SCADA SECURITY ASSESSMENT UNDER CYBER ATTACKS

BY

ASEM ABDO ESMAIL GHALEB

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

SECURITY AND INFORMATION ASSURANCE

DECEMBER 2015

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN 31261, SAUDI ARABIA

DEANSHIP OF GRADUATE STUDIES

This thesis, written by **ASEM ABDO ESMAIL GHALEB** under the direction of his thesis advisor and approved by his thesis committee, has been presented to and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN SECURITY AND INFORMATION ASSURANCE**.

Thesis Committee

[illegible]

Dr. Sami Zhioua (Advisor)

[illegible]

Dr. Ahmad Almulhem (Member)

[illegible]

Dr. Sami Elferik (Member)

[illegible]

Dr. Abdulaziz Alkhoraidly (Member)

[illegible]

Dr. Jameleddine Hassine (Member)

[illegible]

Dr. Abdulaziz Alkhoraidly
Department Chairman

[illegible]

Dr. Salam A. Zummo
Dean of Graduate Studies

عمادة الدراسات العليا
DEANSHIP OF GRADUATE STUDIES

11/1/16

Date

Dedicated to
the soul of my father,
who encouraged me to have high expectations and to fight hard for what I believe and for my dreams to keep coming true. Dad, I feel you are always with me supporting and guiding.
To my beloved mother, for her prayers to me.

ACKNOWLEDGMENTS

First and foremost, my deepest thanks and gratitude go to God for providing me with the means and perseverance to complete this journey. Without His will and generosity, none of this or any other accomplishment would have been possible.

I would like to express my gratitude to my thesis advisor Dr. Sami Zhioua for inspiring me with his passion for exploring new technologies and new research areas. I would like to thank him for ceaselessly providing me with feedback, discussion and guidance and for spending hours on my drafts. My heartfelt thanks are also to my thesis committee members, Dr. Ahmad Almulhem, Dr. Sami Elferik, Dr.Abdulaziz Alkhoraidly and Dr. Jameleddine Hassine for their valuable comments and advice.

Moreover, special appreciation is due to my parents, brothers and sisters, friends, and all who have helped me along the way I can not mention them by name, but they know who they are.

My sincerest gratitude is extended to Yemenmobile for giving me the opportunity to pursue my master and is also to the KFUPM faculty, staff and colleagues.

Thank you all...

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

SCADA	Supervisory Control and Data Acquisition
PLC	Programmable Logic Controller
RTU	Remote Terminal Unit
HLA	High Level Architecture
OMNeT++	Optical Micro-Networks Plus Plus
IDE	Integrated Development Environment
HMI	Human Machine Interface
MTU	Master Terminal Unit
OSI	Open Systems Interconnection
TCP/IP	Transmission Control Protocol/Internet Protocol
IED	Intelligent Electronic Device
Profibus	Process Field Bus
DoS	Denial-of-Service
IT	Information Technology

IDS	Intrusion Detection system
ICS	Industrial Control Systems
MiTM	Man-In-The-Middle

THESIS ABSTRACT

NAME:	Asem Abdo Esmail Ghaleb
TITLE OF STUDY:	SCADA Security Assessment under Cyber Attacks
MAJOR FIELD:	Security and Information Assurance
DATE OF DEGREE:	December 2015

Supervisory Control and Data Acquisition (SCADA) systems are responsible of controlling and monitoring industrial processes and critical infrastructures, such as electricity generation, gas production, and water distribution. In the few past years, several security incidents have been reported on SCADA systems. The consequences of these attacks ranged from small operations disturbance to loss of human lives. Therefore, there is an urgent need to carry out a security analysis of SCADA systems and to design appropriate security solutions. Security testing on live SCADA systems, however, is not practical due to the difficulty and the cost related to the implementation. In addition, evaluation of some security vulnerabilities, such as Denial of Service (DoS) attacks, may lead to the delay or interruption of SCADA services, which is not acceptable for real-time monitoring of critical infrastructures and operating systems. Therefore, such necessary solutions require *extensive testing and validation prior to their implementation. In this thesis, a SCADA simulation environment for testing and evaluating SCADA attacks and mitigation techniques is presented. The simulation environment is designed and developed in such way to allow hybrid architectures (involving simulated as well as physical components). Two realistic SCADA configurations are designed using our proposed environment, namely , water tanks control and smart grid systems. In addition, the testbed allowed us to test successfully a set of serious network attacks on a physical PLC including replay and Man-In-The-Middle (MiTM) attacks.*

ملخص الرسالة

الاسم : عاصم عبده اسماعيل غالب

عنوان الرسالة : تقييم أمنية نظم سكادا تحت الهجمات الإلكترونية

التخصص : أمن وضمان المعلومات

تاريخ التخرج : ديسمبر 2015

أنظمة التحكم الاشرافي وتجميع البيانات(سكادا)، هي الأنظمة المسؤولة في التحكم ومراقبة العمليات الصناعية والبنى التحتية الحساسة، مثل توليد الكهرباء، انتاج الغاز وتوزيع المياه. في السنوات القليلة الماضية، تم تسجيل العديد من الحوادث الأمنية في أنظمة سكادا. وتراوحت عواقب هذه الحوادث مابين خلل في اداء العمليات الصناعية إلى خسائر في الأرواح. لذلك، هناك حاجة ملحة لإجراء تحليل لأمنية نظم سكادا وتصميم الحلول الأمنية المناسبة. ومع ذلك، ليس عملياً اجراء تجارب أمنية على أنظمة سكادا اثناء عملها نتيجة للصعوبة والتكلفة المرتبطة بالتنفيذ. لذلك، من الضروري إجراء اختبارات مكثفة للحلول الأمنية والتحقق منها قبل تطبيقها.

في هذه الأطروحة، سيتم عرض بيئة محاكاة لغرض اختبار الهجمات الأمنية على أنظمة سكادا وكذلك تقنيات كشف هذه الهجمات. حيث تم تصميم وتطوير بيئة المحاكاة بحيث تتيح بناء أنظمة هجينة تجمع بين مكونات مادية واخرى تم محاكاتها.

تم تصميم نموذجين لأنظمة سكادا باستخدام بيئة المحاكاة المقترحة، وهي نظام التحكم في خزانات توزيع المياة. وشبكة الكهرباء الذكية. بالإضافة إلى ذلك، تم استخدام هذه الأنظمة لإختبار مجموعة من الهجمات الأمنية على اجهزة التحكم (PLCs)

CHAPTER 1

INTRODUCTION

This chapter introduces our motivation to pursue this research. It highlights the current security issues affecting SCADA systems and also gives an overview of the main ideas developed in this thesis. It goes further presenting how the introduced concepts provide a better environment for evaluating the security of SCADA systems

1.1 Motivation

SCADA systems are widely used to monitor and supervise critical infrastructure and industrial processes. Such systems are vital for our society and responsible for providing such many services on different fields. They are used in electricity and water distribution, oil and gas production, managing railways and controlling street traffic and in processing and recycling our waste, etc. Therefore, security of such systems cannot be overstated since any interruption or disruption to such systems can have a direct harmful effect on our live.

In the last few years, the number of cyber attacks targeting SCADA systems increased dramatically. According to Dell security annual threat report released in 2015 [1], the number of cyber attacks against SCADA systems doubled in 2014 compared with the number of cyber attacks against SCADA systems in 2013, see figure 1.1. "Since companies are only required to report data breaches that involve personal or payment information, SCADA attacks often go unreported," said Patrick Sweeney, executive director, Dell Security. "This lack of information sharing combined with an aging industrial machinery infrastructure presents huge security challenges that will continue to grow in the coming months and years."

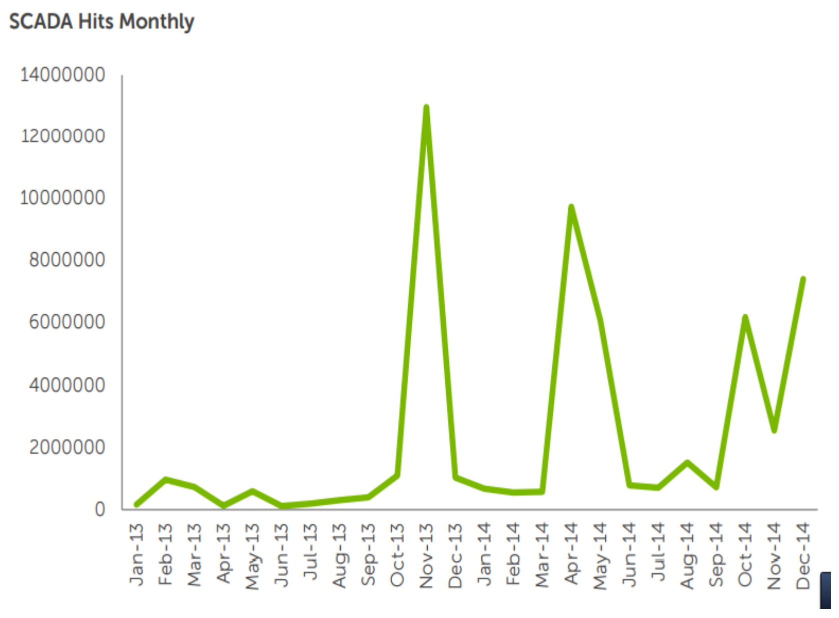


Figure 1.1: SCADA hits monthly [1]

Historically, SCADA components were special-purpose embedded devices connected through a proprietary communication bus. Vendors would typically offer turn key solutions, which would be incompatible with competitors systems. Security was not a main concern in the design of these systems, instead major concerns regarded real-time processing, and event notification [2]. Despite the lack of security features, SCADA vendors and operators believed they could rely on two forms of protection. The first is employing an air gap, that is, a SCADA network would be physically isolated from any other networks, thus making it harder for an attacker to gain access. Secondly, they relied on security through obscurity, that is, vendors and operators believed that very little, if any, information was publicly available about their environments, and this lack of information made their systems secure. Security concerns focused on restricting access to unmanned field networks and on preventing configuration mistakes [3].

A number of SCADA systems are still using legacy devices while they are directly or indirectly connected to the Internet. This is because sharing of real-time information with the business operations has become a necessity for improving efficiency, minimizing costs, and maximizing profits. This, however, exposes SCADA systems to various types of exploitation. Therefore, it is important to identify common attacks and develop security solutions tailored to SCADA systems. However, to do so, it is impractical to evaluate security solutions on industrial systems while in operation because of the difficulty and the cost of implementing standalone SCADA systems, in addition to the potential risk of failure and downtime of SCADA services that may be caused during implementation of security solutions. Therefore, the key problem of developing and improving particular security solutions for SCADA systems is the lack of suitable modeling and testing tools to evaluate those solutions prior to their adoption and implementation. Moreover, most of the SCADA testbeds [4, 5, 6] are proprietary used only by researchers within organizations and the software is not shared for public use. In general, to come up with simulation environment for SCADA systems that can be used for evaluating the security issues, a set of requirements has to be satisfied. First, the simulator should be composed of simple, flexible, and reusable components. Second, the simulator should be extensible and supports easy interconnection with other simulators and/or real modules.

1.2 Thesis Contributions

This thesis provides the following contributions to the field of SCADA systems and their security.

- Design and implementation of SCADA security simulation environment with a key benefit of building SCADA security testbeds that allows the simulation of various SCADA components, security attacks, and security solutions with ability to interface with real physical devices.

- Security analysis of the network communication between Simatic Programmable Logic Controllers (PLCs) and the engineering stations in charge of setting up and configuring them.

- Implementing and carrying out a set of serious network attacks targeting Simatic PLC leading to serious compromise of the PLC.

1.3 Organization of the Thesis

The thesis consists of eight chapters. Chapters 5-7 represent the main contributions of the thesis. The other chapters are respectively the introduction, SCADA systems, SCADA security, SCADA simulation related work and finally the conclusion and future work chapter. The thesis is organized as follows.

Chapter 2 gives introduction about SCADA systems, their architecture, main components and some of the common protocols used by SCADA systems. Chapter 3 provides overview about SCADA security through describing how it is different from IT security, attacks targeting SCADA systems and finally the security solutions for SCADA systems. Literature review of the related work is provided in chapter 4.

Chapter 5 describes implementation details of the introduced simulation environment. This chapter provides overview of the tools and techniques used to implement the simulation environment, the multiple implemented components that shape the simulation environment, the implemented industrial protocols and implemented interfaces that connect the simulation environment with the real external devices.

In order to evaluate the introduced simulation environment, Chapter 6 presents several use cases. It shows how the simulation environment can be used along with available physical devices to construct security testbeds to analyze the security of SCADA systems as well as evaluating security solutions. In chapter 7, one of the testbed presented in Chapter 6, that includes real common PLC, is used to carry out three network attacks leading to serious compromise of typical PLCs. In addition, simulation of some attacks targeting SCADA systems would be described.

Finally, chapter 8 concludes the thesis by summarizing the presented work and provides recommendations that can be considered for pursuing a future work.

CHAPTER 2

SCADA SYSTEMS

2.1 SCADA Architecture

SCADA is a system that operates over communication channels for monitoring and controlling industrial and manufacturing processes and facilities, in addition to providing control over remote equipment. It is a type of Industrial Control Systems (ICS) that monitor and control industrial processes and it is distinguished from other ICS systems in that it is large-scale processes including multiple sites and operating over large distances. SCADA system collects and analyzes realtime data from remote equipment such as pumps, valves, etc. and provides overall remote actuation and control. Today, SCADA systems reached a high level of domination that most of US national infrastructures depend on SCADA systems to a high degree [7].

