

6-13-2013

# Evaluation of Cyber Sensors for Enhancing Situational Awareness in the ICS Environment

Jeremy R. Otis

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

Part of the [Computer and Systems Architecture Commons](#)

---

## Recommended Citation

Otis, Jeremy R., "Evaluation of Cyber Sensors for Enhancing Situational Awareness in the ICS Environment" (2013). *Theses and Dissertations*. 894.

<https://scholar.afit.edu/etd/894>

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



**EVALUATION OF CYBER SENSORS FOR ENHANCING  
SITUATIONAL AWARENESS IN THE ICS ENVIRONMENT**

THESIS

Jeremy R. Otis

AFIT-ENG-13-J-06

**DEPARTMENT OF THE AIR FORCE  
AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

DISTRIBUTION STATEMENT A:  
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED

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

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

AFIT-ENG-13-J-06

EVALUATION OF CYBER SENSORS FOR ENHANCING  
SITUATIONAL AWARENESS IN THE ICS ENVIRONMENT

THESIS

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

Jeremy R. Otis, B.S.C.S.


June 2013

DISTRIBUTION STATEMENT A:  
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED


EVALUATION OF CYBER SENSORS FOR ENHANCING  
SITUATIONAL AWARENESS IN THE ICS ENVIRONMENT

Jeremy R. Otis, B.S.C.S.

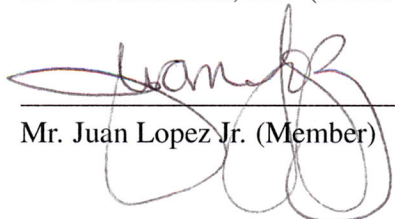
Approved:

  
\_\_\_\_\_  
Maj Jonathan Butts, PhD (Chairman)

7 JUNE 2013  
Date

  
\_\_\_\_\_  
Lt. Col Robinson, PhD (Member)

11 JUNE 2013  
Date

  
\_\_\_\_\_  
Mr. Juan Lopez Jr. (Member)

16 JUNE 2013  
Date

**Abstract**

Industrial Control Systems (ICS) monitor and control operations associated with the national critical infrastructure (e.g., electric power grid, oil and gas pipelines and water treatment facilities). These systems rely on technologies and architectures that were designed for system reliability and availability. Security associated with ICS was never an inherent concern, primarily due to the protections afforded by network isolation. However, a trend in ICS operations is to migrate to commercial networks via TCP/IP in order to leverage commodity benefits and cost savings. As a result, system vulnerabilities are now exposed to the online community. Indeed, recent research has demonstrated that many exposed ICS devices are being discovered using readily available applications (e.g., Shodan search engine and Google-esque queries).

Due to the lack of security and logging capabilities for ICS, most knowledge about attacks are derived from real world incidents after an attack has already occurred. Further, the distributed nature and volume of devices requires a cost effective solution to increase situational awareness. This research evaluates two low cost sensor platforms for enhancing situational awareness in the ICS environment. Data obtained from the sensors provide insight into attack tactics (e.g., port scans, Nessus scans, Metasploit modules, and zero-day exploits) and characteristics (e.g., attack origin, frequency, and level of persistence). The results indicate that the low cost cyber sensors perform sufficiently within the ICS environment. Furthermore, findings enable security professionals to draw an accurate, real-time awareness of the threats against ICS devices and help shift the security posture from reactionary to preventative.

## **Acknowledgments**

My sincerest thanks and appreciation goes out to my supportive wife. I would also like to thank my advisor, Maj Butts, who dedicated much of his time helping me accomplish my academic and professional goals. Finally, I would like to thank my family for their unending support throughout my academic career.

Jeremy R. Otis

## Table of Contents

	Page
Abstract . . . . .	iv
Acknowledgments . . . . .	v
Table of Contents . . . . .	vi
List of Figures . . . . .	ix
List of Tables . . . . .	x
List of Acronyms . . . . .	xi
I. Introduction . . . . .	1
1.1 Background . . . . .	1
1.2 Motivation . . . . .	2
1.3 Problem Statement . . . . .	2
1.4 Approach . . . . .	3
1.5 Scope and Limitations . . . . .	3
1.6 Organization . . . . .	4
II. Literature Review . . . . .	5
2.1 ICS Background . . . . .	5
2.2 ICS Security . . . . .	7
2.3 Logging and Categorization of Data . . . . .	9
2.4 Control Systems Availability . . . . .	10
2.5 Attack Techniques . . . . .	11
2.6 Incidents . . . . .	14
2.6.1 Threat Picture . . . . .	14
2.6.2 Maroochy Shire . . . . .	17
2.6.3 Davis-Besse Nuclear Power Plant . . . . .	17
2.6.4 Worcester Air Traffic Communications . . . . .	18
2.6.5 Stuxnet . . . . .	18
2.7 Cyber Sensors . . . . .	19
2.8 Situational Awareness . . . . .	20
2.8.1 Logging . . . . .	20
2.8.2 Intrusion Detection Systems . . . . .	21



	Page
2.8.3 Analytics . . . . .	22
2.9 Summary . . . . .	22
III. Methodology . . . . .	23
3.1 Problem Definition . . . . .	23
3.2 Goals . . . . .	23
3.3 Hypothesis . . . . .	23
3.4 Environment . . . . .	23
3.4.1 Traffic Generator . . . . .	24
3.4.2 Log Analyzer . . . . .	25
3.4.3 Cyber Sensors . . . . .	25
3.4.4 Communication . . . . .	28
3.5 Evaluation Technique . . . . .	28
3.5.1 Network Traffic . . . . .	28
3.5.2 Performance Metrics . . . . .	30
3.6 Experiments . . . . .	31
3.6.1 Effectiveness . . . . .	34
3.7 Methodology Summary . . . . .	35
IV. Analysis and Results . . . . .	36
4.1 Implementation Details . . . . .	36
4.1.1 Cyber Sensor . . . . .	36
4.1.2 Log Analyzer . . . . .	36
4.1.3 Event Development . . . . .	37
4.2 Initialization Checks . . . . .	37
4.2.1 Functionality Test Case . . . . .	37
4.2.2 Identification Test Case . . . . .	39
4.3 Results . . . . .	39
4.3.1 Functionality Test Case . . . . .	39
4.3.2 Identification Test Case . . . . .	42
4.3.3 Examination . . . . .	44
4.4 Degradation Observations . . . . .	47
4.5 Results Summary . . . . .	48
V. Conclusions . . . . .	50
5.1 Applications . . . . .	51
5.1.1 ICS-CERT Incident Response . . . . .	51
5.1.2 Network Performance Metrics . . . . .	52
5.1.3 Operation Picture . . . . .	52

	Page
5.2 Future Work . . . . .	52
5.2.1 Snort Signatures . . . . .	52
5.2.2 Optimization . . . . .	53
5.2.3 Deployment in Operational Environment . . . . .	53
5.2.4 Additional Capabilities . . . . .	53
5.3 Concluding Remarks . . . . .	54
Appendix A: Test Data . . . . .	55
Appendix B: Scripts and Code . . . . .	58
Appendix C: Configurations . . . . .	62

## List of Figures

Figure	Page
2.1 ICS Layout . . . . .	7
2.2 ICSvsIT . . . . .	8
2.3 Banner from Shodan . . . . .	11
2.4 Map of Available ICS through Shodan . . . . .	12
2.5 Various microcomputers used as cyber sensors. . . . .	19
3.1 Complete Evaluation Environment . . . . .	24
3.2 Gumstix Size . . . . .	26
3.3 Raspberry Pi Sensor . . . . .	27
4.1 Percentage Packet Loss . . . . .	45
4.2 Sensors Compression Times . . . . .	46
4.3 Degradation Trend . . . . .	48
A.1 Full Functionality Test Results . . . . .	56
A.2 Full Identity Test Results . . . . .	57

## List of Tables

Table	Page
2.1 Common Web Attacks . . . . .	13
2.2 Common Authentication Attacks . . . . .	14
2.3 ICS Threats . . . . .	15
2.4 ICS Threats . . . . .	16
3.1 Sensor Specifications . . . . .	28
3.2 Functionality Test Enumeration . . . . .	33
3.3 Identification Test Enumeration . . . . .	34
4.1 Calibrated Rate Ranges . . . . .	38
4.2 Functionality Test Case:Laptop Results . . . . .	40
4.3 Functionality Test Case:Raspberry Pi Results . . . . .	41
4.4 Functionality Test Case:Gumstix Results . . . . .	41
4.5 Total Log Size . . . . .	42
4.6 Identification Test Case:Laptop Results . . . . .	43
4.7 Identification Test Case:Raspberry Pi Results . . . . .	43
4.8 Identification Test Case:Gumstix Results . . . . .	44
4.9 Compression Ratios . . . . .	46
C.1 Functionality Test Randomization . . . . .	63
C.2 Identification Test Randomization . . . . .	64

## List of Acronyms

Acronym    Definition

**AFB** Air Force Base

**CIA** Confidentiality Integrity Availability

**COM** Computer-on-Module

**CPU** Central Processing Unit

**DCS** Distributed Control Systems

**DoS** Denial of Service

**GB** Gigabytes

**GPS** Global Positioning System

**HDMI** High Definition Multimedia Interface

**HTTP** HyperText Transfer Protocol

**HMI** Human Machine Interface

**ICS** Industrial Control Systems

**IDS** Intrusion Detection System

**IED** Intelligent Electronic Device

**I/O** Input/Output

**IT** Information Technology

Acronym    Definition

**KB** Kilobyte

**LAN** Local Area Network

**MAC** Media Access Control

**MB** Megabytes

**MBPS** Megabytes per Second

**MTU** Master Terminal Unit

**NIC** Network Interface Card

**PLC** Programmable Logic Controller

**pps** Packets Per Second

**RAM** Random Access Memory

**RTU** Remote Terminal Unit

**SA** Situational Awareness

**SCADA** Supervisory Control and Data Acquisition

**SCP** Secure Copy

**SQL** Structured Query Language

**SUT** System Under Test

**TCP/IP** Transmission Control Protocol/Internet Protocol

**USAF** United States Air Force

Acronym    Definition

**USB** Universal Serial Bus

**VM** Virtual Machine

**WPAFB** Wright-Patterson Air Force Base

**XSS** Cross Site Scripting

# EVALUATION OF CYBER SENSORS FOR ENHANCING SITUATIONAL AWARENESS IN THE ICS ENVIRONMENT

## I. Introduction

### 1.1 Background

Industrial control systems (ICS) integrate plant and processing data onto computer systems that are accessible via a network. ICS monitor and send limited instructions to facilities dispersed over a large geographical area. These networks include infrastructure, such as power grids, water and waste management, gas, oil pipelines, and more.

Due to the proprietary nature of ICS and the isolation from the rest of the connected world, they have traditionally been considered immune to cyber attack. Unfortunately, academic research, as well as real world incidents, have demonstrated that ICS is anything but immune to cyber attack. Indeed, the adoption of technologies such as TCP/IP and Ethernet, have interconnected the once air-gapped systems. As a consequence, attack surfaces have increased and ICS are experiencing elevated exposure to malicious activity.

Attacks against ICS field devices are discovered primarily after the fact with no means for advanced warning. To improve resiliency and help thwart attacks, it is important to enhance system situational awareness. This paper presents a robust logging capability for TCP traffic in ICS utilizing two popular microcomputers as *cyber sensors*. These sensors provide a low cost logging solution that can be distributed across operational networks to provide early warning indications of looming or in progress attacks.



## **1.2 Motivation**

Industrial control systems offer a variety of unique security challenges [31]. The promise of remote availability to isolated locations has introduced entry points into the systems that cannot be physically safeguarded. These entry points are the result of interconnecting ICS devices and migrating them into public networks, such as the Internet.

Security mechanisms, such as authentication, intrusion detection systems, and network logging are used in traditional information technology (IT) systems to detect and prevent malicious activity. ICS systems, however, lack the necessary resources, such as processing power, memory, or communication capabilities to incorporate these security mechanisms. Additionally, the highly dispersed nature of ICS requires extensive costs to retrofit or deploy security solutions. One of the primary concerns is the lack of network logging capabilities. Without fundamental logging, situational awareness in the ICS domain remains underdeveloped.

## **1.3 Problem Statement**

Situational awareness in ICS environments is limited, there is a need for an inexpensive distributive solution to enhance real-time threat identification. Current awareness of the attacks against ICS are after they occur on real time systems. That awareness relies on the ability to document the extent of the damage and how it was carried out. In order to learn and protect against these threats there is a need to understand the variety of attacks being conducted without damaging production systems. Therefore, a need exists to determine if an inexpensive solution can be deployed and perform sufficiently within the ICS environment. The objective of this research is to provide a solution to enhance situational awareness using inexpensive cyber sensing platforms. Specifically, this research focuses on the the following two goals:

1. Determine if the functionality of the cyber sensors are adequate for ICS environments.

2. Determine if logged traffic can be identified as non-standard or malicious.

These goals will provide a better understanding of the frequency and types of attacks against ICS today. Specifically, this research focuses on capturing interactions between ICS field devices and potential attackers using inexpensive and readily available technology. The solution can be used to gather information and help classify new attacks, as well as serve as an early warning sensor.

The sensors should demonstrate that they can accurately record traffic under four volumes: base, low, medium, high. The recorded traffic should then be compressed and sent to a central collection point where identification analysis can take place. The sensors are compared against a baseline to determine sufficient performance and capabilities. The cyber sensors' performance is considered sufficient or effective if they perform within 5% of the baseline sensor. It should also be demonstrated that any traffic that does not follow expected patterns can be identified as *non-standard traffic*.

#### **1.4 Approach**

For this research a logging algorithm is deployed across three platforms, a baseline laptop, a Gumstix Overo Earth COM, and a Raspberry Pi. The sensors are evaluated based on performance, as well as their ability to capture and identify malicious activity. They are tested in an experimental environment and measurements on their accuracy, timing, and identification are recorded for analysis.

#### **1.5 Scope and Limitations**

The scope of this research focuses on the ability to capture and identify ICS TCP/IP traffic. Note that the traffic is specific to ICS network traffic and not ICS field device protocols. This research is limited to two cyber sensors: Gumstix Overo Earth COM, and the Raspberry Pi. Additionally, one custom Snort signature is used to identify non-standard traffic. The signature is readily modified for application to varying operational

environments. This research is limited to an experimental environment. Based on the large scale and distributed nature of ICS networks, deploying the cyber sensors in an operational environment is beyond the scope of this research. Finally, this research evaluates the feasibility of employing cyber sensors in an ICS environment; optimization of the sensors is reserved for future research.

## **1.6 Organization**

Chapter 2 presents background information about ICS, ICS security, threats to the ICS domain, and related efforts. Chapter 3 presents the methodology used for this research. Chapter 4 presents the results from the experiments and Chapter 5 presents conclusions and future work.

## **II. Literature Review**

### **2.1 ICS Background**

Industrial control systems encompass several types of automated control functionalities, including supervisory control and data acquisition (SCADA) systems and distributed control systems (DCS). These systems control and gather information on assets that are dispersed over large geographical areas associated with electric power grids, oil and natural gas, water management, public transportation, and other critical infrastructures. The following list details the major ICS components [7]:

#### **Master Terminal Unit (MTU)**

The MTU resides in the control center and processes information received from one or more remote terminal units and relays the data to the human machine interface.

#### **Remote Terminal Unit (RTU)**

The RTU gathers information, such as valve settings, sensors, or alarms, from field devices. The data is then stored until the MTU requests it, or likewise sends instructions to field devices that the MTU issues.

#### **Programmable Logic Controller (PLC)**

A PLC is a field device controller that can be programmed to interact with field devices, such as valves, meters, sensors, etc. An example is a PLC that controls a valve on a tank of water. It can read from a sensor and open the valve when a certain criterion is met. Modern PLCs are preconfigured with small web servers that provide the capability to directly connect to the PLC and manipulate its configuration, or view its status [3].

#### **Intelligent Electronic Device (IED)**

IEDs are smart sensors that can communicate with other devices, gather data,

and control field devices. Typically, this requires multiple pieces of hardware to accomplish similar results, but the IED is capable of performing all of these tasks on one device.

