possible. It does not appear that there is any sort of sequence number or timing component to the K-Sequence functionality that prevents replaying of the data, since, that is the exact method used in the current implementation. Therefore, by methodically altering ladder logic programs through DirectSOFT5, and observing the K-Sequence queries and responses the protocol may be reversed.

Considering that the K-Sequence protocol is used to modify memory values, it may simply be the case that a simple record-store-replay done by the emulator is possible. That is, the user modifies the ladder logic program and sends it to the emulator, which stores

the new ladder logic values in *kseqdictionary.txt*. When queried for ladder logic programs in the future, the new values would simply be read from *kseqdictionary.txt* and sent to DirectSOFT5.

#### ***6.4.6 Evaluation of Other SCADA Honeypots.***

Apply the techniques for measuring accuracy developed in this thesis, to any other ICS honeypots, such as the SCADA honeypot and honeywall developed by Project Basecamp [Dig12b]. This requires acquisition of a target PLC to create a baseline. If the studied device is discovered as fingerprintable based on the results, an automated tool can be used to determine if the device is real or an emulation. Enhancements can also be made to make the selected device more accurate.

#### ***6.4.7 Implementation as Evidence Collection Tool.***

As discussed in Section A.3, one application of an industrial control emulator is to perform the same type of function as a fake bomb used by the FBI. In the same capacity, the evidence collection device has potential to be deployed by law enforcement and when activated or accessed by the suspected terrorist, they would be arrested. The device and device log could then be used in court to convict the individual who thought he was accessing an actual network on perhaps a water treatment plant or power company. To prove that it was the suspect who tried to access the device, a complex (and un-guessable) password could be implemented and passed to the suspect. The device would log this password being attempted as well as actions performed on the fake PLC as evidence to convict a suspect.

Rapid deployment and re-configurability are top considerations for this device because it must be able to look like a device on any arbitrary SCADA or ICS network. Enhancements must be made to automate the configuration at the device level to allow for complete device flexibility. The implemented levels of accuracy must also be reconsidered. Since this tool is only a small aspect of the FBI's sting operation, in these applications, it is

more likely that a potential attacker would exclusively use the web interface to perform their actions, therefore extensive and accurate emulation of this portion is likely more important than a high level of packet byte accuracy. Additional important design considerations for this type of device are robustness and scalability, overshadowing cost. Therefore, emulation on a dedicated Laptop PC might be the best approach for accurate emulation.

#### **6.4.8 *Other Areas.***

There are also several less-developed, but potentially interesting research areas involving this implementation of the ICS field device honeypot emulator. Future research could include performance of human experiments to test the authenticity of the device to establish if the emulator feels authentic compared to the target PLC. In addition, testing the implementation of the device on other single board or embedded platforms such as the Raspberry Pi, Beagle Board, or newer and faster Gumstix module. Compare these results to the results of this research. And finally, generalizing the parsing tools to be useful for comparing arbitrary .pcap files. A graphical implementation would be very useful for determining quickly the areas that are not exactly the same. Currently, the pcap parser is very specific to this application and a slight deviation in the order of arrival of response packets requires preprocessing to get packets in the proper order.

#### **6.4.9 *Hardware Emulation.***

Though it has not been exhaustively studied, it may be possible based on open-source knowledge to pick a popular PLC and create a scaled-down hardware emulation. This type of emulation is superior to application or system based emulations because the device will be easily configurable simply based on applying the correct firmware. As a starting point, the hardware must be identified completely and reverse engineered. If hardware and architectures cannot be identified, fingerprinting techniques might be used to determine if the TCP/IP stack is an open-source embedded stack (Like  $\mu$ IP or lwIP [Dun13]) to then determine the hardware. After the hardware is determined, a series of reverse engineering

intensive projects could focus on designing the emulator. This technique is similar to those used to reverse engineer and emulate video game consoles [Sal05].

#### ***6.4.10 Implementation of a Large-Scale Network Using the Emulator.***

A final aspect of future work is to demonstrate the scalability of these devices for field-device level research. One suggestion is to research how many emulations can currently run on one PC without loss of accuracy. In addition, research can be conducted to determine the best approach to scalability: a single virtual machine per emulation, like the *Project Basecamp* SCADA honeypot, or many emulations in a single virtual machine. There is also the *Megatux approach* [Mar09]: The Megatux project claims to implement 1 Million computers from only 4480 Intel processors. This suggests that 223 copies of computers are running on one Intel processor. Investigation would include the possibility of using this technique to implement ICS field device emulators. Finally, an investigation into which portions of an industrial control honeynet should be emulated, and which should be real hardware is warranted. In studies done on Honeyd, emulated network components were shown to have predictable link delays that made fingerprinting honeypots implemented with Honeyd trivial [FYC06].

## References

- [AbW08] M. Abrams and J. Weiss. "Malicious control system cyber security attack case study—Maroochy water services, Australia." 2008. [http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study\\_report.pdf](http://csrc.nist.gov/groups/SMA/fisma/ics/documents/Maroochy-Water-Services-Case-Study_report.pdf).
- [Aut11] DL06 Micro PLC User Manual, 3rd Edition, Rev. B. (03/12/2012), 2011. <http://www.automationdirect.com/static/manuals/d006userm/appxk.pdf>
- [Aut12a] AutomationDirect.com. "Getting started with DirectNET communications." (02/12/2012), 2012. <http://www.automationdirect.com/static/manuals/dadnet/ch2.pdf>
- [Aut12b] Automation Direct.com. DirectLogic 405 PLC. (8/15/2012), 2012. [http://www.automationdirect.com/adc/Shopping/Catalog/Programmable\\_Controllers/DirectLogic\\_Series\\_PLCs\\_\(Micro\\_to\\_Small,\\_Brick\\_-a-\\_Modular\)/DirectLogic\\_405\\_\(Small\\_Modular\\_PLC\)](http://www.automationdirect.com/adc/Shopping/Catalog/Programmable_Controllers/DirectLogic_Series_PLCs_(Micro_to_Small,_Brick_-a-_Modular)/DirectLogic_405_(Small_Modular_PLC)).
- [Aut12c] Automation Direct.com. DirectSOFT5 Programming software. (7/31/2012), 2012. <http://www.automationdirect.com/directsoft>.
- [Ayu10] Netfilter.org. The netfilter.org "iptables" project. 2010. <http://www.netfilter.org/projects/iptables/index.html>
- [Ber12] D. Berman. "Emulating industrial controls system devices using gumstix technology." AFIT Master's Thesis. 2012. <http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA563104>
- [BMS11] W. J. Broad, J. Markoff and D. E. Sanger. "Israeli test on worm called crucial in iran nuclear delay." The New York Times. 2011. <http://www.nytimes.com/2011/01/16/world/middleeast/16stuxnet.html?pagewanted=all>.
- [Boy09] S. A. Boyer, SCADA: Supervisory Control and Data Acquisition. International Society of Automation, 2009.
- [Cap10] Capula Limited. Case study: Alstom power environmental. (04/21/2012), 2010. <http://www.capula.co.uk/capula/files/be/be07dbbf-68e2-4609-ac48-f545f5755e05.pdf>.
- [CKS+06] J. Cappaert, N. Kisserli, D. Schellekens, and B. Preneel. "Self-encrypting code to protect against analysis and tampering," In 1st Benelux Workshop on Information and System Security, (<https://www.cosic.esat.kuleuven.be/publications/article-811.pdf>), 2006.
- [Cla10] M. Clayton. "How stuxnet cyber weapon targeted Iran nuclear plant." The Christian Science Monitor 2010. <http://www.csmonitor.com/USA/2010/1116/How-Stuxnet-cyber-weapon-targeted-Iran-nuclear-plant>.



