Improving SCADA Control Systems Security with Software Vulnerability Analysis

GIOVANNI CAGALABAN, TAIHOON KIM, SEOKSOO KIM
Department of Multimedia
Hannam University
Ojeong-dong, Daedeok-gu, Daejeon 306-791
KOREA
gcagalaban@yahoo.com, taihoonn@empas.com, sskim0123@naver.com

Abstract: - Cyber security threats and attacks are greatly affecting the security of critical infrastructure, industrial control systems, and Supervisory Control and Data Acquisition (SCADA) control systems. Despite growing awareness of security issues especially in SCADA networks, there exist little or scarce information about SCADA vulnerabilities and attacks. The emergence of Internet and World Wide Web technologies integrated with business systems brought a growing concern about the security and safety of the SCADA control systems. This research aims to addresses the issues regarding security and vulnerability testing. Software program was created to simulate the vulnerability testing and carry out assessment methodologies to test existing SCADA software design and implementation. This paper provides an analysis of potential software vulnerabilities, and security concerns including recommendations toward improving security for SCADA control systems. It also provides recommendations on key requirements and features needed for the enhancement of security for SCADA control systems.

Key-Words: - SCADA, control system, security, security framework, vulnerability analysis

1 Introduction

SCADA control systems are common industrial control systems used to gather data from sensors and instruments located at remote sites and to transmit data at a central site for either control or monitoring purposes [1]. With a computer system monitoring and controlling a process which can be industrial, infrastructure or facility-based, SCADA networks were developed with little attention being paid to security. As a result, many SCADA networks may be susceptible to attacks and misuses.

Historically, SCADA control systems were isolated from other information technology systems. Connection to the Internet is new and many specialists agree that exposing control systems to the Internet is not a good idea. However, without any connection to the Internet these systems are still vulnerable to external or internal attackers that can exploit vulnerabilities in software such as operating systems, custom and vendor software, data storage software, databases, and applications. These systems evolved from static to dynamic systems. The increased connectivity to Internet and mobile device technology has also a major impact on control systems architectures. Standardization and use of open market technologies are current requirements in control systems. Modern products are often based on component architectures using commercial off-the-shelf products (COTS) elements as units. This architecture leads to control systems that are becoming “very complex software applications” with the following characteristics [2]: time critical, embedded, fault tolerant, distributed, intelligent, large, open and heterogeneous.

There exist still little or scarce information about SCADA vulnerabilities and attacks, despite the growing awareness of security issues in industrial networks. Regarding the case of information technology security, most owners and operators are often unwilling to release attack or incident data. Yet, these sensitive data are not public repositories of advisories and vulnerabilities in industrial devices unlike information technology products and protocols. Even though some vulnerability testing and research are being conducted in this area, very little has been released publicly.

To solve the limitations in SCADA security, this research aims to perform vulnerability assessment methodologies to test the existing SCADA software design and implementation. With the growing concern about the security and safety of the SCADA control systems, this paper provides a relevant analysis of most important issues and a perspective on enhancing security of these systems. This also discusses key developments that mark the evolution of the SCADA control systems along with the increase. Further, this describes key requirements and features needed to improve the security of the current SCADA control systems.

The rest of this article is organized as follows. Section 2 discusses related works. In section 3, we present the current SCADA configuration. Specifically, a prototypical security testbed is illustrated for an attack scenario along with the corresponding attack algorithm.

Section 4 analyzes the vulnerability of SCADA systems by presenting a vulnerability assessment result. It is followed by the security approaches and key requirements for improving SCADA security in Section 5. Section 6 presents a security framework and finally, section 7 presents the conclusion of the article.

2 Related Works

Numerous ongoing research and assessment activities have revealed an effective methodology for identifying vulnerabilities and developing assessment methods to secure SCADA systems. Software tools used to determine known vulnerabilities in traditional IT systems have been widely available. The market for these vulnerability scanners has been significant and products such as Nessus, FoundScan and Internet Security Scanner (ISS) have been popular with IT administrators trying to locate unpatched computers on their networks.

Several test tools that have been successful in locating new vulnerabilities in network devices based on grammar and fuzzy techniques are also been developed in academic researches. Considerable work has been done by the PROTOS project group [3] and by Tal, Knight and Dean [4]. Each considers the syntax-based generation of protocol data units that translates into a single test packet to be sent to the device under test. Their methods have proven effective in finding vulnerabilities however, they only allow for the construction of simple single-packet test cases.