### **Human Machine Interface (HMI)**

The HMI is composed of both hardware and software that enables operators to have the capability to interact with various field devices attached to the ICS network. It collects and presents data about the system in a user friendly format and can be used to generate reports on the statuses of processes in real time. The HMI also allows the operator to manipulate control processes by sending commands from a centralized location. The centralized location typically requires physical access in order to interact the HMI. However, modern HMI software suites provide remote capabilities that operators and engineers can utilize via the Internet.

### **Data Historian**

The Data Historian is a centralized database used to collect all ICS transactions and processing. This data is used for statistical analysis and process control.

Figure 2.1 shows a general ICS configuration. The HMI, MTU, and Data Historian are located in a control center, and communicate with multiple field sites over various communication channels. The communication infrastructure must span over large areas and are comprised of radio waves, satellite, telephone, and power lines. PLCs or RTUs gather information about the sensors and field devices. The data is then sent to the control center where the MTU collects the information and displays it to the HMI in a format easily readable to the operator [7]. Based on alarm thresholds or set conditions, the HMI or MTU generates an appropriate action. Transactions are gathered for analysis on the data historian.

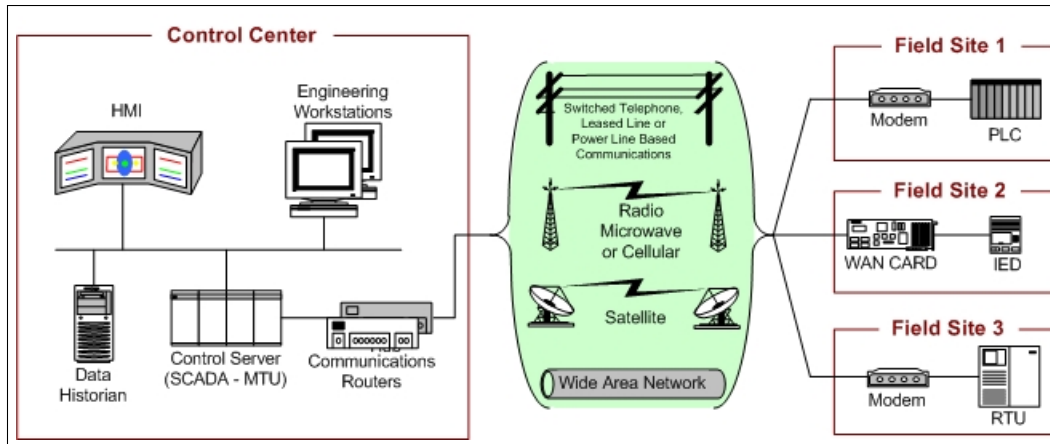


Figure 2.1: ICS general layout [31].

## 2.2 ICS Security

Most ICS in use today were developed prior to the integration of personal computers and the Internet into everyday business operations [31]. Indeed, ICS were fundamentally designed to support availability, reliability, and safety requirements. At the time, security was inherited from the physical isolation of the communication links.

ICS evolved alongside personal computers and the use of the Internet. Lured by the prospect of remote availability and faster communications, ICS started migrating onto the Internet through the adoption of inexpensive TCP/IP components. These adjustments altered the attack surface of ICS, exposing them to new types of threats that increased the likelihood of compromise.

ICS environments have many characteristics that differ from traditional IT systems [15]. IT systems rely on three cornerstones of cyber security: confidentiality, integrity, and availability (CIA). Confidentiality means that information should only be able to be accessed by authorized users. Integrity is concerned with the “trustworthiness, origin, completeness, and correctness of information as well as the prevention of improper or unauthorized modification of information [10].” Availability ensures that the information

is readily accessible to the authorized users that request it. IT systems are designed to encompass all aspects of CIA, but focus mainly on confidentiality and integrity. On the other hand, ICS security is concerned primarily with availability and integrity. ICS must remain available to provide critical services with minimal fault tolerance levels. Since these systems control physical assets, if availability and integrity are not consistent, it could result in severe human injury or impact critical infrastructure assets.

The goals of ICS often conflict with standard IT security practices (e.g., requiring passwords may interfere with emergency actions for ICS). Simply deploying IT security solutions into control systems may not be a viable solution. Figure 2.2 shows some common IT security solutions and how they are addressed in IT domains as opposed to ICS [31].

Security Solution	Information Technology	Industrial Control Systems
Anti-virus	Commonly used	Rare, difficult to implement without affecting availability
Software Patches	Regularly scheduled	Rare, unscheduled, vendor specific
Encryption	Standard practice for data transmission	Difficult to implement without affecting availability
Authentication	Heavily integrated	If even possible, rarely enforced
Secure System Development	Generally accepted and adopted during development	Usually not adopted for development
Technology Support	2-3 years, multiple vendors	10-20 years, same vendor
Logging	Commonly used, easily implemented	Rare, difficult to implement due to limited system resources

Figure 2.2: Difference in IT and ICS security solutions [15].

One security solution that is fundamental in IT is the capability to log and categorize information. Currently, end devices such as PLCs, RTUs, or IEDs have limited resources (e.g., processing power, and memory) and are not designed to execute additional applications. As a result, there are minimal logging mechanisms designed particularly for

ICS. This restricts the identification of early attack indicators and impedes the ability to perform vital forensics when a system has been attacked. The next section details the importance of logging and categorizing data.

### **2.3 Logging and Categorization of Data**

As ICS migrate onto the Internet through the adoption of TCP/IP and Ethernet communications, it is important to monitor traffic for malicious activity. A *network logger* is a device that uses specialized software to capture traffic transmitted over a particular communication medium. Note that the network logger is primarily used to capture data between multiple devices. The SANS Log Management Survey provides a list of benefits for deploying a network logger [28]:

- Detect/Prevent unauthorized access or insider abuse;
- Forensic analysis after compromise;
- Track suspicious behavior;
- Monitor user activity;
- Measure application performance; and
- Ensure regulatory compliance.

On large networks, the amount of information captured by a network logger can be overwhelming and arduous to the analyst. To make this more manageable, filtering is required to categorize and generate a subset of suspicious activity. This is often accomplished using intrusion detection systems (IDS). The goal of IDS are to detect intrusions or suspicious activity [20]. IDS use configuration files, called signatures, that are designed to recognize patterns in network activity. Signatures can be developed on a case-by-case basis depending upon an analyst's needs. IDS detect intrusions for active



network communications and stored log files. For this research, the focus is on the second function: the ability to filter traffic based on a custom signature of logged network traffic. The next section discusses the availability of ICS and a popular myth about their isolation.

## **2.4 Control Systems Availability**

The basis of this research was motivated by a paper entitled, “Quantitatively Assessing and Visualising Industrial System Attack Surfaces,” written by Éireann Leverett [21]. Leverett debunked the popular myth that ICS are not connected to the Internet. Additionally, his work presented logging data from connections and visualized them alongside vulnerability information available from ICS-CERT advisories. His intent was to gather information on ICS through the use of open source information, following the restrictions below:

- Do not interact with any device other than viewing the HyperText Transfer Protocol (HTTP) interface.
- Do not attempt to login to any device, when prompted for credentials, cancel any interaction.
- Do not actively scan the Internet for devices, rather use existing information.

In order to abide by the third restriction, Leverett leveraged the Shodan search engine. Shodan is a search engine much like Google, but instead of searching for websites, Shodan searches for specific computers running certain software [23]. Once a device is found, banners are used (see Figure 2.3) to further categorize device types and running services. For example, Figure 2.3 shows a successful connection to a EnergyICT RTU. Finally, the list of known devices are cross referenced with vulnerability and exploitation databases in order to discern possible weaknesses.

```
HTTP/1.0 200 OK
Date: Sat, 23 Apr 2011 21:1:34 GMT
Content-Type: text/html
EnergyICT RTU 130-D93392-0840
Expires: Sat, 23 Apr 2011 21:1:34 GMT
```

Figure 2.3: An example banner from Shodan [21].

At the end of the two-year experiment period, 7,500 control systems were identified. Of those 7,500, only 17 percent had authentication mechanisms and 20.5 percent had published remote exploits. The results encompassed responses received globally; however, 52 percent of the exposed devices resided in the USA. Figure 2.4 is a visual representation of ICS devices identified during the research.

The research was successful in debunking the myth that ICS are not connected to public networks. The results show that not only are there myriad control systems exposed, but they are also vulnerable to readily available exploits. The research serves as an eye opener for the ICS community, as the awareness of who is connecting to these systems, and for what purpose, is unknown. The next section examines some common attacks available to employ against these connected devices.

## **2.5 Attack Techniques**

There are a multitude of attack vectors for ICS. HMI software is typically deployed on a Microsoft based operating system (e.g., Microsoft Server, Windows XP/Vista/7) and is subject to the inherent operating systems' vulnerabilities, as well as vulnerabilities that may exist in the HMI software. Additionally, Reid Wightman, of Digital Bond, led a team of

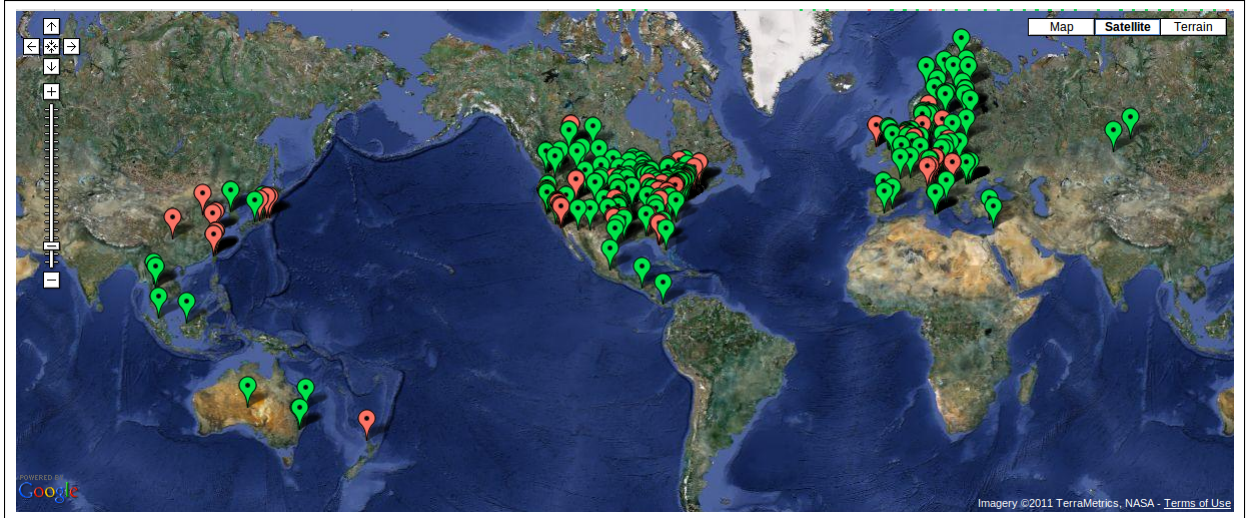


Figure 2.4: Visual representation of available ICS [21].

researchers in Project Basecamp to test the security of specific end devices [33]. The team discovered flaws in the device’s remote services as well as authentication mechanisms.

Many devices in ICS are preconfigured with a default web server that allows an operator to remotely change configurations or view the status of processes. Most of these devices require authentication, namely username/password, in order to make major changes. However, some do not require any authentication whatsoever. The use of these web servers increase the remote availability of ICS devices and, as a result, expose them to common vulnerabilities associated with this feature. Table 2.1 describes some common attacks used against web servers and their impact on ICS.

Attack	Description
Cross Site Scripting (XSS)	Stealing private data through due to improperly validated input, allowing malicious code execution on any machine that views the site. This can be used to steal cookies or session IDs without the knowledge of the user, used in conjunction with session hijacking or replay attacks [30].
SQL Injection	A technique used for manipulating web services that send SQL queries to alter, insert, or delete data in a database. This can be used in ICS to reveal or change information stored in the Data Historian, or any database attached to ICS [29].
Directory Traversal	Occurs when an attacker tries to access restricted files used by the web server. These attacks can be used to bypass authentication and access the devices configuration or password files, as well as to execute arbitrary commands [29].
Denial of Service (DoS)	A technique designed to stop or suspend system service by either overwhelming the server with inputs, or sending malformed packets.
Information Leakage	Used to gather information about a target. Information displayed to users, such as a model number, manufacturer, location, or operating thresholds, can be used by an attacker to find vulnerabilities specific to the device [34].
Fuzz Testing	Implemented by tools called <i>fuzzers</i> , which are programs or scripts that submit some combination of inputs to the test target to reveal how it responds [18]. In ICS networks, fuzz testing can be devastating because most devices are designed to only handle preconfigured inputs. If the inputs deviate, the device could seize up or fail altogether.

Table 2.1: Common attacks used against web servers in ICS.

Vulnerabilities in ICS may occur from flaws, misconfigurations, or improper administration of ICS network connections. These particular vulnerabilities exist in the various authentication mechanisms employed across the ICS domain. Table 2.2 describes some common attacks used to exploit authentication vulnerabilities and their impact on ICS.

<b>Attack</b>	<b>Description</b>
No password	Some devices are configured requiring no authentication or having authentication as an optional feature.
Default Passwords	Most devices that have require authentication are preconfigured with a default password. The password is often in the manual and is available on the vendors website. One of the attackers initial tests would be to see if the default password has been changed, if not they can log in and make changes without the operators knowledge.
Brute Force	Brute force is an attack that enumerates all possible password options. Due to hardware limitations in ICS, there is a tendency to use small passcodes or passwords that can be brute forced in a matter of hours and because there is no logging, the operator has no knowledge of the attack.
Replay Attack	A replay attack is when a malicious actor records authentication information of a successful login, and replays that information to the target posing as the legitimate user.
Session hijacking	The exploitation of a computer session to gain unauthorized access to a devices hosted services. This is often used in conjunction with Cross Site Scripting and replay attacks, with the objective of stealing the cookie or session id of a legitimate session and leveraging it to establish or resume the authorized connection.

Table 2.2: Common attacks used against authentication used in ICS.

## 2.6 Incidents

This section discusses threats and incidents involving ICS. Note, not all impacts resulted from malicious intent, however, the results demonstrate direct physical impact.

### 2.6.1 Threat Picture.

Threats to ICS can come from numerous sources to include intentional, accidental, human error, and even natural disasters. It is important to understand the threat landscape in order to develop proper defenses against it. The Guide to Industrial Control Systems (ICS) Security [31], describes a list of possible ICS threats that are shown in Table 2.3.

Threat Agent	Description
Attackers	Attackers break into networks for the thrill of the challenge or for bragging rights in the attacker community. While remote cracking once required a fair amount of skill or computer knowledge, attackers can now download attack scripts and protocols from the Internet and launch them against victim sites. Thus, while attack tools have become more sophisticated, they have also become easier to use. Many attackers do not have the requisite expertise to threaten difficult targets such as critical U.S. networks. Nevertheless, the worldwide population of attackers poses a relatively high threat of an isolated or brief disruption causing serious damage.
Bot-network operators	Bot-network operators are attackers; however, instead of breaking into systems for the challenge or bragging rights, they take over multiple systems to coordinate attacks and to distribute phishing schemes, spam, and malware attacks. The services of compromised systems and networks are sometimes made available on underground markets (e.g., purchasing a denial of service attack or the use of servers to relay spam or phishing attacks).
Criminal groups	Criminal groups seek to attack systems for monetary gain. Specifically, organized crime groups are using spam, phishing, and spyware/malware to commit identity theft and online fraud. International corporate spies and organized crime organizations also pose a threat to the U.S. through their ability to conduct industrial espionage and large-scale monetary theft and to hire or develop attacker talent.
Foreign intelligence services	Foreign intelligence services use cyber tools as part of their information gathering and espionage activities. In addition, several nations are aggressively working to develop information warfare doctrines, programs, and capabilities. Such capabilities enable a single entity to have a significant and serious impact by disrupting the supply, communications, and economic infrastructures that support military powerimpacts that could affect the daily lives of U.S. citizens.

Table 2.3: Threats against ICS [31].

