Testing and Exploring Vulnerabilities of the Applications Implementing IEC 60870-5-104 Protocol

Zi Bin, Cheah

KTH Electrical Engineering

Master Thesis
Stockholm, Sweden 2008

XR-EE-ICS 2008:021

Testing and Exploring Vulnerabilities of the Applications Implementing IEC 60870-5-104 Protocol

Zi Bin, Cheah

cheah@stud.ntnu.no
zbcheah@kth.se

MASTER'S THESIS

Student's name: Zi Bin, Cheah

Area: Telematics

Title:

Testing and Exploring Vulnerabilities of the Applications Implementing IEC 60870-5-104 Protocol

Description:

Supervisory Control and Data Acquisition (SCADA) networks control critical infrastructures. They play vital roles for utility companies and the process industry including electricity, natural gas, oil, water, etc.This research will analyze, test and identify threats that might be faced by Applications implementing IEC 60870-5-104, which is a de facto industry standard protocol for implementing the parts of SCADA communications.

The protocol enables the Master Station to request data from Substations using pre-defined commands and Substations to respond by transmitting the requested data. IEC 60870-5-104 was not designed with security mechanisms in mind. The protocol itself lacks any form of authentication or encryption. It is made worse when industrial stations have started to connect to the Internet as this will allow conventional TCP/IP-based attacks to be launched.

The Thesis will study the protocol with respect to security analysis and risk. The next phase will be to try to identify and test vulnerabilities. Analysis shall be done to estimate security threats from found vulnerabilities.

Start Date: 15-02-2008
Deadline: 30-06-2008
Submission Date: 30-06-2008

Abstract

IEC 60870-5-104 protocol is an important protocol in the SCADA system. ABB Company uses this protocol for monitoring and managing power utility devices. These devices are interconnected and form part of an important SCADA systems.

As SCADA environment becomes more interconnected to the networked world, the understanding of SCADA and it's associated protocol increases in the public domain. The concept of *security by obscurity* that protected the SCADA environment is no more efficient. In this thesis we look at the many characteristics of power utility SCADA devices and it's possible weaknesses.

This thesis studied the IEC 60870-5-104 protocol. We used a method called “fuzzing” to test the protocol implementation in target devices. This method allows us to inject random or semi-random data into target devices. We also used vulnerability scanners and HTTP scanner to probe the target device. Finally, we performed a TCP/IP based attack on the device.

The author of this thesis has signed an Non Disclosure Agreement with ABB Corporate Research Centre, Oslo. The original thesis should only be read by people who have also signed the Non Disclosure Agreement. This copy of thesis is not the original version and thus is publicly available. The original version involves the test findings and analysis.

Acknowledgment

This project has given me the opportunity to understand the security concerns of the SCADA systems. It also allowed me to look into the IEC 60870-5-104 protocol. The opportunity to be involved in laboratory work is exciting.

I had the privilege to work alongside **Omar Faruk** - whom was also involved in his Masters thesis project in ABB. We carried out discussions and comparisons regarding the IEC 60870-5-104 and DNP3 protocol – which he was working on.

I would also like to thank **Martin Gilje Jaatun** from SINTEF ICT for recommending us to ABB for our Masters thesis. Martin was also very resourceful in commenting on many of our thesis drafts.

I would like to thank ABB Corporate Research Centre staff **Kai Hansen** and **Kevin McGrath** for their support in ABB. Kevin helped us settled in quickly by explaining some new testing concepts and the previous work done in ABB.

I would like to thank the supervisors/examiners in KTH and NTNU - **Teodor Sommestad** and **Mathias Ekstedt** from KTH, and **Svein Knapskog** from NTNU. I appreciate all the feedbacks that I received from all of you.

Table of Contents

Illustration Index

Acronyms

ABB : *Asea Brown Boveri* is a multinational corporation operating mainly in the power and automation technology areas. This thesis is performed in ABB's Corporate Research Centre in Oslo.

AES : *Advanced Encryption Standard* is a security algorithm considerably strong for civilian usage.

APCI : *Application Protocol Control Information* is an upper layer header of IEC 60870-5-104.

APDU : *Application Protocol Data Unit* is the whole IEC 60870-5-104 upper layer packet, consisting of both the APCI and ASDU.

ARP : *Address Resolution Protocol* is a protocol for mapping an IP address to a physical machine address that is recognized in the local network.

ASDU : *Application Service Data Unit* is the upper layer payload of IEC 60870-5-104.

CRC : *Cyclic Redundancy Check* is a method of error detection and correction that is applied to a certain field of data. CRC is an efficient method of accidental error detection, but not deemed secured enough for data integrity check.

DCS : *Distributed Control System* is a series of decentralized control centre which have some degree of autonomy, but are still integrated into a whole system.

DNP3 : *Distributed Network Protocol 3* is a set of communications protocols used between components in DCS/SCADA. It is commonly compared with IEC 60870-5 protocols.

DoS : *Denial of Service* an attempt to make a computer resource unavailable to its intended users. It generally consists of effort to prevent an Internet site or service from functioning efficiently or at all, temporarily or indefinitely.

HTTP : *Hyper Text Transfer Protocol* is a language protocol used when web sites and browsers communicate with each other. It indicates that the Web site interprets HTML and uses it as a part of the Web address.

IANA : Internet Assigned Numbers Authority is an organization that was responsible for the allocation of IP addresses, port numbers, character sets, and so on.

ICMP : *Internet Control Message Protocol* is a protocol used between a host server and a gateway to the Internet to send message control and error-reporting messages.

IDS : *Intrusion Detection System* a system deployment used to detect all types of malicious network traffic and computer usage, especially those that can't be detected by a conventional firewalls.

ISDN : *Integrated Services Digital Network* is a service offered by telephone carriers for the transmission of voice and data.

IEC : *International Electrotechnical Commission* is an organization that sets standards for electronic products and components.

IEEE : *Institute of Electrical and Electronics Engineers* is a professional society for electrical and computer engineers.

IPSec : *Internet Protocol Security* has been developed to support a secure exchange of packets at the IP layer. It has been deployed widely in VPNs.

MAC : *Media Access Control* is a special numbering scheme on a computer network interface card. It is used to create a unique identity for each computer connected to the Internet.

OSI : *Open Systems Interconnect* is an industry wide protocol standard consisting of seven well defined layers. The layers are 1. Physical 2. Data Link 3. Network 4. Transport 5. Session 6. Presentation 7. Application.

PCS : *Process Control System* controls the output of a specific process. It is often use for controlling processes in different industry. Often, PCS are referred to as DCS or SCADA when there are more complex processes involved - including remote management and security.

RPC : *Remote Procedure Call* allows program to make subroutine calls to other systems, usually across the network.

RTU : *Remote Terminal Unit* is a device which interfaces objects in the physical world to a distributed control system or SCADA system by transmitting telemetry data to the system and/or altering the state of connected objects based on control messages received from the system.

SCADA : *Supervisory Control And Data Acquisition* is a software for process control. It gathers data in from remote locations in order to control equipment and conditions. SCADA is used in power plants as well as in oil and gas refining, telecommunications, transportation, and water and waste control.

TCP : *Transmission Control Protocol* is a protocol used along with the Internet Protocol (IP) to send data in the form of packets between computers over the Internet.

TLS : *Transport Layer Security* is a protocol intended to secure and authenticate communications across a public networks by using data encryption.

VPN : *Virtual Private Network* creates a secure tunnel between the points within the VPN. Only devices with the correct "key" will be able to work within the VPN. The VPN network can be residing within a normal company LAN (Local Area Network), and/or over public networks such as Internet.

X.25 : X.25 is a communication protocol for transmitting and receiving packets of data over a network.