SCADA systems can be relatively simple, a single circuit that notifies you of one event, such as self-contained SCADA systems that are built for a given application, and the ones that monitor the environmental conditions of a small building, or more complex, such as a system that monitor all the activity in a nuclear plant.

SCADA systems consist of hardware and software components as depicted in figure 2.1. The hardware components gather data and push it into a computer that has SCADA software. The SCADA software then processes and presents this data in a timely manner and makes the appropriate controlling decisions if necessary. All events are recorded by SCADA into log files stored on a database or sent to a printer. Alarms are raised when conditions become risky or hazardous. These operations are performed by different kinds of SCADA components [8].

The sensors (either digital or analogue) collect data from the managed system or equipment. Remote Terminal Units (RTUs) deployed in the field at specific sites and locations gather reports from sensors and deliver commands to control relays. The collected data is moved through a communications network to the master units, these are larger computers that serve as the central processor for the SCADA system to automatically regulate the managed system in response to the sensors inputs [8].

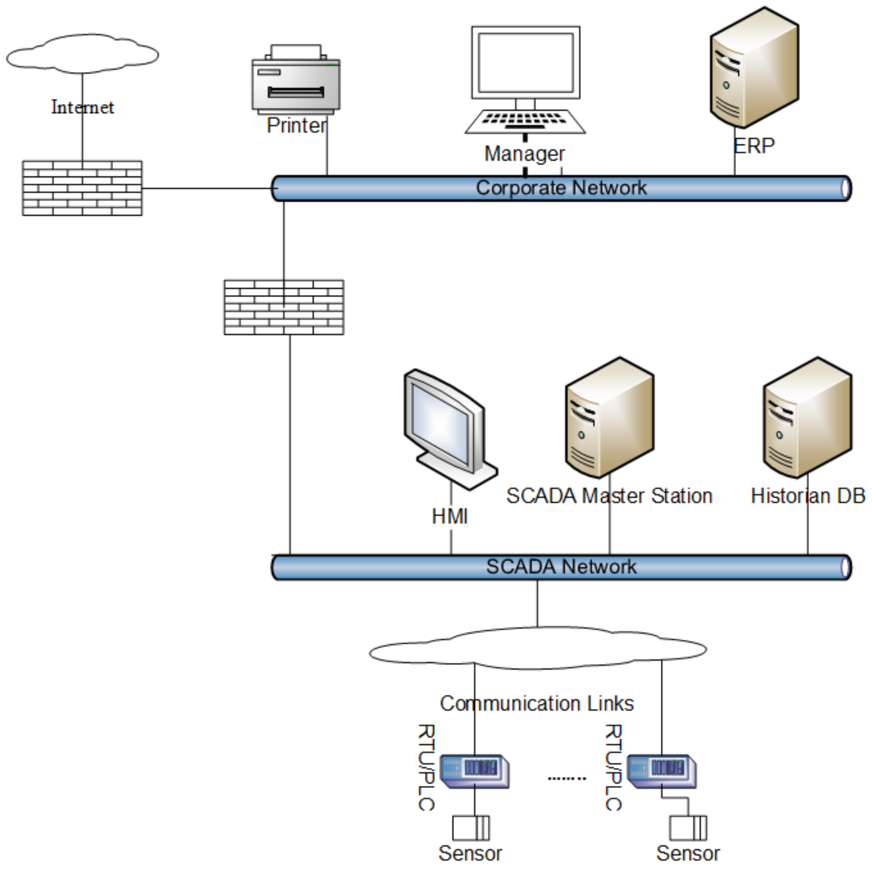


Figure 2.1: Typical SCADA Architecture

2.2 SCADA Components

A typical deployment of a SCADA system contains the following components:

- **Human Machine Interface (HMI):**

  HMI is the means through which data collected and stored is presented to human operators in understandable and comprehensible forms. This includes easy-to-understand screen layouts, detailed schematics, pictorial representation, and animations representing the running states and health of the field equipment and machines. In addition, HMIs enable human operators to interact with the processes through a touch screen, keyboard or both. With HMI, human operators interact with SCADA systems in a simple, clear and easy to understand way. Recently, HMIs have become web-based applications, consequently, users interact with them through web browsers.

- **Master Terminal Unit (MTU):**

  MTU is the repository of the real-time data collected from the remote terminal units and transferred to it through SCADA network. Equivalent to a master unit in a master/ slave architecture. SCADA software on the master station must be able to collect and retrieve data values from the RTUs, store, process and present it to the operator through the HMI and transmits control signals to the remote site. Operator HMIs are connected to the MTU by a LAN/WAN so that the viewing screens and associated data can be displayed for the operators [8]. The processing may include unit conversion, recording or cataloging into tables etc.

- **Remote Terminal Unit (RTU):**

  Remote Terminal Unit (RTU) functions as a slave in the master/slave architecture. Collects and processes I/O in an intelligent manner. The Inputs data from sensors, switches and transmitters is read by RTU and then transmits the data to the MTU in a format understood by SCADA system. The

RTU also converts the SCADA system digital control signals into the suitable form, discrete or analog, understood by the device under control [7]. RTUs are different from PLCs in that they are better suited for wireless communications. This makes RTUs ideal in renewable energy applications like wind and solar farms. RTUs commonly use the Modbus (or some other) protocol to connect to SCADA and HMI software. The data rate between the RTU and controlled device is relatively high[9].

- **Programmable Logic Controller (PLC):**

  Programmable Logic Controller (PLC) is a microprocessor into which a program is fed so that it can control several functions in industrial processes. Over the years, the functionality or the job of the PLC has developed to involve process control, relay control, distributed systems control, motion control, and networking. They were initially invented to replace the electromagnetic relays and the cumbersome wirings in a control circuit. In some modern PLCs, data handling, processing power, storage, and communication capabilities are , to some extent, equivalent to desktop computers. The process logic or sequence of operation is executed as per a software or control logic program. However, PLC differs from a desktop computer. Unlike computers, PLCs have multiple number of inputs and outputs, and are designed for performing a strict and bumpy operations under extreme industrial conditions. They operate under higher or increased temperature ranges, have resistance to vibration and impact, and have immunity to electrical noise. PLCs can be (re)configured using proprietary software installed on a standard computer (typically with Microsoft Windows OS). Reconfiguring the PLC consists in changing the control system software, known also as the programming layer of the PLC. A very common example of the configuration software is the Siemens Simatic Step 7 [10] for Simatic controllers. The software allows engineers to perform three main tasks: (1) write the graphical ladder logic code, (2) compile it to machine code for execution and (3) upload the compiled code to the device.

- **Communication Means:**

  Communication media/methods between the central host computer servers and the remote field-based controllers through which data can be transferred to and from different sites [8]. The communication can be established through wired or wireless networks, Internet or the public telephone network [9].

2.3 SCADA Protocols

In order to establish a communication between any entities, a protocol for that communication has to be defined. A protocol states the form of the messages and the rules for the exchange of messages. High-level models are used to state where the protocols are applied and to break down the functions needed for the sake of sending and receiving messages. The layered architecture model is one of those models widely used in which the elements necessary to establish a communication are separated into layers connected together through interfaces. The Open Systems Interconnection (OSI) and the Transmission Control Protocol/Internet Protocol (TCP/IP) are the two most widely used layered communication models [9].

The OSI model is constructed of 7 layers where each layer uses specific protocols to define its function. In this model, data pass form higher-to-lower level layers such that data is encapsulated by the next layer while passing through layers on the sender node. For instance, a packet of data from a higher layer would be encapsulated in the next lower layer by adding header information. On the receiving node, the reverse process happens, packets pass from lower-to-higher level layers. The encapsulation is stripped from messages while they move from down-to-up layers. The TCP/IP model is constructed of 4 layers and the various capabilities of the Internet are based on the TCP/IP protocols.

Protocols of SCADA systems evolved from propriety hardware and software designed specifically for SCADA systems. The protocols were developed out of necessity to serve the burgeoning market for computer application in real time control situation. Then SCADA protocols inserted versions of Internet and LAN technologies in an effort to take advantages of new networking developments. This resulted in some standardization commonly used in IT environment [9].

During the past three decades, hundreds (150 - 200) of these protocols have been developed for communications based on both serial, LAN and WAN in a wide variety of industries including petrochemical and electrical generation or distribution [11]. Of these, approximately 10 protocols currently dominate the industrial marketplace and include systems such as MODBUS, DNP3, EtherNET/IP, PROFIBUS and Foundation Fieldbus [12]. Following sections would provide a brief description of the commonly used protocols for SCADA systems.

- **MODBUS Protocol**

  MODBUS protocol was developed by Modicon and has become defacto standard communication protocol. Modbus protocol is positioned in the application layer of OSI model and supports client-server communications between

PLCs and other SCADA system components. It defines means for connecting industrial devices, such as PLCs, RTUs, MTU, etc. together and for detecting reporting and errors. MODBUS communication use a master-slave technique in which only the master device can send commands(transactions) called queries. The other devices(the slaves) just respond by providing the requested data to the master or by performing the requested action [9, 13]. Modbus protocol is used in industrial environment for the reasons that it was designed considering industrial applications, it is an open standard protocol and royalty-free, and it is easily deployed and maintained. The development and update of Modbus protocol has been managed by the Modbus Organization [14].

- **DNP3 Protocol**

  DNP3 is an open SCADA protocol used for communication between components in process automation systems. It is widely used by utilities such as water and electricity companies. It plays a crucial role in SCADA systems for the exchange of data and control instructions between master station, RTUs and Intelligent Electronic Devices (IEDs). The typical commands issued by master station are ”open a valve”,”start a motor”, and ”provide data on a particular control station”. The out-stations also provide the master station with information such as status of circuit breaker, pressures, analog signals representing such items as temperatures or powers. DNP3 has also been adapted to Internet technologies by using TCP/IP for exchange of DNP3 messages [9, 15].

- **Profibus Protocol**

  Process Field Bus (Profibus) is a open standard for fieldbus communication in time-critical control and data acquisition applications. It was promoted by German department of education and research and then used by Siemens. Since Profibus is an open standard, it can accommodate devices from different manufacturers. It resides at the application, data link, and physical layers of the OSI model. Profibus has advanced through a handful of revisions and as a result, there are three versions of Profibus: Profibus Fieldbus Message Specification (FMS), Profibus Process Automation (PA), and Profibus Factory Automation (Decentralized Peripherals DP) [9].

CHAPTER 3

SCADA SECURITY

In the recent years, there has been a noticed growth in the number of incidents against SCADA systems. According to a recent report, which was published in 2013 by ICS-CERT (U.S. Department of Homeland Security) [16], the number of intrusions, attacks, scanning, and footprinting activities against the critical infrastructure in U.S. is growing continuously. Just in 2013, ICS-CERT reported and analyzed more than 250 incidents, particularly in the networks of industrial companies. The report stated that, "because reporting of cyber incidents is done on a voluntary basis, it is estimated that many more incidents are occurring but are not reported." Moreover, large number of incidents are not detected due to the lack of sufficient logging and detection capabilities [17]. Another report published by Dell security in 2015 showed that the number of SCADA cyber attacks doubled in 2014 [1]. Therefore, as these systems began to change and became more interconnected, operators, however, began to recognize that cyber security was a real concern that must be addressed in order to maintain the safety and reliability of process control. This chapter highlights SCADA security by describing the difference between SCADA systems and Information Technology (IT) systems security and what are the common attacks targeting SCADA systems and finally, some of the security solutions used for securing SCADA systems would be stated.

3.1 SCADA vs. IT Security

SCADA systems have several characteristics that differentiate them from IT systems in terms of operational priorities and risks. The control components of SCADA systems are optimized to provide real-time performance and reliable service with a reasonable cost. There are little computing cycles and extra memory sufficient to execute other functions such that related to performing security tasks and SCADA systems use unconventional operating systems and software. In addition, the primary and most important goal in IT is to protect the central servers and not the clients. Contrary to IT systems, in SCADA systems, the edge client devices such as PLCs are the backbone of SCADA systems and are of great importance compared to central edges such as data historian servers [18].

The security of information system was not inherited in SCADA protocols. At the time when SCADA protocols were developed, SCADA systems were isolated from the outside world and were working in closed environments. Today, SCADA systems are connected to corporate IT networks and they are using protocols that are targets for attacks in the IT world.

One big challenge that SCADA systems face is the inability to deploy vulnerability countermeasures, that are usually used with IT networks. In IT networks, Antivirus and encryption are commonly employed. Network administrators perform penetration testing and auditing of information security. Employees are trained on information security and hence have an increased awareness. An upgrading or replacing of equipment is done every couple of years. In addition, software is updated and patched on regular basis. On the other hand, none or just a few of these countermeasures are considered in SCADA networks. Antivirus are difficult to be employed in SCADA networks because delays can not be tolerated. The performance of the network is affected negatively with encryption. Penetration testing is rarely done due to the high potentiality of disturbing the control system. Software is patched and updated on infrequent basis because of the need to careful planning and cooperation of the different component vendors. Equipment can stay for years without replacing or upgrading, running applications and operating systems with known vulnerabilities [9]. Table 3.1 summarizes characteristics of SCADA systems vs. IT systems.

IT Systems	**SCADA Systems**
Data loss caused by disturbance can be recovered	Disturbance may lead to disastrous consequences
Delays can be accepted	Real-time actions & responses
Tolerate rebooting & crashes	Must be running 24/7
Antivirus commonly used	Inadmissible delays caused by antivirus
High security awareness and training	Lack of security awareness and training
Encryption commonly used	Encryption rarely used
Regular penetration testing	Rare penetration testing with high care
Regular application of patches	Patches are applied infrequently, carefully and with vendors' cooperation
Replacement of equipment per three-to-five years	Equipment run for decades without replacement
Regular information security audits	Rare information security audits

Table 3.1: IT systems vs. SCADA systems.