- [CNM08] A. Carcano, I. Nai Fovino, M. Masera and A. Trombetta. "Scada malware, a proof of concept." European commission joint research centre. Rome, IT. 2008. [http://critis08.dia.uniroma3.it/pdf/CRITIS\\_08\\_27.pdf](http://critis08.dia.uniroma3.it/pdf/CRITIS_08_27.pdf).
- [Con13] L. Constantin. "Brace for more attacks on industrial systems in 2013." (2/10/2013), 2013. PCworld.com. <http://www.pcworld.com/article/2023249/brace-for-more-attacks-on-industrial-systems-in-2013.html>.
- [Dhs12] U.S. Department of Homeland Security. ICS-ALERT-12-020-05—KOYO ECOM100 MULTIPLE VULNERABILITIES. 2012. [http://www.us-cert.gov/control\\_systems/pdf/ICS-ALERT-12-020-05.pdf](http://www.us-cert.gov/control_systems/pdf/ICS-ALERT-12-020-05.pdf)
- [Dhs12a] U.S. Department of Homeland Security. "Project SHINE." ICS-CERT Monitor. (2/11/2013), Dec 2012. [http://ics-cert.us-cert.gov/pdf/ICS-CERT\\_Monthly\\_Monitor\\_Oct-Dec2012.pdf](http://ics-cert.us-cert.gov/pdf/ICS-CERT_Monthly_Monitor_Oct-Dec2012.pdf)
- [Dig12a] Digital Bond Inc. Koyo / DirectLOGIC ECOM. (2/ 2/ 2012), 2012. <http://www.digitalbond.com/tools/basecamp/koyo-directlogic-ecom/>.
- [Dig12b] Digital Bond Inc. SCADA honeynet. (5/ 27/ 2012), 2012. <http://www.digitalbond.com/tools/scada-honeynet/>.
- [Doe02] U.S. Department of Energy. 21 steps to improve cyber security of SCADA networks. 2002. <http://www.oe.netl.doe.gov/docs/prepare/21stepsbooklet.pdf>.
- [DTO+06] C. Davis, J. Tate, H. Okhravi, C. Grier, T. Overbye, D. Nicol. "SCADA Cyber Security Testbed Development," in Power Symposium, 2006. NAPS 2006. 38th North American, pp. 483-488, 2006.
- [Dug05] D. P. Duggan. Penetration testing of industrial control systems. Sandia National Laboratories. Albuquerque, New Mexico. 2005. [http://energy.sandia.gov/wp/wp-content/gallery/uploads/sand\\_2005\\_2846p.pdf](http://energy.sandia.gov/wp/wp-content/gallery/uploads/sand_2005_2846p.pdf).
- [Dun13] A. Dunkel. uIP and lwIP, (1/21/13), 2013. <http://dunkels.com/adam/>.
- [Dur13] N. Duara. "Mohamed Mohamud, suspected terrorist in oregon christmas tree plot, entrapped by FBI, defense says." The Huffington Post. (17 Jan 2013), 2013. [http://www.huffingtonpost.com/2013/01/12/mohamed-mohamud-terrorist-oregon-christmas-tree-entrapped-trail\\_n\\_2463134.html](http://www.huffingtonpost.com/2013/01/12/mohamed-mohamud-terrorist-oregon-christmas-tree-entrapped-trail_n_2463134.html)
- [Emu12] Emulab.net, Network Emulation Testbed, University of Utah, Utah [www.emulab.net](http://www.emulab.net), 2012.
- [Eyc12] H. Eychenne. Iptables(8) Linux man page. 2012. <http://linux.die.net/man/8/iptables>
- [FAA+09] G. Franceschinis, E. Alata, J. Antunes, H. Beitollah, A. B. Neves, M. Correia, W. Dantas, G. Deconinck, M. Kaaniche, N. Neves, V. Nicomette, P. Sousa and P. Verissimo. "Experimental validation of architectural solutions." 2009. <http://crutial.rse-web.it/content/files/Documents/Deliverables%20P3/WP5-D20-final.pdf>.

- [FMC11] N. Falliere, L. O. Murchu and E. Chien. W32.Stuxnet Dossier. 2011. [http://www.symantec.com/content/en/us/enterprise/media/security\\_response/whitepapers/w32\\_stuxnet\\_dossier.pdf](http://www.symantec.com/content/en/us/enterprise/media/security_response/whitepapers/w32_stuxnet_dossier.pdf).
- [FYC06] X. Fu, W. Yu, D. Cheng, X. Tan, K. Streff, and S. Graham "On Recognizing Virtual Honeypots and Countermeasures," Proceedings of the 2nd annual IEEE International Symposium on Dependable, Autonomic and Secure Computing, 2006.
- [Gao09] United States Government Accountability Office, "Information security: Cyber threats and vulnerabilities place federal systems at risk," Tech. Rep. GAO-09-661T, 2009.
- [Gei10] General Electric Intelligent Platforms. Programmable Control Products, Series 90\*PLC Serial Communications (02/ 10/ 2012). Tech. Rep. GFK-0582D, 2010. [http://support.ge-ip.com/support/resources/sites/GE\\_FANUC\\_SUPPORT/content/staging/DOCUMENT/0/DO225/en\\_US/1.0/gfk0582d.pdf](http://support.ge-ip.com/support/resources/sites/GE_FANUC_SUPPORT/content/staging/DOCUMENT/0/DO225/en_US/1.0/gfk0582d.pdf)
- [Gum12a] Gumstix inc. Gumstix overo COMS open source products. (5/ 25/ 2012), 2012. <https://www.gumstix.com/store/index.php?cPath=33>.
- [Gum12b] Gumstix inc. Gumstix overo- computer-on-module. gumstix.org. 2012. [http://gumstix.org/images/overo\\_signals\\_latest.pdf](http://gumstix.org/images/overo_signals_latest.pdf).
- [Gum12c] Gumstix inc. Factsheet: Overo EarthSTORM COM. Gumstix.org.2012. [https://www.gumstix.com/store/product\\_info.php?products\\_id=264](https://www.gumstix.com/store/product_info.php?products_id=264).
- [Hod11] H. Hodson. "Hackers accessed city infrastructure via SCADA – FBI." Information Age 2011. <http://www.information-age.com/channels/security-and-continuity/news/1676243/hackers-accessed-city-infrastructure-via-scada-fbi.shtml>.
- [HoL10] S. Hong and M. Lee. Challenges and direction toward secure communication in the SCADA system. Presented at Communication Networks and Services Research Conference (CNSR), 2010 Eighth Annual. 2010.
- [Hos12] Hosteng.com. Ethernet Software Developers Kit source code for ECOM100s. 2012. Available by request at: [http://www.hosteng.com/Ethernet\\_SDK.htm](http://www.hosteng.com/Ethernet_SDK.htm).
- [HoW05] J. van der Hoeven, H. van Wijngaarden. "Modular emulation as a long-term preservation strategy for digital objects." Koninklijke Bibliotheek, The Hague, The Netherlands. (1/16/2013) 2005. <http://iaw.europarchive.org/05/papers/iaw05-hoeven.pdf>.
- [Ine03] INEEL, Newsletter: Need to Know, vol 3(2), pp. 1-3, INEEL Establishes National SCADA Testbed, Idaho, <http://www.inl.gov/nationalsecurity/newsletter/docs/jan2003.pdf>, 2003.
- [Jae08] JAEC. Modbus protocol. (4/ 22/ 2012), 2008. <http://www.jaec.info/Home%20Automation/Protocols-buses-house/Modbus-Protocol/modbus-protocol.php>.

- [Ken09] H. Kenchington , Factsheet: National SCADA Test Bed, Washington, DC [http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB\\_Fact\\_Sheet\\_FINAL\\_09-16-09.pdf](http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/NSTB_Fact_Sheet_FINAL_09-16-09.pdf), 2009.
- [Kna11] E. D. Knapp, Industrial Network Security: Securing Critical Infrastructure Networks for Smart Grid, SCADA, and Other Industrial Control Systems. Waltham, MA: Elsevier, 2011.
- [Koy12] Koyo Electronics Inc. Factsheet: History. <http://www.automationdirect.com/static/press/koyo.html>. 2012.
- [Lee05] B. Lee. "Authenticity, Accuracy and Reliability: Reconciling Arts-related and Archival Literature." University of Windsor. Ontario Canada. 2005. [http://www.interpares.org/display\\_file.cfm?doc=ip2\\_aar\\_arts\\_lee.pdf](http://www.interpares.org/display_file.cfm?doc=ip2_aar_arts_lee.pdf) ).
- [Lev11] E. P. Leverett. Quantitatively Assessing and Visualizing Industrial System Attack Surfaces. 2011. <http://www.cl.cam.ac.uk/~fms27/papers/2011-Leverett-industrial.pdf>.
- [Lyo09] G.F. Lyon. Nmap Network Scanning. Chapter 8: Remote OS Detection. nmap.org. <http://nmap.org/book/osdetect.html#osdetect-intro>. 2009.
- [Lyo12] G.F. Lyon. Nmap security scanner v5.12, <http://nmap.org/>. 2012.
- [Mar09] J. Markoff, "Researchers try to stalk botnets used by hackers," (2/8/2013), 2009. [http://www.nytimes.com/2009/07/28/science/28comp.html?\\_r=0](http://www.nytimes.com/2009/07/28/science/28comp.html?_r=0).
- [Met12] Metasploit Project, Metasploit framework. <http://www.metasploit.org>. 2012
- [Mod06] Modbus.org. "MODBUS over serial line specification and implementation guide V1.02." www.Modbus.org. 2006. [http://www.modbus.org/docs/Modbus\\_over\\_serial\\_line\\_V1\\_02.pdf](http://www.modbus.org/docs/Modbus_over_serial_line_V1_02.pdf).
- [Mot07] Motorola. "SCADA systems." 2007. [http://www.motorola.com/web/Business/Products/SCADA%20Products/\\_Documents/Static%20Files/SCADA\\_Sys\\_Wht\\_Ppr-2a\\_New.pdf](http://www.motorola.com/web/Business/Products/SCADA%20Products/_Documents/Static%20Files/SCADA_Sys_Wht_Ppr-2a_New.pdf).
- [MPH12] J. Marzulli, R. Parascandola, B. Hutchinson. "Federal reserve bomb plot busted." New York Daily News. (16 Jan 2013). 2012. <http://www.nydailynews.com/new-york/man-arrested-federal-reserve-bomb-plot-article-1.1185825>
- [MVD10] T. Morris, R. Vaughn, Y. Dandass. "A testbed for scada control system cybersecurity research and pedagogy." Mississippi State University, Mississippi. <http://www.ece.msstate.edu/~morris/Morris-Abstract-19.pdf>. 2010.
- [Nes12] Nessus 5. <http://www.tenable.com/products/nessus>. 2012.
- [Oud10] L. Oudot, "Fighting Internet Worms with Honeypots," 2010. <http://www.symantec.com/connect/articles/fighting-Internet-worms-honeypots>.
- [PeP09] D. Peck, D. Peterson. "Leveraging ethernet card vulnerabilities in field devices." Proceedings of SCADA Security Scientific Symposium, Miami, USA. 2009.