SCADA protocol vulnerabilities, specifically the Modbus protocol, was analyzed by Byres [5] and he suggested the use of attack trees to define a series of attacker goals, determine possible means to achieve that goal and identify the weak links of the system. He identified some robustness issues the lack of command and session structure as well as simplistic framing technique.

Currently, Modbus-based SCADA systems have no existing solutions that address specifically the Modbus protocol over Ethernet links. Despite the inherent lack of security in the design of Modbus, no security tools exist that are geared toward the detection of malicious Modbus traffic. Despite efforts to improve and provide guidance to help ensure program activities address real control system security issues, still there are very little security tools that have been released publicly.

3 SCADA Network Configuration

The transmission of data and control commands between a Master Terminal Unit (MTU) and a Remote Terminal Unit (RTU), referred to as SCADA communications, is carried over a variety of media, including Ethernet, corporate frame relay, fiber channel, Cellular Digital Packet Data (CDPD) systems, microwave signals, direct satellite broadcast, and many licensed or unlicensed radio systems. As shown in Figure 1, a typical SCADA architecture usually consists of a central computer system that is generally redundant or fault tolerant and communicates using one or more possible existing network technologies, to various RTUs that are connected with the field-based process equipments.

SCADA systems are exposed to the same cyberspace threats as any business system because they share the common vulnerabilities with the traditional information technology systems. Most SCADA systems are not protected with appropriate security safeguards.

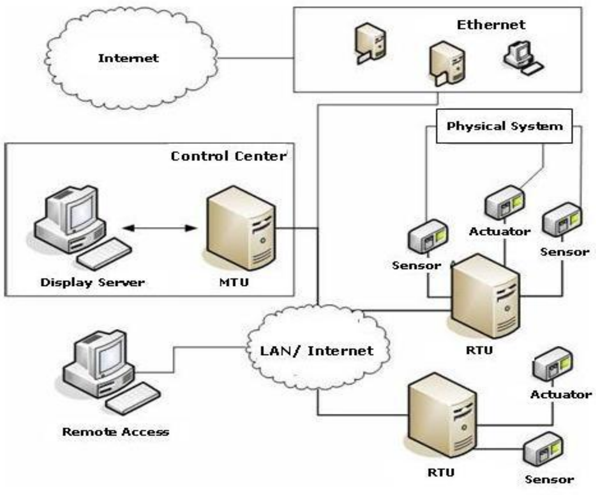

Figure 1. Typical SCADA Architecture

3.1 SCADA Security Testbed

This research designs a simple prototypical SCADA security testbed to be used for vulnerability assessment. A prototypical SCADA master and slave programs were used to simulate the serial communication between a SCADA master station and RTUs or slaves. This will allow assessment of vulnerabilities and security configurations of SCADA software used in industries.

The prototypical security testbed consists of one MTU that communicates with several RTUs. The system used SCADA software for process monitoring and control. RTUs are running a Modbus communication driver to communicate and exchange data with the MTU. Security attacks were simulated using direct access to the infrastructure. Figure 2 shows the prototypical SCADA security testbed for vulnerability analysis.

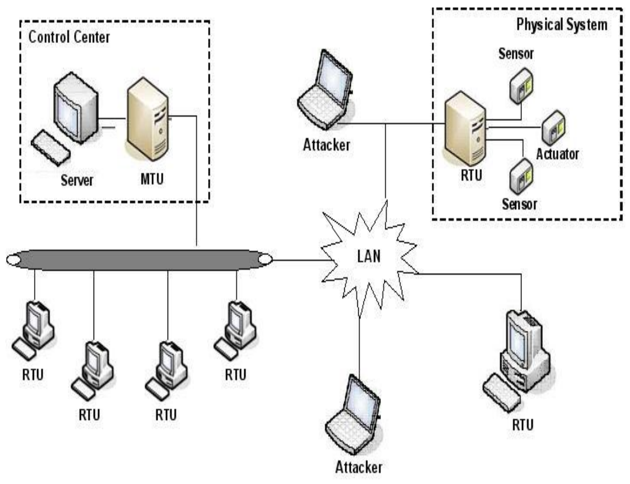

Figure 2. Prototypical SCADA Security Testbed

Security vulnerabilities and attacks could be at different levels – software controlling or controlled device, application, storage, data access, LAN, enterprise, Internet, communications as shown in Figure 1. SCADA systems are now adopting Web technology (ActiveX, Java, etc.). However, Web applications are an interesting target for cyber attacks that are increasingly automated. Web is the dominant development platform for software, but Web-based secure software is immature. In an average month, Web vulnerabilities accounted for 60% of the total vulnerabilities counted during that month of 2006 [6].