1 Introduction

Supervisory Control and Data Acquisition (SCADA) networks control critical infrastructures. They play vital roles for utility companies and the process industry including electricity, natural gas, oil, water, etc. This thesis will analyze, test and identify threats that might be faced by applications implementing IEC 60870-5-104. IEC 60870-5-104 is a de facto industry standard protocol for SCADA communications implementation over TCP/IP stack.

The protocol enables the master station to request data from substations using pre-defined commands and substations to respond by transmitting the requested data. IEC 60870-5-104 was not designed with security mechanisms in mind. The protocol itself lacks any form of authentication or encryption. It is made worse when industrial stations have started to connect to the Internet as this will allow conventional TCP/IP-based attacks to be launched.

The primary technique used to test vulnerabilities of IEC 60870-5-104 is fuzzing. Fuzzing is a method that allows semi random or random generated packets to be sent to the target machines – in hope of crashing or malfunctioning it. Other techniques include scanners to analyze vulnerability the devices. We also performed Denial of Service attacks.

We will focus on testing the effect it has on target devices. The outcome can be used by ABB's software development team to improve the reliability or robustness of their devices.

1.1 Problem Statement

The security of SCADA systems has come into question as they are increasingly seen as extremely vulnerable.

There are several weaknesses that make such systems susceptible.

- The lack of concern in security and authentication in the design, deployment and operation of existing networks.
- The mistaken belief that they have the benefit of "*security by obscurity"* through the use of specialized protocols and proprietary interfaces.
- The mistaken belief that they are secure because they are supposedly physically secured
- The mistaken belief that they are secure because they are supposedly disconnected from the Internet.
- Knowledge gap between IT professionals and process control professionals is also a factor. Both have different understanding expertise when it comes to processes and technology.

Some of the aforementioned protections are already defeated. Also, systems nowadays are becoming more interconnected to the Internet. These are signs that the "barrier-to-intrude" becomes lower by the day.

To make matters more serious, there are extra incentives for the blackhat hacking community to penetrate and disrupt *SCADA* systems. By attacking these networks, the damages are more "real" as power, oil, water, traffic, amongst others become interrupted. [1] According to Jonathan Pollet [2], over 90% of major *SCADA* and automation vendors have all of their manuals and specifications online to the general public.

The examples below describe errors in power industry has happened in the past.

- **America's Davis-Besse power plant**

  The Nuclear Regulatory Commission confirmed that in January 2003, the Microsoft SQL Server worm known as Slammer infected a private computer network at the idled Davis-Besse nuclear power plant in Oak Harbor, Ohio, disabling a safety monitoring system for nearly 5 hours. In addition, the plant's process computer failed, and it took about 6 hours for it to become available again. [4]

- **North America's 2003 blackout**

The Northeast Blackout of 2003 was a massive widespread power outage that occurred throughout parts of the Northeastern and Midwestern United States, and Ontario, Canada on August 14, 2003.

A silent failure of the alarm function is listed as one of the direct causes of a blackout that eventually cut off electricity to 50 million people in eight states and Canada that estimated the cost of financial losses at $6 billion USD. [5]

In U.S., critical infrastructure businesses account for more than 2 million business establishment in the U.S., which is roughly 17% of all U.S. establishments. Combined, they account for 40% of the GDP. The cost of poor services including power and network failure can cost up to $188 billion per year. [6]

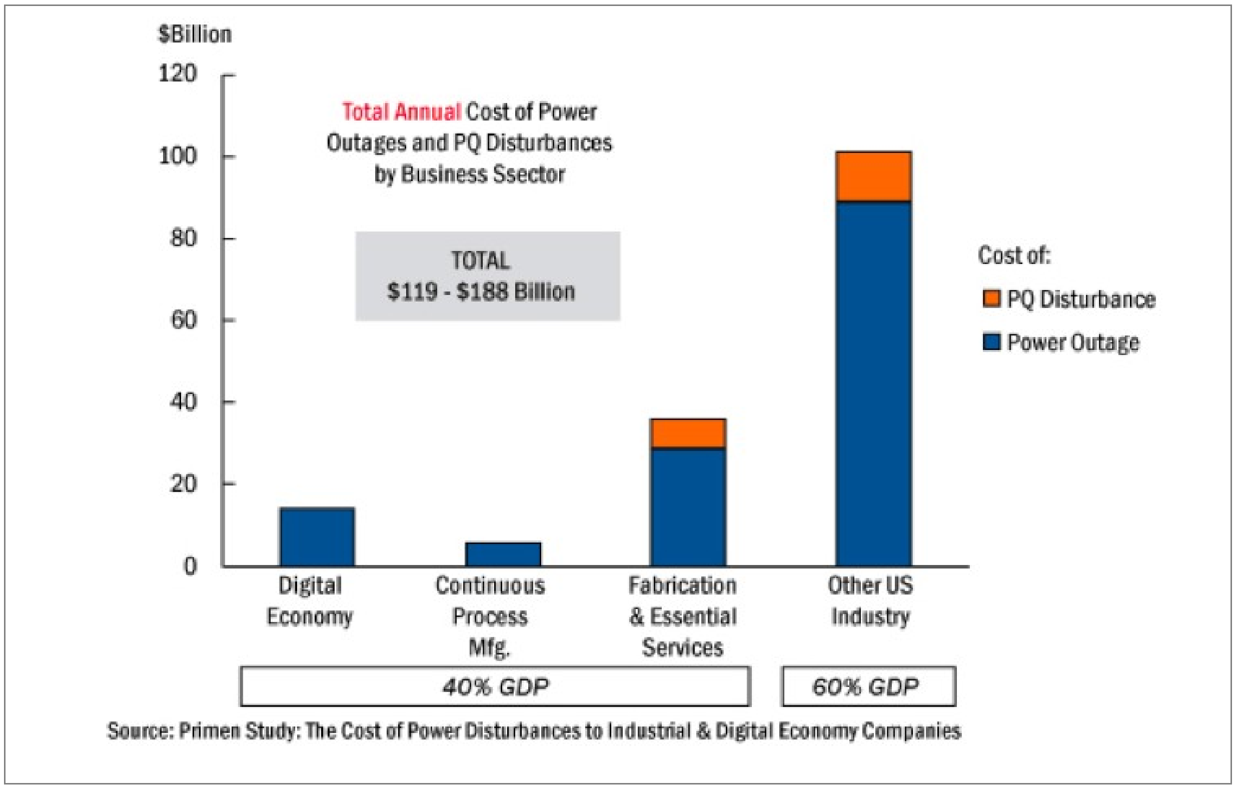


Figure 1: Total Annual Cost of Power Outages by Business Sector [6]

These incidents show that if malicious attacks are targeted at similar *SCADA* system, the attacker might be able to create the same outcomes as elaborated by the two example above.

Furthermore, we realized that malicious attacks on *SCADA/PCS* are happening. As these systems become more connected, they become more and more exposed.

1.2 Research Questions

The goal of this thesis is to access weaknesses in the IEC60870-5-104 protocol. Weaknesses can come directly from the software implementation of the IEC 60870-5-104 protocol, or from weaknesses not directly related to the protocol, but nevertheless exposes computer security weaknesses of the device.

We will try to answer the questions:

1. Is IEC 60870-5-104 a secure protocol, if not what are it's inherent weaknesses?

2. What are the test cases that we want to create that allows us to test IEC 60870-5-104's implementation on target devices?

3. Are target devices implementing IEC 60870-5-104 protocols immune to malformed/ erroneous data?

4. What kind of attacks are IEC 60870-5-104 protocols susceptible to? What weaknesses do the attack take advantage of?

5. Can secure coding practices overcome such problems?

6. Apart from fuzzing, what are the other options to discover vulnerabilities in IEC 60870-5-104 implementation?

Will vulnerability scanners that are freely available on the Internet be able to discover any weaknesses?