- [Pet11] D. G. Peterson. S4 presentation by Eireann Leverett. 2011. <http://www.digitalbond.com/2012/02/09/s4-video-denial-of-service-ics-on-the-internet/>.
- [Plc12] Plcdev.com. Schneider electric modicon history. (05/28/2012), 2012. [http://www.plcdev.com/schneider\\_electric\\_modicon\\_history](http://www.plcdev.com/schneider_electric_modicon_history).
- [PoF05] V. Pothamsetty and M. Franz. SCADA HoneyNet project: Building honeypots for industrial networks. (5/27/2012), 2005. <http://scadahoneynet.sourceforge.net/>.
- [Pos81] J. Postel. Internet control message protocol, rfc 792. internet engineering task force, september 1981. <http://www.ietf.org/rfc.html>.
- [PrH07] N. Provos and T. Holz, Virtual Honeypots: From Botnet Tracking to Intrusion Detection. Addison-Wesley Professional, 2007.
- [QMJ+09] C. Queiroz, A. Mahmood, H. Jiankun, Z. Tari, Y. Xinghuo, "Building a SCADA Security Testbed," in Network and System Security, 2009. NSS '09. Third International Conference on Computing and Processing, pp. 357-364. 2009.
- [RaB12] R. Radvanovshy, J. Brodsky, SHINE Project. (2/11/2013), 2012. <http://news.infracritical.com/pipermail/scadasec/2012-December/010162.html>.
- [Rap12] Rapid7.com. Press Release, "New Metasploit Module to Exploit GE PLC SCADA Devices," January 19, 2012, 2012. <http://www.rapid7.com/news-events/press-releases/2012/2012-new-metasploit-module-to-exploit.jsp>.
- [Ras11] F. Y. Rashid. Russian firm dumps SCADA zero-day exploits into the wild. (04/22/2012), 2011. <http://www.eweek.com/c/a/Security/Russian-Firm-Dumps-SCADA-ZeroDay-Exploits-into-the-Wild-685614/>.
- [Rob12] P. Roberts. Project basecamp adds stuxnet-type attack module to metasploit. (04/29/2012), 2012. [http://threatpost.com/en\\_us/blogs/project-basecamp-adds-stuxnet-type-attack-module-metasploit-040512](http://threatpost.com/en_us/blogs/project-basecamp-adds-stuxnet-type-attack-module-metasploit-040512).
- [Roc12] Rockwell Automation, Factsheet: "GuardPLC safety control programming software." Milwaukee, Wisconsin <http://www.ab.com/en/epub/catalogs/3377539/5866177/5985760/6388285/6388301/5985774/5985818/Introduction.html>, 2012.
- [Rot00] J. Rothenberg. "Using emulation to preserve digital documents". Koninklijke Bibliotheek, The Hague, The Netherlands. (16 January 2013), 2000. <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.23.6652&rep=rep1&type=pdf>.
- [Rot99] J. Rothenberg. "Avoiding technological quicksand: finding a viable technical foundation for digital preservation. Washington, DC: CLIR. 1999. <http://www.clir.org/pubs/abstract/pub77.html>
- [Sal05] N. Salmoria. "MAME project history." (1/17/2013), 2005. <http://mamedev.org/history.html>.
- [San11] R. Santamarta. "Trojans: attacking the grid + advantech vulnerabilities." 2011. <http://seclists.org/bugtraq/2011/Mar/214>.

- [Saw11] J. H. Sawyer. "0-day SCADA exploits released, publicly exposed servers at risk." Dark Reading. 2011. [http:// www.darkreading.com/ blog/ 231601549/ 0-day-scada-exploits-released-publicly-exposed-servers-at-risk.html#](http://www.darkreading.com/blog/231601549/0-day-scada-exploits-released-publicly-exposed-servers-at-risk.html#).
- [SFK11] K. Stouffer, J. Falco and K. Ken. "Guide to supervisory control and data acquisition (SCADA) and industrial control systems security." technical report. National Institute of Standards. 2011. [http:// csrc.nist.gov/ publications/ nistpubs/ 800-82/ SP800-82-final.pdf](http://csrc.nist.gov/publications/nistpubs/800-82/SP800-82-final.pdf).
- [Sho10] Shodanhq.com. Manpage: SHODAN. (1 May 2012), 2010. [www.shodanhq.com/ help](http://www.shodanhq.com/help).
- [Sie12] Siemens Automation, Factsheet: "LOGO! software overview", Princeton, New Jersey (<http://www.automation.siemens.com/mcms/programmable-logic-controller/en/logic-module-logo/logo-software/Pages/Default.aspx>),2012.
- [Sla03] J. Slats. "Emulation: context and current status." Digital Preservation Testbed, The Hague, The Netherlands, 2003. [http:// en.nationaalarchief.nl/ kennisbank/ emulation-context-and-current-status-2003](http://en.nationaalarchief.nl/kennisbank/emulation-context-and-current-status-2003) (accessed 16 January2013)
- [Sof12] Softwaretoolbox.com. 2012. [http:// support.softwaretoolbox.com/ ci/ fattach/ get/ 12282/ 0/ filename/Communicating+to+GE+Series+One+PLCs.pdf](http://support.softwaretoolbox.com/ci/fattach/get/12282/0/filename/Communicating+to+GE+Series+One+PLCs.pdf)
- [Spi03a] L. Spitzner. "Honeypots definitions and values of honeypots." (05/ 08/ 2012), 2003. [http://www.tracking-hackers.com/ papers/ honeypots.html](http://www.tracking-hackers.com/papers/honeypots.html).
- [Spi03b] L. Spitzner. "Honeypots: Tracking Hackers." Addison-Wesley, 2003.
- [Ste93] R. Stevens. "TCP/ IP illustrate (vol. 1): the protocols." chapter 20, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1993.
- [Syn12] Synfin.net. TCP replay tool. [http:// tcpreplay.synfin.net/](http://tcpreplay.synfin.net/)
- [Tho05] Thornton. "A user's guide to configuring serial ports for DirectLogic PLCs: a two part series." Automationnotebook.com. (03/ 12/ 2012). Tech Thread, Issue 3. 2005. [http:// www.automationnotebook.com/ 2005-Issue-3/ tech-thread-January2005.html?learnsite](http://www.automationnotebook.com/2005-Issue-3/tech-thread-January2005.html?learnsite)
- [Vau09] R. Vaughn. Presentation: "Information assurance teaching, service and research." Mississippi State University, 2009. Mississippi [http://www.cse.msstate.edu/~cse8011/ presentations /Security\\_Overview\\_Seminar\\_09.pdf](http://www.cse.msstate.edu/~cse8011/presentations/Security_Overview_Seminar_09.pdf).
- [Vmw13] Vmware.com. "Why choose VMware?" (1/17/2013), 2013. [http://www.vmware.com/ products/ workstation/ overview.html](http://www.vmware.com/products/workstation/overview.html)
- [Wad11] S.M. Wade. "SCADA Honeynets: The attractiveness of honeypots as critical infrastructure security tools for the detection and analysis of advanced threats." (1/17/2013), 2011. <http://lib.dr.iastate.edu/cgi/viewcontent.cgi?article=3130&context=etd>
- [Wig12] R. Wightman. "Koyo/ automation direct vulnerabilities." 2012. [http:// www.digitalbond.com/blog/ 2012/ 02/ 08/ koyoautomation-direct-vulnerabilities/](http://www.digitalbond.com/blog/2012/02/08/koyoautomation-direct-vulnerabilities/)

[ZhN03] P. Zheng, L.M. Ni. "EMPOWER: a network emulator for wireline and wireless networks," INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications.IEEE Societies , vol.3, no., pp. 1933- 1942 vol.3, 30 March-3 April 2003,[http://gargoyle.arcadia.edu/Mathcs/zhengpei/publications/INFOCOM03\\_zheng.pdf](http://gargoyle.arcadia.edu/Mathcs/zhengpei/publications/INFOCOM03_zheng.pdf).

## **Appendix A: Additional Applications for Industrial Control Emulators**

### **A.1 Sensor in Industrial Networks**

ICS cyber sensors can be compared to production-class honeypots, which directly enhance the security of operational networks. Production-class honeypot systems primarily function as security tools in conjunction with firewalls and Intrusion Detection and Prevention Systems (IDS/IPS) managed by the same IT staff that operates the corporate network. When activity is logged on a production honeypot system, the information is used to quickly identify and respond to an active threat or newly identified vulnerability. The IT staff responds relatively quickly in response to honeypot activity, the same way they might respond to a barrage of malicious packets detected by the IDS.