3.2 SCADA Attacks

Increasingly, SCADA systems are connected to corporate networks, in order to maximize benefits by enabling leaders to track and control real-time production, watch changes in the production and react accordingly. Moreover, Ethernet, TCP/IP, Wireless technologies such as IEEE 802.x and Bluetooth have been adopted in SCADA systems. However, such interconnection exposes SCADA or industrial control systems to attacks. The control network can be penetrated by remote attackers without having any physical access by exploiting the gateways vulnerabilities between SCADA and corporate networks. Moreover, the recent trend for using commercial and open source software decreased the development and deployment costs. On the other side, the attackers require less knowledge about the operation of the control system than was required when proprietary protocols and hardware were prevalent [19].

Cyber attacks on SCADA systems can take a route through connections to Internet or enterprise networks or through connections with other networks such as connection to satellite and wireless networks. The most common attacks against SCADA systems can be classified as follows [18].

- Protocol vulnerabilities
- Network backdoors
- Field devices attacks
- Database attacks
- Man-in-the-middle attacks
- Time provision and synchronization attacks

With the successful penetration of SCADA systems, the following malicious acts can be performed [18, 8, 20]

- System shutdown through using Denial-of-Service (DoS)
- Compromise the function of RTUs/PLCs
- Get access to master stations and gain control of system by planting Malware
- Retrieve SCADA system passwords by log keystrokes from operators
- Gain access to SCADA systems
- Shut down the control devices such as PLCs using, for example, replay attack
- Send modified incorrect data to master stations by spoofing RTUs
- Preform wrong actions by spoofing control stations such as PLCs
- Disturb communications between operator stations and control stations

3.3 SCADA Security Solutions

With the security threats and challenges that face SCADA systems, there is an increased interest for techniques and tools that could be adopted in order to improve the security of SCADA systems. Figure 3.1 illustrates the typical defenses or security countermeasures employed in corporate/IT network and their relationship with the SCADA network. Recently, security mechanisms for SCADA systems have been adopted, many of these are used in IT security.

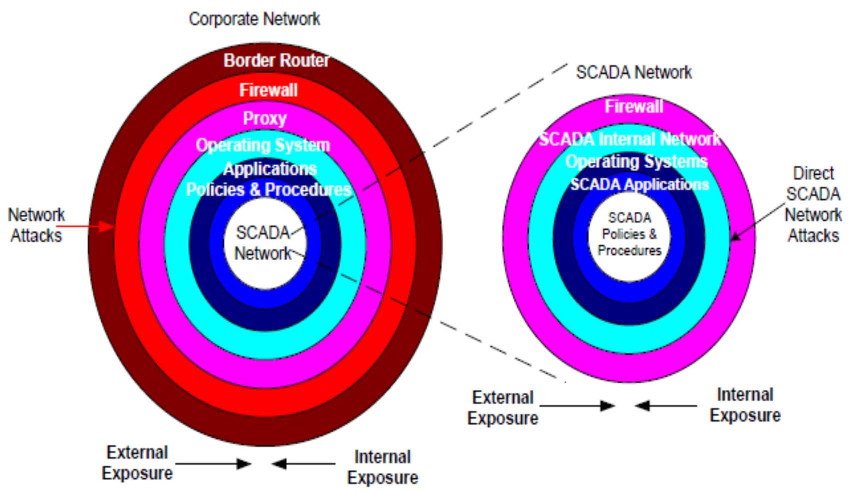


Figure 3.1: Security defenses relationship between IT and SCADA networks [8]

Following are a set of techniques that could be implemented in order to improve the cyber security of SCADA systems [9].

- Analyze SCADA network and its nodes for vulnerabilities
- Use network encryption, strong authentication and isolate SCADA network from unnecessary external connections
- Disable all unnecessary services
- Adopt firewalls compatible with SCADA protocols
- Install and configure Intrusion Detection systems (IDSs)
- Integrate patch management with SCADA systems
- Apply configuration management of SCADA network, software and hardware

- Develop and implement risk assessment, security audits and incidents response plans
- Conduct security awareness programs and training

CHAPTER 4

SCADA SIMULATION RELATED WORK

A number of research papers focused on developing simulation environments of SCADA systems. In this chapter, we present work related to developing simulation environments for the sake of investigating SCADA security issues. McDonald et al. [21] described a virtual control system environment developed at Sandia National Laboratories for investigating SCADA vulnerabilities in the filed of energy systems. As set of assumptions, that have to be made for developing hybrid models, have been discussed. Simulated RTUs interacted with simulated power systems (PowerWorld server) were used to represent the control system. While both simulated and real components were included in the cyber layer. However, the proposed environment is constrained on Power Systems and does not support a wide variety of physical processes.

Chabukswar et al. [22] concentrated on demonstrating the use of Command and Control Wind-Tunnel (C2WT), framework used more widely in research, with the aim to simulate DDOS-like attacks on a plant and its control system as well as to analyze the effects on different routers. The C2WT framework is based on High Level Architecture (HLA) and it was designed to facilitate the development of large-scale simulations. It uses the Generic Modeling Environment and employs model-based design techniques and graphical interface to allow integration of diverse simulation engines. The authors use the NetworkSim, which is based on OMNeT++ to simulate the communication protocols and the Simulink to model the domain specific processes. Moreover, they developed a Simulink function to synchronize the model with the Run-Time Infrastructure allowing Simulink to progress only when the RTI allows it. They use timed-stepped synchronization, while keeping an appropriate small time-step size in order to minimize event timing errors introduced by exchanging events between Simulink and HLA. However, the authors work and the used platform were designed mainly for power plant simulations and not typical SCADA systems.

Queiroz et al. [23] proposed a SCADA simulation tool (SCADASim) developed for SCADA security studies. Their objective was to examine the effect of attacks in real devices and applications by using a simulated environment. Attacks that are supported include denial of service, man in the middle, eavesdropping, and spoofing. They use the OMNET++ to simulate the network and they exploit the socket based integration of OMNET++ to allow the integration of the external devices using deployed gates. Finally, they deploy malicious attacks (denial of service and spoofing) scenarios to evaluate the framework, and they demonstrate how the attacks are affecting the process of using legitimate requests. However, the work was limited to simulated attacks, and it does not provide a framework to launch attacks from outside the simulation framework. In addition, the authors did not mention the possibility of integrating detection techniques. The hardware components were simulated using MATLAB/Simulink and no real physical hardware was mentioned except sensors and actuators.

The work by Chunlei, Lan, and Yiqi [24] proposes a reference architecture. It consists of several layers and components that represent the enterprise network, the OPC server and client, the SCADA protocols tester, the RTUs, the field sensors and actuators as well as the industrial infrastructure. Their prototype implementation targets the security analysis and assessment of SCADA systems. The architecture is extensible and adaptable and it is mainly based on NS2. Moreover, in order to allow the integration with real networks they exploit the capabilities of emulation feature of NS2. The latter has the ability to inject traffic from the simulator into a live network and to simulate a desired network between real applications in real-time. For the simulation framework, they use real PLC/RTUs and sensors/actuators, as well as industrial and open source systems for OPC client/server implementation. Finally, for the evaluation of the simulation framework, they implemented attack scenarios that compromise the security of SCADA system and they developed methods to analyze and assess the impact of these attacks. On the other hand, all SCADA components such as PLCs, RTUs are real components and only the enterprise network has been simulated. This doubts its ability to be used in tests that require large infrastructure.

Almalawi [25] proposed SCADA simulation framework (SCADAVT) for building a SCADA testbed based on virtualization. CORE emulator has been used to build the framework. The essential SCADA components such as protocols, I/O modules, and simulators of field devices have been integrated through the plug-in service available in CORE emulator. Moreover, in order to simulate water distribution systems, a server has been introduced with the use of the dynamic link library (DLL) of EPANET-a modeling tool for simulating water movement and quality behavior within pressurized pipe network. In addition, the simulated server can be used to simulate any topology of water network systems and can be manipulated by a custom TCP-based protocol. They presented a case study to show how the testbed can be used to monitor and control any automated processes. Two attacks (DDoS and integrity) have been described to demonstrate how attacks can disrupt supervised processes.

Mahoney and Gandhi [26] research has constructed an integrated framework for simulating control system and monitoring of regulatory compliance in near real-time. SCADASiM framework allows the reconstruction of SCADA network components at the abstraction level needed for the 7 monitoring of the system aspects. The authors have used Autonomous Component Architecture (ACA) to model SCADA network and in order to monitor the regularity compliance, their research involved using of a new language called ADACS.

CHAPTER 5

SCADA SECURITY TESTBED

This chapter presents implementation details of a simulation environment intended to study and analyze SCADA systems security. The system is called: SCADA-SST. First, an overview of the tools used to implement security environment is provided in addition to a discussion about why these tools in particular are chosen to develop the proposed environment. Then it proceeds to present the design and development details in the following sections.

5.1 OMNeT++

SCADA-SST environment has been implemented using OMNeT++ network simulator in combination with INET framework that contains all libraries needed to build communication network models and implements the most common Internet protocols, such as TCP, IP, UDP, MAC protocols, etc. [27]. OMNeT++ was chosen to build SCADA-SST for several reasons. First, this work focuses on TCP/IP protocol stack, which are included in OMNeT++. OMNeT++ is an open source and generic simulation engine and provides the ability to integrate with external real devices. In addition, OMNeT++ is commonly used in the research field and this has the benefit of building on top of others work.

OMNET++ is an extensible, modular, component-based and object-oriented discrete event simulation library and framework primarily for building network simulators written in C++ [28]. The main components of OMNET++ are the modules, which can be simple modules or compound modules grouping several simple modules together, which communicate with each other by exchanging messages. The modules communication can occur either directly through messages or through input and output gates as depicted in figure 5.1.

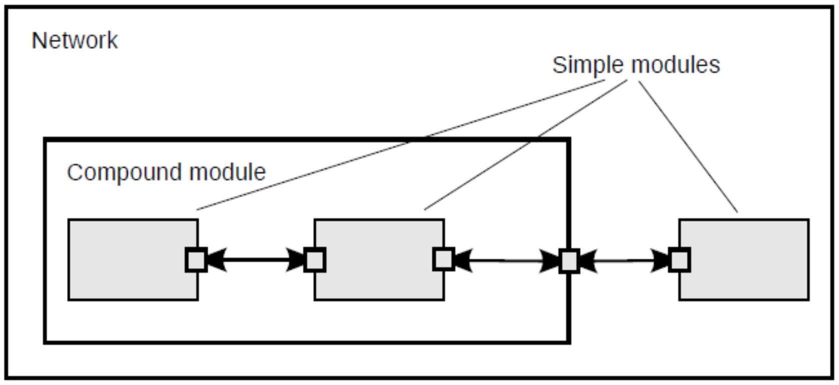


Figure 5.1: OMNeT++ modeling [29]

In OMNeT++, simple modules can be combined together in a hierarchy of levels making it possible to construct complex simulation components. Figure 5.2 shows an example of a compound module StandardHost constructed by combining several modules together.

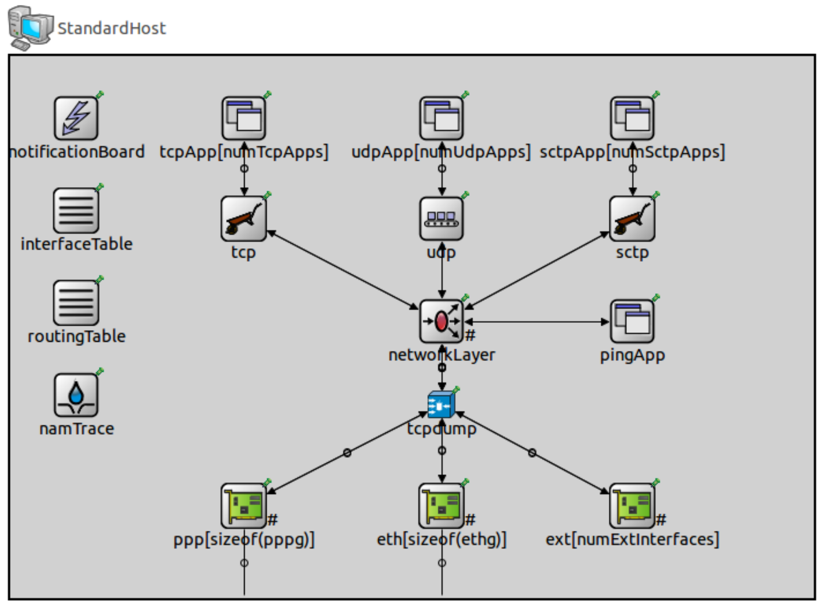


Figure 5.2: OMNeT++ compound module example

OMNeT++ comes with Integrated Development Environment (IDE) based on the Eclipse platform with extended new editors, views, wizards, and additional functionality for the sake of creating and configuring models, running patch executions, and evaluating the simulation results.

In OMNeT++, the general approach of simulation implementation (modeling) can be summarized in the following steps:

1. **Define the structure:**

   The simulated module structure and the various network topologies are defined using NED language. This can be done using a text editor or in the graphical editor of the eclipse-based OMNeT++ simulation IDE.

2. **Define the behavior:**

   The behavior of the module is programmed in C++ using the simulation kernel and class library.

3. **Define runtime parameters:**

   The OMNeT++ specific configuration and the module parameters are grouped in the *omnetpp.ini* configuration file which can describe several simulation runs with different parameters.

4. **Run the simulation:**

   After building the simulation, it can be run through either the command line batch or the interactive graphical user interface.