1.3 Areas Covered

1. **SCADA Security**

   The *SCADA* systems are built using public or proprietary communication protocols which are used for communicating between a master unit and one or more RTUs. The *SCADA* protocols provide transmission specifications to interconnect substation computer. We discuss why it is important to have a secured *SCADA* system.

2. **IEC 60870-5-104**

   IEC 60870-5-104 uses an open TCP/IP interface to communicate with devices in an intranet and the Internet. It is a standard industrial communications protocol that is often used on control networks. We look into this protocol, and other relevant protocols. We discuss why it is important to have IEC 60870-5-104 with built-in security mechanism, rather than depending on external protection such as *VPN*. We also discuss ongoing initiative to implement the security features into the protocol.

3. **Fuzz Testing**

   A fuzzer is a program that attempts to discover security vulnerabilities by sending random input to an application/device. If the target contains a vulnerability that can leads to an exception, crash or even delay, it can be determined that a vulnerability has been discovered. In the thesis, we use fuzzing techniques to inject data into our target devices. This is done by creating *test cases,* injecting data continuously on the target, and observing the outcome.

4. **Vulnerability Scanning**

   Vulnerability scanners are programs that scan a computer or device to discover vulnerabilities. These errors can range from known common protocols flaws, to operating systems backdoors. We use the vulnerability scanners to scans target devices and records vulnerabilities found. We also performed a simple DoS attack on a target device.

5. **Secure Coding**

   Results from f*uzz testing* and *vulnerability scanning* will be submitted to ABB. If there are vulnerabilities found, ABB can patch the problems with firmware updates.

1.4 Keywords

SCADA, PSC, security, industrial protocol, IEC 60870-5-104, Black box testing, fuzz testing, fuzzing, vulnerability scanner.

1.5 Limitations

As we are fuzzing the target device without knowledge of the program source code, we are performing blackbox testing. Blackbox testing might be weak in discovering all *cases* in the code, such as leaving certain *if-else* condition statements unchecked.

We are testing IEC 60870-5-104. Weaknesses found might not be reproducible in other devices implementing the same protocol. Also certain vulnerabilities found on the test device might be due to other flaws rather than the IEC 60870-5-104 protocol flaw.

Fuzz testing is not a substitute for exhaustive testing or formal methods. Fuzzing also do a poor job at finding vulnerabilities related to information disclosure, encryption flaws and any other vulnerability that does not cause the program to crash.

1.6 Report Structure

In this section, we present the outline of this thesis. This includes some pertinent questions about the thesis topics – the backgrounds, the problems we are addressing, the questions we try to answer, and the limitations that come with the thesis project.

In the **2nd chapter**, we give an explanation on concepts that are common in the power industry – including the tools they used. We then discussed security concepts. Last but not least, we look into IEC 60870-5-104 and relevant protocols in brief.

In the **3rd chapter**, we talk about fuzzing. We discuss this concepts and it's methods and application. We dedicate the later part of this chapter to introduce vulnerability scanners.

After looking at IEC 600870-5-104 in brief in chapter 2, we now dive into the specifications in the **4th chapter**. The last part of the chapter discusses an example packet.

Chapter 5 discusses the theory aspect of fuzzing, while **Chapter 6** shows the fuzzing results using two fuzzers.

Chapter 7 shows results of two Vulnerability scanners. Next a DoS results is presented.

Finally **Chapter 8** discusses all the test results and tries to answer the research questions.

2 Theory

Below we discuss some concepts that are relevant to our thesis.

2.1 Process Control

Process control is a discipline that deals with controlling the output of a specific industrial process. For example, a process control can include controlling the heating of a room temperature. This process has the objective to reach and maintain a defined temperature, say 15 degrees. Here, the temperature is the controlled variable. At the same time, it is the input variable since it is measured by a thermometer and used to decide whether to heat or not to heat. The desired temperature of 15 degrees is the set point. The state of the heater (e.g. the setting of the valve allowing hot water to flow through it) is called the manipulated variable since it is subject to control actions.

Larger and more complex systems can be controlled by a DCS or SCADA system.

2.2 SCADA and DCS

SCADA is the acronym for "Supervisory Control and Data Acquisition". *DCS* is the acronym for "Distributed Control Systems". SCADA is referred as a large-scale, distributed process control system.

Typically, SCADA involves a the communication between a Remote Terminal Unit (RTU) and Operator Station(s). The communication can be linked by a local Intranet, or over the Internet. A simplified version of SCADA system is shown in figure below. [7]

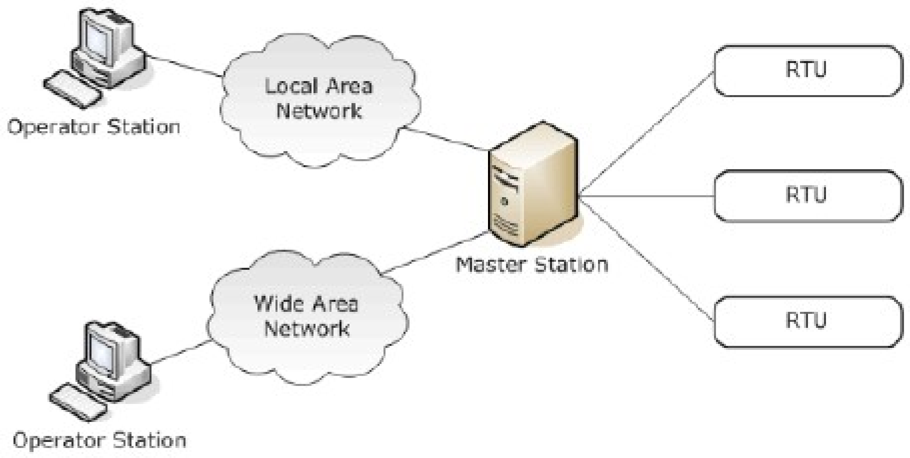


Figure 2: SCADA Systems Involving Operator Station, Master Station and RTUs [7]

Both *SCADA* and *DCS* describe and relate to the same type of industrial control systems. *SCADA* is used in American journals and in the power producing industry. *DCS* is the equivalent European abbreviation and is used by the gas and oil industry. [8] Both *SCADA* and *DCS* term are used interchangeably in this thesis.

2.3 Power Utility System

This project's objective is to study vulnerabilities that exists in target devices implementing the protocol IEC 60870-5-104. These devices are used in the power utility area of a SCADA system. We now take a look at the power utility system, and it's security characteristics.

Power utility systems have a wide variety of functions which are crucial to the day-to-day running of the electrical power utility. These functions include identifying faults, isolating them and restoring service, circuit breaking, recloser control, feeder reconfiguration, line switching, voltage control, load management, automatic meter reading, automatic generation control, economic dispatch, simulation and emulation, capacitor bank switching, voltage regulator monitoring, transformer temperature, and metering. [9]

Knowledge of electric utility power system monitoring and control is critical to operating and managing the power system. As discuss in *Problem Statement*, previous blackout has caused enormous losses to the industry. This endangers people's life.

When *SCADA* was first created, there was little or no connection to other networks. As *SCADA* develops, *SCADA* devices today are interconnected by communication links including - optical fiber, radio, telephone line, microwave, satellite, and ethernet. The IEEE standard C37.1-1994 specifies the communication topologies used in SCADA. They vary from simple a point-to-point to composite architectures with one single master station, multiple sub-master (slave master) stations and multiple RTUs[9]. In other words, SCADA will continue to become more available through many mediums, thus exposing to attacks through different mediums.

We would like to highlight some security characteristic of the Power Utility System :

- A denial-of-service in power systems for critical functions or prioritized functions could cause critical failure to the system.

- Many communication channels used in the power industry are narrowband, thus not permitting some of the overhead needed for certain security measures, such as encryption and key exchanges. This is particularly true for IEC 60870-5-101 that uses serial link. IEC 60870-5-104 is somehow free from this constraint as it uses a network connection of possibly up to 100mbps.