Threat Agent	Description
Insiders	The disgruntled insider is a principal source of computer crime. Insiders may not need a great deal of knowledge about computer intrusions because their knowledge of a target system often allows them to gain unrestricted access to cause damage to the system or to steal system data. The insider threat also includes outsourcing vendors as well as employees who accidentally introduce malware into systems. Insiders may be employees, contractors, or business partners. Inadequate policies, procedures, and testing can, and have led to ICS impacts. Impacts have ranged from trivial to significant damage to the ICS and field devices. Unintentional impacts from insiders are some of the highest probability occurrences.
Phishers	Phishers are individuals or small groups that execute phishing schemes in an attempt to steal identities or information for monetary gain. Phishers may also use spam and spyware/malware to accomplish their objectives.
Spammers	Spammers are individuals or organizations that distribute unsolicited e-mail with hidden or false information to sell products, conduct phishing schemes, distribute spyware/malware, or attack organizations (e.g., DoS).
Spyware/malware authors	Individuals or organizations with malicious intent carry out attacks against users by producing and distributing spyware and malware. Several destructive computer viruses and worms have harmed files and hard drives, including the Melissa Macro Virus, the Explore.Zip worm, the CIH (Chernobyl) Virus, Nimda, Code Red, Slammer, and Blaster.
Terrorists	Terrorists seek to destroy, incapacitate, or exploit critical infrastructures to threaten national security, cause mass casualties, weaken the U.S. economy, and damage public morale and confidence. Terrorists may use phishing schemes or spyware/malware to generate funds or gather sensitive information. Terrorists may attack one target to divert attention or resources from other targets.
Industrial Spies	Industrial espionage seeks to acquire intellectual property and know-how by clandestine methods

Table 2.4: Threats against ICS (cont) [31].

### ***2.6.2 Maroochy Shire.***

Between February 28, 2000 and April 23, 2000, at least 46 attacks were conducted on sewage equipment in Australia [2]. Vitek Boden worked for Hunter Watertech, an Australian firm specializing in installing SCADA radio-controlled sewage equipment for the Maroochy Shire Council in Queensland, Australia. After leaving Hunter Watertech, Boden applied for a job directly with the Maroochy Shire Council and was turned down. Seeking revenge on both the Council and Hunter Watertech, Boden drove around the area with stolen radio equipment attacking sewage facilities on at least 46 separate occasions. The results were devastating. Over 800,000 liters of raw sewage was dumped into local parks, rivers, and creeks turning water black and killing local marine life. Normally, these systems are set up to alarm engineers if such malfunctions occur, however, Boden disabled alarms at four of the major pumping stations. This allowed him to carry out the attacks with a relatively low chance of being caught. The attack was carried out with insider knowledge and motivated by revenge. The system under attack had no existing cyber security policy, or procedure, to deal with such a threat and, as a result, had no cyber security defenses in place to thwart the attack.

### ***2.6.3 Davis-Besse Nuclear Power Plant.***

In January 2003, the Slammer Worm infected the Davis-Besse nuclear power plant in Ohio, disabling safety monitoring software for five hours [11]. The worm was introduced via a private contractor's laptop. Once within the plant's boundaries, it replicated and spread to factory systems over a T1 line. Investigators later found that the line between the contractor's laptop and the plant's computers bypassed the firewall entirely. Fortunately, the plant had already been shut down for maintenance, so the worm did not pose a serious risk at the time.



#### ***2.6.4 Worcester Air Traffic Communications.***

In March 1997, a teenager in Worcester, Massachusetts disabled phone services using a dial-up modem [9]. The disruption caused phone blackouts to the air control tower, airport security, airport fire department, and weather service. Additionally, the tower's main radio transmitter, and transmitters that control runway lights, were disabled, as well as a printer used to monitor flight progress. In addition to the airport, phone services to over 600 homes and businesses were knocked out in the nearby town of Rutland.

#### ***2.6.5 Stuxnet.***

Stuxnet has received a considerable amount of attention from researchers and media around the world. It is one of the most complex pieces of malware discovered thus far [14]. This particular threat propagated through IT systems, but its final goal was to target and reprogram SCADA devices. Stuxnet was not discovered until July 2010, but is thought to have been implemented one year prior to its discovery. The worm spread over traditional IT networks and contained many different exploits including zero-days, forged certificates, default password attacks, and rootkits. Once the worm gained access to a computer it would look for a certain SCADA PLC controlling programs and would rewrite instructions to carry out different operations. It would also send signals to the operator, telling them that all systems are operating as normal. One of its main functions was to alter equipment processes in order to damage materials. The result slowed production down considerably and costing the facility an untold amount of financial losses. Stuxnet is the first of its kind, propagating through commercial networks and containing multiple payloads depending on which system it infected. Stuxnet required extensive insider knowledge of specific devices and protocols, indicating a well funded and well trained adversary. The post-stuxnet consensus indicated that the author was most likely a nation state targeting other countries infrastructure.

## 2.7 Cyber Sensors

As stated previously, it can be expensive and arduous to deploy any security solution in ICS environments. Most of the security mechanisms mentioned in Section 2.2 are used on computing platforms that provide extensive system resources, such as dedicated servers, desktop workstations, or laptops. The size and cost of these platforms hinder their ability to be distributed among large scale networks. This research leverages tiny computer modules referred to as cyber sensors. Cyber sensors use fully functional ‘micro’ computers that are designed to run a modern operating system. Most of these devices are about the size of a credit card or smaller, (see Figure 2.5), and bear the capabilities for standard input/output (I/O) (e.g., keyboard, mouse, video output and sound output) and communication (e.g., USB, Ethernet, or WiFi). These cyber sensors can be configured to execute the same applications as full scale computers as long as the applications system resources requirements do not exceed those of the sensor. Cyber sensors, however, have limited resources, so while they do not perform well executing multiple resource intensive applications, they excel in one or two smaller applications. This makes them ideal candidates for small robust sensors in the ICS domain. The sensors used for this research are discussed further in Section 3.4.3.



Figure 2.5: Various microcomputers used as cyber sensors.

## 2.8 Situational Awareness

This section describes related research for increasing situational awareness in the field of ICS. Dr. Mica Endsley defines situational awareness as, “the perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future [13].” There are three functions that a system must perform to achieve situational awareness: (i) it must sense its environment; (ii) it must take raw data and assemble it into a meaningful understanding of its environment; and (iii) it must use its current understanding to predict the future [25].

### 2.8.1 Logging.

The first function, the system must sense its environment, can be accomplished with the use of extensive logging mechanisms, such as network loggers and honeypots.

Morris and Pavurapu [24] developed an embedded device that can be attached to networks at the PLC or RTU level and capture traffic on specific protocols. The logging solution provides the capability to securely record and store traffic without affecting the performance of the end device. It does not, however, provide TCP/IP capability nor the flexibility to be distributed over different areas of an ICS network.

Chandia, *et al.*, [8] proposed a forensics architecture that can be used on ICS communications. In this architecture, agents capture traffic at three levels and forward all traffic to a centralized server for storage and analysis. Level 1 agents capture traffic at the control station, or master node location. Level 2 agents collect information at intermediate locations in the network between Level 1 and Level 3. Level 3 agents capture network traffic at the end device, such as PLCs, RTUs, or IEDs. It is important to note that all communications from the centralized storage server to the agents occur on a separate isolated network. While Morris and Pavurapu’s embedded device is best suited for level 3 collection, it is not suited for full situational awareness across the levels.

Some ICS vendors offer logging features for their RTU, IED, and PLC devices. For instance, Microsystems, Inc offers SCADAPack 350 and SCADAPack 357 [12]. While they provide the capability for the RTU to be connected to external storage and log process data, they do not allow for logging network transactions going to and from the RTU.

Another important research area to note is the use of honeypots in ICS. Honeypots are typically deployed in IT systems and are designed to impersonate production devices. All traffic going to and from the honeypot is logged and stored for further analysis. It is important to note that because these are fake systems, there should be no legitimate reason to connect to the device, therefore, all data collected is suspicious. Digital Bond [6] has developed a honeynet that is capable of emulating five popular ICS protocols using virtual machines. A honeynet is simply a network of honeypots. This research is a proof-of-concept to demonstrate that honeypots can be used in ICS. However, it lacked the robustness to be deployed and interacted with in production environments. Dustin Berman [4] furthered honeypot research by successfully emulating a PLC on a Gumstix Overo COM (cyber sensor). The research indicates that the device was able to respond correctly to a number of different connections (e.g., common protocols, fingerprinting, MAC address resolution, and invalid traffic) and successfully captured all network traffic involved. This research provides the capability to capture information at the PLC level and can be distributed across an ICS network to aggregate data. Honeypots can be a valuable asset for increasing situational awareness in control systems, but they are limited in that they only record data going to and from that device.

### ***2.8.2 Intrusion Detection Systems.***

The second function of situational awareness, it must take raw data and assemble it into a meaningful understanding of its environment, can be accomplished with the use of pattern analyzers, such as intrusion detection systems.

Digital Bond has developed an IDS signature package that encompasses four different ICS communication protocols [5]. The signatures are designed to integrate with pre-deployed intrusion detection systems, and can identify unauthorized requests, malformed protocol requests, and dangerous commands. Note that while these signatures are useful in identifying attacks, they require a pre-existing sensor infrastructure that is not common in ICS today.

Fovino *et al.* [16] present a state-based intrusion detection system designed specifically for SCADA architectures. The proposed system blends together traditional signature based IDS with an innovative state analysis technique. The solution keeps track of the state of a control system and monitor it for malicious activity. While the results show promise, it does not allow monitoring traffic beyond the DNP3 and Modbus protocols.

### **2.8.3 Analytics.**

The third function of situational awareness, using current understanding to predict the future, can be accomplished through analyzing the data gathered from logging and IDS for predictable behavior. Because this function relies on previously recorded data, analytics in the ICS environment is lacking, however, the data gathered from real world incidents has been used to develop some threat detection capabilities [5]. Note that looking at future threats is outside the scope of this research, however, the data collected can be used to identify those threats.

## **2.9 Summary**

This chapter explains ICS, ICS security, and the importance of logging and categorizing information within the network. It describes how accessible control systems are to the public, the types of attacks that can occur, and incidents that have resulted. This chapter also details ICS current abilities to log network traffic and the affect it has on situational awareness. The next chapter discusses the methodology used to evaluate the effectiveness of the logging solution created as part of this research.

### **III. Methodology**

#### **3.1 Problem Definition**

Current logging capabilities for ICS field devices are inadequate or often nonexistent. Most devices are designed for specific functionalities, and do not have the necessary computing power to carry out proper logging. Without the ability to record activity, the ICS environment lacks the situational awareness required to gather intelligence on targeted attacks.

#### **3.2 Goals**

The goal of this research is to demonstrate that logging capabilities can be implemented using a low cost solution without suffering a loss of performance, given limited resources. This research focuses on two functionalities: (i) performance evaluation of each sensor (functionality test) and (ii) ability to distinguish malicious traffic from standard traffic (identification test).

#### **3.3 Hypothesis**

It is expected that inexpensive computing systems perform adequately as cyber sensors by logging ICS field traffic and differentiating targeted malicious traffic from legitimate ICS traffic. It is expected that the inexpensive cyber sensors will perform within 5% of the baseline laptop device and identify malicious traffic.

#### **3.4 Environment**

This section contains the hardware and software specifications used in this experiment. Figure 3.1 shows the evaluation environment. Traffic that conforms to standard ICS parameters is generated to emulate operational packets sent to an ICS field device. Note that the actual field device is not within the scope of this research, the interest is in the network

traffic destined for the device. The sensors are placed in promiscuous mode in order to receive, log, and process the packets. Captured traffic is then forwarded to a Log Analyzer for analysis and to discern malicious or unexpected activities. The Log Analyzer enables centralized processing, while providing a capability to distribute the sensors throughout an operational environment.

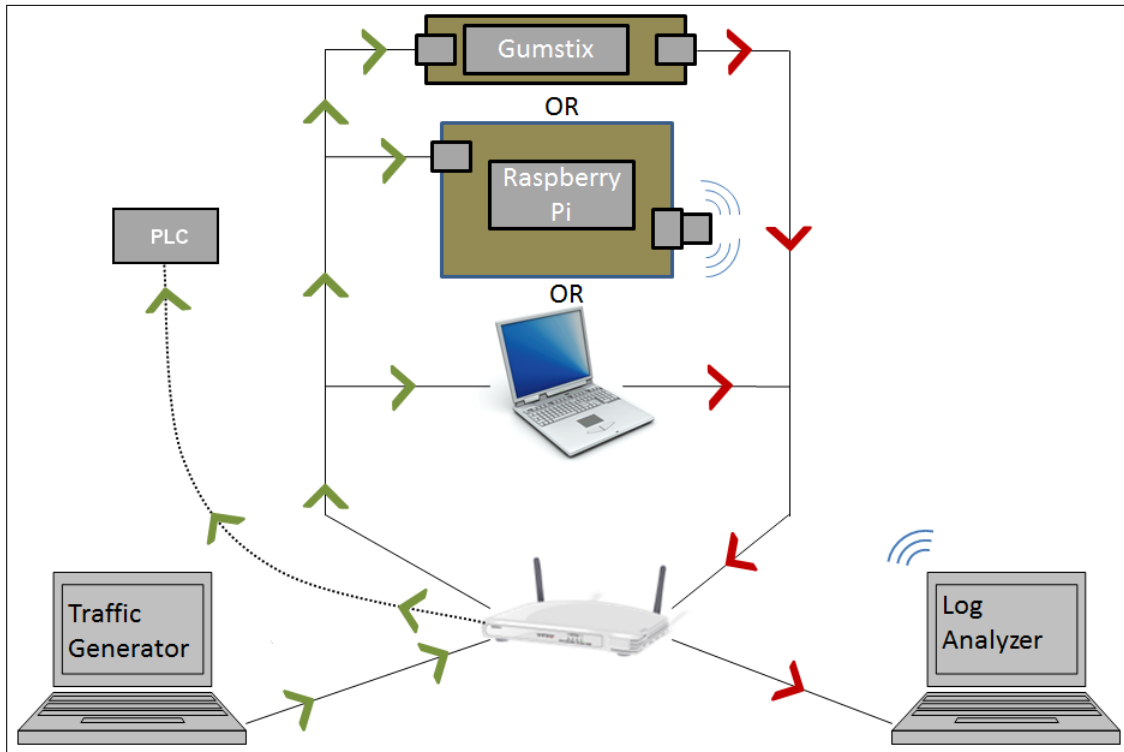


Figure 3.1: Evaluation environment.

### 3.4.1 Traffic Generator.

The Traffic generator is a virtual machine (VM) running 32-bit Ubuntu 12.04 with a 20GB hard drive, 4GB of memory, and 1 core of an i7-M640 CPU at 2.80 GHz. The software, Nping, generates standard ICS network traffic. Nping is an open-source tool that provides the capability to generate raw random packets for numerous protocols and can be used for network stress testing [22]. The software, netcat, generates a password replay

attack that is consistent with exploiting an ICS field device. Netcat is a networking utility which reads and writes data across a network [17].

### ***3.4.2 Log Analyzer.***

Similar to the Traffic Generator, the Log Analyzer is a VM running 32-bit Ubuntu 12.04, with a 20GB hard drive, 4GB of memory, and 1 core of an i7-M640 CPU at 2.80GHz. This machine is used as a repository for all logs generated by each sensor. The tool, Snort, examines the traffic logs for malicious activity. Snort is an open source network intrusion detection system developed by Sourcefire that can be customized to include unique attack signatures and detection [27]. For this research, Snort signatures are developed that identify traffic that does not conform to specified operating parameters.

### ***3.4.3 Cyber Sensors.***

The cyber sensors are placed in promiscuous mode, connected to the router, and receive traffic generated for the PLC device. The logging software is tcpdump version 4.2.1. Tcpdump is a command-line network packet analyzer that utilizes the libpcap library [19]. The traffic is stored on the sensor until it reaches a specified size, then uses bzip2 to compress the traffic and secure copy (scp) to transfer the compressed traffic to the Log Analyzer for further analysis. Note that the transfer occurs on the back channel using the secondary network interface to enhance operational security.