Unlike SCADA honeypots for attack-landscape research, cyber sensor honeypots do not warrant as much effort in maintenance and design. Because the honeypot emulator devices themselves have no production value, any activity on them is considered suspicious and is investigated.

Honeypots can also prevent attacks both directly and indirectly. Depending on the intention of the honeypot, software might be used to detect and slow down or even stop attacks being carried out by worms [Oud10]. Worms are automated viruses that seek to spread and infect as many hosts as quickly and efficiently as possible. Honeyd is one such honeypot implementation that takes active measures to slow down and stop worms from spreading, giving IT administrators more time to react to the attack. Honeyd is an example of what is known as a low-interaction honeypot which emulates limited functionality and responds to a limited set of attacker commands. In this way the worm loses time on the honeypot by attacking non-production computers [Oud10].

## **A.2 Laboratory Research**

Industrial control emulators in a laboratory environment have many of the same advantages over simulations as using real hardware. Accurate emulators not only implement the responses to user queries in the same manner as their real-hardware counterparts, but also emulate the timing profile. When timing and protocol responses have been accurately emulated, the devices can be used in the same way as real hardware for research experiments with less cost, setup and maintenance effort.

Accurate emulators also have several advantages over real hardware devices. In addition to reduced cost, emulators can be much more robust because error conditions such as buffer overflows and bad firmware uploads can be recovered from easily. While the real device might be permanently damaged by such vulnerabilities, an emulated device is virtually impervious because the services have control over which vulnerabilities are ignored or duplicated, depending on the purpose of the emulator and severity of the vulnerability.

Uses for emulated research networks include the study of automated malware propagation, network reactions to wide scale scanning, denial of service (DoS) techniques, pivot attacks from human machine interfaces to devices or devices to devices, and more.

## **A.3 Evidence Collection Tool**

Apart from cyber sensing, an industrial control field device or PLC emulator can be used as an evidence collection device. This application is comparable to the way the Federal Bureau of Investigation (FBI) uses fake bombs to catch terrorists in the act of executing acts of terrorism [Dur13, MPH12]. Since this application has never been implemented before, and is out of scope of this research, it is further developed as future work in Section 6.4.7.

### ***A.3.0.1 Education.***

In the classroom, an industrial control emulator can teach the basics of SCADA and Industrial ICS with the goal of base-level knowledge. Currently, the only way to gain hands



on experience with industrial control networks is to implement an actual system which, as previously mentioned, can be prohibitively expensive. The same is true for features that are particular to physical devices like uploading and downloading of ladder logic programs. Though programming simulators are available through several manufacturers to learn and test ladder logic programs, there is no known way to test network communications and functionality without having an actual industrial control device to perform those operations [Sie12, Roc12].

An industrial control emulator used for education would not necessarily need to be a perfectly accurate emulation. Since the device is an acknowledged emulator, timing, protocol deviations or simplifications, and incomplete implementations may be acceptable. However, if the goal is to simulate an ICS network that communicates with arbitrary real devices on universal protocols, timing and protocol deviations might not be tolerated. Furthermore, the implementation of device failure conditions, or known vulnerabilities may not be an important feature in educational emulators.

Educational emulators can be used to develop skills and understanding of tools and techniques used to conduct attacks on industrial control networks. These emulators can be used to extend the capabilities of tools like the Nmap [Lyo12] network scanner and the Metasploit [Met12] remote exploitation tool from traditional IT networks to industrial control networks.

In this way, an attack on an educational emulator demonstrates the parallels with traditional IT attacks using the same toolset, but also illustrates the physical effects of an attack on an ICS network. An emulator that responds to scans and attacks from tools like Nmap and Metasploit in the same way as the target device, provides hands-on ICS attack experience without putting an actual device at risk. The emulators could be remotely configured by instructors to implement any number of device configurations and vulnerabilities while recording the techniques and strategies the students use to exploit.

To respond in the same way as the target device, industrial control emulators require a high level of accuracy to fool fingerprinting and port scanning tools, as well ensure the tools that work on the target device also work successfully on the emulator. However, even a lower-accuracy emulator has value in a training environment. For example, the student could be tasked with identifying a networked ICS device as a honeypot among several real devices, employing techniques such as network scanning and protocol dissection to investigate unfamiliar industrial protocols.

### Appendix B: Features of Related Research

Implementation Details	Implementation Version			
Year	2013	2012	2008	2004
Author/Group	R. Jaromin (AFIT)	D. Berman(AFIT)	D. Peterson (Digital Bond)	V. Pothamsetty, M. Franz (CIAG)
Platform	Gumstix	Gumstix	2 Virtual Machines or Real PLC	PC
Publically Known Vulnerability	Yes	No	No	No
Accuracy Data	Yes	Incomplete	None	None
Representative PLF Fingerprint	Actual Koyo PLC	Generic Device	Actual Modicom PLC	Generic PLC
Special Requirements	iptables, queuing	iptables	3 NICs, 2 VMs, Real PLC	Honeyd, iptables
Target Device	Koyo Directlogics PLC Generic	Modbus TCP device	Modicom Quantum PLC	Generic PLC
Platform/Portability	Gumstix	Gumstix	Requires 2 VMs or 1 VM and a Real PLC	Honeyd Based
Logging	Out-Of-Band	Out-Of-Band	Out-of-Band	Local
Configurable	Device-level only	Device-level only	Device-level only	Device-level only
Fails Like Target Device	Some	No	No	No
Protocols Supported	HAP, Modbus, HTTP	Modbus	FTP, Telnet, HTTP, SNMP, Modbus	FTP, Telnet, HTTP, Modbus

<b>Communications Medium</b>	Ethernet	Ethernet	Ethernet	Ethernet
<b>Applied Research</b>	Planned future work	None	[Wad11] S. Wade	[FAA+09] G. Franceschinis, et al.

### Appendix C: Iptables Rules in *iptables\_rules.sh*

<b>Rule 1:</b>	Result	"For all incoming packets on eth0 that have not been marked with 0x02 send them to queue number 3"
	Rule	<code>iptables -t mangle -A INPUT -m mark ! --mark 0x2/2 -j NFQUEUE --queue-num 3 -i eth0</code>
	Resource	Queue number 3 user-space program is <b>logging.o</b>
<b>Rule 2:</b>	Result	"for all incoming TCP packets on eth0, with TCP destination port 22 (SSH), accept"
	Rule	<code>iptables -A INPUT -p tcp --dport 22 -m tcp -j ACCEPT -i \$INTERFACE_LOGGING</code>
	Resource	n/a
<b>Rule 3:</b>	Result	"For all incoming TCP packets on eth0 with destination port 80, send them to queue number 1"
	Rule	<code>iptables -A INPUT -p tcp --dport 80 -m tcp -j NFQUEUE --queue-num 1 -i eth0</code>
	Resource	Queue number 1 user-space program is <b>infilter.o</b>
<b>Rule 4:</b>	Result	"For all incoming TCP packets on eth0 with destination port 1, send them to queue number 1"
	Rule	<code>iptables -A INPUT -p tcp --dport 1 -m tcp -j NFQUEUE --queue-num 1 -i eth0</code>
	Resource	Queue number 1 user-space program is <b>infilter.o</b>
<b>Rule 5:</b>	Result	"For all incoming TCP packets on eth1 with source port 502 (Modbus), accept"
	Rule	<code>iptables -A INPUT -p tcp --sport 502 -m tcp -j ACCEPT</code>
	Resource	n/a
<b>Rule 6:</b>	Result	"For all incoming TCP packets on eth0 with destination port 502 (Modbus), accept"
	Rule	<code>iptables -A INPUT -p tcp --dport 502 -m tcp -j ACCEPT</code>
	Resource	n/a
<b>Rule 7:</b>	Result	"For all incoming TCP packets on eth0 that do not match a previous rule, reject with TCP reset"
	Rule	<code>iptables -A INPUT -p tcp -j REJECT --reject-with tcp-reset -i eth0</code>
	Resource	n/a