- Most systems and equipments are located in wide-spread, unmanned, remote sites with no access to the Internet. This makes key management and some other security measures difficult to implement.

- Many systems are connected by multi-drop communication channels, so normal network security measures might not work.

- Wireless communications are becoming widely used for many applications. Wireless communications could proof easier to compromise as there are prevalence of WEP hacking in the wireless community. Also, noises produced in a wireless environment can corrupt the communication links.

2.4 Concept of Security

This chapter introduces some theory of security.

2.4.1 Security

Computer security is a branch of technology known as information security as applied to computers. Security can involve many facades, the primary three are *Confidentiality*, *Integrity*, *Available*. *Figure 3 [7]* shows the *CIA* triad.

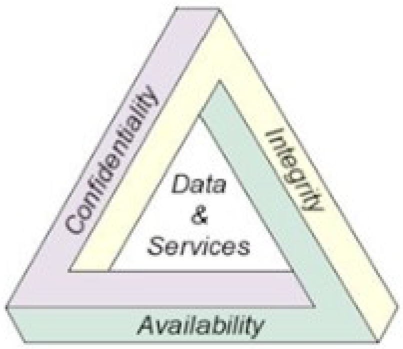


Figure 3: Computer Security Triad– Confidentiality, Availability and Integrity

Other elements beyond the main three are, amongst others, *non-repudiation* and *robustness*. Each of this aims to achieve different purpose of security.

Confidentiality ensures that information is accessible only to those authorized to have access. The field of cryptography helps to achieve security by providing confidentiality of data/identity.

Integrity tries to ensure that data is "whole" or complete, the condition in which data are identically maintained during any operation (such as transfer, storage or retrieval), the preservation of data for their intended use, or, relative to specified operations, the a priori expectation of data quality. Put simply, data integrity is the assurance that data is consistent and correct.

Availability means that the information must be available when it is needed. Data might be stored safely in a remote area, but without the ability to retrieve it, the data is as good as gone. Availability ensures that data is available anytime.

Non-repudiation implies ones intention to fulfill their obligations to a contract. It also implies that one party of a transaction cannot deny having received a transaction nor can the other party deny having sent a transaction

Robustness is the quality of being able to withstand stresses, pressures, or changes in procedure or circumstance. A system, organism or design may be said to be "robust" if it is capable of coping well with variations. Robustness is related to availability since a robust system is almost always available for data delivery/processing.

2.4.2 Vulnerability

In this chapter, we discuss two common vulnerabilities that exist. They are *buffer overflow* and *format string attack*.

Buffer Overflow

A buffer overflow, or buffer overrun, is an anomalous condition where a process attempts to store data beyond the boundaries of a fixed-length buffer. The result is that the extra data overwrites adjacent memory locations. The overwritten data may include other buffers, variables and program flow data, and may result in erratic program behavior, a memory access exception, program termination (a crash), incorrect results or, especially if deliberately caused by a malicious user, a possible breach of system security.

A detailed buffer overflow scenario is explained in [10].

Format string attack

The Format String exploit occurs when the submitted data of an input string is evaluated as a command by the application. In theory, the attacker can execute code in an application.

The attack could be executed when the application doesn't validate properly the submitted input- such inputs are called un-sanitized input. For example, a format strings parameter, like %x, is inserted in the posted data, the string is parsed by the Format Function the conversion specified in the parameters is executed. However, the Format Function is expecting more arguments as input, and if these arguments are not supplied, the function could read or write the stack.

To understand the attack it's necessary to explain the components that constitute it. They are: [11]

- The Format Function is an ANSI C conversion function, like printf, fprintf, which converts primitive variable of the programming language in a human readable string representation.
- The Format String is the argument of the Format Function and is an ASCII Z string which contains text and format parameters, like: printf ("The magic number is: %d\n", 1911);
- The Format String Parameter, like %x %s defines the type of conversion of the format function.

Format bugs were first noted in 1990 in the fuzz testing work done at the University of Wisconsin.

2.5 Industrial Network Model

Chapter 2.5 discusses the model that IEC 60870-5-104 is modeled upon, which is the *EPA model*.

2.5.1 OSI Model

OSI is a seven layer protocol model. It is the abbreviation for *Open System* Interconnect. This model consists of the *Application*, *Presentation*, *Session*, *Transport*, *Network*, *Data Link*, and *Physical* layers. A layer is a collection of related functions that provide services to the layer above it and receives service from the layer below it.

2.5.2 EPA Model

Around the same time, when *OSI* layer model emerged. A three layer version of this model was proposed by the *IEC* to provide a simplified hierarchical model that would be suitable as a basis for *SCADA* communications. This is known as the *EPA* model- which stands for *Enhanced Performance Architecture* [12]

In the figure 4, it shows the relationship of OSI and EPA.[13]

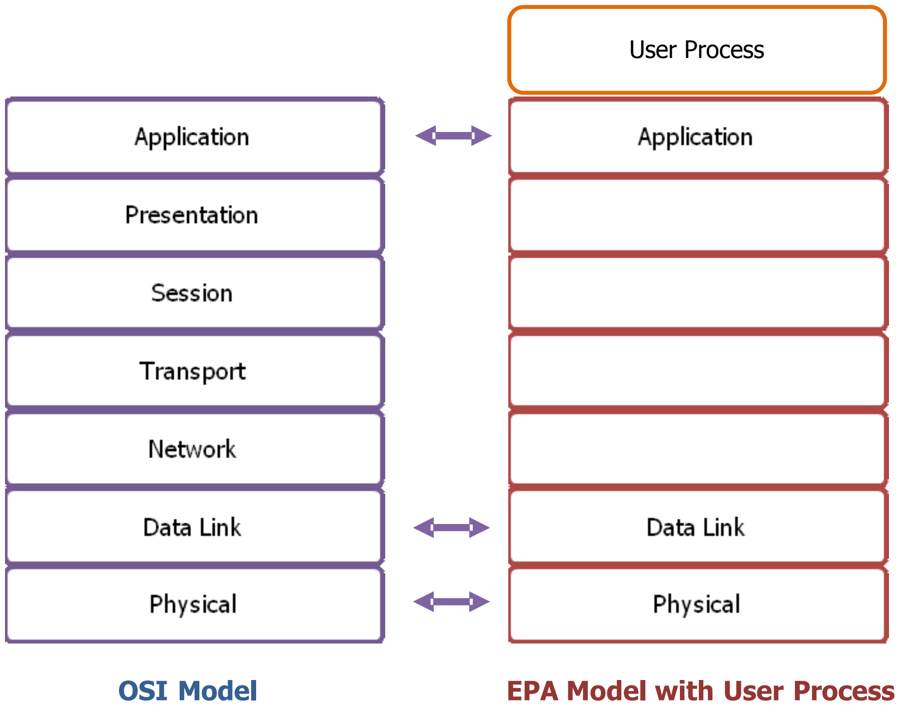


Figure 4: OSI & EPA Model Comparison

2.6 Network Protocols

In the early 1990s two separate open standard network protocols for *SCADA* were developed by different organizations. These provided both protocols with some areas of similarity in their procedures for handling communications at the lower data link level. In their higher level functionality and data objects though, they are entirely different. In summary, two open protocol standards have emerged. These are: [12]

- DNP3
- lEC 60870-5-101 (IEC 60870-5-104 for network transmission)

These are the two open communication protocols that provide for interoperability between systems for telecontrol applications that are now competing within the world market. Today DNP has a strong following in North America, South America, South Africa, Asia and Australia, while lEC 60870-5-101(IEC60870-5-104 for network transmission) is strongly supported in the European region. [12]

2.6.1 IEC 60870-5

IEC 60870-5 is a communication profile for sending basic telecontrol messages between two systems. Telecontrol messages are "remote control" messages.