The **Gumstix Overo Earth COM** is a small computer that, not surprisingly, is about the size of a stick of gum (see Figure 3.2). It runs a Linux based platform leveraging the Open Embedded framework and costs approximately \$149 [1]. The Gumstix board has a ARM Cortex-A8 600MHz processor, 512MB of memory, and microSD card slot for non-volatile storage. For this research, an 8GB microSD card is used. Gumstix do not natively possess I/O capabilities and must leverage expansion boards to extend interactive operation such as bluetooth, GPS, 802.11 wireless, and Ethernet. For this research, the Tobi-Duo (an additional \$79, see Figure 3.2) expansion board is used to provide dual NIC functionality.



A primary NIC captures live network traffic and a secondary NIC sends compressed traffic logs, on the back channel, to the Log Analyzer for post process analysis. Both the primary and secondary NICs operate at 10/100 MBPS. The operating system on the Gumstix is Linux 2.6.34. Appendix C.2 provides specific configuration details used in this research.

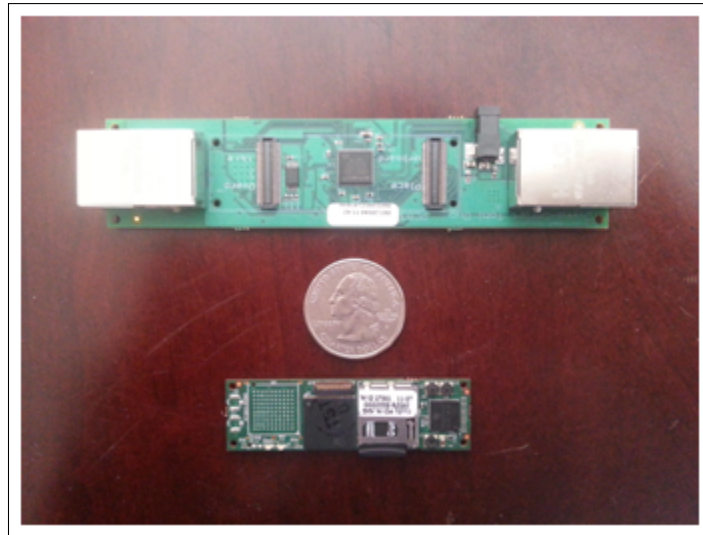


Figure 3.2: Tobi Duo expansion board (top), Gumstix Overo Earth (bottom).

The **Raspberry Pi Model B** is a microcomputer depicted in Figure 3.3 and costs \$35 [26]. It uses an ARM1176 800MHz processor, 512MB of memory, and SD card slot for non-volatile storage. For this research an 4GB SD card is used. Unlike the Gumstix, the Raspberry Pi does not require expansion boards to provide basic I/O capabilities; it comes with a single NIC and various I/O ports such as USB, HDMI, and audio out. For this research, a USB wifi dongle is used, specifically Wi-Pi (an additional \$16.50), to extend the Raspberry Pi's communication capabilities to match the Gumstix ability for dual communication. The primary 10/100 MBPS NIC is used as the capture interface, and the Wi-Pi is used to send compressed logs on the back channel. The Wi-Pi supports 802.11n wireless networks, and is capable of speeds of 150 MBPS. The operating system native to

the Raspberry Pi is Linux kernel 3.2.27. Appendix C.3 provides the specific configuration details.

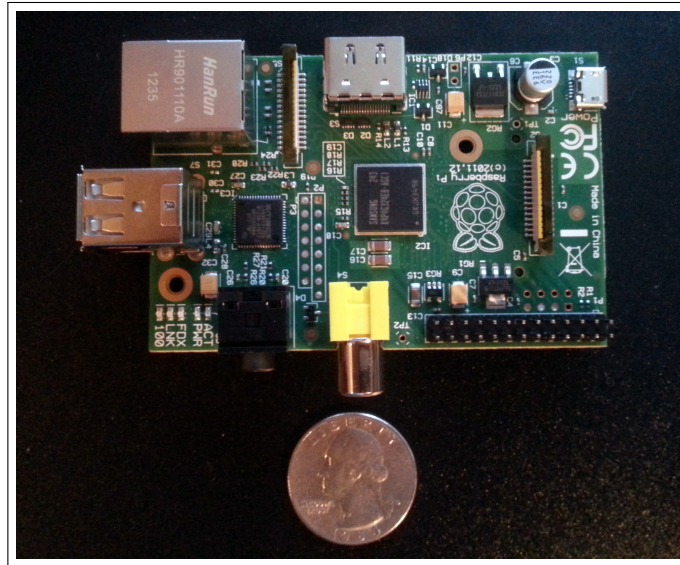


Figure 3.3: Raspberry Pi.

A **standard laptop (LAPT)** is a VM running 32-bit Ubuntu 12.04. It has a 20GB hard drive, 4GB of memory, and has been allocated 1 core of an i7-2620M CPU at 2.70GHz. These particular specifications in a laptop typically can cost between \$700 to \$1000 and is considered standard. The laptop uses a primary 10/100/1000 MBPS Ethernet connection to capture traffic, and a 802.11n 300 MPBS wireless interface to send compressed traffic on the back channel.

Table 3.1 provides a specification summary of all sensors. The CPU determines the overall system speed, RAM affects the speed in which the system can execute processes, and the NIC affects the speed of network traffic sent and received.

The laptop serves as the baseline to evaluate the performance of the cyber sensors. The laptop is consistent with a system deployed in traditional IT systems for logging capabilities.

<b>Cyber Sensor</b>	<b>CPU</b>	<b>Storage</b>	<b>RAM</b>	<b>NIC (MBPS)</b>	<b>OS</b>	<b>Tot. Cost</b>
Gumstix Overo Earth	600 MHz	8GB	512MB	10/100 Ethernet 10/100 Ethernet	Linux 2.6.34	\$228
Raspberry Pi	800 MHz	4GB	512MB	10/100 Ethernet 150 Wireless	Linux 3.2.27	\$51.50
Laptop PC	2.8 GHz	20GB	4GB	10/100/1000 Ethernet 300 Wireless	Ubuntu 12.04	\$700-\$1000

Table 3.1: Sensor specifications.

#### **3.4.4 Communication.**

A Trendnet (TEW-672GR) router connects all sensors for communication. The router is capable of 10/100/1000 MBPS connections on LAN, and 300 MBPS on wireless. The Gumstix sensor connects both NICs via wired Ethernet connection. The Raspberry Pi sensor connects via one wired Ethernet and one wireless connection. The LAPT connects via one Ethernet connection and one wireless connection. Note that the capability of the router exceeds that of the sensors so it is expected that it will not impact packet loss in this research. This router is chosen because it is consistent with specifications used in an operational environment.

### **3.5 Evaluation Technique**

The three cyber sensors are evaluated for performance characteristics. The Gumstix and Raspberry Pi sensors are selected because they are the leading technology for inexpensive, portable, and reliable computer systems [1]. The laptop serves as a baseline in which to compare the sensor's performance. This section discusses the techniques used to evaluate the cyber sensors' capabilities.

#### **3.5.1 Network Traffic.**

Network traffic is used to test the cyber sensors' packet capturing limitations. Network traffic consists of packet rate, event rate, and traffic type. Aspects are varied to measure performance.

## **Packet Rate**

Packet rate is the rate of traffic, measured in packets per second, sent to the sensors. For this research four values are used: base, low, medium, and high. Note that the packet rates were derived from examination of packet captures of actual production ICS. The base rate is defined at 1 packet per second (pps). Low traffic is 10 pps and is representative of an operational liquid pipeline system. Medium traffic represents 100 pps and is associated with more active ICS networks including small electric power grids or waste water facilities. High rate is chosen to represent an overly active network associated with large electric power grids or HVAC systems and is defined as 1000 pps.

## **Event Rate**

Rate of events is the number of malicious events targeting the field device. The three event rates low, medium, and high represent: 5, 10, and 20 events per hour respectively. Note that the signature is designed to alert on specific traffic patterns and is not dependent on when they occur, so for an added level of control during testing, the intervals between malicious events were not randomized. This ensures consistency across the different tests for evaluating identification of malicious traffic.

## **Traffic Type**

Traffic type describes the type of traffic generated and sent to the sensors. There are two traffic types: standard or non-standard. Standard traffic is generated using Nping to deliver ICS payloads via the HTTP (port 80) protocol. HTTP is a common protocol for ICS to use when serving up management web sites to the engineer. These web sites contain the ability to configure the devices and are often vulnerable to basic attacks. Note that the traffic used in this research was derived from actual production systems to represent traditional ICS packet size and protocols. Non-standard traffic is traffic that does not conform to predefined operational parameters and is indicative

of malicious events. For malicious events a replay attack is used to generate an unauthorized administrator login to a PLC's web server configuration page. It is important to point out that due to the design of many ICS components, traffic does not necessarily have to be malicious for it to damage or degrade service. In this research a known vulnerability for a particular series of PLCs (MicroLogix 1400) is used to serve as non-standard traffic to demonstrate a likely real world scenario [32].

### **3.5.2 Performance Metrics.**

It is vital that each sensor remains available and perform as intended. These following metrics are used to evaluate the outcome of the experiments.

#### **Accuracy**

##### **Packets Logged**

For each packet sent, the same packet must be logged. In the functionality test, Nping is used to calculate how many packets are generated, and tcpdump on the cyber sensor is used to calculate how many packets have been captured. The metric recorded is [No. of packets received] / [No. of packets sent].

##### **Event Logged**

In the identification test, a script is used in conjunction with netcat to generate the desired amount of malicious events. The logged traffic confirms the number of events received. The metric recorded is [No. of events received] / [No. of events sent].

##### **Compressed**

Once tcpdump logs 2MB of packets, it compresses the log file and rotates to a fresh file. In order to test the sensor's compression performance thoroughly, a larger, 20MB file, is used during the high rate. Compression takes place on the

sensor and is measured by listing and comparing files. The metric recorded is [No. of compressed log files] / [No. of total log files].

### **Transferred**

Once the log files are compressed, they are transferred to the Log Analyzer. A comparison is performed between the files sent from the cyber sensor and the files received on the Log Analyzer. The metric recorded is [No. of compressed files received] / [No. of compressed files sent].

### **Identification**

The main metric gathered in the Identification Test Case is how many events are identified. Snort is used with a custom filter to identify all events that occurred. The metric is [No. of events identified] / [No. of events sent].

## **Timing**

### **Compression**

The time to compress a file depends on the sensor's resources and may vary when other resource intensive processes are running. As each file is compressed, the time command available in Linux is used to determine how long the process takes to execute, and is measured in seconds. The metric recorded is [No. of seconds to compress].

## **3.6 Experiments**

The experiments examine the cyber sensor's functionality and identification capability. The functionality test is designed to determine if the sensors can perform sufficiently under different rates of traffic that is measured in packets per second. Additionally, it is designed to examine the sensor's ability to adequately compress and transfer log files of varying size. The traffic identification test is designed to determine if suspicious traffic can be accurately recorded and identified. Note that ICS traffic is predictable and consists of a

small subset of traffic functionality. Any traffic not conforming to expected traffic patterns and functionality are identified as suspicious. The ease of developing Snort signatures ensures that the ability to define expected traffic patterns is readily expandable to meet specific ICS operational environments.

### **Functionality Test**

There are 12 tests consisting of 4 varying packet rates (base, low, medium, and high) for the 3 cyber sensors shown in Table 3.2. Each test is repeated 3 times for statistical significance using a 95% confidence interval. The order of each test is randomized to rule out equipment anomalies. Note that Appendix C.1 shows the steps to ensure randomization. The following steps are used to conduct the tests:

1. Tcpcmdump is started on the cyber sensor under test.
2. A script is executed on the Traffic Generator to start sending packets at the specified rate, using Nping, to the sensor. Note that the script is set to run for 1 hour to ensure a sufficient amount of traffic for analysis.
3. Once the capture traffic reaches the size of 2MB for the base, low, and medium rates or 20MB for the high rate, it is compressed using bzip2 and transferred to the Log Analyzer using scp.

All scripts record the outputs of each process and performance metrics are gathered for each test. The performance metrics associated with this test are accuracy and timing. Accuracy consists of packets logged, log files compressed, and compressed files transferred. Timing consists of compression time.

### **Traffic Identification Test Case**

There are 9 tests consisting of 3 event rates (low, medium, high) for the 3 cyber

<b>Parameters</b>	<b>Rate</b>	<b>Type</b>	<b>Service</b>	<b>Platform</b>
<i>Levels</i>	<i>Low, Med, High</i>	<i>Standard</i>	<i>Web</i>	<i>LAPT, Raspberry Pi, Gumstix</i>
1	Base	Standard	Web	LAPT
2	Low	Standard	Web	LAPT
3	Medium	Standard	Web	LAPT
4	High	Standard	Web	LAPT
5	Base	Standard	Web	Raspberry Pi
6	Low	Standard	Web	Raspberry Pi
7	Medium	Standard	Web	Raspberry Pi
8	High	Standard	Web	Raspberry Pi
9	Base	Standard	Web	Gumstix
10	Low	Standard	Web	Gumstix
11	Medium	Standard	Web	Gumstix
12	High	Standard	Web	Gumstix

Table 3.2: Functionality test enumeration.

sensors which is shown in Table 3.3. The following steps are used to conduct the tests:

1. Tcpdump is started on the cyber sensor.
2. The same script used in the functionality test is used to generate standard traffic at a medium rate to serve as an operating environment. Note that medium traffic was selected to serve as white-noise and should not diminish identification accuracy during post process analysis.
3. Netcat is started on the sensor to accept incoming HTTP requests.
4. Another script is executed on the Traffic Generator to start sending malicious events at the specified rate. For this research, a password replay attack is generated and sent using netcat. The script is set to run for 1 hour in order to generate a sufficient amount of traffic for analysis.



5. Once the capture traffic reaches the size of 2MB for the base, low, and medium rates or 20MB for the high rate, it is compressed using bzip2 and transferred to the Log Analyzer using scp.
6. The Log Analyzer unzips and filters the traffic to remove standard traffic using Snort and a custom signature filter.

All scripts record the outputs of each process and performance metrics are gathered for each test. The performance metric associated with this test is accuracy. Accuracy consists of events logged and events identified.

<b>Parameters</b>	<b>Rate</b>	<b>Type</b>	<b>Service</b>	<b>Platform</b>
<i>Levels</i>	<i>Low, Med, High</i>	<i>Non-Standard</i>	<i>Web</i>	<i>LAPT, Raspberry Pi, Gumstix</i>
1	Low	Non-Standard	Web	LAPT
2	Medium	Non-Standard	Web	LAPT
3	High	Non-Standard	Web	LAPT
4	Low	Non-Standard	Web	Raspberry Pi
5	Medium	Non-Standard	Web	Raspberry Pi
6	High	Non-Standard	Web	Raspberry Pi
7	Low	Non-Standard	Web	Gumstix
8	Medium	Non-Standard	Web	Gumstix
9	High	Non-Standard	Web	Gumstix

Table 3.3: Identification test enumeration.

### **3.6.1 Effectiveness.**

The results of the functionality and traffic identification tests for the two cyber sensors are compared to the baseline laptop. Acceptable results may be case dependent based on the financial limitations of the organization, however, for this research all performance metrics within 5% of the baseline laptop are considered effective. The performance metrics that will be compared are percent of packets logged, compressed, transferred, and identified, as well as the time to compress the log files.

### **3.7 Methodology Summary**

With the rapid push to integrate ICS into traditional IT networks, it is necessary to employ security solutions. The approach taken in this methodology is to integrate localized logging capabilities into these networks. ICS networks are the backbone of providing critical services across the globe. Since the ability to monitor attacks is limited, this effort determines if logging capabilities can be implemented cost effectively. The design of this experiment consists of two main cases. The first case tests 4 rates using standard traffic against the HTTP service for all 3 sensors, and is repeated 3 times. The second case tests similar parameters differing only with 3 rates. Each experiment is designed to provide metrics on performance that are used to evaluate the effectiveness of each cyber sensor ultimately deciding if the sensor performs at an effective level.

## IV. Analysis and Results

This chapter discusses research results. Implementation details for the evaluation environment are discussed, followed by initialization checks run prior to each test. Next are the results of the functionality and identification tests for each sensor: standard laptop, Raspberry Pi, and Gumstix. An examination of the test results is discussed, followed by a summary concluding the chapter.

### 4.1 Implementation Details

This section describes the implementation details of the cyber sensors, Log Analyzer, and malicious event development.