3.2 Attack Scenario

In this research, we assume that the prototypical SCADA system uses Modbus communication as represented by RS232, RS422 and RS485 communication standard. Modbus protocol was used since it lack inherent security that any moderately skilled hacker would be able to carry out a large variety of attacks if system access can be achieved.

Here, we utilized RS485 and RS232 standards to establish and examine the communication between SCADA Master and RTUs. In Modbus communication, there are two options possible: the installation of interface devices (PCI or PCMCIA type) and use of an RS485 communication converter connected with an RS232 standard interface connected to a computer.

RS232 communication device was used as a medium for serial transmission of data between SCADA master and slave. A man-in-the-middle physical configuration is shown in Figure 3 wherein the SCADA master and slave is connected with an intruder that eavesdrops on the network traffic. The man-in-the-middle computer serves as an intruder to perform sniffing and fault injection through the use of a developed program. The goal of this attack was to analyze the communication link between the SCADA communication port and the RTU and develop a means to send software-injected faults to change state in the either the master or slave operation.

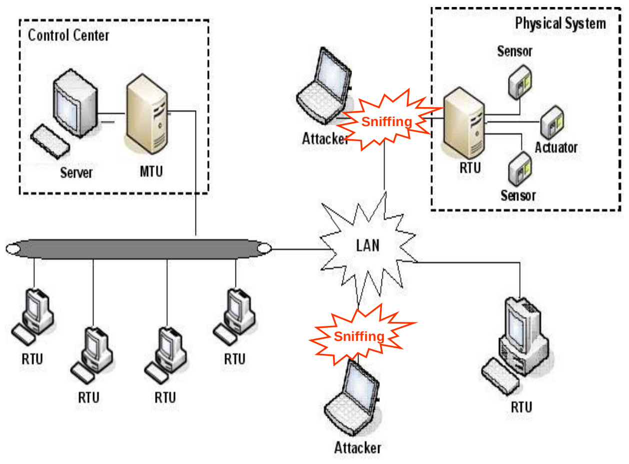

Figure 3. Sniffing Attack

In this approach, various attacks which include sniffing, replay and spoofing which include faults that are injected by a program into the system thereby changing its current status. Changing the status of the system environment during analysis would provide assessment results on how it responds and whether there will be security vulnerabilities that can be detected. If not, then the whole system is considered secure and reliable.

3.3 Attack algorithm

In order to determine the security vulnerabilities, it is necessary to inject faults through software that manifest themselves as security defects in systems at the application software level. One thing to note is that those faults should emulate the real system faults appropriately. The elements to be considered are internal elements and external elements. Internal elements refer to those elements that are part of the program's code and data such as variables and stack. While external elements are elements that are external to a program's code and data such as file, media, other party, and network.

One factor to consider in making and implementing a secure system is the nature of having shared elements. A program is not alone in accessing and changing these internal and external elements. Other factors, such as other users, may access and change the whole elements as well.

To provide high confidence in the validity of the security vulnerabilities caused by faults to be injected, the approach described here models the software system at a high level. Software implemented fault injection at this level emulate what a "real" attack scenarios are being done.

To implement fault injection, the software approach created can be outlined in the following steps. The program was written in C language, a language which supports serial interface and communication. The attack algorithm is shown in order below:

1. Set the variables **counter** and **size** initially to 0.
2. For each test case, do step 3 to 9.
3. For each interaction point in the execution trace, decide if the program asks for an input. If there is no input, only inject direct environment faults; if there is an input; inject both direct and indirect way of injecting faults.
4. Decide the object where faults will be injected.
5. For each fault in the list, inject it before the interaction point for the direct environment faults; inject each fault after the interaction point for the indirect environment faults since in this case, we want to change the value the internal entity receives from the input.
6. Increase **size** by 1.
7. Detect if security policy is violated. If violated, increase **counter** by 1.
8. Calculate interaction coverage. If the test adequacy criterion for interaction is satisfied then stop else repeat steps 3-9 until the adequacy criteria for interaction coverage is achieved.
9. Divide **counter** by **size** yielding to obtain the vulnerability assessment score for the application program.

4 Vulnerability Analysis

To assess the possible security vulnerabilities, some method of assessing and rating the risk of any vulnerability is needed. The impact in this case is an expression of the likelihood that a defined threat will exploit a specific set of vulnerability of a particular attractive target to cause a given set of consequences. Sniffing activities which include injected faults seems to only have little cost or apprehension concerns.