The IEC Technical Committee 57 (Working Group 03) has developed a protocol standard for telecontrol, teleprotection, and associated telecommunications for electric power systems. The result of this work is IEC 60870-5. Five documents specify the base IEC 60870-5: [13]

1. **IEC 60870-5-1** Transmission Frame Formats
2. **IEC 60870-5-2** Data Link Transmission Services
3. **IEC 60870-5-3** General Structure of Application Data
4. **IEC 60870-5-4** Definition and coding of Information Elements
5. **IEC 60870-5-5** Basic Application Functions

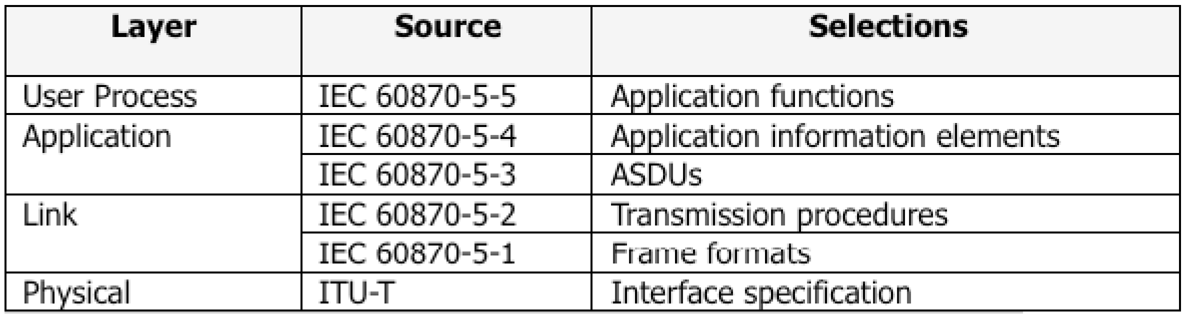
The IEC Technical Committee 57 has also generated companion standards:

1. **IEC 60870-5-101** defines all necessary application level functions and data objects to provide for telecontrol applications operating over wide area networks, using low bandwidth bit-serial communications.
2. **IEC 60870-5-102** and **IEC 60870-5-103** provides data types and functions to support electrical protection systems.
3. **IEC 60870-5-104** defines operation of the transmission protocol over networks using standard transport profile specifying the TCP/IP protocol.

This companion is related to IEC 60870-5-101, as it replaces the message transport sections with a network version, leaving the application level functions largely unaltered.

2.6.2 IEC 60870-5-101

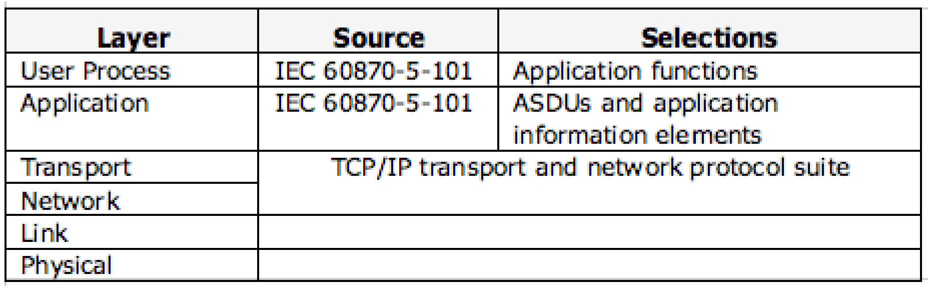
IEC 60870-5-101 completely specifies the whole IEC 60870-5 protocol structure. It does this by referring to the main sections of the IEC 60870-5 standard, and by making particular selections from the options that may be available within those selections.[14] As shown in figure 5.



Layer	Source	Selections
User Process	IEC 60870-5-5	Application functions
Application	IEC 60870-5-4	Application information elements
	IEC 60870-5-3	ASDUs
Link	IEC 60870-5-2	Transmission procedures
	IEC 60870-5-1	Frame formats
Physical	ITU-T	Interface specification

Figure 5: Matching IEC 60870-5-X -to- EPA Layer [14]

As we discussed later in the thesis, when IEC 60870-5-104 companion is implemented, **the lower layer of IEC 60870-5 is removed, and replaced by TCP/IP protocol suits** (only keeping the user process and application layer of IEC 60870-5-101 with IEC 60870-5-104 specification of TCP/IP usage). The figure below illustrates this.



Layer	Source	Selections
User Process	IEC 60870-5-101	Application functions
Application	IEC 60870-5-101	ASDUs and application information elements
Transport	TCP/IP transport and network protocol suite	
Network		
Link		
Physical		

Figure 6: IEC 60870-5-104: IEC60870-5-101 Siting On Top of TCP/IP Stack [14]

2.6.3 IEC 60870-5-104

IEC 60870-5-104 is a companion standard of IEC 60870-5. It uses standard transport profiles, this standard defines the use of an open TCP/IP-interface to transport data. Routers which include the different WAN-types (for example. X.25, Frame Relay, ISDN etc) may be connected via a common TCP/IP LAN interface to transfer IEC 60870-5-104 packets.

In the figure below, it shows the end-to-end transmission of a SCADA system using IEC 60870-5-104 as the protocol. As the figure shows, at the application level, IEC 60870-5-104 uses the specification already defined by IEC 60870-5-101. The difference is that IEC 60870-5-104 uses TCP/IP stack as transport while IEC 60870-5-101 uses a serial link.

The wide area technology backbone can be X.25, Frame Relay or ISDN as long as in can carry TCP/IP based packets. This is extremely useful as, in the future, SCADA systems might be deployed over networks deploying different backbone.