#### *4.1.1 Cyber Sensor.*

A logging algorithm was developed and deployed across all three sensors. The logging algorithm consisted of executing tcpdump, bzip2, and scp in a particular order with specified configurations. Tcpdump is set to capture raw packets on port 80, that have a source IP of the Traffic Generator. Once the traffic log reaches the specified size, it closes the file, executes a script, and rotates to a fresh file. The script executes bzip2 to compress the closed log file, transfers it to the Log Analyzer, and deletes the compressed file if the transfer was successful. The transfer occurs on the back channel communication using scp and is encrypted with a 2048 bit RSA public/private key pair. Note that the software versions for tcpdump, bzip2, and scp were the same on all three cyber sensors. Further details are described in Appendix C.

#### *4.1.2 Log Analyzer.*

The Log Analyzer is a VM running Ubuntu and Snort. As the logs are received from the cyber sensors, they are decompressed using bzip2, and merged using mergecap. Mergecap is a tool included in the libpcap library that merges multiple log files into one.

Snort is then executed on the merged file, to filter out all standard traffic, leaving only the malicious events. The signature used in this research alerts on any traffic that does not conform to standard traffic. Using this signature, Snort alerts on any traffic trying to access the administrative page on a PLC web server. Specifically, the signature alerts when the traffic data contains the username ‘administrator,’ indicating that an attempt to log in with administrator credentials has occurred. The Log Analyzer’s complete configuration and Snort signatures is in section B.3.

#### ***4.1.3 Event Development.***

For this research a man-in-the-middle attack was chosen to represent a malicious event, specifically a password replay attack. Wireshark was used to capture traffic from a Micro Logix 1400 PLC during a successful administrative login to the device’s configuration web page. The page request was captured and used as the replay attack payload, sent by the Traffic Generator. The Traffic Generator initiates a connection with the cyber sensor via netcat, and transmits the payload. Specific event development details are provided in Appendix B.

## **4.2 Initialization Checks**

Prior to starting each test case, initialization and validation checks were executed to ensure the systems were operational and functioning properly.

### ***4.2.1 Functionality Test Case.***

The following initialization and validation checks were executed for the functionality test case:

- Network communications;
- The Traffic Generator sends intended traffic at the designated rate;
- The sensors capture, compress traffic, and transferring logs; and

- The Log Analyzer accepts transferred logs.

Network communications were tested by sending ICMP ping messages to each sensor. A response from the sensor indicates proper configuration to communication on the network.

To ensure the Traffic Generator was sending intended traffic, sample packets were generated using Nping, captured on the cyber sensor under test, and inspected using Wireshark. During testing, Wireshark’s packet inspection similarly ensured no data loss within the packet. Nping’s rate flag indicates the rate in which to send packets, measured in packets per second. It was noted that there can be differences between the set rate and the actual rate the traffic is generated. This discrepancy can depend on many variables such as packet-size, type, or interface speed. Due to these variables the tool had to be calibrated for each sensor in order to achieve the value closest to the desired rate (1, 10, 100, 1000). This was accomplished by incrementing Nping’s rate flag until the actual rate converged on the desired rate. The calibration resulted in a range of packets per seconds for each rate as shown in Table 4.1. Note that during the test cases, the average packet per second never exceeded the bounds of the calibrated ranges.

<b>Rate Desired (pps)</b>	<b>Actual Rate (pps)</b>
Base (1)	1
Low (10)	9-10
Medium (100)	70-85
High (1000)	730-770

Table 4.1: Calibrated rate ranges.

The traffic sent during the initialization check for the Traffic Generator was captured on each sensor, compressed, and sent to the Log Analyzer where it was tested for data loss using the checksum feature in the TCP protocol. The Log Analyzer received logs

transferred from each sensor successfully. Additionally, the number of logs sent by the cyber sensors was compared to the number of logs received by the Log Analyzer, ensuring 100% accuracy.

#### ***4.2.2 Identification Test Case.***

The following initialization and validation checks were added for the identification test case:

- The Traffic Generator sends non-standard traffic; and
- The Log Analyzer executes Snort to identify malicious events.

This was validated by sending one malicious event to one of the sensors, that was then logged and sent to the Log Analyzer where Wireshark was used to confirm its contents. Snort was operating properly on the Log Analyzer and capable of identifying malicious events. Snort was executed against the single event generated in the previous check to validate successful operation.

### **4.3 Results**

This section describes results of the functionality test and identification test for the cyber sensors.

#### ***4.3.1 Functionality Test Case.***

The main goal of this test was to evaluate the functionality and performance of each cyber sensor. It examined how the three sensors perform under varying traffic rates (base, low, medium, and high) to ensure they could log traffic, compress logs, and transfer compressed logs appropriately. The metrics associated with these goals are percent of packets logged, compressed, and transferred, as well as time to compress the logs files. A detailed list of all results and calculations is provided in Appendix A.

Table 4.2 shows the consolidated results for the laptop. Traffic was successfully logged during all rates except high, in which there was a slight loss of traffic logged. Captured

traffic was successfully compressed and transferred with 100% accuracy for all rates. Note that the longest compression time of 3.92 seconds was on a 20MB file captured during the high rate.

During traffic logging, it was observed that the base rate did not generate 2MB of data. On average, this resulted in bzip2 compressing a 320KB file instead. It is expected to see faster compression times for the base rate, due to a smaller original file size. Additionally, it was also observed that bzip2's compression ratio varied slightly. Compressing the 2MB files used in the low and medium rates resulted in file sizes ranging from 81KB to 82KB. It was expected to see similar compression times between the two rates. However, upon further investigation, it was noted that the low rate had an average file size of 81KB, whereas the medium rate had an average file size of 82KB. Due to the higher compression ratio used in the low rate, a longer compression time is expected.

<b>Rate</b>	<b>Logged</b>	<b>Compressed</b>	<b>Compression Time</b>	<b>Transferred</b>
Base (1)	100.00%	100.00%	0.09s	100.00%
Low (9-10)	100.00%	100.00%	0.40s	100.00%
Medium (70-85)	100.00%	100.00%	0.33s	100.00%
High (730-770)	99.93%	100.00%	3.92s	100.00%

Table 4.2: LAPT functionality results.

Table 4.3 shows the results for the Raspberry Pi. Both the base and low rates were logged with 100% accuracy; however, the cyber sensor's logging capability degraded as the rate increased. The longest compression time was 119 seconds for a 20MB file. Note that the base rate does not generate 2MB of traffic, instead the compression was done on logs with an average file size of 320KB. The sensor compressed and transferred all captured packets.

<b>Rate</b>	<b>Logged</b>	<b>Compressed</b>	<b>Compression Time</b>	<b>Transferred</b>
Base (1)	100.00%	100.00%	1.53s	100.00%
Low (9-10)	100.00%	100.00%	11.07s	100.00%
Medium (70-85)	99.61%	100.00%	11.57s	100.00%
High (730-770)	97.14%	100.00%	119.00s	100.00%

Table 4.3: Raspberry Pi functionality results.

Table 4.4 shows the results for the Gumstix Overo. At the base rate, the sensor logged, compressed, and transferred all packets. As the rate of packets increased, percentage of the logged traffic decreased. At the high rate of traffic generation, the sensor successfully logs 94.28%. For compression time, the high rate generated a 20MB file, resulting in a 158.27 second compression time. Once again, the base rate did not generate 2MB of traffic, and resulted in compressing a 320KB log file in 1.78 seconds. Because both the low and medium rates generated a 2MB log file, it was expected that their compression time would be similar. However, on average, the low rate of traffic compressed data with a higher ratio, resulting in an 80KB file, whereas the medium rate produced an 82KB file. Due to the higher compression ratio, it is expected to see longer delays for the low rate of traffic.

<b>Rate</b>	<b>Logged</b>	<b>Compressed</b>	<b>Compression Time</b>	<b>Transferred</b>
Base (1)	100.00%	100.00%	1.78s	100.00%
Low (9-10)	99.77%	100.00%	15.16s	100.00%
Medium (70-85)	96.36%	100.00%	13.95s	100.00%
High (730-770)	94.28%	97.37%	158.27s	100.00%

Table 4.4: Gumstix functionality results.

Although this test leverages a central log storage device, the Log Analyzer, the cyber sensors have the potential to store log files for an extended period of time if the back channel is not available or if the sensor is used primarily for forensic analysis. Table 4.5 shows the



average size of the generated traffic that is stored on the sensors, ordered by rate of traffic. Note that the file sizes for all sensors is similar. For example, the laptop captured 245MB of traffic during the high rate of traffic for 1 hour, which was compressed to 8.92MB for local storage. Since the high rate of traffic generates the most amount of data, it is expected that the sensors are capable of logging over 4 days of traffic per every 1GB of available storage. The sensors ability to store log file for longer periods of time is dependent on the size of the non-volatile storage used (microSD or SD card), and have the potential to be expanded further via the use of USB hard drives. The implications of this observation are such that these sensors possess the capability to store large amounts over an extended period of time before being collected for analysis.

<b>Sensor</b>	<b>Rate</b>	<b>Raw File Size</b>	<b>Compressed File Size</b>
Laptop	Base (1)	320KB	16KB
	Low (9-10)	2.86MB	137KB
	Medium (70-85)	25MB	1.04MB
	High (730-770)	245MB	8.92MB
Raspberry Pi	Base (1)	320KB	16KB
	Low (9-10)	2.86MB	137KB
	Medium (70-85)	22.5MB	1.04MB
	High (730-770)	241MB	8.92MB
Gumstix	Base (1)	320KB	16KB
	Low (9-10)	2.85MB	137KB
	Medium (70-85)	22.4MB	1.03MB
	High (730-770)	230MB	8.91MB

Table 4.5: Total log sizes based on rate.

#### ***4.3.2 Identification Test Case.***

The main goal of this test was to evaluate the ability to capture and identify non-standard traffic, referred to as malicious events. A captured event is one that is logged on the sensor, compressed, and transferred to the Log Analyzer with no loss of data. A malicious event is successfully identified when the Log Analyzer filters and identifies non-

standard traffic by using Snort with custom signatures. The metrics associated with this goal are the percentage of events logged and percentage of events identified.

Table 4.6 shows malicious events for the laptop. In each instance, 100% of events were logged. This indicates that the events were captured on the laptop, compressed, and transferred to the Log Analyzer without loss of data. 100% of events were accurately identified by the Log Analyzer signatures.

<b>Rate</b>	<b>Logged</b>	<b>Identified</b>
Low (5)	100.00%	100.00%
Medium(10)	100.00%	100.00%
High (20)	100.00%	100.00%

Table 4.6: LAPT identification results.

Table 4.7 shows results for the Raspberry Pi. 100% of malicious events were logged and identified.

<b>Rate</b>	<b>Logged</b>	<b>Identified</b>
Low (5)	100.00%	100.00%
Medium (10)	100.00%	100.00%
High (20)	100.00%	100.00%

Table 4.7: Raspberry Pi identification results.

Table 4.8 shows results for the Gumstix. The sensor logged 100% of malicious events for low and medium rates. At the high rate 98.33% of events were logged. As noted during the packet capturing process from the functionality test, it is expected to see some event loss during the high as well. The cyber sensor did, however, successfully identify all events that were captured.

<b>Rate</b>	<b>Logged</b>	<b>Identified</b>
Low (5)	100.00%	100.00%
Medium (10)	100.00%	100.00%
High (20)	98.33%	100.00%

Table 4.8: Gumstix identification results.

### 4.3.3 Examination.

Due to the varying nature of ICS networks, it is important that the all sensors perform effectively under different workloads and environments. Figure 4.1 compares the percentage of packet loss between the three cyber sensors. Note that the 95% confidence interval generated for each value is too small to be accurately represented in the figure. The laptop performed consistently with little packet loss during all rates, the Raspberry Pi started showing an increase of packet loss at the high rate, whereas the Gumstix started showing a larger increase of packet loss at the medium rate. The Raspberry Pi outperformed the Gumstix sensor by 2.86% at the high rate. This may become a factor in some environments, but in either situation the sensors performed at or above 94.28% under the highest rate of traffic tested. Note that for this research a cyber sensor is only considered effective if it performs within 5% of the baseline. For packets logged, the Gumstix sensor is not considered effective at the high rate of traffic.

Although not optimal, the findings demonstrate an ability to successfully log traffic and provide a capability that currently is void in ICS security. Additionally if these sensors are deployed on an ICS network with low to medium traffic, they would capture traffic with little to no data loss.

Figure 4.2 compares the compression time between the three cyber sensors. On all sensors, the base rate generated a 320KB file, low and medium produced a 2MB file, whereas high generated a 20MB file. During the base, low, and medium rates both sensors

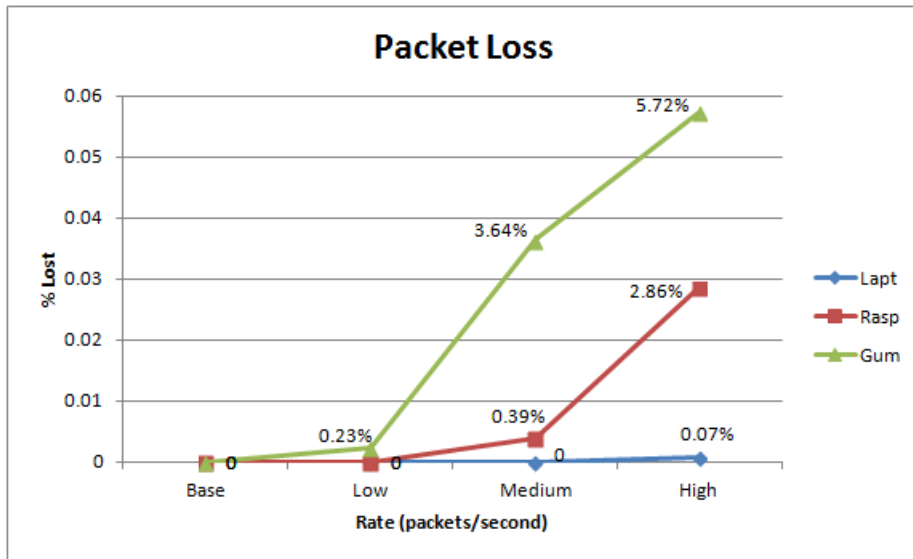


Figure 4.1: Percentage of packet loss by cyber sensor.

performed similarly. However, during the high rate, larger differences between the sensors were observed, which is indicative of the sensor's limited resources. On average, it takes 119 seconds to compress a 20MB log file on the Raspberry Pi, and 158.27 seconds on the Gumstix sensor. Note that when compared to the baseline laptop, both the Raspberry Pi and Gumstix sensors do not perform within 5% on any rate of traffic, and therefore are not considered effective for compression time.

Upon further investigation it was discovered that the Gumstix sensor had a higher rate of compression than both the Raspberry Pi and standard laptop. Table 4.9 shows the compression ratio measured by percent of the original file size. Note that because the Gumstix compresses at a higher rate it takes longer for the process to execute. If these sensors are used for logging and forensic analysis, then the compression time is negligible and a 5% loss may be acceptable. However, if the sensor is used to react to an attack in real-time on a network equivalent to the high rate (e.g., 700 packets per second), the analyst may have over a two-and-a-half minute delay to understand and react accordingly.

Note that since the Raspberry Pi is restricted to use a WiFi dongle as its second network communication interface, it is limited to wireless communication speeds, in this case 150 MBPS. The result could introduce an additional time delay and may be a consideration when deploying it particular environments.