5. **Evaluate the simulation:**

   The simulation results are recorded in output vectors and scalar files. Those results can be processed or visualized using the appropriate tools.

5.2 Testbed Design

This section introduces the overall structure of SCADA-SST and its main components. Details of the implementation will be discussed subsequently in the following sections. Before starting with designing the proposed simulation environment that meets the needs of SCADA security testbeds, the requirements of a typical SCADA security testbed were specified. The need is for a testbed that can be used to launch various security tests either to study the effect of different attacks on SCADA systems or to test the validity and impact of detection and mitigation techniques. Then the design phase stated the architecture of the simulation environment and the main components needed to construct the simulation environment in addition to classifying theses components according to their job and the site in which they are used. Finally, the development phase of the simulation environment was established according to the output of the design phase.

During the design phase, the specifications of typical SCADA systems in general and the features of the proposed SCADA simulation environment in particular have been taken into consideration. In SCADA systems, there are two types of networks, corporate/IT network and SCADA network. Special field devices and SCADA components are employed in SCADA network. Corporate network connects combination of nodes of different types. Some of these nodes are special SCADA nodes and the rest are normal nodes that are seen usually in IT networks. Moreover, two kinds of protocols will be employed in the simulation environment. Internet protocols, such as TCP, IP, UDP, etc. and SCADA special industrial protocols. In addition to these specifications related to SCADA systems, the simulation environment was designed while considering producing extensive, generic, flexible, reusable and easy-to-use components. Moreover, the ability to connect simulation environment along with physical devices was taken into account. Which means that a testbed can be constructed of a combination of some simulated nodes and other real physical devices communicating with each other.

The general architecture of the SCADA simulation environment is designed in four different layers as shown in figure 5.3. The first layer at the bottom of the hierarchy is composed of SCADA industrial specific modules or real world field devices such as PLCs, RTUs, etc. The second layer is composed of thoroughly designed coordination modules which are responsible for the interconnection as well as the critical time synchronization between the modules. In addition, the second layer contains the interfacing modules which are responsible for establishing communication with real external devices. The third layer encompasses the simulation of the different network protocols and modules constructing simulation components. The top layer consists of the generic applications running inside simulation or GUI and scripts that allow the user to interface with the developed environment.

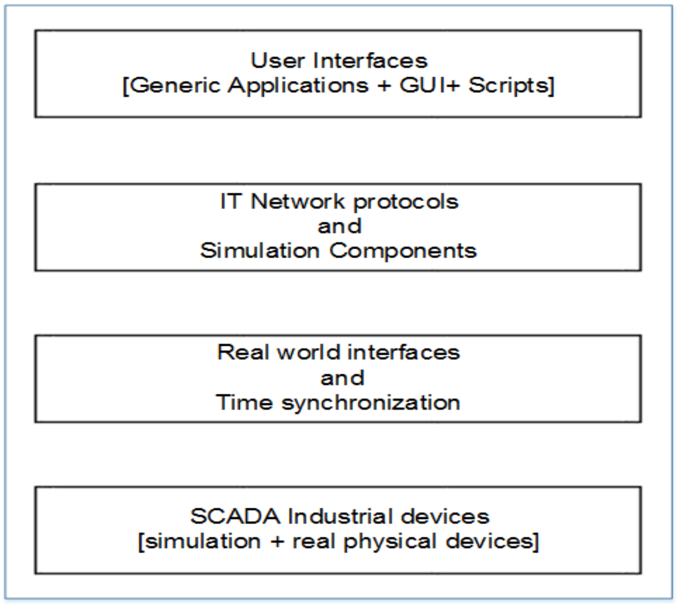


Figure 5.3: SCADA-SST simulation environment architecture

5.2.1 Testbed Components

In order to implement a simulation environment that can be used to model different SCADA architectures, there is a need to model main SCADA components and the communication networks to connect those components. Following is a list of various components constructing SCADA-SST with a specification of what have been done as part of this work.

- **SCADA Components:**

  OMNeT++ has been used to model main SCADA components, namely, PLC, RTU, and MTU. In addition, we have used HTML, JavaScript and VS.NET(WPF) to develop the HMIs used inside SCADA-SST

- **SCADA Protocols:**

  OMNeT++ has been used to simulate Modbus/TCP protocol

- **TCP/IP Stack Protocol:**

  The INET framework implementation of the TCP/IP stack protocol has been used in proposed simulation environment.

- **Real Hardware/Software Interfaces:**

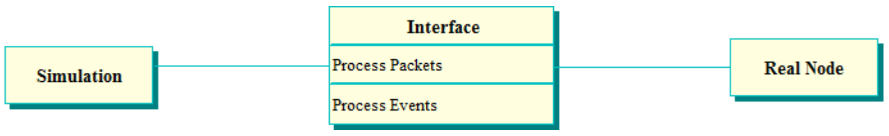


Figure 5.4: SCADA-SST interfaces design

In order to connect real hardware/software with simulated modules inside SCADA-SST, a set of interfaces have been implemented. They are working as translators that take packets coming from the outside world and convert them to events; the means of communication in OMNeT++. The design of SCADA-SST main components and the communication between them is shown in figure 5.5.

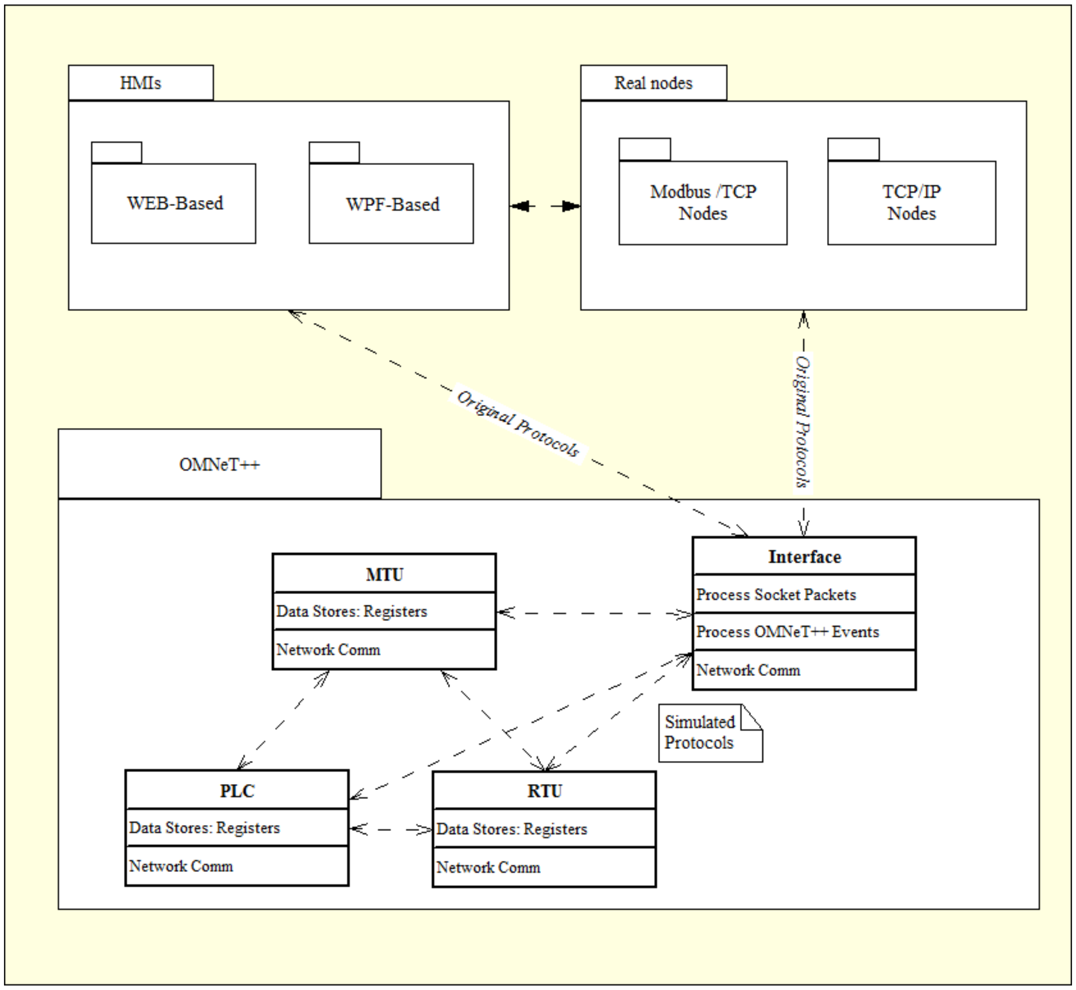


Figure 5.5: SCADA-SST components and communication

5.3 Testbed Implementation

SCADA-SST was implemented for the sake of building SCADA security testbeds. Therefore, these security testbeds should resemble the typical SCADA systems. Figure 5.6 depicts the typical layout of SCADA-SST.

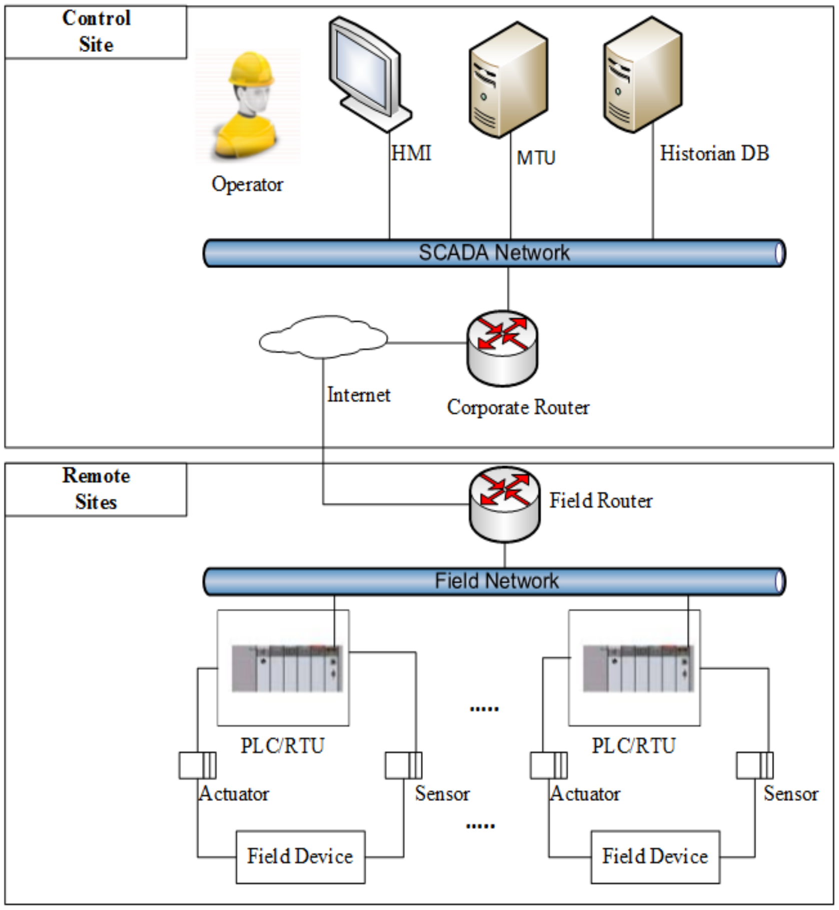


Figure 5.6: SCADA-SST environment layout

SCADA-SST has been implemented on the top of INET framework mentioned in section 5.1 and ReaSE tool [30] which is used for creation of realistic environments. According to the layout of the environment depicted in figure 5.6, the implemented simulation environment will consist of a set of modules that simulate various SCADA components, e.g., PLC, RTU, etc., a collection of protocols; either those protocols that are used inside the simulation environment to connect the different modules together or those protocols used to communicate with the outside world, e.g., modbus protocol. The implemented environment will involve a custom scheduler as well as interfaces to integrate with real hardware and applications.

The development process followed the model depicted in figure 5.7. First, the behavior of every component, needed to be simulated, is studied and analyzed in order to fully understand what is needed from this component to do or how it should behave. Then the simulation code is written according to the output of the behavior analysis phase. After writing the behavior code of the simulated component, the simulated component is tested to see if it is simulated in the right way. Finally, the properly simulated component is integrated with the whole simulation environment and become ready to use.

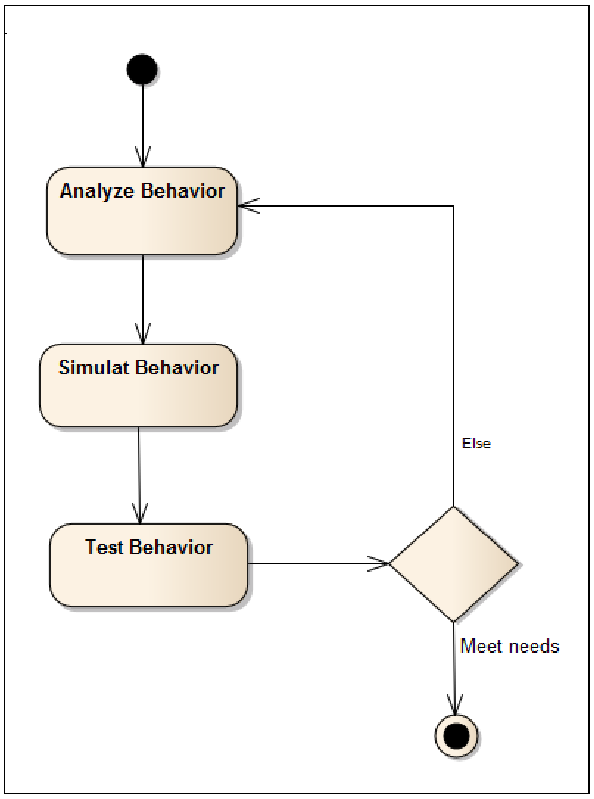


Figure 5.7: SCADA-SST development model

In OMNeT++, the term module is used to refer to any simulated node or protocol. Network components in OMNeT++ are described with the Network Description Language. With *modules* and *channels* to connect modules, two types of modules exist: simple modules and compound modules. Simple modules implement the behavior of a certain node or protocol and are, therefore, responsible for all activity occurring in a component. Their behavior is defined by user in C++ source code files. Compound modules are containers for simple modules and may contain any number of simple modules. The network node is comprised of the modules and their connections.