<b>Rule 8:</b>	Result	"For all incoming UDP packets on eth0 with destination port 28784 (HAP), accept"
	Rule	iptables -A INPUT -p udp --dport 28784 -m udp -j ACCEPT -i eth0
	Resource	n/a
<b>Rule 9:</b>	Result	"For all incoming ICMP packets on eth0, send them to queue number 1"
	Rule	iptables -A INPUT -p icmp -m icmp -j NFQUEUE --queue-num 1 -i eth0
	Resource	Queue number 1 user-space program is <b>infilter.o</b>
<b>Rule 10:</b>	Result	"For all incoming UDP packets on eth0 with destination port 1, send them to queue number 1"
	Rule	iptables -A INPUT -p udp --dport 1 -m udp -j NFQUEUE --queue-num 1 -i eth0
	Resource	Queue number 1 user-space program is <b>infilter.o</b>
<b>Rule 11:</b>	Result	"For all incoming UDP packets that do not match a previous rule, reject with ICMP port unreachable"
	Rule	iptables -A INPUT -p udp -j REJECT --reject-with icmp-port-unreachable -i eth0
	Resource	n/a
<b>Rule 12:</b>	Result	"For all outgoing TCP packets on eth0 with destination port 80, that have not been marked with 0x1, send them to queue number 2"
	Rule	iptables -t mangle -A OUTPUT -p tcp --sport 28784 -m mark ! --mark 0x1/1 -m udp -j NFQUEUE --queue-num 2 -o eth0
	Resource	Queue number 2 user-space program is <b>outfilter.o</b>
<b>Rule 13:</b>	Result	"For all outgoing TCP packets on eth0 with destination port 80, that have been marked with 0x1, accept"
	Rule	iptables -t mangle -A OUTPUT -p tcp --sport 80 -m mark --mark 0x1/1 -m tcp -j ACCEPT -o eth0
	Resource	n/a
<b>Rule 14:</b>	Result	"For all outgoing UDP packets on eth0 with destination port 28784 (HAP), that have not been marked with 0x1, send them to queue number 2"
	Rule	iptables -t mangle -A OUTPUT -p udp --sport 28784 -m mark ! --mark 0x1/1 -m udp -j NFQUEUE --queue-num 2 -o eth0
	Resource	Queue number 2 user-space program is <b>outfilter.o</b>

<b>Rule 15:</b>	Result	"For all outgoing UDP packets on eth0 with destination port 28784 (HAP), that have been marked with 0x1, accept"
	Rule	iptables -t mangle -A OUTPUT -p udp --sport 28784 -m mark --mark 0x1/1 -m udp -j ACCEPT -o eth0
	Resource	n/a
<b>Rule 16:</b>	Result	"For all outgoing packets on eth0, set the TTL to 255"
	Rule	iptables -t mangle -A POSTROUTING -j TTL --ttl-set 255 -o eth0
	Resource	n/a
<b>Rule 17:</b>	Result	"For all outgoing TCP packets on eth0, send them to queue number 3"
	Rule	iptables -t mangle -A POSTROUTING -m mark ! --mark 0x2/2 -j NFQUEUE --queue-num 3 -o eth0
	Resource	Queue number 3 user-space program is <b>logging.o</b>

**Appendix D: Web and HAP Protocol Functions Implemented in the ICS field device emulator**

<b>Function</b>	<b>Accessible through</b>
Download ladder logic	DirectSOFT5
Disable Protection via Password	DirectSOFT5
Read ladder logic dynamic status	DirectSOFT5
Read CPU information	NetEdit3, DirectSOFT5
Read Firmware Revision info	NetEdit3, Web browser, DirectSOFT5
Read Booter Revision info	NetEdit3, Web browser, DirectSOFT5
Read PWB Revision info	NetEdit3, Web browser, DirectSOFT5
Read PLD Revision info	NetEdit3, Web browser, DirectSOFT5
Read CPU Revision info	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module ID	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module Name	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module Description	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module IP Address (Manually)	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module Subnet Mask (Manually)	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module Subnet Gateway (Manually)	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module IP Address (DHCP)	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module Subnet Mask (DHCP)	NetEdit3, Web browser, DirectSOFT5
Read / Configure Module Subnet Gateway (DHCP)	NetEdit3, Web browser, DirectSOFT5
Read / Read Only Web Config	NetEdit3, Web browser
Read / Enable Web Server	NetEdit3
Test CPU Access	NetEdit3, DirectSOFT5
Password protection	DirectSOFT5
Bruteforce password cracking	Metasploit
Nmap OS Fingerprint Scan	Nmap
Device Polling	NetEdit3, DirectSOFT5
Addressed Queries	NetEdit3, DirectSOFT5



Read Device info	NetEdit3, DirectSOFT5
Read CCM Data (select operations)	DirectSOFT5
Write CCM Data (select operations)	DirectSOFT5
Read K-Sequence Data (select operations)	DirectSOFT5
Read Version Info	NetEdit3, DirectSOFT5
Read Ethernet Statistics	NetEdit3, Web browser

## Appendix E: Non-deterministic Fields

### E.1 Non-deterministic Fields

These are fields whose values change from query to query depending on the state of the querying computer, not necessarily the device (PLC or Emulator), or that cannot be predicted (such as the Initial Sequence Number).

#### *E.1.1 Non-Deterministic Bytes in Web Response.*

Table D.1: Ethernet Header Fields.

<b>Offset(Hex)</b>	<b>Description</b>	<b>Response Packet (1-8)</b>
0x0B	Ethernet Address (MAC) 6 <sup>th</sup> byte	1,2,3,4,5,6,7,8

Table D.2: IP Header Fields.

<b>Offset(Hex)</b>	<b>Description</b>	<b>Response Packet (1-8)</b>
0x18	IP CRC 1 <sup>st</sup> byte	1,2,3,4,5,6,7,8
0x19	IP CRC 2 <sup>nd</sup> byte	1,2,3,4,5,6,7,8
0x1D	IP Source byte 4	1,2,3,4,5,6,7,8

Table D.3: TCP Header Fields.

<b>Offset(Hex)</b>	<b>Description</b>	<b>Response Packet (1-8)</b>
0x24	TCP Destination Port byte 1	1,2,3,4,5,6,7,8
0x25	TCP Destination Port byte 2	1,2,3,4,5,6,7,8
0x26	TCP Sequence Number byte 1	1,2,3,4,5,6,7,8
0x27	TCP Sequence Number byte 2	1,2,3,4,5,6,7,8
0x28	TCP Sequence Number byte 3	1,2,3,4,5,6,7,8
0x29	TCP Sequence Number byte 4	1,2,3,4,5,6,7,8
0x2a	TCP Acknowledgement Number byte 1	1,2,3,4,5,6,7,8
0x2b	TCP Acknowledgement Number byte 2	1,2,3,4,5,6,7,8
0x2c	TCP Acknowledgement Number byte 3	1,2,3,4,5,6,7,8
0x2d	TCP Acknowledgement Number byte 4	1,2,3,4,5,6,7,8
0x32	TCP CRC byte 1	1,2,3,4,5,6,7,8
0x33	TCP CRC byte 2	1,2,3,4,5,6,7,8

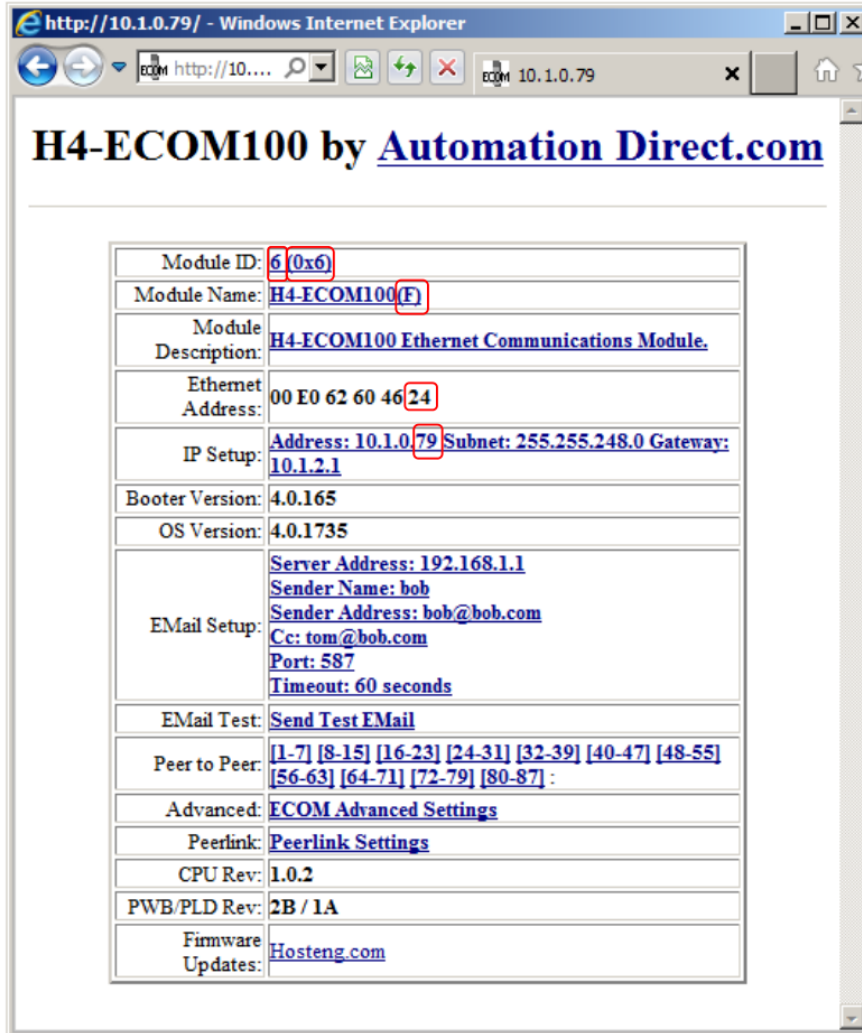


Figure C.1. Example of homepage served from the Gumstix emulator device by webserver.py

Table D.4: HTTP Data Fields:

Offset(Hex)	Description	Response Packet (1-8)
0x184	Module ID address displayed in Webpage	3
0x189	Module ID displayed in Webpage in Hex	3
0x1DD	(F)ake or (R)eal designator in Module Name on Webpage	3
0x9E	Ethernet address displayed in Webpage	4
0xED	IP address displayed in Webpage	4
0xEE	IP address displayed in Webpage	4

## E.2 Non-deterministic Bytes in HAP Response

Table E.5: Ethernet Header Fields

<b>Offset(Hex)</b>	<b>Description</b>
0x0B	Ethernet Address (MAC) 6 <sup>th</sup> byte

Table E.6: IP Header Fields

<b>Offset(Hex)</b>	<b>Description</b>
0x18	IP CRC 1 <sup>st</sup> byte
0x19	IP CRC 2 <sup>nd</sup> byte
0x1D	IP Source byte 4

Table E.7: UDP Header Fields

<b>Offset(Hex)</b>	<b>Description</b>
0x28	UDP CRC 1 <sup>st</sup> byte
0x29	UDP CRC 2 <sup>nd</sup> byte

### E.3 Non-deterministic Fields in Nmap OS Fingerprint

Table E.8: UDP Header Fields

Section	Field	Description
OS Guess	MAC Address	6 <sup>th</sup> Byte
Scan Details	SP†	Sequence Prediction Index
Scan Details	ISR†	TCP ISN Counter Rate
Scan Details	UN‡	Unused ICMP port unreachable field nonzero
TCP Sequence Prediction	TCP Sequence Prediction†	Equal to base 10 version of SP
<p>†These fields are based on the TCP initial sequence numbers in response packets. Although they are non-deterministic, they are still considered relevant fields with regard to accuracy, and their matching is considered important.</p> <p>‡This field is non-deterministic, but is still considered relevant with regard to accuracy, and its matching is considered important.</p>		

## Appendix F: Partial Factorial Design

Scenario	Platform	Request Type	Request Frequency	Service	Workload Gen
1	PC	Std Web	slow	Web	web-tool
2	PC	Std Web	PLC Break	Web	web-tool
2a	PC	Std Web	PC Break	Web	web-tool
3	PC	Non-std Web	slow	Web	nmap
4	PC	Non-std Web	PLC Break	Web	nmap
5	PC	Std HAP	slow	HAP	hap-tool
6	PC	Std HAP	PLC Break	HAP	hap-tool
6a	PC	Std HAP	PC Break	HAP	hap-tool
7	PC	Non-std HAP	slow	HAP	Metasploit
8	PC	Non-std HAP	PLC Break	HAP	Metasploit
8a	PC	Non-std HAP	PC Break	HAP	Metasploit
9	Gum	Std Web	slow	Web	web-tool
10	Gum	Std Web	PLC Break	Web	web-tool
10a	Gum	Std Web	Gum Break	Web	web-tool
11	Gum	Non-std Web	slow	Web	nmap
12	Gum	Non-std Web	PLC Break	Web	nmap
13	Gum	Std HAP	slow	HAP	hap-tool
14	Gum	Std HAP	PLC Break	HAP	hap-tool
14a	Gum	Std HAP	Gum Break	HAP	hap-tool
15	Gum	Non-std HAP	slow	HAP	Metasploit
16	Gum	Non-std HAP	PLC Break	HAP	Metasploit
16a	Gum	Non-std HAP	Gum Break	HAP	Metasploit
17	PLC	Std Web	slow	Web	web-tool
18	PLC	Std Web	PLC Break	Web	web-tool
19	PLC	Non-std Web	slow	Web	nmap
20	PLC	Non-std Web	PLC Break	Web	nmap
21	PLC	Std HAP	slow	HAP	hap-tool

22	PLC	Std HAP	PLC Break	HAP	hap-tool
23	PLC	Non-std HAP	slow	HAP	Metasploit
24	PLC	Non-std HAP	PLC Break	HAP	Metasploit

## Appendix G: Custom Tool Description and Validation

### G.1 Accuracy Parser: `pcap_accuracy_parse.py`

#### *G.1.1 Description.*

The accuracy parser compares two similar .pcap files to determine their byte differences. The first .pcap file is the baseline file from the target PLC that captured the packet conversation between the workload generator and the PLC, the second .pcap file captures the conversation between the workload generator and the emulator for the same experimental conditions.

This Python script works by separating the first .pcap file into streams, and measuring byte differences between each stream (in the case of this experiment, 1000 streams). Since these responses are all from the target PLC, the only differences should be non-deterministic bytes indicated in Appendix C, a report of the deterministic (or non-changing) bytes is generated. The .pcap from the emulator is processed the same way. Finally the two reports for the target PLC and the emulator bytes are compared to each other to generate a summary for all queries, which shows the differences at the byte level for every packet all 1000 responses.

#### *G.1.2 Validation.*

The program is run two ways to validate its functionality. The first ran the same file for both input parameters. The results are that all bytes matched. Next, a two captures for the same query are taken, and the bytes are compared manually to ensure the results produced by the tool are accurate. The manual verification concluded that the bytes that are dissimilar are in fact different between the two captures, and the deterministic bytes are actually the same.



## **G.2 Workload Generators: *htmlget\_mt* and *hap\_mt***

### ***G.2.1 Description.***

These files generate the standard web and HAP workloads at nominal timing rates for a variable number of queries. Both generators are designed using raw Linux sockets, and therefore, many aspects are definable, as a result, however, certain intricacies such as the TCP handshake must be implemented manually. The number of successful connections, response time, and moving average of response times are displayed real-time as the programs execute.

### ***G.2.2 Validation.***

Both tools are validated using the packet capture accuracy parsing tool, *pcap\_accuracy\_parse.py*, already validated. The procedure for each tool is the same and is shown in Figure G.1.

A packet capture is taken for one web or HAP request using the predefined tool (Internet Explorer 9.0.8112 for web, and NetEdit3 for HAP) to make one query to the target PLC. A second capture is taken under exactly the same conditions at least one minute later for a total of two captures from the baseline tool. The accuracy parser is run on both of these packet captures to identify packet bytes that are non-deterministic for a single query. A packet capture is then taken for a single query using the custom workload generator, and this packet capture is compared to the first capture using the base program using the accuracy parser. The non-deterministic bytes are then compared to those measured from step 2. If all the bytes are the same, then the workload generator is validated.

Using this procedure 35 bytes are found to be non-deterministic for the web browser query, all deterministic bytes matched when the web browser capture is compared to the *htmlget\_mt* capture. For the NetEdit3 query (after accounting for MAC and IP), 8 bytes are non-deterministic and all deterministic bytes matched when the NetEdit3 capture is compared to the *hap\_mt* capture.

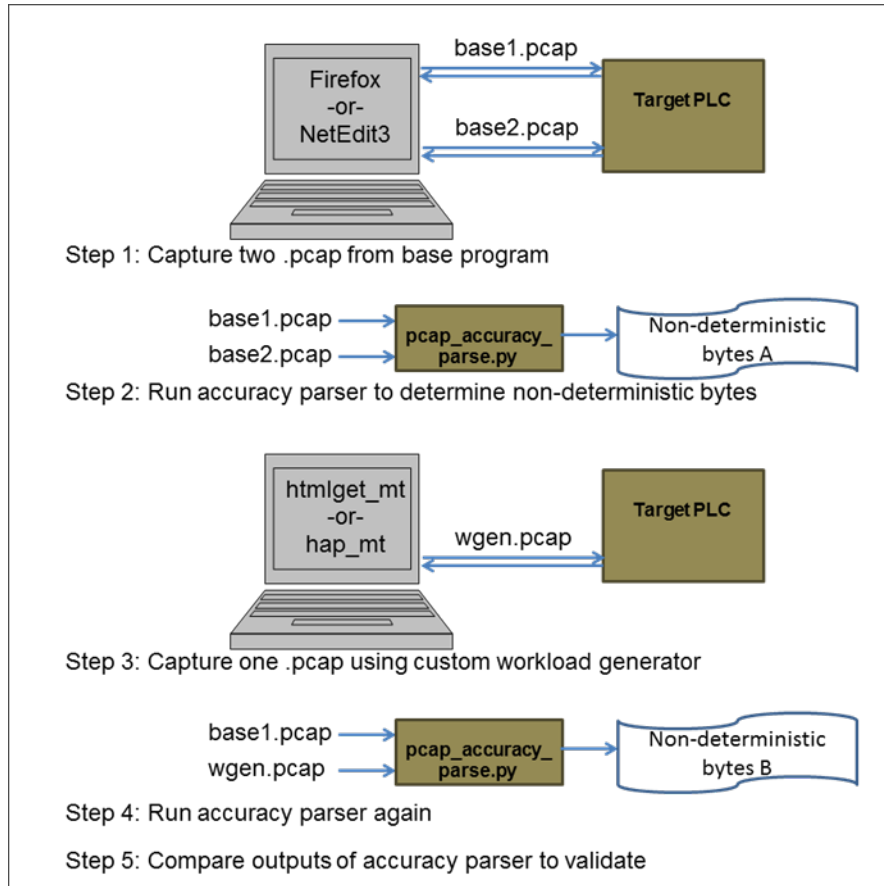


Figure G.1: Validation of workload generators.