The purpose of the of the analysis is to determine the values associated with the goal of attack to give a better understanding which also reflect the classification of the faults to compromise the whole system. These also indicate where security recommendations are required. Table 1 shows the results of the vulnerability assessment of sniffing activities and compromising the SCADA system.

Impact analysis of the various activitiess such as sniffing, replay and spoofing including fault injection indicates that the avenues of attack depend on the ability of the attacker to gain SCADA access and identify the existing protocol. If sufficient security measures are put in place to block all possible intrusion points into the SCADA system, then the chances of a successful attack are greatly reduced. Unfortunately, in this research the predominant security effort in most SCADA facilities tends focus on attacks via the Internet or through the business network. This leaves open attacks from other intrusion points such as remote field stations, the SCADA transmission infrastructure, or wireless control network connections.

Attacker's Goal	Attack Type	Impact Rate	Security Recommendation
Gain SCADA system and media	Internal	Low	Authentication and integrity
Compromise Master HMI	Internal	High	Authentication and session
Gain SCADA through remote access	External	Low	Authentication and transmission media
Network access on master	External	Very Low	Confidentiality and authentication
Disable master	Internal	Moderate	Authentication and integrity
Master attack through slave	External	Moderate	Frame format and authentication
Disable slave	Internal	Very High	Frame format and authentication
Sniff through media	Internal	Very Low	Confidentiality
Sniff on protocol session	Internal	Moderate	Confidentiality

Table 1. Vulnerability Assessment

Vulnerability analysis provides important and critical information for different attack scenarios. For example, these include information such as type and the impact of the attack that would allow an attacker to successfully break the system. Based on the information, the plausibility of each of the attacks can be considered. If one or more attacks are estimated to be likely or possible with improvement in technology over time, the vulnerability analysis would indicate that the current security model must be upgraded or revised. Additionally, various security threats are also identified. Each threat is then analyzed by constructing various scenarios revealing what an attacker can do to the SCADA system.

5 Improving the SCADA Security

SCADA control systems are becoming more complex and there is a greater need of compliance for safety, quality of service, and security of systems and data. SCADA security design and information security management can be improved by applying a wide range of control principles and methods as well productivity control, involving decision making under uncertainty with increased levels of decision support. Therefore, the improvements for SCADA security have to be broad - at the systems level - and detailed - at the component level.

5.1 Holistic Approach to Software Security

SCADA software security must be viewed holistically. It is achieved through the combination of effective people, process and technology with none of these three on their own capable of fully replacing the other two entities. This also means that just like software quality in general, software security requires that we focus on security throughout the application's life cycle.

5.2 Software Security Key Requirements

SCADA systems need to have a strategy that supports not only management of knowledge and training but security knowledge that include policy, standards, design and attack patterns, threat models, code samples, reference architecture and security framework [7]. Key requirements for software security are discussed in the following.

Security requirements in the software development cycle

Developing security architecture and engineered approach to the problem are recommended by Saydjari [8] because current technology is not enough to prevent cyber attacks. Developing requirements for control systems with security features and use of simulation models based on a framework could improve the definition of requirements and reveal problems early in the software development cycle.

Compliance to standards for software development

Software development for control systems can be improved by following documents such as NIST published guidelines SCADA Security, Configuration, Guidelines, general assessment methods and tools for SCADA vulnerabilities and Holzman's rules [9].

6 Security Framework

Defining the security framework will helping the analysis of root causes of vulnerabilities. In this paper, SCADA security framework will be defined as a common reference to view software security problems and solutions. Security issues include the following:

Authorization & Authentication: Considered here are those that deal with appropriate mechanisms to enforce access control on protected resources in the system. Authorization flaws could result in either horizontal or vertical privilege escalation. The usage of strong protocols to validate the identity of a user or component is considered here. Further, issues such as the possibility or potential for authentication attacks such as brute-force or dictionary based guessing attacks also fall within this context.

User & Session Management: This concerns how a user's account and session is managed within the application. The quality of session identifiers and the mechanism for maintaining sessions are some of the considerations here. Similarly, user management issues such as user provisioning and de-provisioning, password management and policies are also covered as part of this category.

Configuration Management: This will consider all issues surrounding the security of configuration information and deployment. For instance, any authentication and / or authorization rules embedded in configuration files or how the framework and application deal with error messages.

Data Protection in Storage & Transit: This includes handling of sensitive information such as social security numbers, user credentials or credit card information. It is also covers the quality of cryptographic primitives being used, required / minimum key lengths, entropy and usage of industry standards and best practices.

Data Validation: This is responsible for the most well known bugs and flaws including buffer overflows, SQL injection and cross site scripting. Length, range, format and type checking for inputs and outputs are considerations here.