In order to develop communication networks as they can be found in SCADA systems, new compound modules for SCADA components are implemented. The following SCADA components are taken into account [31]: Remote Terminal Unit (RTU), Programmable Logic Controller (PLC), Master Terminal Unit (MTU), and Human Machine Interface (HMI). These SCADA components are derived from INET's standard modules, meaning they are treated by the simulation kernel as if they were INETs *StandardHost* module.

In general, RTU and PLC take over the same tasks [32], therefore, they have been modeled or simulated identically but are present in SCADA-SST as individual components. The behavior of SCADA-SST components is written in C++ this allows dynamic binding of the C++ programs describing the behavior of those modules. Those simulated components communicate with each other via simulated network packets and make decisions based on the packets they receive.

The simulated PLC component, for example, is shown in figure 5.8. The PLC is a compound module, that consists of several simple modules. The PLC is also derived from the *StandardHost* and so INET's implementation of the OSI stack is clearly visible inside the TCP/IP connectivity group. The Data Link Layer is represented by the simple module labeled *eth[i]* on the bottom. Packets received at *eth[i]* are first passed to the Network Layer protocol and then to the Transport Layer protocol, where TCP/IP is used. The Application Layer implements the behavior of the PLC via user-defined C++ code. At this point, the C++ applications describing the PLC's behavior are integrated into the PLC compound module. In the PLC component, there are other INET modules visible on the left side: NotificationBoard, InterfaceTable and RoutingTable. These modules are responsible for proper packet routing inside the simulation.

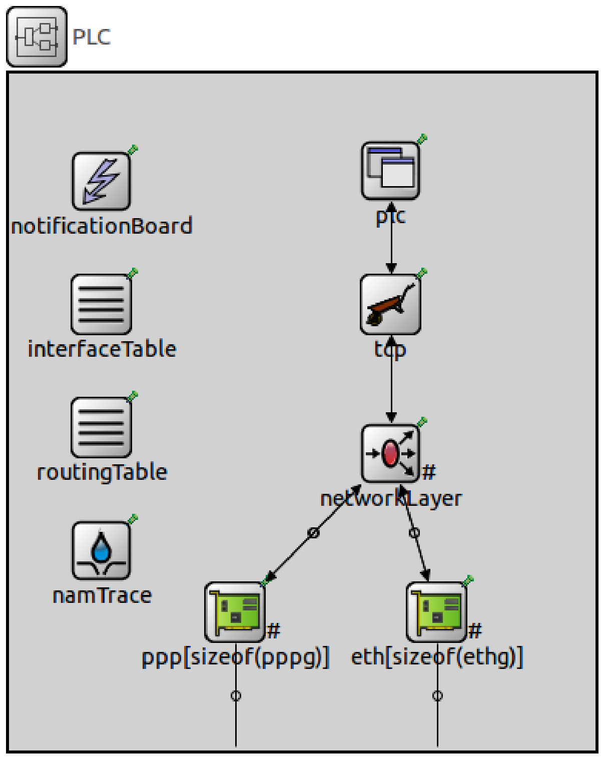


Figure 5.8: SCADA-SST's PLC component

Simulated components are always an approximation of the modeled entity since real world objects are inherently complex and require assumptions to reduce their complexity in order to derive a feasible model, that can be used in computer simulations. The assumptions for the different SCADA and network components, that are made during modeling, are discussed at this point. The SCADA components are based, to some extent, on their descriptions provided in Section 2.2. The communication in the modeled network uses TCP/IP. The simulated components are capable of communication via UDP and ICMP as well, however, this is not used in the modeled network. Specialized components for PLC and RTU are present in the simulation library of SCADA-SST but inside the simulations they are treated as identical components.

The implementation of SCADA components mentioned above was done using OMNeT++. However, HMIs used in this work were implemented as real external applications either web-based applications or WPF applications. The WPF-based HMIs have been developed using Visual Studio 2012 [33] and OPC Systems.NET [34]. While the web-based HMIs have been developed using HTML and javascript.

The use of the implemented environment to test effects of various attacks on a typical SCADA system and to evaluate the various mitigation techniques can be done using one of two possible ways. The first method is to simulate the attacks and mitigation techniques under study and run them as part of the simulation environment. The second method enables us to use real attacks binaries already implemented and take benefits of existing software or hardware and launch them from outside the simulation environment targeting either a component inside the simulation environment or an external physical component. The latter method can be used when we do not have the sufficient time and information about attacks and mitigation techniques under study so that we can not simulate them.

OMNeT++ Integrated Development Environment (IDE) enables the users of SCADA-SST to easily build the needed network topologies without writing any piece of code, or a little bit of code if any, using the drag and drop functionality of the IDE's interactive graphical editor and then setting some topology configuration parameters such as simulation runtime parameters and real external devices external information.

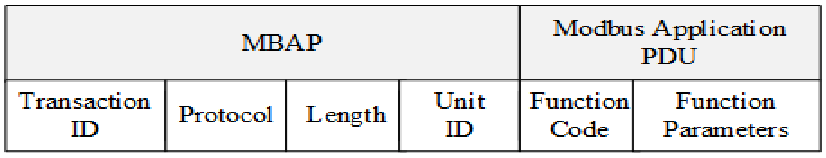
5.4 Protocol Implementation

In SCADA systems, there are two types of used protocols. The well-known Internet protocols commonly used inside corporate networks and special SCADA industrial protocols. In SCADA-SST, both types are needed to simulate realistic

SCADA testbeds. We have used INET implementation of Internet protocols such as TCP, IP, UDP, MAC, etc. while doing some modification to parts of these protocols in order to make them behave like real protocols especially when running malicious attacks. For instance, to launch DoS attack against TCP connections we need to limit the number of concurrent connections and drop all new connection requests when the number of active connections reaches this limit. On the other hand, there is no existing framework that implements SCADA protocols. Thus, part of this thesis contribution focuses on implementing SCADA protocols. Initially, Modbus/TCP protocol has been implemented . In this work, the implementation of Modbus/TCP protocol is based on the libmodbus library [35] stable version v3.0.6. The implementation details of this protocol are given in the following section.

5.4.1 Modbus/TCP Simulation

Modbus protocol is based on messaging and is widely used to establish master-slave communications between industrial devices.



MBAP				Modbus Application PDU	
Transaction ID	Protocol	Length	Unit ID	Function Code	Function Parameters

Figure 5.9: Modbus TCP architecture

Because of the nature of Modbus protocol, messaging structure, the implementation of this protocol is independent of the underlying physical layer. The basic structure of Modbus message contains the slave address, the command, the data, and check sum as depicted in figure 5.9. The work of this protocol is based on two concepts: request and response. In the request, the function code tells the target slave what kind of action to perform. Along with the function code, the request may contain additional data that the slave may need to perform. For example, if the function code was 'read_register', then the request must contain additional information about which holding registers to read. Moreover, the request check sum field enables the slave to check the integrity of the message contents. on the other hand, the response contains copy of the function code sent with the request or an error code in case of error is happening. In addition, the data bytes contain the requested data or a description of the error [36, 13, 14, 37, 38].
Modbus protocol supports eight function codes which tell the slave what action to do as stated in table 5.1. Table 5.2 shows an example of a request to read registers 0-1 from slave device 1. The response for the request would be as in table 5.3.

The class diagram of the of the modeled modbus/tcp, adjusted from libmodbus library, is given in figure 5.10.

Function code	meaning
01	read coil status
02	read input status
03	read holding registers
04	read input registers
05	read single coil
06	write single register
15	write multiple coils
16	write multiple registers

Table 5.1: Modbus function codes

trans id	unit	function	starting	registers	error check
00 00	00 01	00 03	00 00	00 02	LRC (FA)

Table 5.2: Modbus request example

5.5 Interfacing with External Devices

External devices and applications can be integrated with OMNeT++ using one of three possible ways [39, 23]. In the first method we can use source code integration which requires the modification of OMNeT++'s source code. The second method is the shared library integration where a library used by both OMNeT++ and the external application is developed. The third method is through the use of socket connections. i.e. commonly used network protocols are employed for communication. In order to use the first method, knowledge of OMNeT++'s implementation is required. The source code needs to be recompiled using the specified build

trans id	unit	function	byte count	data	data	error
00 00	00 01	00 03	00 04	00 05	00 06	LRC (E D)

Table 5.3: Modbus response example

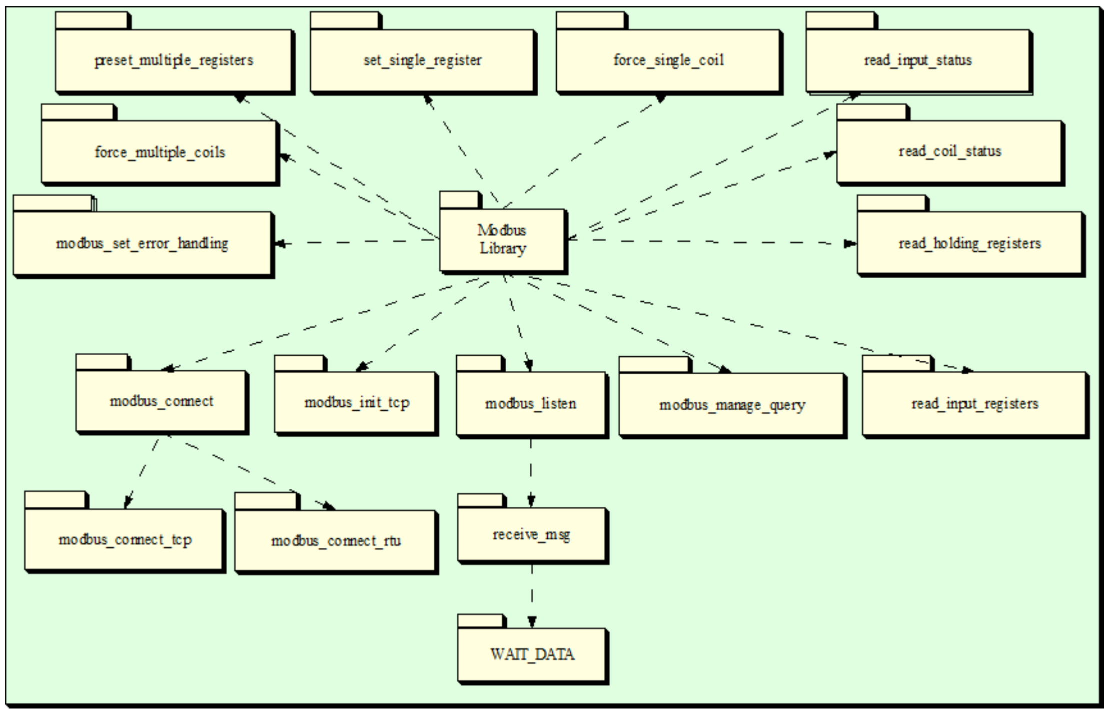


Figure 5.10: Modbus protocol implementation library

environment. The external application must be integrated into this environment, which limits the flexibility during design and development specially when connecting with physical devices. Moreover, mastering OMNeT++'s implementation details takes a steep learning curve. The second method is similar to the first one but it does not require the use of the build environment. However, the use of OMNeT++'s interfaces is mandatory, which has the same drawbacks as the first method. In the third way, the minimum changes are required and it imposes the minimal restrictions on development. In addition, it is the most flexible for connecting with physical devices. The integration between SCADA-SST and the outside world is implemented using sockets.

5.5.1 Interface Concept

One of the requirement of the SCADA-SST simulation environment is to design it in such way that external real devices may be connected. For this to be satisfied, interfaces have been implemented so that they are used in the simulation environment whenever we need to connect with external real devices. An interface has been implementedto connect with any device using Modbus/TCP protocol. Other interfaces are implemented to connect with devices using Internet protocols TCP and UDP as extensions of the INET classes "CSocketRTScheduler" and "ExtInterface".

5.5.2 Interface Implementation

Figure 5.11 shows the class diagram of this work implementation of the modbus interface used to connect with physical components via modbus protocol. In this implementation, the ModbusFace() routine is responsible for initializing and creating active instance of the modbus interface. Here, the IP address of the corresponding physical device is specified. The Listen() routine is a procedure that keeps listening and receives messages from external world. The processMSG() receives messages from Listen(), process them and forward messages to simulated modules.