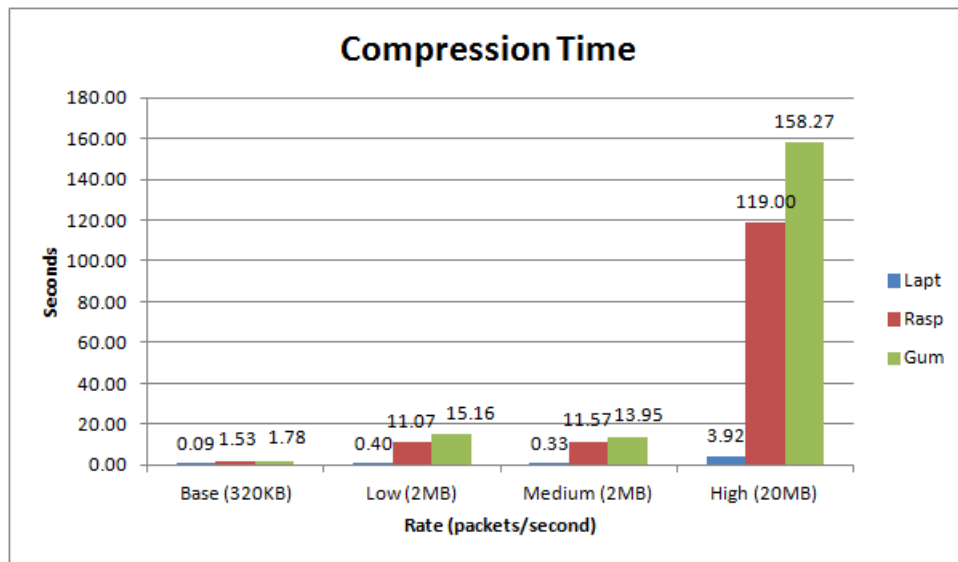


Figure 4.2: Compression times by sensor.

Sensor	320KB	2MB	20MB
Laptop	5.2%	4%	3.4%
Raspberry Pi	5.2%	4%	3.4%
Gumstix	5.2%	3.9%	3.3%

Table 4.9: Compression ratio for each sensor measured by percent of original file size.

Because of the noted loss during the packet capturing process, it was expected to see some loss during the event capturing process as well. This is evident in the high rate of events for the second test. It is important to note that so long as the event was captured it was always identified. This means that the events that were captured by the sensor,

compressed, and transferred to the Log Analyzer were identified 100%. For all sensors, Snort was able to identify all of the events that were transferred to it. All three sensors captured 100% of events except for the Gumstix, which lost one event during the high rate. This indicates that the most important function of the sensor is the ability to accurately log traffic.

All of these factors should be taken into account when assessing if either of the cyber sensors will perform sufficiently, as the success of the sensor depends on the environment in which it is deployed.

#### **4.4 Degradation Observations**

During the functionality test, it was observed that in the worse case scenario across both sensors, only a 6% degradation occurred at the highest rate of traffic. Asset owners may be curious to know the threshold of the sensors, and where they begin to fall below an acceptable performance. After the functionality and identification tests were completed, a pilot evaluation was conducted to examine the impact of traffic rates beyond those tested. This was accomplished by adding multiple Traffic Generators to increase the rate of traffic, and recording the logging performance of each sensor.

Figure 4.3 shows the degradation trend between the Raspberry Pi and Gumstix sensors. One interesting observation is that the Gumstix sensor starts out-performing the Raspberry Pi at around 1,000 packets per second. As demonstrated in the functionality test, the opposite was observed during traffic rates below 800 packets per second. This indicates that as the rate of traffic increases, the Gumstix sensor may fail more gracefully than the Raspberry Pi. The Raspberry Pi starts performing below 50% at approximately 2,200 packets per second, whereas the Gumstix falls below 50% until at approximately 2,800 packets per second.

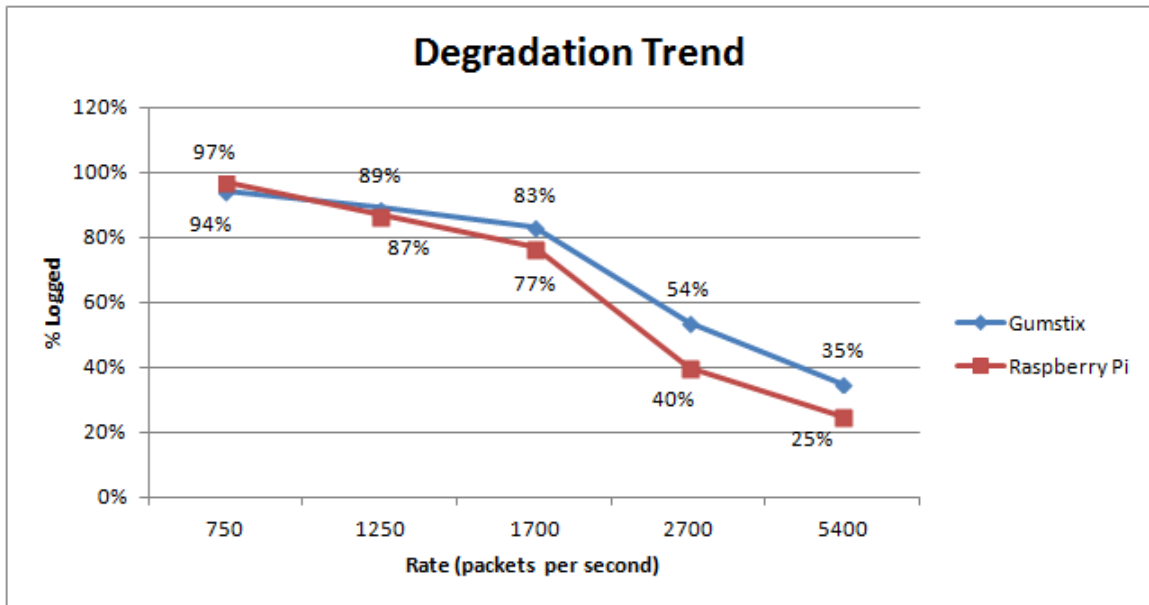


Figure 4.3: Degradation trend.

#### 4.5 Results Summary

Both the Gumstix and Raspberry Pi cyber sensors were successful in capturing traffic with slight performance degradation as the rate of traffic was increased. They were also successful in identifying the malicious event used in this research with 100% accuracy. Both sensors are considered effective at successfully compressing and transferring log files. The Raspberry Pi is considered effective at logging traffic, whereas the Gumstix sensor failed to be effective during the high rate of traffic. Both, the Raspberry Pi and Gumstix sensors failed to be considered effective at compression times when compared to the baseline laptop. As the results indicate, the Raspberry Pi out-performed the Gumstix sensor in all tests. The performance of the Raspberry Pi along with its lower cost indicates that this sensor is the optimal choice when deployed on ICS networks that has a low to medium amount of network traffic.

The overall results are promising for deploying a sensor that captures traffic and forwards it to a centralized system for analysis. The findings also demonstrate a

requirement to optimize the processing at a higher rate of traffic to reduce the number of packets that are not logged.



## V. Conclusions

This research introduced a robust logging capability that utilizes small, inexpensive sensors. The highly distributive nature and low cost of these sensors offer situational awareness for ICS environments where it is currently lacking. Indeed, the sensors afford the ability to capture and analyze traffic for suspicious activity. Employment of sensors into production environments helps identify and subsequently categorize patterns associated with malicious behavior. This data can be used to gain insight into attacker tactics (e.g., port scans, vulnerability scans, and zero-day exploits) and characteristics (e.g. attack origin, frequency, and level of persistence), as well as provide early warning capabilities needed to thwart future attacks.

This research tested the functionality and identification capability of the Raspberry Pi and Gumstix sensors. The Raspberry Pi was successful in capturing 100% of the traffic on the base and low traffic rates, with marginal performance degradation as the traffic rate increased to high. The Gumstix sensor was successful at capturing 100% of the traffic on the lowest rate of traffic, and slightly degraded in performance as the traffic rate increased. It is important to note that other factors, such as compression and transfer time, may become bottlenecks when the sensors are used for real-time analysis. The findings indicate that the Gumstix sensor will work best in ICS networks with low rates of traffic, and that further optimization is needed for high rate systems. However, the Raspberry Pi out-performed the Gumstix sensor in packet logging, log compression accuracy, and compression time. It is best suited for networks with low to medium traffic rates and, based on cost effectiveness, is the optimal sensor for most environments. Note that the transmission rate of the Raspberry Pi sensor for back channel communication is limited to wireless speeds, and may have additional delays in certain environments. Both the Raspberry Pi and Gumstix sensors successfully identified 100% of malicious events.

During the high traffic rate, for the Gumstix, the logging capability was not within 5% of the baseline laptop and according to the prerequisite goals of this research it is not considered effective. Additionally, due to the sensors' limited resources, the compression time for both sensors is not considered effective, however, there is a potential for optimization in future work. Note that based on the organizational requirements and environment, anything over 5% may be an acceptable loss.

In summary, both the Raspberry Pi and Gumstix sensors can perform in most ICS networks with little to no data loss. The sensor's ability to be distributed at any layer of ICS networks to gather data on network attacks can provide the necessary foundation to increase situational awareness in the ICS domain.

## **5.1 Applications**

Some applications of these sensors can be put to use immediately without further optimization. These applications include deploying with ICS-CERT incident response teams, gathering network performance metrics, and building a state-based operational picture of a ICS network.

### ***5.1.1 ICS-CERT Incident Response.***

When an ICS-CERT incident response team is contacted by ICS facilities alerting to a possible compromise, their first action is typically to send a responder out to assess the situation. In essence, the responder collects data on the ICS environment and sends it back to ICS-CERT for further forensic analysis. Instead of sending a responder to assess the state of the ICS environment, cyber sensors can be shipped overnight and connected throughout the network to collect traffic remotely to send logs back for further analysis. The sensors can then be left on the network to continue gathering data and send back to ICS-CERT for ongoing analysis.

### ***5.1.2 Network Performance Metrics.***

Another application of the sensors can be to gather network performance metrics on ICS networks. The network traffic can be collected and analyzed to answer questions such as:

- Are devices performing as expected?
- What is the time delay between two points in the network?
- Is device traffic in compliance with regulatory guidelines?
- What time of day is the network utilized the most?
- Where are the most active parts of the network, and how can they be optimized to ensure stability?

### ***5.1.3 Operation Picture.***

Asset owners can deploy cyber sensors into their ICS networks to gather data and build a standard traffic history. The standard traffic history can be used to develop a ‘safe state’ composed of healthy network activity. If an incident were to occur, new data can be compared to the safe state in order to understand when and how the network transitioned into an unsafe state. Additionally, the sensors are already in place before an attack occurs, providing incident responders the pertinent information they need to resolve the issue.

## **5.2 Future Work**

Future work includes the development of additional attack signatures for ICS, optimizing the sensors, deployment in operational environments, and providing additional capabilities.

### ***5.2.1 Snort Signatures.***

Follow on work for the cyber sensors includes developing more attack signatures for ICS used by Snort to detect malicious activity. These signatures should encompass multiple

protocols and should be narrowly tailored to ICS environments. In addition to attack signatures, another area of research is expanding white-listing capabilities for predictable ICS traffic. Due to the predictable nature of ICS traffic, a white-listing approach can be used to identify any traffic that does not conform to expected patterns. Developing a solution that employs both attack signatures and white-list signatures will improve event detection for known and unknown attacks.

### ***5.2.2 Optimization.***

During this research, there was data loss during heavy traffic loads. Optimizing the cyber sensors' performance may be necessary before employing them in ICS environments with high rates of traffic. Some areas that show potential for optimization are: the logging algorithm, tcpdump, bzip2, and the operating system. During the evaluation, a pilot study was performed to examine the degradation trend of each sensor. An in-depth analysis of the failure rate of these sensors should be conducted to determine how they perform under extreme rates of traffic such as denial-of-service attacks.

### ***5.2.3 Deployment in Operational Environment.***

Implementing cyber sensors into an operational environment may introduce additional restrictions that affect the sensor's ability to log, compress, and transfer traffic adequately. Employing the cyber sensors in an operational ICS environment will further evaluate their capabilities. Additionally, determining the most effective placement of the sensors throughout an ICS network should be evaluated.

### ***5.2.4 Additional Capabilities.***

Follow on work for the cyber sensors includes expanding their initial capabilities beyond network traffic logging, such as firewalls and log repositories. Firewalls can be installed on the sensors to further regulate traffic between ICS end devices. Additionally, the sensors can be configured to be cost effective log servers that store various log files, such as event logs, systems logs, and application logs.

### **5.3 Concluding Remarks**

Situational awareness is an inherent problem in the ICS domain. This research demonstrated a method to employ inexpensive cyber sensors into ICS environments that will provide operators and asset owners more detailed analysis. As further research is conducted and adopted among the control systems community, the findings will enable security professionals to draw accurate awareness of the threats against ICS and help shift the security posture from reactionary to preventative.