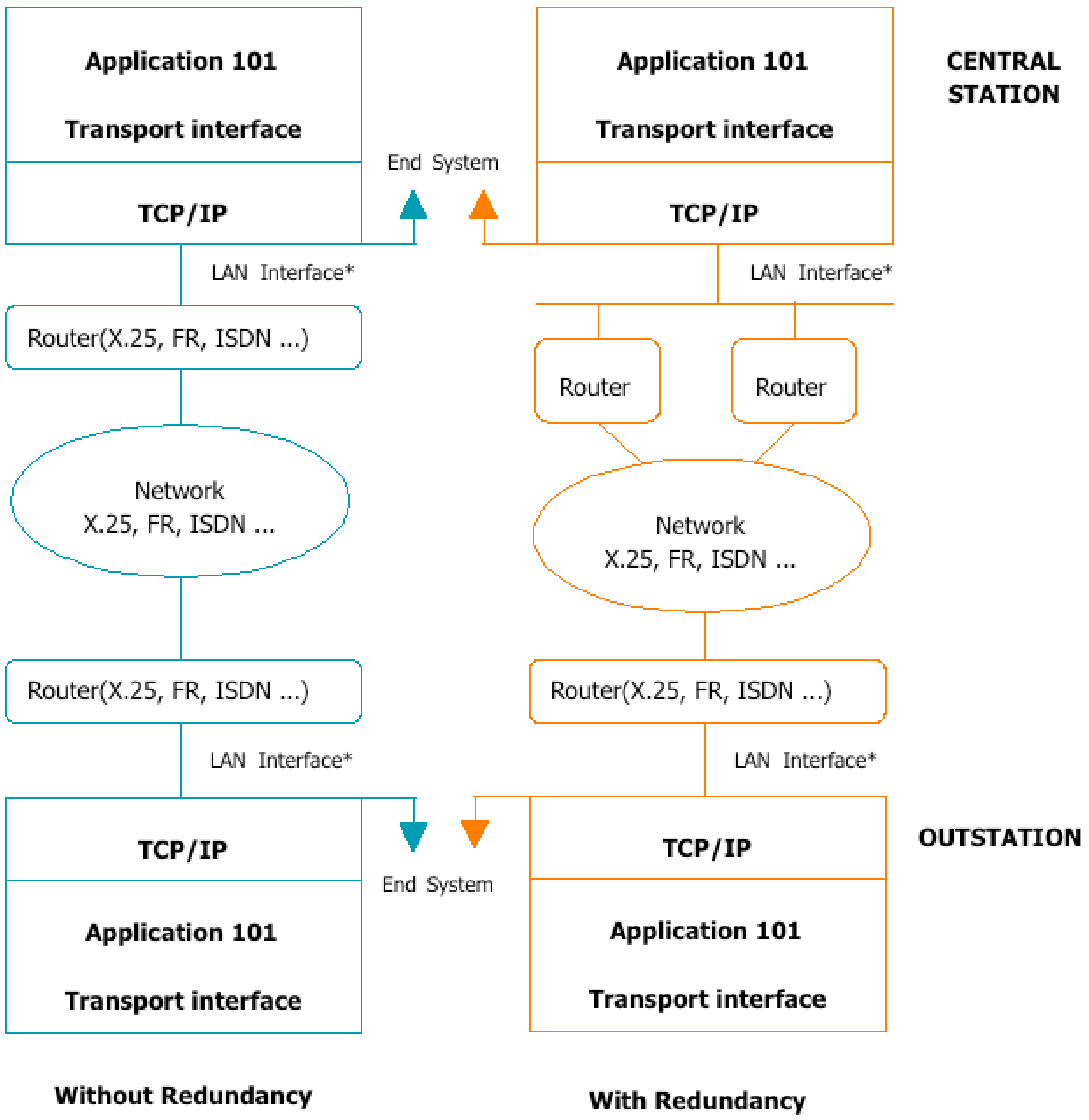


Figure 7: IEC 60870-5-104 end-to-end transmission [14]

In *chapter 4*, we take a look at the protocol more in-depth.

2.6.4 DNP3

DNP3 is closely related to the *SCADA* systems as it is seen as an alternative to IEC 60870-5-104.

DNP3 (Distributed Network Protocol) is a set of communications protocols used between components in process automation systems. Its main use is in utilities such as electric and water companies.

DNP3 is, in standard networking terms, a layer 2 protocol. It provides multiplexing, data fragmentation, error checking, link control, prioritization, and layer 2 addressing services for user data.

It makes particularly heavy use of Cyclic Redundancy Checks (CRCs) embedded in its data packets, in an attempt to deal with the very noisy environments in which it is typically used. [15]CRC is the only security mechanism implemented internally by DNP3. IEC 60870-5-104 does not even have CRC. However, IEC 60870-5-104 can make use of the TCP/IP's CRC check instead.

Like IEC 60870-5-104, many modern applications can now carry DNP3 messages over TCP/IP.

3 Testing

In this chapter, we discuss testing, fuzzing methodology and tools used. We also briefly discuss how fuzz tests are performed. Lastly we touch on *DoS* attack.

3.1 Testing Method

Test methods can be broadly divided into *whitebox* testing and *blackbox* testing.

3.1.1 Whitebox Testing

Whitebox testing **gives the tester an internal perspective of the system's buildup**. This may include access and understanding of the target machine, operating systems and source codes.

By looking at the codes, the tester can test out all logic paths of the code. He may also figure out programming flaws that might be exploited by different test cases.

3.1.2 Blackbox Testing

Blackbox testing **gives the tester and external perspective of the system.** He might not know how the target is constructed form the inside. He will only be able to probe the systems from the outside. In software testing, blackbox methods imply that the tester does not have the source code. Therefore he will need to build his test cases based on his other inference on the software.

While this method can uncover unimplemented parts of the specification, one cannot be sure that all existent paths are tested.

In our attempt to test 60870-5-104 devices, we will deploy several blackbox testing methods which are - *equivalence partitioning* and *boundary value analysis*. These techniques will be discussed further in *chapter 3.3.*

3.2 Fuzzing

Fuzzing is a software testing technique that provides random data to the target. A target can be a web application, computer host, or even a mobile phone. Anything that is capable of accepting the fuzz data is a possible target.

Fuzzing can be entirely random, whereby a *test case* constructs random data that is being fed to the target. Since test data are random, there is a high probability that the test cases might fail to find any weaknesses. However, the nature of fuzzing allows test cases to construct thousands of test data in a short period of time, thus **allowing random data test at a high speed**. This, in return, might produce discoveries.

Some individual prefers to have higher control over the test data. This **reduces randomness but increases control.** This is sometimes being regarded as *Mutation Analysis***.** *Mutation Analysis* carefully models the state, structure, and semantics of the protocol and maximizes the damage that can be caused in the shortest possible time.

Fuzz testing is effective in finding odd oversights and defects which normal system testing fails to uncover. This is because the "randomness" of fuzzing allows bugs to be uncovered accidentally, many case through black box testing scenarios .

However, fuzz testing is not a substitute for exhaustive testing or formal methods. Fuzzing also do a poor job at finding vulnerabilities related to information disclosure, encryption flaws and any other vulnerability that does not cause the program to crash.

3.2.1 Fuzz Methods

Fuzz testing or fuzzing is **a software testing technique** that provides random data ("fuzz") to the inputs of a program. If the program fails (for example, by crashing, or by failing built-in code assertions), the defects can be noted.

here are two forms of fuzz testing:

- *Valid fuzz* attempts to assure that the random input is reasonable, or conforms to actual production data.

- *Simple fuzz* usually uses a pseudo random number generator to provide input.

A combined approach uses valid test data with some proportion of totally random input injected.

By using all of these techniques in combination, fuzz-generated randomness can test the un-designed behavior surrounding a wider range of designed system states.

3.2.2 Fuzz Applications

There are many fuzzing applications, two fuzz applications are selected for this thesis. A collection of fuzzing tools can be found in [16]

- **GPF**

  GPF uses captured packet sessions (from libpcap) to construct a protocol description from real traffic. Users can then configure various types of injected faults, manually modify the capture file, and define custom functions to deal with dynamic data.

  We will further discuss GPF in Section 7.4. GPF runs on Linux machines. [17]

- **Sulley**

With Sulley, we need to construct a *test case* and also define the *requests*. Sulley's strength lies in the ability to define *requests* to have stronger control over the fuzzing process. Sulley's p*rocess monitor*'s ability to monitor test environment is also something GPF does not have.

We will further discuss Sulley in Section 7.2. Sulley runs on Windows. The Sulley information can be found in [18].

3.2.3 Fuzz Test Cases

Test cases are constructed to test the target. The target can be a server, software, web application and etc. A *test case* is a set of conditions or variables under used by a tester to determine if a requirement or use case is satisfied. It may take many test cases to determine that a requirement is fully satisfied.

In fuzzing analogy, a test case is often the test scripts. In software engineering domain, however, test case and test scripts are distinctively divided.

3.3 Fuzz Test Technique

In this thesis, tests are used to determine the robustness of our test target, which are devices running IEC 60870-5-104 protocols. As part of the test methodology, We discuss *equivalence partitioning* and *boundary value analysis*.

3.3.1 Equivalence Partitioning

It is a software testing related technique with the goal of :

- Reducing the number of test cases to a necessary minimum.
- Selecting the right test cases to cover all possible scenarios.

Equivalence Partitioning of IEC 60870-5-104 can be broadly divided into three.

- Consider each field for **APCI** and define valid and invalid input for each field.
- Consider each field for **APDU** and define valid and invalid input for each field.
- Consider each field for **both APCI and APDU** and define valid and invalid input for each field.

In order to reduce the test cases to a necessary minimum, and to select the right test case to cover all possible scenarios, fuzzer contributes an important aspect.

3.3.2 Boundary Value Analysis

Boundary Value Analysis serves as an extension to *Equivalence Partitioning,* often applying values to an input which is out of bound. According to [19], the common test conditions for *boundary value analysis* includes

- zero value
- negative value
- infinity value
- file space overflow
- multiple update of a single file
- table entries missing

For the instance of protocol testing, a fuzzer inputs value that is out of bound to protocol fields.