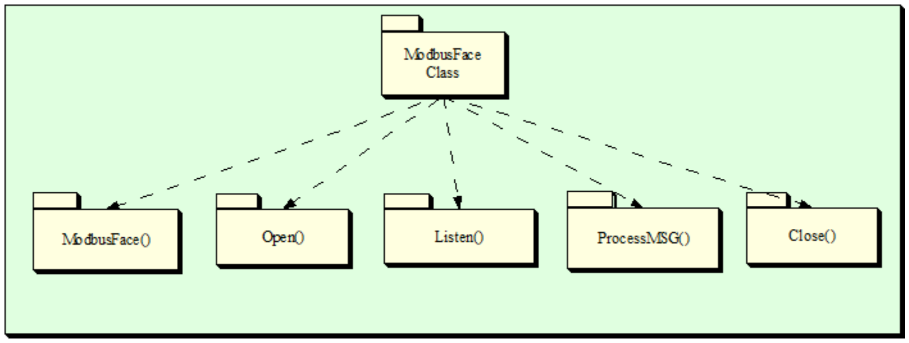


Figure 5.11: Modbus interface structure

The integration between the simulated modules and real external devices can be done either between those industrial devices that communicate using industrial protocol Modbus/TCP, or devices that work inside the corporate network using the well-known Internet protocols, TCP and UDP. To connect with an external physical devices, an interface node is added to the simulation network topology and setting a parameter 'externalIP' to the real IP address of the external physical device we want to communicate with. The selected interface node must be compatible with communication protocol used in the physical device. For example, to connect with a physical PLC, we must select modbus interface node.

In order to manage the communication of OMNeT++ with external world, a custom scheduler has been implemented. The sending and receiving of the packets from and to the external hardware and applications is still done with sockets, which are programmed using the Windows Socket API (WSA or Winsock). The scheduler is necessary for the propagation of the packets, i.e., translating the external packets to internal, simulated packets and vice versa. The scheduler of SCADA-SST is based on the cSocketRTScheduler, which is presented as a demo application of OMNeT++.

The ability to integrate the simulation environment with external real devices increased the functionality of the implemented environment since it would be possible to evaluate the security of different SCADA components using different network topologies. With this feature, we can use the simulation environment to construct different SCADA topologies taking benefit of the existing real devices and replacing the missing SCADA components or corporate/IT devices with simulated ones.

CHAPTER 6

SCADA-SST USE CASES

In this chapter, two realistic SCADA configurations are designed using the proposed environment. The first configuration is for water distribution control in which a hybrid architecture of simulated and physical components is presented. The second configuration is of a smart grid system.

6.1 Water Distribution Use Case

In this case, the SCADA system presented is a combination of simulated components, physical hardware and real applications.

The scenario resembles a SCADA system that is used to control the water distribution in water distribution station. In this case, we have two water tanks, the first tank, Tank1, is filled from the water source and then used to distribute water. The second tank, Tank2, is used as an extra tank to reduce the pressure on Tank1. In case the water level in Tank1 reached a pre-specified level, a transfer

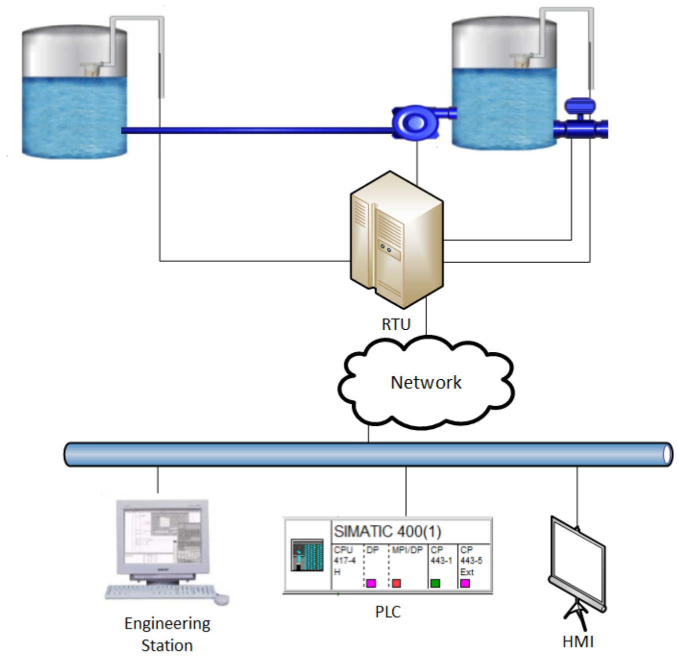


Figure 6.1: Water distribution network topology

pump between Tank1 and Tank2 would be opened to pass the water from Tank1 to Tank2. In case the water level in both tanks exceeded the permitted water level, a drain valve on Tank2 would be opened as depicted in the HMI of this SCADA system shown in figure 6.2.

The RTU collects data from the two tanks' water-level sensors. The PLC is connected with the RTU. It reads data from the RTU, processes it, then sends it to the HMI. Moreover, The PLC receives actions from HMI's operators and direct commands that reflect those actions to the field devices through RTU. The engineering station is used to configure the PLC and to write PLC programs. The configuration details of each component is described as follows.

- PLC

  The PLC used in this scenario is a real Siemens SIMATIC S7-400 PLC that has been integrated with the simulated components using modbus/tcp protocol. It has been configured to work as a master device to collect data from the simulated RTU and then sends it to the HMI. Moreover, it forwards operator's actions to the RTU.

- RTU

  The RTU is simulated using SCADA-SST simulation environment and is used to gather information from the tanks' sensors and works as a slave device listening on port 502 for coming requests from the PLC and then responds by providing the requested data.

- Engineering Station

  Windows 7 host equipped with Siemens Simatic PCS7 V8.0 software which is a programming and configuration environment for Siemens PLCs. Here, it is used to configure the physical PLC and connect it with the simulated components.

- Water-Level Sensors

  An implementation of external application is used to simulate the two sensors. The first sensor is used to measure the water level in Tank1 and the second sensor is used to measure the water level in Tank2. This external application that simulates the two sensors is connected with the RTU in the simulation environment.

- HMI

  The HMI has been developed as real web-based application to receive updates about the status of the water tanks from PLC and presents it in graphical way as shown in figure 6.2. In addition, the implemented HMI allows operators to interact with the controlled field devices by starting/stopping transfer pump or opening/closing the drain valve. The HMI is running outside the simulation environment and is connected through the ethernet network with the PLC working outside the simulation environment.

This scenario shows how we can construct SCADA testbeds by employing the existing physical hardware, real applications and SCADA-SST simulation environment. This configured SCADA testbed will be used in Chapter 7 to conduct network security analysis of the communication between the PLC and the engineering station.

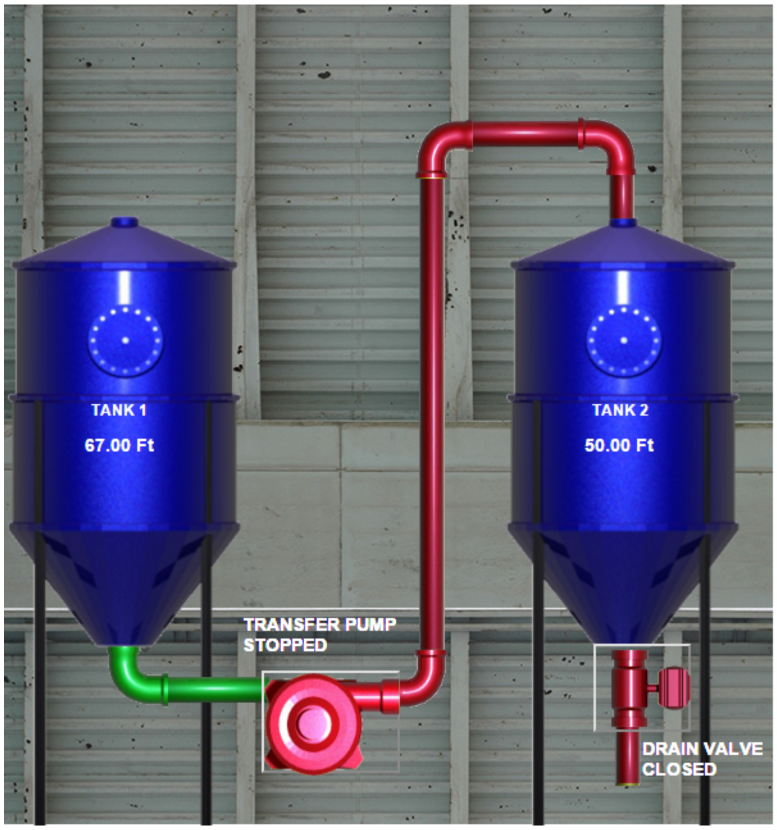


Figure 6.2: Water distribution HMI

6.2 Smart Grid Use Case

In this case, the SCADA system presented had been configured using simulated components, and real applications.

The scenario exemplifies a smart grid system that is used to control electricity consumption by smart homes. In this case, we have two smart homes, for simplicity, each home is equipped with a smart electric meter that measures the power consumption and sends these data to electric company. According to the received data, the electric company would be able to do the needed actions in order to enhance the power supplied to the smart home.

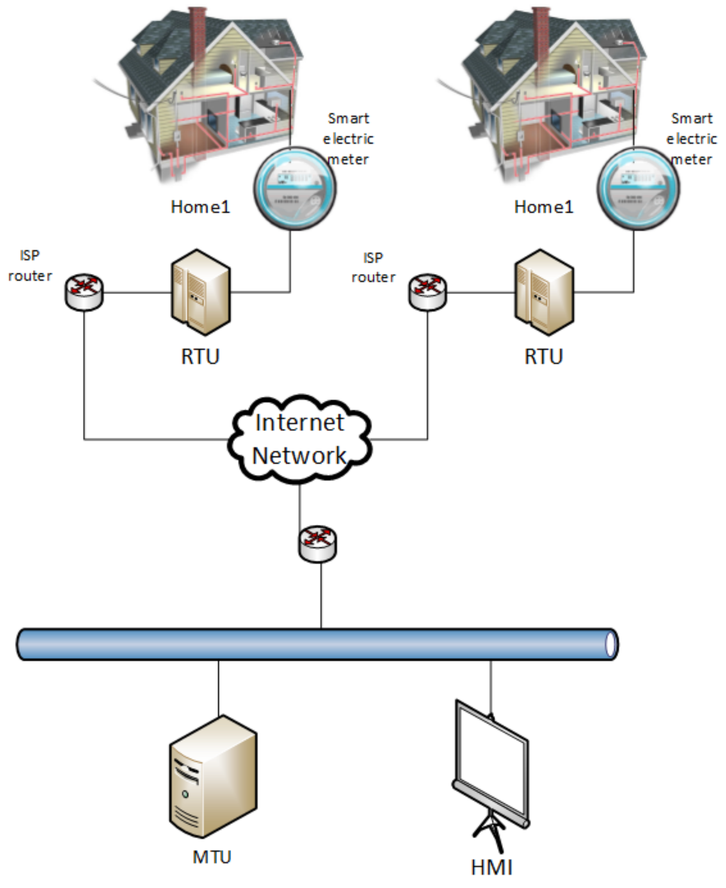


Figure 6.3: Smart Grid system network topology

The RTU collects data from the electric meter. The MTU is connected with the RTU. It reads data from the RTU, then sends it to the HMI. The configuration details of each component is described as follows.

- MTU

  The MTU used in this scenario is a SCADA-SST's simulated component that has been configured to work as a master device to collect data from the simulated RTU and then sends it to the HMI.

- RTU

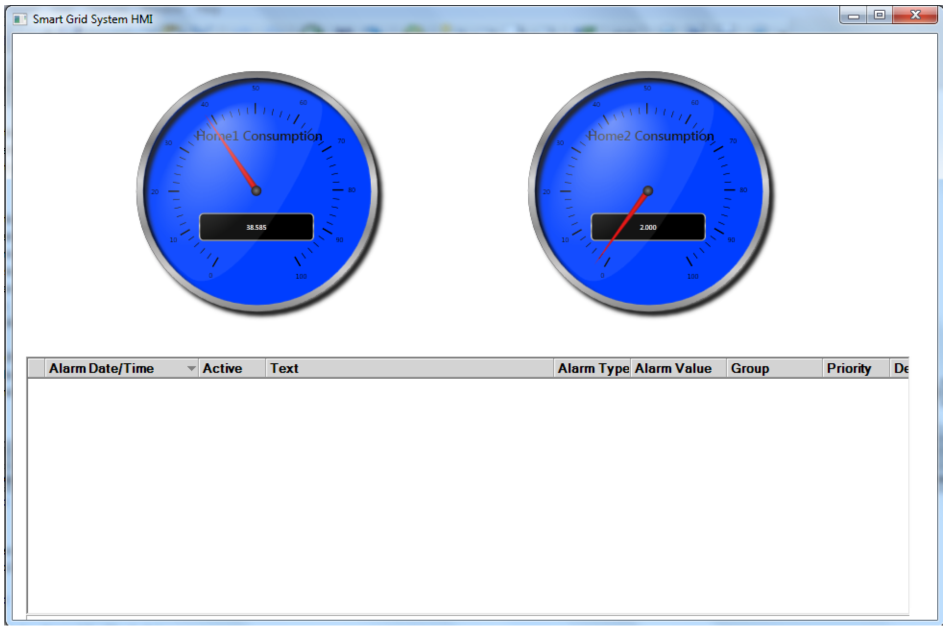
  The RTU is simulated using SCADA-SST simulation environment and is used to gather information from a smart home's electric meter and works as a slave device listening on port 502 for coming requests from the MTU and then responds by providing the requested data.

- Electric Meter

  An implementation of external application is used to simulate the electric meter. This external application simulates power consumption reads and is connected with the RTU in the simulation environment.

- HMI

  The HMI has been developed as a real WPF application to receive updates about the power consumption by the smart homes from MTU and presents it in graphical way as shown in figure 6.4. The HMI is running outside the simulation environment and is connected through the ethernet network with the MTU.