### G.3 Timing parsers: `web_time_parse.py` and `udp_time_parse.py`

#### G.3.1 Description.

The two time parser files take .pcap files as inputs, and separate the files into individual streams based on the to and from ports in the .pcap capture. This method is valid because the custom workload generators are explicitly give a unique source port to use for each query, and reusing the source ports is not allowed.

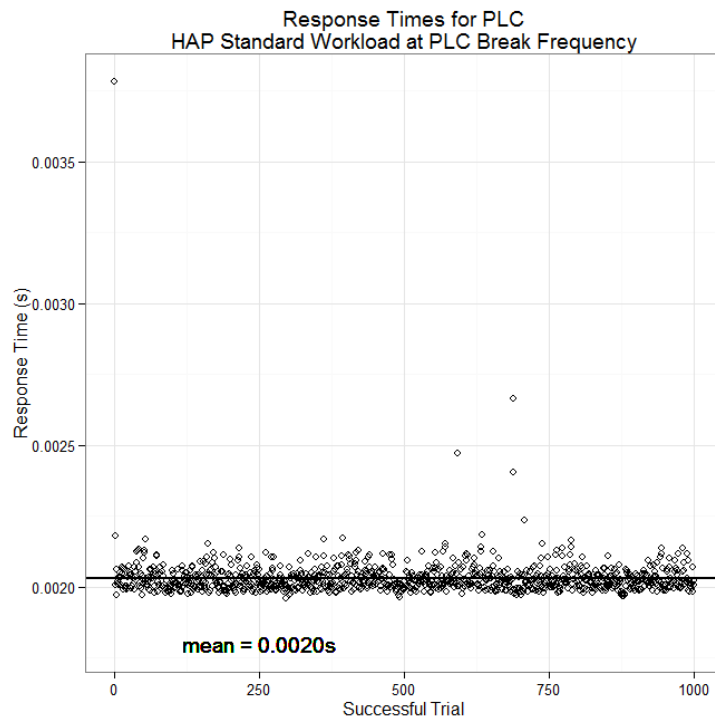
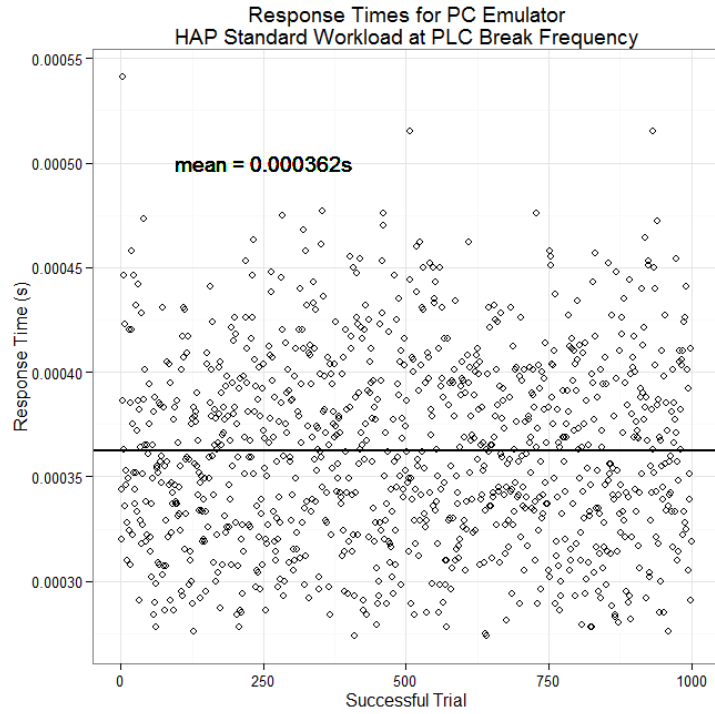
The difference between first packet per stream and the last packet is generated and averaged together for all successful streams for a mean response time. This procedure works with the standard web and HAP response captures and the HAP non-standard

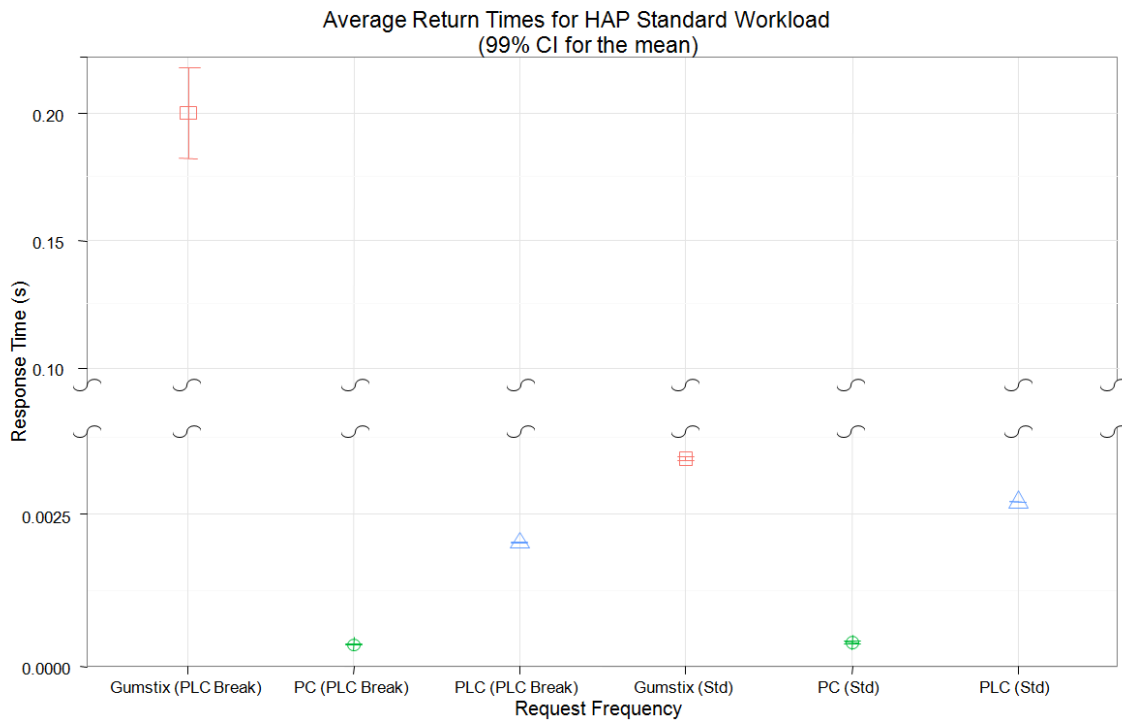
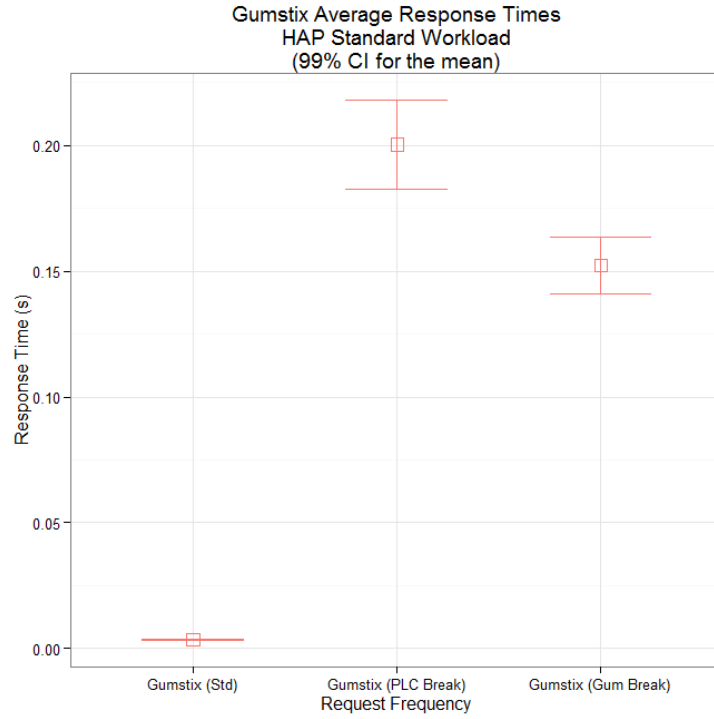
(Metasploit) workload. It does not work for the web non-standard (Nmap) workload because the packets are sent using random source ports and sent to destination ports in non-sequential order.