Auditing and Logging: This concerns with how information is logged for debugging and auditing purposes. The security of the logging mechanism itself, the need and presence of an audit trail and information disclosure through log files are all important aspects.

6.1 Building the security framework

Vulnerability analysis can delineate a security framework that has a potential to guard against the attacks, which are threats to SCADA systems. In this research, software vulnerability analysis is presented as effective method for testing SCADA security framework. Figure 4 shows the proposed security framework. It is a modified version described by Stoneburner [10]. The figure shows an effective calculation of risk which requires a definition of mishap and identification of potential harm to safety. This is calculated as an impact of hazard multiplied by likelihood of mishap event happening.

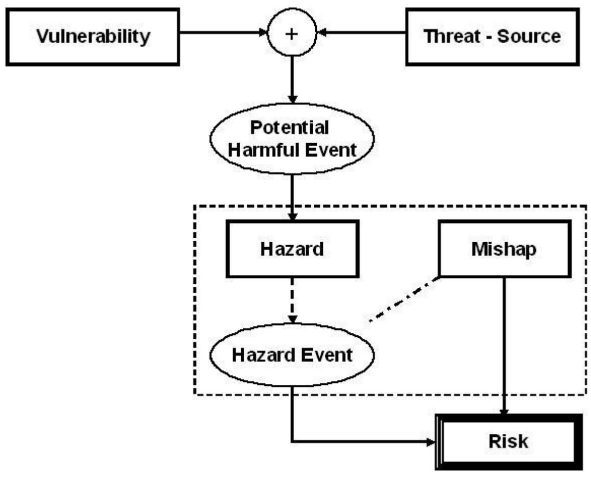

Figure 4. Security Framework

The effective calculation of risks for SCADA systems bridges the gap between security and safety. With the aid of integrated computer systems, the line that delineates security and safety is almost negligible.

7 Conclusion

The study indicated that the use of these vulnerability assessment in SCADA communications can significantly reduce the vulnerability of these critical systems to malicious cyber attacks, potentially avoiding the serious consequences of such attacks. The results of the study also indicate that the software implemented fault injection can be a very useful tool for modeling threats and vulnerabilities in a wide variety of systems especially SCADA control systems. However, the approach is not without its limitations. Lightweight approaches to threat modeling are useful for protocol designers, vendors, and users in an area that needs more exploration.

Implementing security features as those described above ensures higher security, reliability, and availability of control systems. Thus organizations need to reassess the SCADA control systems and risk model to achieve in depth defense solutions for these systems. The increasing threats against SCADA control systems indicate that there should be more directions in the development of these systems. A strategy to deal with cyber attacks against the nation’s critical infrastructure requires first understanding the full nature of the threat. A depth defense and proactive solutions to improve the security of SCADA control systems ensures the future of control systems and survivability of critical infrastructure.

Future researches need to be done for software security improvement. A truly effective security enhanced software development lifecycle has many parts to it and while they can be built over time, it can be achieved only when all the parts are in place. In getting there though, each additional part will result in a marked improvement in the security quality of your software applications.

Corresponding Author

Seoksoo Kim (sskim0123@naver.com)

References:
[1] http://en.wikipedia.org/wiki/SCADA
[2] Sanz, R., Arzen, K. E., Trends in software and control. *IEEE Control Systems Magazine*, Vol. 23, No. 3, 2003, pp. 12-15.
[3] Andrews, M., The state of web security. *IEEE Security & Privacy*, Vol. 4, No. 4, 2006, pp. 14-15.
[4] Steven, J., Adopting an enterprise software security framework. *IEEE Security & Privacy*, Vol. 4, No. 2, 2006, pp. 84- 87.
[5] Byres, E., Understanding Vulnerabilities in SCADA and Control Systems, October 2004
[6] Andrews, M., The state of web security. *IEEE Security & Privacy*, Vol. 4, No. 4, 2006, pp. 14-15.
[7] Steven, J., Adopting an enterprise software security framework. *IEEE Security & Privacy*, Vol. 4, No. 2, 2006, pp. 84- 87.
[8] Saydjari, O. S., Defending cyberspace. *IEEE Computer*, Vol. 35, No. 12, 2002, pp. 125.
[9] Holzmann, G. J., The power of 10: Rules for developing safety-critical code. *IEEE Computer* Vol. 39, No. 6, 2006, pp. 95-97.
[10] Stoneburner, G., Toward a unified security/safety model. *IEEE Computer*, Vol. 39, No. 8, 2006, pp. 96-97.