Alarm Date/Time	Active	Text	Alarm Type	Alarm Value	Group	Priority	De

Figure 6.4: Smart Grid System HMI

CHAPTER 7

IMPLEMENTED ATTACKS

This chapter exposes a network security analysis of the communication between PLCs and the engineering stations. Using the water distribution testbed discussed in Chapter 6, a number of three network attacks have been carried out leading to serious compromise of typical PLCs. In the subsequent sections, for every implemented attack we describe: (i) the behavior of the attack, (ii) the scripts and tools used to configure the attack, and (iii) the nodes used to launch the attack.

Major PLC manufacturers (Siemens, Allen-Bradley, Phoenix Contact, etc.) provide efficient software environments to program and configure their PLCs. The programs are written in a variety of languages including graphical languages such as Ladder logic. PLC programs need to be efficient, lightweight and guarantee secure communication with the other field devices once deployed.

Programming a PLC consists of uploading the written program to the PLC after it has been developed and tested at an engineering station. Typically, the engineering station is connected to the PLC with Ethernet. This communication can be point-to-point involving a simple Ethernet cable between the PLC and the engineering station or is a part of a network including other stations. Because the PLC program is in charge of controlling how the PLC works and commands field devices, the upload procedure should be performed in a secure way. An adversary who can interfere with this uploading procedure can launch a variety of attacks ranging from DOS to seizing full control of the PLC.

Simatic PCS7 is the programming environment for Siemens PLCs. It is a comprehensive software suite offering a variety of features to configure control systems, in particular PLCs. The software provides a graphical user interface for simple operation and clear display of configuration data. In order to upload a new configuration program, an engineering station with Simatic PCS7 software communicates with the PLC through Ethernet and using COTP (Connection Oriented Transport Protocol).

COTP protocol is not commonly used and is based on a very old specification (RFC 905 [40]). Very scarce documentation about the protocol is publicly available and few attempts were made to reverse engineer it [41]. Although COTP protocol has been replaced by TCP in most applications, it is still being used by Simatic PCS7 software. This can be seen as a manifestation of security-by-obscurity which is common protection measure in ICS.

The three main commands that the engineering station with Simatic PCS7 software can send to the PLC are the following:

- Start Command: turns the PLC on, assuming it is currently turned off. The start command is typically used when re-programming the PLC. In particular, the start command packets are sent along with the new PLC program packets.
- Stop Command: turns off the PLC.
- Check Status: enquires about the current status of the PLC.

Using the water distribution control testbed, described in Chapter 6, equipped with a common PLC, namely, Siemens S7-400 in addition to its corresponding configuration software, namely, Simatic PCS7 8.1, we successfully carried out three network security attacks, as described in the following sections, which allowed to interfere with the PLC-PCS7 communication and send arbitrary commands to the PLC. The three security attacks, namely, replay, Man-In-The-Middle (MiTM), and command modification, are common IT network security attacks, but they are not typically used to interfere with PLC-PCS7 communication.

7.1 Replay Attack

The first implemented PLC network attack is a typical replay attack. The attack consists of 3 steps: starting a PCS7 command (stop, start, etc.), capturing the packets, and replaying the captured packets at a later time. The captured packets corresponding to a given command are first processed by filtering out any packets that are not part of the commands traffic. Since PCS7-PLC communication uses the COTP protocol (port 102), any other packets are filtered out. In addition, only packets in the PCS7-PLC direction are kept (packets in the opposite direction are filtered out). The cleaned traffic for each command is then stored in a pcap file.

Initially, tcpreplay [42] suite is used to replay the recorded packets (cleaned pcap file). tcpreplay suite comes with different tools such as tcpprep (packets pre-processor that isolates packets in each direction), tcprewrite (pcap file editor which rewrites packet headers), tcpreplay (replays pcap files onto the network), etc. Using these tools, the pcap file is pre-processed before replaying by changing the source IP address and recomputing the checksum value in each packet. Once the pre-processed pcap file is replayed on the PLC, most of the packets are discarded by the PLC and the replay attack fails. After investigation, it turns out that the packets are discarded for two main reasons. First, the sequence (SEQ) and acknowledgement (ACK) numbers in the replayed packets are not changed. Consequently, the TCP/IP kernel at the PLC tags those packets as duplicates and discards them. Second, tcpreplay tool replays the packets in the pcap file one after the other without waiting for any response from the PLC. Hence, the PLC receives some packets out of the proper sequence and discards them. This problem has been recently observed by Maynard et al. [43].

To overcome these problems and to guarantee that the replayed packets are accepted by the TCP/IP kernel at the PLC, we resorted to write a customized python script using scapy [44]. Scapy is a powerful packet manipulation program written in python and hence can be easily used in python scripts. It features a variety of packet manipulation capabilities including: sniffing and replaying packets in the network, network scanning, tracerouting, etc. However, the most useful scapy features for our replay attack are the ability to rewrite the sequence and acknowledgement numbers and to match requests and replies.

Dealing with the duplicate sequence and acknowlgement numbers consists of recalculating these numbers and rewriting them with scapy. Manipulating packet headers using scapy is straightforward since any packet field is simply accessible by the dot operator (e.g. ip.src, tcp.flags, rcv[TCP].seq). Initially, random sequence and acknowledgement numbers are chosen. Then, at each packet sending, the numbers are incremented and added to the next packet. Replaying packets in the appropriate sequence and time requires waiting for the response of some packets before releasing the next packet. Scapy provides several variants of the Send function which is in charge of sending a packet in the network. For packets not requiring a response (e.g. Acknowlegement packet), the simple sendp function is used. The sendp function takes as input the packet as well as the network interface. For packets requiring a response, several functions can be used:

– *sr:* Send and receive packets at layer 3

– *sr1:* Send packets at layer 3 and return only the first answer

– srp: Send and receive packets at layer 2

– *srp1:* Send and receive packets at layer 2 and return only the first answer

– *srloop:* Send a packet at layer 3 in loop and print the answer each time

– *srploop:* Send a packet at layer 2 in loop and print the answer each time

In our program, we used srp1 function because there is always one single response packet sent by the PLC. Algorithm 1 shows the core of the python script using the scapy features. The REPLAY subroutine takes as input the pcap file, the network interface, the attackers IP address and port number. In addition, arbitrary values are chosen to initialize the ACK and RSTACK numbers. The for loop inside the subroutine goes through the packets one by one. For each packet, TCP checksums are removed (line 7) so that the network interface card recalculates newer values, the source IP and port numbers are updated (lines 8 and 9), the sequence numbers are incremented (lines 11 and 19), the packet is replayed using either sendp function (for SYN and RST packets) or the srp1 function (lines 13 and 18).

Algorithm 1 Replay a sequence of captured packets using Scapy

```
 1: function REPLAY(pcapfile, eth_interface, srcIP, srcPort)
 2:     $recvSeqNum \leftarrow 0$
 3:     $SYN \leftarrow True$
 4:     for packet in rdpcap(pcapfile) do
 5:         $ip \leftarrow packet[IP]$
 6:         $tcp \leftarrow packet[TCP]$
 7:         del ip.chksum
 8:         $ip.src \leftarrow srcIP$
 9:         $ip.sport \leftarrow srcPort$
10:         if tcp.flags == ACK or tcp.flags == RSTACK then
11:             $tcp.ack \leftarrow recvSeqNum + 1$
12:             if SYN or tcp.flags == RSTACK then
13:                 sendp(packet, iface=eth_interface)
14:                 $SYN \leftarrow False$
15:                 continue
16:             end if
17:         end if
18:         $rcv \leftarrow srp1(packet, iface = eth\_interface)$
19:         $recvSeqNum \leftarrow rcv[TCP].seq$
20:     end for
21: end function
```

The above python program has been tested using two attack scenarios. In the first scenario, the replay attack was launched from the same host (IP address) used for the capture, that is, the host with PCS7 software. In the second scenario, the replay attack was launched from a different host on the same network, that is, the attacker machine with Kali. In each scenario, two types of commands are tried, namely, start and stop. The replay attack was successful in both scenarios for both types of commands. Hence, an unknown attacker machine (without PCS7 software) on the same network can turn the PLC ON or OFF by simply replaying a start or stop command. This clearly might cause significant damage to a SCADA system.

7.2 Man-In-The-Middle (MiTM) Attack

The communication between PCS7 host and the PLC uses COTP over Ethernet. Ethernet protocol uses Address Resolution Protocol (ARP). Hence, theoretically the communication is vulnerable to Man In The Middle (MiTM) attacks through ARP Poisoning.

In a switched Ethernet network, a host A who tries to communicate with a host B (with a known IP address) needs its physical address (MAC). The MAC address can be obtained by broadcasting an ARP request to all hosts in the network. In a normal scenario, only host B will send a response with the correct IP-MAC pair. In an attack scenario, an attacker (host C) in the same network will send a fake response with a false IP-MAC claiming to be the owner of B's IP address. Typically, the attacker floods the network with its fake response forcing the victim host (A) to accept the false pairing and ignore the correct one sent by host B. ARP poisoning is typically launched between two hosts allowing the attacker to insert himself as a tunnel between the two victims and consequently sniff all packets between them.

In our scenario, an ARP poisoning MiTM attack is implemented between the PCS7 host and the PLC using ettercap tool [45]. The attack is successful and all the packets exchanged between the PCS7 and PLC are tunneled through the attacker host (Kali). A MiTM attack can be passive or active. A passive version consists in simply observing the traffic of the PLC and hence breaking the confidentiality of the commands sent to the PLC. An active version is more dangerous since it allows the attacker to tamper with the packets and commands and consequently interfere with the normal operation of the system.

7.3 Stealth Command Modification Attack

The third attack is a combination of replay and MiTM attacks which aims at sniffing the traffic between the PCS7 and PLC and then interfering with sent commands by replaying other commands in a stealth way. Through this attack, an adversary can completely change the behavior of the SCADA system since sending a command leads to the execution of another command.

The attack goes through three main steps: MiTM attack, command detection, and replay of a false command. Figure 7.1 illustrates the attack. Initially, the attacker (Kali) starts by launching a MiTM attack to place himself between the PCS7 and the PLC exactly as described in the previous section. Then, it stays in an idle state observing the traffic passively and waiting for commands sent by the PCS7 host to the PLC (Step 1 in Figure 7.1). For the sake of command detection in the network, Snort intrusion detection system (IDS) [46] is used. Snort is a signature-based network IDS which allows to detect patterns of traffic inside the network. Currently, Snort is configured to detect two types of commands, namely, start and stop. As soon as the attacker detects a command from the PCS7 host to the PLC (Step 2), a different command will be replayed to the PLC (Step 4). That is, if a start command is detected, the attacker replays a stop command to the PLC. If a stop command is detected, the attacker replays a start command (with a different PLC program) to the PLC. However, it is easy to notice that if the attacker interferes with a start command to make it a stop command (or the opposite), the PCS7 will quickly notice that something is wrong. To make the attack as stealth as possible, the attacker continues the communication with the PCS7 host while impersonating the PLC (Step 3). So for the PCS7 host the communication appears to be perfectly normal. This technique has been used by Stuxnet [47] in its famous attack on Iran's nuclear facility. Indeed, to make the attack stealth, Stuxnet recorded normal frequency values. Then, at attack time, it played those recorded frequencies to make the monitoring system believes that centrifuges are operating as normal [47, 48].

Snort is an IDS which allows only to detect known patterns in the net-

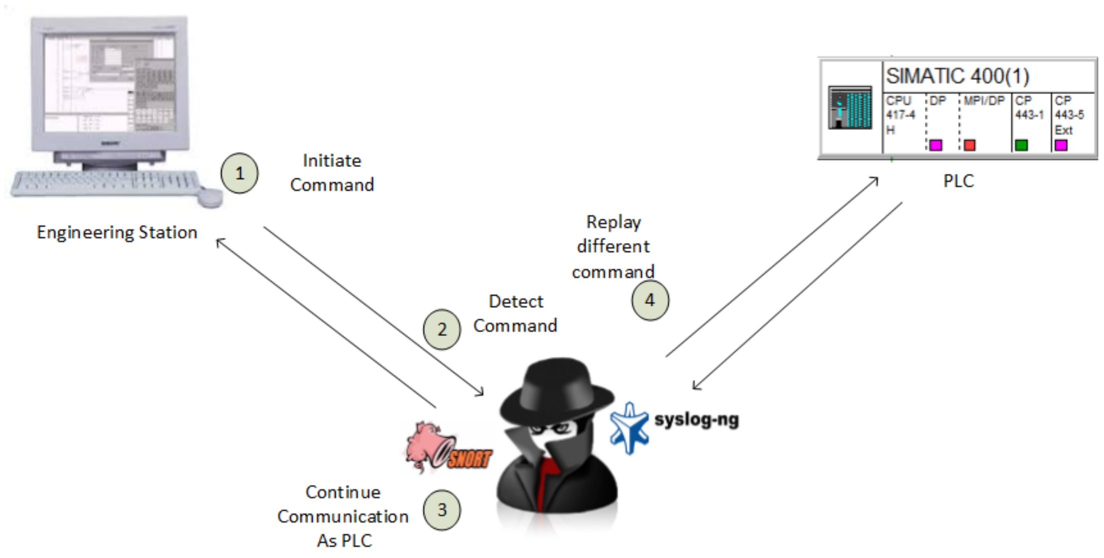


Figure 7.1: Stealth command modification attack

work traffic. However, the stealth command modification attack requires the launching of the replay attack (python program) as soon as a command is detected in the traffic. To fill this gap, Snort is configured to log alerts to Syslog-ng utility [49]. In turn, Syslog-ng is configured to trigger the replay attack upon the reception of appropriate Snort alerts.

CHAPTER 8

CONCLUSION AND FUTURE WORK

This chapter summarizes the work of this thesis. In addition, it outlines possibilities for future work and addresses open problems.