3.4 Other Test Tools

Apart from fuzzers, we also use different test tools. In this section, we discuss these test tools.

3.4.1 Vulnerability Scanner

A vulnerability scanner is a computer program designed to search for and map a system's weaknesses. The steps below illustrates the process it goes through.

1. Scan for host or applications that are alive. This includes looking for active IP addresses and ports. A list of host and open/filtered ports are compiled.

2. Using finger-printing techniques, the type of applications that runs on each ports are determined. This includes determining the Operating System of the host too.

3. From a list of known vulnerabilities, the scanner may be able to report if some of the applications/systems are vulnerable to attacks.

4. Some scanners will provide the option to launch attacks on the vulnerable services.

The purpose of a vulnerability Scanner is to :

- Detect vulnerabilities. It is always important to run vulnerability scanners as your attacker might be running similar scanners. Patching vulnerabilities found by scanners are crucial to prevent attacks.

There are a lists of different scanners out there. We will use several well-known ones to test target devices.

- **Nessus**
  We use Nessus 3 developed by Tenable Network Security [22] .Nessus is a popular vulnerability scanner, based of survey in [23]

  Nessus does not plugin to test IEC protocols.

- **Retina**
  Another popular scanner is Retina. We are using Trial Version 5.9.4.
  The trial version can be obtained from [24]

A list of well-known vulnerability scanners is compiled by *Insecure.Org* at [25]

3.4.2 DoS Attack

A denial-of-service attack is an attempt to make a computer resource unavailable to users. SYN Flooding, FIN_WAIT_2 Flooding, ICMP flooding and *ARP* spoofing can be used to launch Denial of Service attack.

TCP uses three-way handshake (*SYN*, *SYN-ACK*, *ACK*) to establish a session. In the case of SYN-Flooding, the attacker sends several packets but does not send the "ACK" back to the server or spoofing the source IP address of *SYN* message, server sends *SYN-ACK* to false IP address and never receives *ACK*.

ARP spoofing can stop the traffic to the spoofed target system. This is done through masquerading as the recipient to deceive the sender, and masquerading as the sender to deceive the recipient. Upon success, the recipient will not be able to receive data from the sender as it is now sent to the attacker. The attacker can also escalate the attack to Man-in-the-Middle attack by modifying data from the sender and send it to the recipient.

We also used a program to create FIN-WAIT-2 states. FIN-WAIT-2 represents waiting for a connection termination request from the remote TCP. [26]. It is similar to SYN-flood attack whereby resources has been allocated by the victim, thus causing a resource hog.

ICMP flood is another kind of DoS attack that sends large amount of PING data to the victim, with the intention of hogging the TCP/IP stack resources. We used a device called Mu4000[27] to perform SYN-flooding and ICMP flooding.

3.4.3 Web Server Scanner

A web server scanner is a program that performs a list of tests against a web server. We will be able to test it against 560CMU04 communication unit's web server, which is WindWeb1.0. We used Nikto as the scanner. According to Nikto's website [28] *"Nikto is an Open Source (GPL) web server scanner which performs comprehensive tests against web servers for multiple items, including over 3500 potentially dangerous files/CGIs, versions on over 900 servers, and version specific problems on over 250 servers."*

3.5 Other Efforts

There are some similar testing efforts being carried out by other parties.

The British Columbia Institute of Technology (BCIT) is one of a few groups to track industrial cybersecurity incidents. The BCIT Industrial security Incident Database (ISID) contains information regarding security related attacks on process 'control and industrial networked systems.[29]

Another attempt is done by ISA by developing a fifteen step Security Life Cycle Model. It is outline in "Technical Report ISA-TR99.00.02- 2004: Integrating Electronic Security into the Manufacturing and Control Systems Environment"[30]. This report provides a framework for developing an electronic security program and provides a recommended organization and structure for the security plan. The Security Life Cycle Model is a fifteen-step process that covers all areas of security management from initial goal setting to post deployment re-evaluation of security counter measures.

4 Fuzz

In chapter 5, We discussed the fuzzing steps and considerations when fuzzing.

4.1 Fuzz Steps

Fuzz testing can be divided into three steps - Defining *Test Case*, Running *Test Cases* and *Monitoring*. We discuss these three steps in regards to GPF and Sulley.

4.1.1 Define Test Case

A test case consists of codes/inputs that tries to define variables that can be used on the target – this include the type of input, the size of input, the speed of the input and more. For example, if we write a test case to test a HTML webpage's security. We can define text boundaries that can be input into the webpage's form from 0 – infinity. Therefore when the test is performed, the fuzzer will try to input '0' into the form, and it will then input '1','2','3'....'10000000000'. There is a possibility that a value, maybe an infinity value will crash the HTML form's request. A "smarter" approach is by using *Boundary Value Analysis*, as described in *chapter 3.3.2*.

There are cases where a test case definition is not needed. This happens when a tester chooses to test the target simply by randomly fuzzing the target. In GPF, there is an option to simply fuzz the target without even defining/writing a test case. In this instance, the fuzzing will be entirely random. In GPF, it is called *Pure Fuzz*. Pure Fuzz is performed in *chapter 6.2.1*.

A second option is to use existing data packets as test cases. These data packets are used as test cases and fuzzing is performed based on those data packets structure. For instance, using existing data packets of IEC 60870-5-104, we try to perform a buffer overflow fuzzing exercise. In Sulley, d*efining test cases* involve *Building Requests*. Essentially, *Building Request* is to write a test case based on protocol specifications. For example, the instance of fuzzing a HTTP GET "*GET /index.html HTTP/1.1*" is used :

```
from sulley import *
s_initialize("HTTP BASIC")

s_group("verbs", values=["GET", "POST"])
if s_block_start("body", group="verbs"):
        s_delim(" ")
        s_delim("/")
        s_string("index.html")
        s_delim(" ")
        s_string("HTTP")
        s_delim("/")
        s_string("1")
        s_delim(".")
        s_string("1")
        s_static("\r\n\r\n")
s_block_end()
```

Figure 8: Code Snippet for fuzzing HTTP GET

We imported all the methods under Sulley folder using **import *** statement. Then the name of the request is defined as **"HTTP BASIC"** using **s_initialize** method. Now we can start adding blocks and data necessary to build HTTP request. **s_group** method allows us to connect a block to a group primitive to specify that the block should cycle through all possible mutations for each value within the group. In our case we used **"GET"** and **"POST"**. The original http.py contains more values like **"HEAD"**, **"TRACE"** etc. New blocks are defined and opened with **s_block_start()** and closed with **s_block_end()**. Delimiters and strings are defined using **s_delim** and **s_string** method. Delimiter mutation includes repetition, substitution and exclusion. Finally **s_static** method is used to append two newlines after each http request according to protocol specifications. **s_static** method adds static non mutating value. [32]

4.1.2 Run Test Case

After specifying the test case, we fuzz the target with the test case.

4.1.3 Monitor Process

Monitoring process achieves two targets.

- Determine if vulnerability has been discovered by testing if the target has crashed, or produce other anomalies.

- Record down the test that causes the vulnerability in the hope of reproducing them.

Sulley has its internal network monitor. It helps to make sure that the tests are all being captured. GPF does not have a network monitor. We will have to rely on *TCPDump[33]* or *Wireshark[34]* to achieve that. Even then, we still have to search for the right packet that crashed the device.

TCPDump and *Wireshark* are both network packet capture softwares.

4.2 IEC Fuzz Consideration

There are a few ways that might improve the fuzzing of IEC 60870-5-104 protocol. The steps are:

- Make sure that the APCI's *length field* is same as the real APDU length.

- Focus on the application layer of IEC *(forgoing TCP/IP).*

- Start with fuzzing header, then followed by fuzzing header and payload.