## **Appendix A: Test Data**

Test 1: Functionality

Parameters		Frequency	Type	Service	Platform	Metrics															
Levels	Run Order	Low, Med, High	Standard	Web	LAPT, Raspberry Pi, Gumstix	Logged					Accuracy										
						Sent	Received	Lost	Rate	% Logged	Possible	Actual	Lost	% Compressed	Timing (s)	Sent	Received	Lost	% Transferred	Timing (s)	
Run 1	1	8	Base	Standard	Web	LAPT	3588	3588	0	0.998	100.00%	1	1	0	100.00%	0.1	1	1	0	100.00%	5.33
	2	9	Low	Standard	Web	LAPT	32962	32962	0	9.16	100.00%	2	2	0	100.00%	0.40	2	2	0	100.00%	5.40
	3	1	Medium	Standard	Web	LAPT	289074	289074	0	80.3	100.00%	14	14	0	100.00%	0.34	14	14	0	100.00%	5.34
	4	4	High	Standard	Web	LAPT	2281910	2281091	819	633.63	99.96%	13	13	0	100.00%	3.93	13	13	0	100.00%	5.42
	5	5	Base	Standard	Web	Raspberry Pi	3590	3590	0	0.997	100.00%	1	1	0	100.00%	1.54	1	1	0	100.00%	5.89
	6	6	Low	Standard	Web	Raspberry Pi	32985	32985	0	9.16	100.00%	2	2	0	100.00%	13.18	2	2	0	100.00%	6.20
	7	10	Medium	Standard	Web	Raspberry Pi	269158	268057	1101	74.45	99.59%	10	10	0	100.00%	11.57	10	10	0	100.00%	7.00
	8	2	High	Standard	Web	Raspberry Pi	2885320	2809215	76105	780.28	97.36%	13	13	0	100.00%	119.47	13	13	0	100.00%	7.69
	9	7	Base	Standard	Web	Gumstix	3591	3591	0	0.998	100.00%	1	1	0	100.00%	1.77	1	1	0	100.00%	5.53
	10	11	Low	Standard	Web	Gumstix	32976	32911	65	9.14	99.80%	2	2	0	100.00%	14.24	2	2	0	100.00%	5.55
	11	3	Medium	Standard	Web	Gumstix	269378	259713	9665	72.14	96.41%	12	12	0	100.00%	16.14	12	12	0	100.00%	5.54
	12	12	High	Standard	Web	Gumstix	2880308	2722593	157715	756.23	94.52%	12	11	1	91.67%	152.36	11	11	0	100.00%	5.68
Run 2	1	2	Base	Standard	Web	LAPT	3592	3592	0	0.998	100.00%	1	1	0	100.00%	0.08	1	1	0	100.00%	5.26
	2	6	Low	Standard	Web	LAPT	32960	32960	0	9.16	100.00%	2	2	0	100.00%	0.41	2	2	0	100.00%	5.32
	3	5	Medium	Standard	Web	LAPT	287599	287599	0	79.89	100.00%	14	14	0	100.00%	0.33	14	14	0	100.00%	5.33
	4	1	High	Standard	Web	LAPT	2839703	2837578	2125	788.22	99.93%	13	13	0	100.00%	3.95	13	13	0	100.00%	5.39
	5	9	Base	Standard	Web	Raspberry Pi	3589	3589	0	0.997	100.00%	1	1	0	100.00%	1.52	1	1	0	100.00%	6.31
	6	3	Low	Standard	Web	Raspberry Pi	32985	32985	0	9.16	100.00%	2	2	0	100.00%	10.03	2	2	0	100.00%	5.92
	7	12	Medium	Standard	Web	Raspberry Pi	257664	256883	781	71.35	99.70%	12	12	0	100.00%	11.58	12	12	0	100.00%	6.04
	8	10	High	Standard	Web	Raspberry Pi	2856934	2774770	82164	770.7	97.12%	13	13	0	100.00%	118.85	13	13	0	100.00%	6.75
	9	4	Base	Standard	Web	Gumstix	3590	3590	0	0.997	100.00%	1	1	0	100.00%	1.79	1	1	0	100.00%	5.54
	10	8	Low	Standard	Web	Gumstix	32982	32894	88	9.13	99.73%	2	2	0	100.00%	14.24	2	2	0	100.00%	5.55
	11	11	Medium	Standard	Web	Gumstix	260238	251746	8492	70	96.74%	12	12	0	100.00%	12.84	12	12	0	100.00%	5.54
	12	7	High	Standard	Web	Gumstix	2825782	2664948	160834	740.15	94.31%	13	13	0	100.00%	154.92	13	13	0	100.00%	5.73
Run 3	1	9	Base	Standard	Web	LAPT	3593	3593	0	0.998	100.00%	1	1	0	100.00%	0.09	1	1	0	100.00%	5.30
	2	6	Low	Standard	Web	LAPT	32983	32983	0	9.16	100.00%	2	2	0	100.00%	0.39	2	2	0	100.00%	5.35
	3	11	Medium	Standard	Web	LAPT	289426	289426	0	80.39	100.00%	14	14	0	100.00%	0.33	14	14	0	100.00%	5.28
	4	4	High	Standard	Web	LAPT	2804118	2801896	2222	778.3	99.92%	13	13	0	100.00%	3.89	13	13	0	100.00%	5.40
	5	10	Base	Standard	Web	Raspberry Pi	3590	3590	0	0.997	100.00%	1	1	0	100.00%	1.52	1	1	0	100.00%	5.90
	6	2	Low	Standard	Web	Raspberry Pi	32980	32980	0	9.16	100.00%	2	2	0	100.00%	10.00	2	2	0	100.00%	6.00
	7	12	Medium	Standard	Web	Raspberry Pi	257109	255972	1137	71	99.56%	12	12	0	100.00%	11.56	12	12	0	100.00%	5.90
	8	1	High	Standard	Web	Raspberry Pi	2813396	2726859	86537	757.4	96.92%	13	13	0	100.00%	118.68	13	13	0	100.00%	6.90
	9	7	Base	Standard	Web	Gumstix	3590	3590	0	0.997	100.00%	1	1	0	100.00%	1.79	1	1	0	100.00%	5.54
	10	8	Low	Standard	Web	Gumstix	32975	32902	73	9.14	99.78%	2	2	0	100.00%	17.01	2	2	0	100.00%	5.57
	11	5	Medium	Standard	Web	Gumstix	267887	257040	10847	71.4	95.95%	12	12	0	100.00%	12.86	12	12	0	100.00%	5.54
	12	3	High	Standard	Web	Gumstix	2833559	2663997	169562	741.47	94.02%	13	13	0	100.00%	167.54	13	13	0	100.00%	5.75
<b>Averages</b>																					
1		1	Standard	Web	LAPT	3591.00	3591.00	0.00	1.00	100.00%	1.00	1.00	0	100.00%	0.09	1.00	1.00	0	100.00%	5.30	
2		9-10	Standard	Web	LAPT	32968.33	32968.33	0.00	9.16	100.00%	2.00	2.00	0	100.00%	0.40	2.00	2.00	0	100.00%	5.36	
3		70-85	Standard	Web	LAPT	288699.67	288699.67	0.00	80.19	100.00%	14.00	14.00	0	100.00%	0.33	14.00	14.00	0	100.00%	5.32	
4		730-770	Standard	Web	LAPT	2641910.33	2640188.33	1722.00	733.38	99.93%	13.00	13.00	0	100.00%	3.92	13.00	13.00	0	100.00%	5.40	
5		1	Standard	Web	Raspberry Pi	3589.67	3589.67	0.00	1.00	100.00%	1.00	1.00	0	100.00%	1.53	1.00	1.00	0	100.00%	6.03	
6		9-10	Standard	Web	Raspberry Pi	32983.33	32983.33	0.00	9.16	100.00%	2.00	2.00	0	100.00%	11.07	2.00	2.00	0	100.00%	6.04	
7		70-85	Standard	Web	Raspberry Pi	261310.33	260304.00	1006.33	72.27	99.61%	11.33	11.33	0	100.00%	11.57	11.33	11.33	0	100.00%	6.31	
8		730-770	Standard	Web	Raspberry Pi	2851883.33	2770281.33	81602.00	769.46	97.14%	13.00	13.00	0	100.00%	119.00	13.00	13.00	0	100.00%	7.11	
9		1	Standard	Web	Gumstix	3590.33	3590.33	0.00	1.00	100.00%	1.00	1.00	0	100.00%	1.78	1.00	1.00	0	100.00%	5.54	
10		9-10	Standard	Web	Gumstix	32977.67	32902.33	75.33	9.14	99.77%	2.00	2.00	0	100.00%	15.16	2.00	2.00	0	100.00%	5.56	
11		70-85	Standard	Web	Gumstix	265834.33	256166.33	9668.00	71.18	96.36%	12.00	12.00	0	100.00%	13.95	12.00	12.00	0	100.00%	5.54	
12		730-770	Standard	Web	Gumstix	2846549.67	2683846.00	162703.67	745.95	94.28%	12.67	12.33	0.33	97.37%	158.27	12.33	12.33	0	100.00%	5.72	

Figure A.1: Functionality Test Full Results

Test 2: Traffic Identification												
Parameters		Frequency	Type	Service	Platform	Metrics						
Levels	Run Order	High, Med, Low	Non-Standard	Web	PC, Raspberry Pi, Gumstix	Logging Accuracy				Identification Accuracy		
						Sent	Logged	Lost	% Logged	Identified	% Identified	
Run 1	1	3	Low	Non-Standard	Web	LAPT	5	5	0	100.00%	5	100.00%
	2	4	Medium	Non-Standard	Web	LAPT	10	10	0	100.00%	10	100.00%
	3	7	High	Non-Standard	Web	LAPT	20	20	0	100.00%	20	100.00%
	4	1	Low	Non-Standard	Web	Raspberry Pi	5	5	0	100.00%	5	100.00%
	5	6	Medium	Non-Standard	Web	Raspberry Pi	10	10	0	100.00%	10	100.00%
	6	9	High	Non-Standard	Web	Raspberry Pi	20	20	0	100.00%	20	100.00%
	7	8	Low	Non-Standard	Web	Gumstix	5	5	0	100.00%	5	100.00%
	8	2	Medium	Non-Standard	Web	Gumstix	10	10	0	100.00%	10	100.00%
	9	5	High	Non-Standard	Web	Gumstix	20	20	0	100.00%	20	100.00%
Run 2	10	8	Low	Non-Standard	Web	LAPT	5	5	0	100.00%	5	100.00%
	11	3	Medium	Non-Standard	Web	LAPT	10	10	0	100.00%	10	100.00%
	12	7	High	Non-Standard	Web	LAPT	20	20	0	100.00%	20	100.00%
	13	5	Low	Non-Standard	Web	Raspberry Pi	5	5	0	100.00%	5	100.00%
	14	6	Medium	Non-Standard	Web	Raspberry Pi	10	10	0	100.00%	10	100.00%
	15	1	High	Non-Standard	Web	Raspberry Pi	20	20	0	100.00%	20	100.00%
	16	4	Low	Non-Standard	Web	Gumstix	5	5	0	100.00%	5	100.00%
	17	2	Medium	Non-Standard	Web	Gumstix	10	10	0	100.00%	10	100.00%
	18	9	High	Non-Standard	Web	Gumstix	20	19	1	95.00%	19	100.00%
Run 3	19	1	Low	Non-Standard	Web	LAPT	5	5	0	100.00%	5	100.00%
	20	7	Medium	Non-Standard	Web	LAPT	10	10	0	100.00%	10	100.00%
	21	5	High	Non-Standard	Web	LAPT	20	20	0	100.00%	20	100.00%
	22	6	Low	Non-Standard	Web	Raspberry Pi	5	5	0	100.00%	5	100.00%
	23	3	Medium	Non-Standard	Web	Raspberry Pi	10	10	0	100.00%	10	100.00%
	24	8	High	Non-Standard	Web	Raspberry Pi	20	20	0	100.00%	20	100.00%
	25	4	Low	Non-Standard	Web	Gumstix	5	5	0	100.00%	5	100.00%
	26	2	Medium	Non-Standard	Web	Gumstix	10	10	0	100.00%	10	100.00%
	27	9	High	Non-Standard	Web	Gumstix	20	20	0	100.00%	20	100.00%
Averages	<b>Averages</b>											
	1		5	Non-Standard	Web	LAPT	5	5	0	100.00%	5	100.00%
	2		10	Non-Standard	Web	LAPT	10	10	0	100.00%	10	100.00%
	3		20	Non-Standard	Web	LAPT	20	20	0	100.00%	20	100.00%
	4		5	Non-Standard	Web	Raspberry Pi	5	5	0	100.00%	5	100.00%
	5		10	Non-Standard	Web	Raspberry Pi	10	10	0	100.00%	10	100.00%
	6		20	Non-Standard	Web	Raspberry Pi	20	20	0	100.00%	20	100.00%
	7		5	Non-Standard	Web	Gumstix	5	5	0	100.00%	5	100.00%
	8		10	Non-Standard	Web	Gumstix	10	10	0	100.00%	10	100.00%
9		20	Non-Standard	Web	Gumstix	20	19.67	0.33	98.33%	19.67	100.00%	

Figure A.2: Identity Test Full Results



## Appendix B: Scripts and Code

### B.1 Traffic Generator

#### nping.sh

This script executes Nping to generate traffic at a desired rate.

---

```
1 sudo nping -tcp --data-length 21 -c $1 --rate $2 $3 &
2 sleep 1h;
3 sudo killall -2 nping;
```

---

#### replay.sh

This script sends the replay attack payload to the device.

---

```
1 #!/bin/bash
2 c=1
3 freq=$2
4 while [ $c -le $freq ]
5 do
6 if [ $freq == 5 ]; then
7 t=10
8 fi
9 if [ $freq == 10 ]; then
10 t=5
11 fi
12 if [ $freq == 20 ]; then
13 t=2
14 fi
15 echo "Run: _$c"
16 sudo nc -n -i 5 $1 80 -c 'cat payload' | grep HTTP
17 ((c++))
18 sleep $t"m";
19 done
```

---

#### payload

This is the payload used in the replay attack.

---

```
1 GET /svr_set.htm HTTP/1.1
2
3 Host: 192.168.0.102
4
```

```
5 Proxy-Connection: keep-alive
6
7 Authorization: Digest username="administrator", realm="1766-
  L32AWA_B/11.00", nonce="
  a4b8c8d7e0f6a7b2c3d2e4f5a4b7c5d2e7f", uri="/svr_set.htm",
  algorithm=MD5, response="4d0f097f3dbbf6bcaa5f9841a3019dfa"
  , qop=auth, nc=00000001, cnonce="b963861c153b5233"
8
9 Accept: text/html, application/xhtml+xml, application/xml;q
  =0.9, */*;q=0.8
10
11 User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit
  /537.17 (KHTML, like Gecko) Chrome/24.0.1312.57 Safari
  /537.17
12
13 Referer: http://192.168.0.100/navtree.htm
14
15 Accept-Encoding: gzip, deflate, sdch
16
17 Accept-Language: en-US, en;q=0.8
```

---

## Nping Result

This is an example of the output of Nping after execution.

---

```
1 Max rtt: 66.310ms | Min rtt: 0.017ms | Avg rtt: 0.017ms
2 Raw packets sent: 2885320 (176.005MB) | Rcvd: 2885108
  (132.715MB) | Lost: 212 (0.01%)
3 Tx time: 3600.94005s | Tx bytes/s: 0.00 | Tx pkts/s: 0.00
4 Rx time: 3601.94005s | Rx bytes/s: 0.00 | Rx pkts/s: 0.00
5 Nping done: 1 IP address pinged in 3600.94 seconds
```

---

## B.2 Log Analyzer

### unzipNmerge.sh

This script unzips log files and merges them.

---

```
1 bunzip2 $1*
2 mergcap $1* -w $2
3 bzip2 $1*
```

---

## B.3 Snort Signatures

### runSnort.sh

This script runs Snort to identify malicious events.

---

```
1 sudo snort -r $1 -A full -c ~/snort/local.rules -l ~/snort/
  logs/
```

---

### local.rules

This is the black-list signature file used to identify the replay attack.

---

```
1 alert tcp any any -> any 80 (msg:"Admin_Login"; content:"
  administrator"; sid:1000111; nocase)
```

---

### white.rules

This is the white-list signature file used to identify the replay attack.

---

```
1 alert tcp any any -> any 80 (msg:"Non-standard_Traffic";
  window:!1480; flags:!R; sid=1000119)
```

---

### alertCount.sh

This script was used to count malicious events after Snort filtered them out.

---

```
1 cat alert_5" | grep Login | wc -l
2 cat alert_10" | grep Login | wc -l
3 cat alert_20" | grep Login | wc -l
```

---

## B.4 Gumstix, Raspberry Pi, and Standard Laptop

### tcpdump.sh

This script executes tcpdump on the sensor.

---

```
1 sudo tcpdump port 80 -nnXvvSs 0 -C $4 -w $1 -z ./zipNship.sh
  -i eth0 and src $2 and dst $3 &
2 sudo sleep 62m;
3 sudo killall -2 tcpdump;
```

---

### zipNship.sh

This script compresses and sends the log files to a centralized server.

---

```
1  #!/bin/sh
2
3  bzip2 $1
4
5  if [ -f $1.bz2 ]
6  then
7    scp -i /home/pi/.ssh/id_rsa $1.bz2 user@$2:/home/user/
      tcplogs
8    STATUS=$?
9  else
10   echo "zip_file_does_not_exist"
11   exit
12 fi
13 if [ $STATUS -eq 0 ]
14 then
15   rm -f "$1.bz2"
16 else
17   echo "scp_failed"
18 fi
```

---

### **nc-listen.sh**

This script starts netcat as a listener on each sensor.

---

```
1  while true; do sudo nc -lvp 80; done
```

---

### **Tcpdump Output**

This is an example output of tcpdump after execution.

---

```
1  2281091 packets captured
2  2281911 packets received by filter
3  820 packets dropped by kernel
```

---

## **Appendix C: Configurations**

### **C.1 Experiment Randomization**

These steps were taken to randomize the tests conducted in this research. Randomizing the test ensures statistical significance from equipment anomalies.

1. Open Excel
2. Number 1-12 in one column
3. Use the Rand() function to generate random numbers in the second column, ensure each number (1-12) has an associated random number
4. Order the two columns by the second column (random number) and use the first column as randomized run order.
5. Repeat for Identity Test. See Table C.1 for the Functionality Test and Table C.2 for the Identification Test.

Once the randomization is complete, the randomized integers (1-12) are used as the run order show in Appendix A. For example, the first test during run 1 of the functionality test case is for the laptop using a medium rate of traffic.

Run1		Run2		Run3	
8	0.033961	2	0.019748	9	0.072526
9	0.156853	6	0.067562	6	0.151557
1	0.202113	5	0.069656	11	0.271843
4	0.25293	1	0.092817	4	0.284646
5	0.332345	9	0.471312	10	0.324353
6	0.3596	3	0.603972	2	0.365958
10	0.379342	12	0.668211	12	0.491657
2	0.391605	10	0.732998	1	0.534157
7	0.419383	4	0.745249	7	0.733333
11	0.5819	8	0.746738	8	0.865366
3	0.847273	11	0.771163	5	0.950034
12	0.861974	7	0.777119	3	0.963456

Table C.1: Randomized table for Functionality Test

Run1		Run2		Run3	
3	0.398512	8	0.180933	1	0.038384
4	0.414042	3	0.259183	7	0.180268
7	0.632219	7	0.282256	5	0.315046
1	0.029468	5	0.312254	6	0.33644
6	0.590719	6	0.344602	3	0.337531
9	0.92956	1	0.349315	8	0.501056
8	0.663341	4	0.39797	4	0.525709
2	0.061786	2	0.540214	2	0.680244
5	0.521138	9	0.635688	9	0.923471

Table C.2: Randomized table for Identification Test

## C.2 Gumstix Setup

### I. BUILDING OVERO OPEN EMBEDDED IMAGE

The following directions are an excerpt from the guide (<http://gumstix.org/software-development/open-embedded/61-using-the-open-embedded-build-system.html>) [1]:

1. Build a new machine with the Ubuntu 10.10 x86 ISO file to act as the development laptop.
  - (a) <http://releases.ubuntu.com/10.10/ubuntu-10.10-desktop-i386.iso>
2. Once booted, use the Update Manager to update the default packages. Do not upgrade to Ubuntu 11.04 or other versions.
3. Open the synaptic package manager and select the following packages for install:
  - (a) git
  - (b) subversion
  - (c) gcc
  - (d) build-essential
  - (e) help2man
  - (f) diffstat
  - (g) texi2html
  - (h) texinfo
  - (i) libncurses5-dev
  - (j) cvs
  - (k) gawk
  - (l) python2.7-dev



(m) python-pysqlite2

(n) unzip

(o) chrpath

(p) ccache

4. sudo dpkg-reconfigure dash

(a) Answer **No** when asked whether you want to install dash as /bin/sh.