8.1 Conclusion

In this thesis, a simulation environment, SCADA-SST, for building SCADA security testbeds is developed. As security issues affecting SCADA systems have become intensive, the need to have security testbeds where security researchers and specialists can conduct various security scenarios is immense. The security tests are for the sake of enhancing the security of SCADA systems and providing better security solutions for current systems. Therefore, a simulation environment that can provide a close to real simulations for security analysis is highly appreciated. In this work, the introduced simulation environment has been evaluated by configuring two realistic SCADA security testbeds in which a combination of simulated and physical SCADA components were used synchronously. The introduced environment was developed to involve generic and extensible components and to support hybrid architectures as well. The work overcomes the issues and weaknesses of the most relevant approaches found in the literature. For example, the components were designed to be generic enough and extensible in order to be used in building different SCADA architectures. The components were implemented with a generic behavior so that they can be adopted to build different SCADA architectures without considering any restrictions to specific systems with the ability to extend the behavior of any component by dynamic binding of C++ class libraries. In addition, the environment provides the ability to launch cyber attacks, in order to test their effects, either from inside the simulation or using real binaries of cyber attacks. In the evaluation, one of the most common PLCs has been used along with the simulated components. On the other hand, there is a number of SCADA protocols used for the communication inside SCADA systems and in this work, Modbus/TCP has been modeled and there still a need to model further SCADA protocols to be able to enhance the security of most SCADA systems. In addition, this work has modeled the main SCADA components, namely PLC, RTU, MTU and HMI because of their common uses in SCADA systems.

Another key contribution of this thesis is the study and analyze of the security of the network communication between typical SIMATIC PLCs and the engineering stations mainly used for PLCs configuration and re-programming. The water tanks control testbed was used to conduct this analysis. It was demonstrated that a SCADA system can be seriously compromised by mounting network attacks targeting PLCs. PLCs are very common components in SCADA systems. They sit between HMIs and field devices and are in charge of sending commands and receiving data to/from field devices. Since a PLC is programmable, it can be completely compromised by loading a malicious control program. Through the detailed description and implementation of three attacks (Replay, MITM, and command modification), we showed that the communication between the PLC and the engineering station can be compromised leading to serious SCADA system instability. We showed that, with open source tools and simple python scripts, one can mount successful attacks. In particular, programming and configuration traffic directed to PLCs may be replayed, sniffed, and/or modified.

8.2 Future Work

The work of this thesis highlights the possibilities for further work in various directions. Following is a list of open topics that may be considered in our future work.

- Developing and/or testing detection techniques of the network attacks presented in this thesis in addition to other cyber attacks targeting SCADA systems.

- Simulating further SCADA industrial protocols in the simulation environment.

- Analyzing the security of further SCADA components and protocols while involving other commonly used SCADA special components and/or softwares from different vendors.

REFERENCES

[1] "Dell security annual threat report." https://software.dell.com/docs/2015-dell-security-annual-threat-report-white-paper-15657.pdf, 2015. [Online; accessed May-2015].

[2] M. Cheminod, L. Durante, and A. Valenzano, "Review of security issues in industrial networks," *Industrial Informatics, IEEE Transactions on*, vol. 9, pp. 277–293, Feb 2013.

[3] R. R. R. Barbosa, *Anomaly detection in SCADA systems: a network based approach*. University of Twente, 2014.

[4] T. C. for SCADA Security Sandia National Labs, "National scada testbed." http://energy.sandia.gov/energy/ssrei/gridmod/cyber-security-for-electric-infrastructure/scada-systems/testbeds/national-scada-testbed. [Online; accessed March-2015].

[5] N. Laboratory, "National scada test bed program." https://www.inl.gov/scada/. [Online; accessed March-2015].

[6] G. Deconinck, H. Beitollahi, G. Dondossola, F. Garrone, and T. Rigole, "Testbed deployment of representative control algorithms," *Deliverable D9, Project CRUTIAL EC IST-FP6-STREP*, vol. 27513, 2008.

[7] P. D. P. A. H. S. L. G. M. M. W. T. Jack Wiles, Ted Claypoole and J. H. Windle, *Techno Security's Guide to Securing SCADA*. ELSEVIER, 2008.

[8] NATIONAL COMMUNICATIONS SYSTEM, *Supervisory Control and Data Acquisition (SCADA) Systems, TECHNICAL INFORMATION BULLETIN 04-1*, october 2004 ed.

[9] R. L. Krutz, *Securing SCADA systems*. John Wiley & Sons, 2005.

[10] Siemens, "The simatic pcs7 process control system, tech. rep.." , April 2013.

[11] A. G. Association *et al.*, "Cryptographic protection of scada communications; part 2: retrofit link encryption for asynchronous serial communications," tech. rep., AGA Report, 2005.

[12] E. J. Byres, M. Franz, and D. Miller, "The use of attack trees in assessing vulnerabilities in scada systems," in *Proceedings of the International Infrastructure Survivability Workshop*, Citeseer, 2004.

[13] MODICON, Inc., Industrial Automation Systems, *Modicon Modbus Protocol Reference Guide*, june 1996 ed.

[14] I. Modbus Organization, "Modbus home page." http://www.modbus.org. [Online; accessed Augus-2015].

[15] ABB, *DNP3 Communication Protocol Manual*, february 2011 ed.

[16] I. C. S. C. E. R. Team, "Trends in incident response in 2013," 2013.

[17] M. Haney and M. Papa, "A framework for the design and deployment of a scada honeynet," in *Proceedings of the 9th Annual Cyber and Information Security Research Conference*, pp. 121–124, ACM, 2014.

[18] B. Zhu, A. Joseph, and S. Sastry, "A taxonomy of cyber attacks on scada systems," in *Internet of Things (iThings/CPSCom), 2011 International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, pp. 380–388, Oct 2011.

[19] A. A. Cárdenas, S. Amin, and S. Sastry, "Research challenges for the security of control systems.," in *Hot topics in security*, 2008.

[20] J. Pollet, "Innovative defense strategies for securing scada and control systems," *Technical Papers of ISA*, vol. 459, pp. 115–128, 2005.

[21] T. C. S. R. H. C. Michael J. McDonald, Gregory N. Conrad, "Cyber effects analysis using vcse-promoting control system reliability," *andia National Laboratories Report (SAND2008-5954)*, 2008.

[22] R. Chabukswar, B. Sinópoli, G. Karsai, A. Giani, H. Neema, and A. Davis, "Simulation of network attacks on SCADA systems," in *First Workshop on Secure Control Systems*, 2010.

[23] C. Queiroz, A. Mahmood, and Z. Tari, "SCADASim a framework for building SCADA simulations," *Smart Grid, IEEE Transactions on*, vol. 2, no. 4, pp. 589–597, 2011.

[24] W. Chunlei, F. Lan, and D. Yiqi, "A simulation environment for SCADA security analysis and assessment," in *Measuring Technology and*

Mechatronics Automation (ICMTMA), 2010 International Conference on, vol. 1, pp. 342–347, IEEE, 2010.

[25] A. Almalawi, Z. Tari, I. Khalil, and A. Fahad, "SCADAVT-a framework for SCADA security testbed based on virtualization technology," in *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pp. 639–646, IEEE, 2013.

[26] W. Mahoney and R. A. Gandhi, "An integrated framework for control system simulation and regulatory compliance monitoring," *International Journal of Critical Infrastructure Protection*, vol. 4, no. 1, pp. 41–53, 2011.

[27] *INET framework 2.1 for OMNeT++.Manual*, version 4.6 ed., 2014.

[28] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 60, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[29] Andrs Varga and OpenSim Ltd., *OMNeT++ User Manual*, version 4.6 ed., 2014.

[30] T. Gamer and M. Scharf, "Realistic simulation environments for ip-based networks," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, p. 83, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[31] V. M. Igure, S. A. Laughter, and R. D. Williams, "Security issues in scada networks," *Computers & Security*, vol. 25, no. 7, pp. 498–506, 2006.

[32] C. Ie, "Programmable controllers—part 3: Programming languages," 2003.

[33] "Microsoft visual studio professional 2012." http://www.microsoft.com/en-sa/download/details.aspx?id=30682.

[34] "Opc systems.net." https://www.opcsystems.com.

[35] "Open source c library of modbus protocol for linux, mac os x, freebsd, qnx and win32, v3.0.6." http://libmodbus.org, 2014. [Online; accessed Augus-2015].

[36] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, 2008.

[37] I. Modbus, "Modbus application protocol specification v1. 1a," *North Grafton, Massachusetts (www. modbus. org/specs. php)*, 2004.

[38] I. Modbus, "Modbus messaging on tcp/ip implementation guide v1. 0b," *North Grafton, Massachusetts (www. modbus. org/specs. php)*, 2006.

[39] C. P. Mayer and T. Gamer, "Integrating real world applications into omnet++," *Institute of Telematics, University of Karlsruhe, Karlsruhe, Germany, Tech. Rep. TM-2008-2*, 2008.

[40] N. W. Group, "Iso transport protocol specification, tech. rep.." , April 1984.

[41] G. Devarajan, "Unraveling scada protocols: Using sulley fuzzer," in *Defon 15 Hacking Conf*, 2007.

[42] "tcpreplay." http://tcpreplay.synfin.net/.

[43] P. Maynard, K. McLaughlin, and B. Haberler, "Towards understanding man-in-the-middle attacks on iec 60870-5-104 scada networks," in *Proceedings of the 2nd International Symposium on ICS & SCADA Cyber Security Research 2014*, pp. 30–42, BCS, 2014.

[44] P. Biondi, "Scapy." http://www.secdev.org/projects/scapy.

[45] ALor and NaGA, "Ettercap." http://ettercap.sourceforge.net.

[46] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks.," in *LISA*, vol. 99, pp. 229–238, 1999.

[47] N. Falliere, L. O. Murchu, and E. Chien, "W32. stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, 2011.

[48] S. Zhioua, "The middle east under malware attack dissecting cyber weapons," in *Distributed Computing Systems Workshops (ICDCSW), 2013 IEEE 33rd International Conference on*, pp. 11–16, IEEE, 2013.

[49] R. Gerhards, “The syslog protocol,” 2009.

Appendices

Appendix A: Installation & Configuration

1. **Installation**
   In order to use SCADA-SST, you must have installed OMNeT++ and the according version of INET framework
   - **Install OMNeT++**
     Install the prerequisite packages by executing the following command:

```
$ sudo apt-get install build-essential gcc g++ bison flex perl \
tcl-dev tk-dev blt libxml2-dev zlib1g-dev openjdk-7-jre \
doxygen graphviz openmpi-bin libopenmpi-dev libpcap-dev
```

Extract the downloaded OMNeT++ files:

```
$ tar zxvf omnetpp-4.2.1-src.tgz
```

Setup the environment variables:

```
$ cd omnetpp-4.2.1
$ . setenv
```

Then run configure:

```
$ ./configure
```

Finally, compile the simulator:

```
$ make
```

Make sure that OMNeT++ has been installed successfully through running one of the sample simulations packaged with OMNeT++.

```
$ cd samples/dyna
$ ./dyna
```

– **Install INET Framework**

Download the corresponding version of INET

```
https://inet.omnetpp.org
```

Extract the downloaded files:

```
tar xjf inet20111118-src.tar.gz
```

Then start the OMNeT++ IDE by entering

```
$ omnetpp
```

Then import INET as an existing project

```
File -> Import... -> General -> Existing Projects into Workspace
```

Select the directory you previously extracted the INET archive to as root directory. Now INET should appear within the list of projects.

```
Press Finish
```

Start compiling by entering

```
Ctrl + B
```

- **Install SCADA-SST**

  Extract SCADA-SST files

```
tar xzf SCADA-SST-src.tar.gz
```

  Start the OMNeT++ IDE and import SCADA-SST as an existing project

```
File -> Import... -> General -> Existing Projects into Workspace
```

  Select the directory you previously extracted the SCADA-SST archive to as root directory. Now SCADA-SST should appear within the list of projects.

```
Press finish
```

  Start compiling by entering

```
Ctrl + B
```

2. **Connecting SIMATIC S7 PLCs with SCADA-SST**

   To integrate physical Siemens SIMATIC PLC with simulated nodes; either master or slave nodes; inside SCADA-SST kindly follow the instructions in the following document.
   http://www.controltechnology.com/Files/common-documents/application$_notes/Communi$
   $to - Simatic - S7 - using - open - modbus$

Vitae

- Asem Abdo Esmail Ghaleb
- Born on July 1st, 1985 in Taiz, Yemen.
- Obtained Bachelor of Science (BSc) degree in Computer Science from Taiz University, Yemen in July 2007. Very Good with honor.
- Working for Yemen Mobile since March 2011 as Database and ERP Administrator
- Worked for Social Fund for Development-Yemen from August 2007 to February 2011 as Senior IT Officer.
- Submitted this thesis to fulfil the requirements of his Master degree in Security and Information Assurance from King Fahd University of Petroleum & Minerals.
- **Research Interest:** Computer & Network Security, Industrial Control Systems Security, Machine Learning.
- **Publications**
  El-Alfy, El-Sayed M., and Asem Ghaleb. "Biobjective NSGA-II for optimal spread spectrum watermarking of color frames: Evaluation study." Computational Intelligence in Cyber Security (CICS), 2014 IEEE Symposium on. IEEE, 2014.
- **Seminars**
  SCADA Cyber Security
  King Fahd University of Petroleum and Minerals November 25, 2015
- **Contact Information:**
  E-mails: g201305010@kfupm.edu.sa
  aalmekhlafy@gmail.com
  linkedin: https://www.linkedin.com/in/asem-ghaleb-02286832