In our fuzzing test we have performed four tests.

- GPF's ***random fuzz.*** It is called Pure Fuzz. There is no pattern in the fuzzing
- GPF's ***buffer overflow****.* We use an IEC file sample file as the base file, and perform buffer overflow.
- Sulley **Fuzzing for IEC heade**r.
- Sulley **Fuzzing for IEC header and payload**.

5 Conclusion

In this project we have discussed the weaknesses of SCADA. Implementing open standards protocol such as IEC60870-5-104 will be good for the standardization of industry, but the lack of security concern will undermine SCADA in the long run.

We have tested target devices that are implemented in SCADA environment. We have implemented three kinds of attacks – fuzzing which is targeted at the new weaknesses, especially in code-level handling. Secondly, we tried TCP/IP based attacks and thirdly scanning of target devices.

We concluded that fuzzing is long and exhausive and do not always yield results. It is, however, good at discovering new vulnerabilities. Current vulnerability scanners and web scanners are effective in finding existing weakness in their database.

This report copy is the public version, the original version that shows the test finding can only be viewed upon signing a Non-Disclosure Agreement with ABB.

The thesis "**Testing and Exploring Vulnerabilities of the Applications Implementing IEC 60870-5-104 Protocol**" has been a fruitful experience. We have learned many issues with SCADA, understood fuzzing, and carried out many exciting experiments.

Bibliography

1: ABM Omar Faruk; Zi Bin Cheah, Identifying and Responding to External Threats in a PCS Network - NTNU Academic Project,, December 2007

2: Pollet J. , Developing a solid SCADA security strategy, Sensors for Industry Conference, 2002. 2nd ISA/IEEE, Nov. 2002

3: COM600 Station Automation Series COM600 3.1 User Guide, http://library.abb.com/global/scot/scot229.nsf/veritydisplay/5f258baf94795608c12573b8002d0c0d/$File/COM600_usg_756125_ENd.pdf

4: United States Government Accountability Office, Critical Infrastructure Protection - Multiple Efforts to Secure Control Systems Are Underway, but Challenges Remain,www.gao.gov/new.items/d08119t.pdf, 2007

5: Tracking the blackout bug, http://www.securityfocus.com/news/8412

6: MuDynamics, The Business Case and Return on Investment for Deploying the Mu-4000 SecurityAnalysisPlatform,http://www.mudynamics.com/resources/collaterals_noreg/tco_roi.pdf, 2008

7: Patrick P. Tsang; Sean W. Smith, YASIR: A Low-Latency, High-Integrity Security Retrofit for Legacy SCADA Systems , Conference Proceedings, 23rd International Information Security Conference (SEC 2008), September 2007

8: Jan Tore Sørensen, Security in Industrial Networks - NTNU Masters Thesis ,, 2007

9: Edward Chikuni; Maxwell Dondo, Investing the Security of Power System SCADA, Conference proceedings, AFRICON, Sept. 2007

10: Windows OS Security: Analysis of Buffer Overflow Attacks, www.windowsecurity.com/articles/Analysis_of_Buffer_Overflow_Attacks.html

11: Format String Attack, http://www.owasp.org/index.php/Format_string_attack

12: Gordon Clarke; Deon Reynders; Edwin Wright, Practical Modern SCADA Protocols, 2003

13: Deon Reynders; Steve Mackay; Edwin Wright MIPENZ; Steve Mackay, Practical Industrial Data Communications, http://www.amazon.com/Practical-Industrial-Data-Communications-Techniques/dp/0750663952,2004

14: International Electronical Commision, Telecontrol Equipment and Systems - Part 5-104:

Transmission Protocol - Network access for IEC 60870-5-101 using standard transport profiles, 2006

15: DNP User Groups, DNP3 Specification Volume 4 Data Link Layer Version 2.0.1, 2007

16: Fuzzing Software, www.fuzzing.org/fuzzing-software

17: GPF Fuzzer Homepage, http://www.vdalabs.com/tools/efs_gpf.html

18: Sulley Fuzzer Homepage, http://www.fuzzing.org/category/sulley/

19: Testing via Boundary Value Analysis, http://blogs.ittoolbox.com/eai/implementation/archives/testing-via-boundary-value-analysis-17126

20: Remote Terminal Unit RTU560, http://library.abb.com/GLOBAL/SCOT/scot258.nsf/VerityDisplay/BB6F87E103E20D49C125722E0078920E/$File/X1982Sales_brochures_RTU560_271006.pdf

21: Station Computer COM615, http://www.abb.com/Product/seitp328/3cc7d0b2a99f2f40c1257188002facd2.aspx

22: Nessus Download, http://www.nessus.org/nessus/

23: Top 100 Network Security Tools, http://sectools.org/

24: Retina Vulnerability Scanner, http://www.eeye.com/html/Products/Retina/index.html

25: Top 10 Vulnerability Scanners, http://sectools.org/vuln-scanners.html.

26: DARPAnet, RFC 793 : Transmission Control Protocol, www.faqs.org/rfcs/rfc793.html, 1981

27: Nikto HTTP Server Scanner, http://www.cirt.net/nikto2

28: A.A. Creery; E.J. Byres, Industrial cybersecurity for a power system and SCADA networks - Be secure, Industry Applications Magazine, IEEE , vol.13, no.4, July-Aug. 2007

29: ANSI/ISA, ANSI/ISA-TR99.00.02-2004 Integrating Electronic Security into the Manufacturing and Control Systems Environment, http://www.isa.org/Template.cfm?Section=Shop_ISA&Template=/Ecommerce/ProductDisplay.cfm&Productid=7380, 2004

30: TCP Service on Port 2877, http://www.auditmypc.com/port/tcp-port-2877.asp

31: Sulley Fuzzer Tutorial, http://folk.ntnu.no/faruk/thesis/sulley -tutorial-omar.pdf

32: TCPDump Network Packet Capture, http://www.tcpdump.org/

33: Wireshark Network Scanner, http://www.wireshark.org/

34: TCPReplay Network Packet Replay, http://tcpreplay.synfin.net/trac/

35: Mu-4000 Product Overview, http://www.mudynamics.com/products/overview.html

36: BackTrack Linux Operating System, http://www.remote-exploit.org/backtrack.html

37: CERT: Secure Coding, http://www.cert.org/secure-coding/

38: DNP3 Secure Podcast, http://www.digitalbond.com/index.php/2007/06/10/secure-dnp3-podcast/

39: D. G. Fink; H. W. Beaty, Standard Handbook for Electrical Engineers.New York, , 1978

40: Munir Majdalawieh; Francesco Parisi -Presicce; Duminda Wijesekea, Distributed Network Protocol Security (DNPSec) security framework, Conference Proceedings, 21st Annual Computer Security Applications Conference, December 2005

41: Security Standards from WG15 in IEC TC57, www.intelligrid.info/IntelliGrid_Architecture/Technology_Analysis/Anl_Security_Protocol.htm

42: Frances Cleveland, IEC TC57 WG15 Security Standard,https://www.pcsforum.org/library/files/1129579675-White_Paper_on_Security_Standards_in_IEC_TC57_ver_5.pdf, 2005

43: B. Harris; R. Hunt, TCP/IP security threats and attack methods , Computer Communications, Volume 22, Number 10, 1999

44: S.M. Bellovin, Security Problems in the TCP/IP Protocol Suite , Publication, ACM SIGCOMM Computer Communication Review, April 1989

45: Maria B. Line; Martin Gilje Jaatun; Zi Bin Cheah; A. B. M. Omar Faruk; Håvard Husevåg Garnes; Petter Wedum, Penetration Testing of OPC as part of Process Control Systems, in Proceedings of the 5th International Conference on Ubiquitous Intelligence in Computing, Oslo, Norway, Springer LNCS 5061, 2008