5. mkdir -p ~/overo-oe

6. cd ~/overo-oe

7. git clone git://gitorious.org/gumstix-oe/mainline.git org.openembedded.dev

8. cd org.openembedded.dev

9. git checkout --track -b overo-2011.03 origin/overo-2011.03

10. cd ~/overo-oe

11. git clone git://git.openembedded.org/bitbake bitbake

12. cd bitbake

13. git checkout 1.12.0

14. cd ~/overo-oe

15. cp -r org.openembedded.dev/contrib/gumstix/build .

16. cp ~/.bashrc ~/bashrc.bak

17. cat ~/overo-oe/build/profile >> ~/.bashrc

18. Close the Terminal window and open a new one.

19. `gedit ~/overo-oe/org.openembedded.dev/recipes/images/omap3-console-image.bb`
  - (a) Save and close the window
20. `bitbake omap3-console-image`
21. The Overo file system is built at: `~/overo-oe/tmp/deploy/glibc/images/overo/omap3-console-image-overo.tar.bz2`
22. The Overo OE Linux Kernel is built at: `~/overo-oe/tmp/deploy/glibc/images/overo/uImage-overo.bin`

## **II. PARTITIONING BOOTABLE SD CARD FOR OVERO IMAGE**

The next five sections are excerpts from the two guides, (<http://gumstix.org/create-a-bootable-microsd-card.html>) and (<http://gumstix.org/how-to/70-writing-images-to-flash.html>) [1]:

1. `df`
2. `umount /media/`
3. `umount /`

## **III. DEPLOYING OVERO IMAGE**

1. On the development laptop:
2. Delete the current file structure, if any, on the EXT3 partition of the micro SD card
  - (a) `sudo nautilus`
  - (b) Edit >Preferences >Behavior >Check Include a Delete command that bypasses Trash
  - (c) Select rootfs
  - (d) Select all files >Right Click >Delete

3. Copy the contents of `/overo-oe/tmp/deploy/glibc/images/overo/omap3-console-image-overo.tar.bz2` into the rootfs partition of the micro SD card.
4. On the micro SD card FAT partition:
  - (a) Delete uImage
  - (b) Copy `uImage-<kernel version>-overo.bin` into /
  - (c) Rename `uImage-<kernel version>-overo.bin` to uImage

#### **IV. BOOTING OVERO IMAGE CONSOLE**

1. Power off the Overo board.
2. Insert the newly created micro SD card into the micro SD slot of the Overo board.
3. Connect a USB cable between the Console mini USB B port on the Overo board and the development laptop with ckermit installed.
4. On the development laptop create a file called `overo_serial.cfg`

```
set line /dev/ttyUSB0 (Note: 0 might changed)
set flow-control none
set carrier-watch off
set speed 115200
set reliable
fast
set prefixing all
set file type bin
set rec pack 4096
set send pack 4096
set window 5
connect
```

5. Open a terminal and type:
  - (a) kermi
  - (b) take overo\_serial.cfg
6. Power on the Overo board. You should see the boot sequence displayed on the terminal. It will finish with a prompt to login.
7. Enter “root” as the username to log in.
8. To exit kermi:
  - (a) ctrl-/-c
  - (b) Type: exit

## **V. COMPILING TCPDUMP FOR OVERO IMAGE**

1. On the development laptop:
  - (a) bitbake tcpdump
2. Packages will be built in: /overo-oe/tmp/deploy/glibc/ipk/armv7a
3. Copy the packages onto the Overo EXT3 partition
  - (a) sudo scp ./tcpdump\_<version number>.ipk <overo IP address>:/home/root
4. On the Overo console, install the package
  - (a) opkg install ./tcpdump\_<version number>.ipk

## **VI. REMOVING UNWANTED PACKAGES**

1. update-rc.d f ntpd remove
2. update-rc.d f avahi-daemon remove

3. update-rc.d f portmap remove

## **VII. CONFIGURING THE OVERO BOARD TO WORK WITH THE TOBI DUO**

1. This is only needed if you have used the Tobi board to set up the Overo Board
2. Place the Overo Board on the Tobi Duo Expansion Board and power on the board.
3. Once the board has come online (Detected by the blue light on the CPU stops flashing) unplug the board and place the board back on the Tobi Expansion board.
4. Turn on the Overo
5. vi /etc/udev/rules/70-persistent-net.rules
6. There should now be three net device () lines in this file eth0 eth2.
7. Edit the eth1 line so that NAME=eth0
8. Edit the eth2 line so that NAME=eth1
9. If you restart the Overo with the Tobi-Duo extension you should now be able to SSH to 172.16.1.10.

### **C.3 Raspberry Pi Setup**

#### **I. DOWNLOAD THE RASPBERRY PI OPERATING SYSTEM**

The following are excerpts from the guide (Guide: <http://www.raspberrypi.org/quick-start-guide>) [26]:

1. The recommended OS is called Raspbian. Download it here:  
<http://downloads.raspberrypi.org/images/raspbian/2012-12-16-wheezy-raspbian/2012-12-16-wheezy-raspbian.zip>

## **II. UNZIP THE FILE**

1. Right click and choose “Extract All”
2. Follow the instructions – you should end up with a .img file

## **III. DOWNLOAD THE WIN32DISKIMAGER SOFTWARE**

1. Download win32diskimager-binary.zip (currently version 0.6) from:  
<https://launchpad.net/win32-image-writer/+download>
2. Unzip it in the same way you did the Raspbian .zip file
3. You now have a new folder called win32diskimager-binary  
You are now ready to write the Raspbian image to your SD card.

## **IV. WRITING RASPBIAN TO SD CARD**

1. Plug your SD card into your PC
2. In the folder you made in section III, run the file named Win32DiskImager.exe  
(in Windows Vista, 7 and 8 we recommend that you right-click this file and choose “Run as administrator”).
3. If the SD card (Device) you are using isnt found automatically then click on the drop down box and select it
4. In the Image File box, choose the Raspbian .img file that you downloaded
5. Click Write
6. After a few minutes you will have an SD card that you can use in your Raspberry Pi

## **V. BOOTING YOUR RASPBERRY PI FOR THE FIRST TIME**

1. On first boot you will come to the Raspi-config window

2. Change settings such as timezone and locale if you want
3. Finally, select the second choice: `expand_rootfs`, and say 'yes' to a reboot
4. The Raspberry Pi will reboot and you will see `raspberrypi` login:
  - (a) Type:`pi`
  - (b) You will be asked for your Password
  - (c) Type:`raspberry`
5. You will then see the prompt:`pi@raspberry ~$`
6. Start the desktop by typing:`startx`

## Bibliography

- [1] “Gumstix”. 2013. URL [www.gumstix.com](http://www.gumstix.com).
- [2] Abrams, Marshall and Joe Weiss. “Malicious Control System Cyber Security Attack Case Study Maroochy Water Services, Australia”. *McLean, VA: The MITRE Corporation*, 2008.
- [3] Bailey, David and Edwin Wright. *Practical SCADA for industry*. Newnes, 2003.
- [4] Berman, Dustin, Jonathan Butts, Barry Mullins, and Juan Lopez Jr. *Emulating Industrial Control System Field Devices Using Gumstix Technology*. Technical report, 2012.
- [5] Bond, Digital. “Quickdraw SCADA IDS”. 2011. URL <http://www.digitalbond.com/tools/quickdraw/>.
- [6] Bond, Digital. “SCADA Honeynet”. 2011. URL <http://www.digitalbond.com/tools/scada-honeynet/>.
- [7] Boyer, Stuart A. *SCADA: supervisory control and data acquisition*. International Society of Automation, 2009.
- [8] Chandia, Rodrigo, Jesus Gonzalez, Tim Kilpatrick, Mauricio Papa, and Sujeet Sheno. “Security Strategies for SCADA Networks”. Eric Goetz and Sujeet Sheno (editors), *Critical Infrastructure Protection*, volume 253 of *IFIP International Federation for Information Processing*, 117–131. 2007.
- [9] Cheah, Zi bin and Omar Faruk. “Identifying and Responding to External Threats in a PCS Network”. *Norwegian University of Science and Technology*, 2007.



- [10] Chirillo, John and Edgar Danielyan. *SUN Certified Security Administrator for Solaris 9 and 10 Study Guide*. McGraw-Hill, 2005.
- [11] Commission, USNR et al. “Potential Vulnerability of Plant Computer Network to Worm Infection”. *NRC Information Notice*, 14, 2003.
- [12] Electric, Schneider. “SCADAPack 350, 375”. 2011. URL [www.schneider-electric.com](http://www.schneider-electric.com).
- [13] Endsley, Mica R. “Toward a theory of situation awareness in dynamic systems”. *Human Factors: The Journal of the Human Factors and Ergonomics Society*, 37(1):32–64, 1995.
- [14] Falliere, Nicolas, Liam O. Murchu, and Eric Chien. “W32. stuxnet dossier”. *White paper, Symantec Corp., Security Response*, 2011.
- [15] Fergus, Donald J. “Industrial Control System Security Current Trends and Risk Mitigation”. *White Paper, Intekras Inc.*, 2009.
- [16] Fovino, I.N., A. Carcano, T. De Lacheze Murel, A. Trombetta, and M. Masera. “Modbus/DNP3 State-Based Intrusion Detection System”. *Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on*, 729–736. 2010.
- [17] Giacobbi, Giovanni. “The GNU netcat project”. *software available at <http://netcat.sourceforge.net>*, 2006.
- [18] of Homeland Security, Department. “Security in the Software Lifecycle: Making Software Development Processes and Software Produced by Them More Secure”. *Security in the Software Lifecycle*, 2006.

- [19] Jacobson, Van, Craig Leres, and Steven McCanne. *Tcpdump*, 2009. URL <http://www.tcpdump.org/>.
- [20] Kemmerer, R.A. and G. Vigna. “Intrusion detection: a brief history and overview”. *Computer*, 35(4):27–30, 2002.
- [21] Leverett, Eireann P. “Quantitatively Assessing and Visualising Industrial System Attack Surfaces”. *University of Cambridge, Darwin College*, 2011.
- [22] Martin-Garcia, L. and Fyodor. “Nping - Network packet generation tool”. 2012. URL <http://nmap.org/nping/>.
- [23] Matherly, John C. “SHODAN the computer search engine”. 2013. URL <http://www.shodanhq.com/help>.
- [24] Morris, T. and K. Pavurapu. “A retrofit network transaction data logger and intrusion detection system for transmission and distribution substations”. *Power and Energy (PECon), 2010 IEEE International Conference on*, 958–963. 2010.
- [25] Okolica, James, J Todd McDonald, Gilbert L Peterson, Robert F Mills, and Michael W Haas. “Developing systems for cyber situational awareness”. *2nd Cyberspace Research Workshop*, 46. 2009.
- [26] Pi, Raspberry. “An ARM GNU/Linux box for 25”. *Take a byte*, 2012. URL <http://www.raspberrypi.org/>.
- [27] Roesch, Martin and Chris Green. “Snort users manual”. *Snort Release*, 1(3), 2003.
- [28] Shenk, Jerry. “SANS Sixth Annual Log Management Survey Report”. 2010.
- [29] Singhal, Anoop, Theodore Winograd, and Karen Scarfone. “NIST Special Publication 800-95”. *Guide to Secure Web Services*, 2007.

- [30] Spett, Kevin. “Cross-site scripting”. *SPI Labs*, 2005.
- [31] Stouffer, Keith, Joe Falco, and Karen Scarfone. “Guide to industrial control systems (ICS) security”. *NIST Special Publication*, 800:82, 2011.
- [32] US-CERT/NIST. “CVE-2012-6440”. *National Vulnerability Database*, 2013.
- [33] Wightman, Reid, Dillon Beresford, Jacob Kitchel, and Rubin Santamarta. “Project Basecamp”. *Digital Bond*, 2012.
- [34] Wilhelm, Thomas. *Professional penetration testing: Creating and operating a formal hacking lab*. Syngress, 2009.

# REPORT DOCUMENTATION PAGE

*Form Approved*  
OMB No. 0704-0188

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

<b>1. REPORT DATE</b> (DD-MM-YYYY) 13-06-2013		<b>2. REPORT TYPE</b> Master's Thesis		<b>3. DATES COVERED</b> (From — To) Oct 2010-Jun 2013	
<b>4. TITLE AND SUBTITLE</b>  Evaluation of Cyber Sensors for Enhancing Situational Awareness in the ICS environment				<b>5a. CONTRACT NUMBER</b>	
				<b>5b. GRANT NUMBER</b>	
				<b>5c. PROGRAM ELEMENT NUMBER</b>	
				<b>5d. PROJECT NUMBER</b>	
				<b>5e. TASK NUMBER</b>	
				<b>5f. WORK UNIT NUMBER</b>	
<b>6. AUTHOR(S)</b>  Otis, Jeremy R.,					
<b>7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)</b> Air Force Institute of Technology Graduate School of Engineering and Management (AFIT/EN) 2950 Hobson Way WPAFB, OH 45433-7765				<b>8. PERFORMING ORGANIZATION REPORT NUMBER</b>  AFIT-ENG-13-J-06	
<b>9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)</b> Department of Homeland Security, ICS-CERT Neil Hershfield 900 N. Stuart St. Apt. 75 Arlington, VA 22203				<b>10. SPONSOR/MONITOR'S ACRONYM(S)</b>	
				<b>11. SPONSOR/MONITOR'S REPORT NUMBER(S)</b>	
<b>12. DISTRIBUTION / AVAILABILITY STATEMENT</b> DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED					
<b>13. SUPPLEMENTARY NOTES</b> This work is declared a work of the U.S. Government and is not subject to copyright protection in the United States.					
<b>14. ABSTRACT</b> Industrial Control Systems (ICS) monitor and control operations associated with the national critical infrastructure (e.g., electric power grid, oil and gas pipelines and water treatment facilities). These systems rely on technologies and architectures that were designed for system reliability and availability. Security associated with ICS was never an inherent concern, primarily due to the protections afforded by network isolation. However, a trend in ICS operations is to migrate to commercial networks via TCP/IP in order to leverage commodity benefits and cost savings. As a result, system vulnerabilities are now exposed to the online community. Indeed, recent research has demonstrated that many exposed ICS devices are being discovered using readily available applications (e.g., Shodan search engine and Google-esque queries). Due to the lack of security and logging capabilities for ICS, most knowledge about attacks are derived from real world incidents after an attack has already occurred. Further, the distributed nature and volume of devices requires a cost effective solution to increase situational awareness. This research evaluates two low cost sensor platforms for enhancing situational awareness in the ICS environment. Data obtained from the sensors provide insight into attack tactics (e.g., port scans, Nessus scans, Metasploit modules, and zero-day exploits) and characteristics (e.g., attack origin, frequency, and level of persistence). The results indicate that the low cost cyber sensors perform sufficiently within the ICS environment. Furthermore, findings enable security professionals to draw an accurate, real-time awareness of the threats against ICS devices and help shift the security posture from reactionary to preventative.					
<b>15. SUBJECT TERMS</b> ICS logging, Gumstix logging, Raspberry Pi, situational awareness, ICS					
<b>16. SECURITY CLASSIFICATION OF:</b>			<b>17. LIMITATION OF ABSTRACT</b>	<b>18. NUMBER OF PAGES</b>	<b>19a. NAME OF RESPONSIBLE PERSON</b> (ENG)
a. REPORT	b. ABSTRACT	c. THIS PAGE			<b>19b. TELEPHONE NUMBER</b> (include area code)
U	U	U	UU	91	(937) 255-3636 ext. 4332