### ***G.3.2 Validation.***

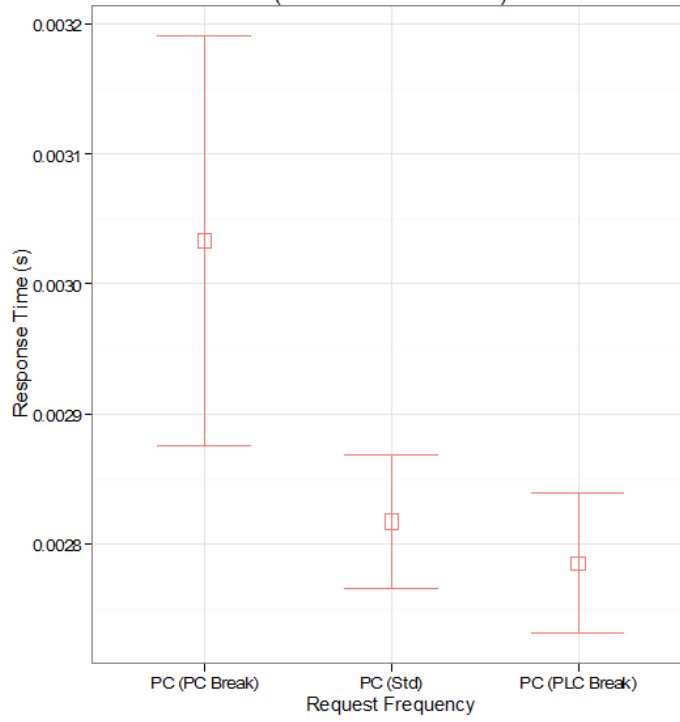
To provide a preliminary validation, the response time values generated by the time parsers are compared to the response time values generated by the workload generators. The primary validation is completed by hand and compared the calculated values from the timing parsers to those seen for the timestamp value for the packets in a graphical packet capture program. A subset for both the web and HAP standard workload responses are successfully validated.

## Appendix H: Additional Results for HAP standard Workload

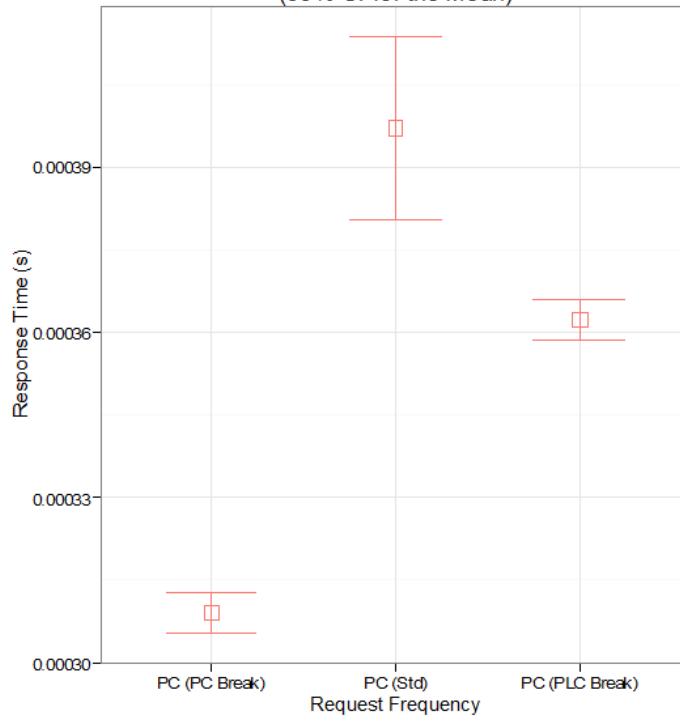




PC Average Response Times  
Web Standard Workload  
(99% CI for the mean)



PC Average Response Times  
HAP Standard Workload  
(99% CI for the mean)



## Appendix I: Configuring the Gumstix Emulator

### I.1 Loading Provided Images

The emulator can be placed onto a secure digital (SD) card via an image created of the final working copy on of the Gumstix emulator following these steps. Most of the hard work is done by shell scripts. This guide assumes a USB SD card reader is attached to the Linux development computer.

1. Insert the target SD card into USB SD card reader.
2. Partition the target SD card with "boot" and "rootfs" partitions. (The *mksdcard.sh* script is in the *Images* directory of the Resources Disc included that accompanies this document).

```
#build partitions for the new SD card:  
$sudo mksdcard.sh /dev/sdb overo minimal  
#mount new partitions  
$sudo mount -t vfat /dev/sdb1 /media/boot  
$sudo mount -t ext3 /dev/sdb1 /media/rootfs
```

3. From a command shell, execute `sudo nautilus` to get super-user privileges for the file navigator. Use this program to delete all current files in the boot and rootfs partitions of the target SD. Or, from the command shell:

```
#delete everything in the /media/rootfs/  
$cd /media/rootfs  
$sudo rm -r *.*  
$cd /media/boot  
$sudo rm -r *.*
```

4. From a command shell, navigate to the *Images* directory of the Resources Disc included that accompanies this document.

5. Execute `sudo ./load_image.sh emulator`. It will take several minutes for the files to be copied onto the target SD card.
6. To do this manually, the commands are:

```
#Copy boot files onto first partition:  
$sudo cp MLO /media/boot/MLO  
$sudo cp u-boot.bin /media/boot/u-boot.bin  
$sudo cp uImage /media/boot/uImage  
#Expand the root file system archive on to the second partition:  
$sudo tar xaf rootfs.tar.bz2 --strip-components 2 -C /media/rootfs
```

(Two other images are included:(1) Angstrom Linux image with both Ethernet ports configured on Tobi-Duo (2) Angstrom Linux image with iptables already installed)

## **I.2 Building From Scratch**

Berman's previous effort developed a detailed set of instructions for building an operational Gumstix device from scratch with most of the required components needed for this follow-on effort [Ber12 (Appendix-A)].

## **I.3 Saving New Images**

A script is developed to easily save Gumstix images for quick Gumstix development. It is also located in the *Images* directory of the Resources Disc included that accompanies this document.

1. Insert the target SD card into USB SD card reader.
2. From a command shell, navigate to the *Images* directory of the Resources Disc included that accompanies this document.
3. Execute `sudo ./make_image.sh <image_name>`. It will take several minutes for the files to be copied and compressed from the target SD card.



4. To do this manually, the commands are:

```
#Copy boot files from first partition:
```

```
$sudo cp /media/boot/MLO ./MLO
```

```
$sudo cp /media/boot/u-boot.bin ./u-boot.bin
```

```
$sudo cp /media/boot/uImage ./uImage
```

```
#Compress the root file system archive from the second partition:
```

```
$sudo tar caf ./<image_name>.image.tar.bz2 /media/rootfs
```

# 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.**

<b>1. REPORT DATE (DD-MM-YYYY)</b> 21-03-2013		<b>2. REPORT TYPE</b> Master's Thesis		<b>3. DATES COVERED (From — To)</b> Jul 2011-Mar 2013			
<b>4. TITLE AND SUBTITLE</b>  Emulation of Industrial Control Field Device Protocols				<b>5a. CONTRACT NUMBER</b>			
				<b>5b. GRANT NUMBER</b> HSHQDC-11-X-00089			
				<b>5c. PROGRAM ELEMENT NUMBER</b>			
				<b>5d. PROJECT NUMBER</b>			
				<b>5e. TASK NUMBER</b>			
<b>6. AUTHOR(S)</b>  Jaromin, Robert M., Captain, USAF				<b>5f. WORK UNIT NUMBER</b>			
				<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765		<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  AFIT-ENG-13-M-27	
				<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Department of Homeland Security ICS-CERT POC: Neil Hershfield, DHS ICS-CERT Technical Lead ATTN: NPPD/CSC/NCSD/US-CERT Mailstop: 0635, 245 Murray Lane, SW, Bldg 410, Washington, DC 20528 Email: ics-cert@dhs.gov phone: 1-877-776-7585		<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b> DHS ICS-CERT	
				<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>			
<b>12. DISTRIBUTION / AVAILABILITY STATEMENT</b> <b>DISTRIBUTION STATEMENT A.</b> APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED							
<b>13. SUPPLEMENTARY NOTES</b> This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.							
<b>14. ABSTRACT</b> It has been shown that thousands of industrial control devices are exposed to the Internet, however, the extent and nature of attacks on such devices remains unknown. The first step to understanding security problems that face modern supervisory control and data acquisition (SCADA) and industrial controls networks is to understand the various attacks launched on Internet-connected field devices. This thesis describes the design and implementation of an industrial control emulator on a Gumstix single-board computer as a solution. This emulator acts as a decoy field device, or <i>honeypot</i> , intended to be probed and attacked via an Internet connection. Evaluation techniques are developed to assess the accuracy of the emulation implemented on the Gumstix and are compared against the implementation on a standard PC and the emulation target, a Koyo DirectLogic 405 programmable logic controller. The results show that both the Gumstix and PC emulator platforms are very accurate to the workloads presented. This suggests that a honeypot implemented on a Gumstix emulator and a standard PC are both suitable for applications in SCADA attack-landscape research.							
<b>15. SUBJECT TERMS</b> ICS SCADA Emulator Gumstix Honeypot PLC							
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>	<b>18. NUMBER OF PAGES</b>	<b>19a. NAME OF RESPONSIBLE PERSON</b>		
<b>a. REPORT</b>	<b>b. ABSTRACT</b>	<b>c. THIS PAGE</b>			Dr. Barry E. Mullins (ENG)		
U	U	U	UU	185	<b>19b. TELEPHONE NUMBER (include area code)</b> (937) 255-3636 x7979 Barry.Mullins@afit.edu		