Identifying Vulnerabilities in Industrial Control System Protocols using Mutation-based Fuzzing

Aditi P. Thakur[1]*, Bharati A. Dixit[2]

***Corresponding Author: -** Aditi P. Thakur

*[1] School of Computer Engineering and Technology, Dr. Vishwanath Karad MIT World Peace University, Pune., E-mail:- 1032200049@mitwpu.edu.in

[2] School of Computer Engineering and Technology, Dr. Vishwanath Karad MIT World Peace University, Pune., E-mail:-bharati.dixit@mitwpu.edu.in

Abstract

As we are heading towards the future of technological developments, we see that industrial automation is getting to the forefront. The combination of information and operation technology is making remarkable wonders. But the price to be paid for this revolution is the security of old mechanisms as they were not built to work with such advanced systems. Programmable Logic Controllers (PLC) are being used in the industry to control field devices for a long time now. These devices tend to be easy targets for cyber-attacks. Stuxnet and Triton are well-known examples of PLCs being compromised. Many of those PLCs are still working on the proprietary protocols. Through this research, we try to acknowledge the security risks of such proprietary protocols. These protocols are being used by many devices which support ethernet connectivity. We develop a plugin that consists of two utilities: packet sniffer and fuzzing section. The sniffer is used to analyze packets. It studies various header fields which can become a target for manipulation. The information gathered using this sniffer is then fed to the fuzzing section to perform a fuzz test. We implement the mutation-based fuzzing approach to study various security aspects of these protocols. Using this plugin we can study various security risks possessed by protocols. This will further help in securing the communication channels in industrial control systems.

Keywords - PLC, SCADA, ICS Security, Fuzzing, ICS Protocols

DOI Number: 10.14704/nq.2022.20.9.NQ440058 **Neuro Quantology 2022; 20(9):544-551**

INTRODUCTION

The term "industrial control systems" (ICS) refers to the integration of hardware and software with network connectivity to support critical infra-structure. ICS technologies include supervisory control and data acquisition (SCADA), distributed control systems (DCS), industrial automation and control systems (IACS), programmable logic controllers (PLCs), programmable automation controllers (PACs), remote terminal units (RTUs), control servers, intelligent electronic devices (IEDs), and sensors. SCADA is a monitoring and control system used in industrial settings. SCADA systems are critical because they collect and process real-time data, which aids in efficiency [1]. SCADA systems are now present in every other area, including power plant production and operation, water and wastewater facilities, transportation sectors, and Other critical infrastructures.

SCADA systems rely on Programmable Logic Controllers (PLC). PLCs (programmable logic controllers) are computerized devices that automate industrial, mechanical, and electrical tasks. Controlling machinery in industries is a common use. It performs the critical duty of establishing a link between the HMI and the field devices. Various protocols are used in this connection. A few open-source protocols used in ICS include Modbus, Profibus, Profinet, and others. However, because the operating technology has been in use for a long time, there are still many devices that employ proprietary protocols.

1. REQUIREMENTS OF CYBERSECURITY IN ICS

In the past, most machinery and technological Components were dumb, and those that were

computerized utilized proprietary protocols. The networks to which they belonged kept them air-gapped and insulated from the outside world. This has changed over time, and today's ICS components are frequently, whether directly or indirectly, connected to the internet. Because of improvements in smart sensor technologies and wireless networking, the combination of operational technology (OT) with information technology (IT) has become desirable and cost-effective. Despite the benefits of enhanced speed, improved responsiveness to changing situations, and increased reliability that IT/OT convergence has brought, security problems persist. Such events include the Stuxnet attack, Logic Locker, and others.

2.1 Purdue Model

The Purdue model is an ICS security structural paradigm that encompasses physical processes, sensors, supervisory controls, operations, and logistics. Purdue University's Theodore J. Williams and others created the Purdue Enterprise Reference Architecture in the 1990s. It describes the many levels of infrastructure used in industries, as well as their security elements.

This study focuses on protocols used to communicate between Level 1 and Level 2 devices, such as PLCs and HMIs. These protocols provide a Client-Server architecture, in which the HMI device serves as the client and the PLC serves as the server. The primary purpose of our research is to identify vulnerabilities in various such protocols using a mutation-based fuzzing technique.

2.2 Fuzzing

Fuzz testing, often known as fuzzing, is an automated approach to detecting software problems. The main idea behind this approach is to generate a large number of bogus inputs, send them to the target being tested, and then look for any unexpected responses. A fuzzer is a piece of software that does these functions. Fuzzers are categorized into three forms based on how the incorrect inputs are generated: mutation-based fuzzers, generation-based fuzzers, and evolutionary fuzzers. Mutational fuzzers create distorted inputs by randomly altering samples of genuine inputs. Generational fuzzers are more complex and are employed when previous information about the target system is available. They create inputs from scratch while well understanding the protocol's structure. Evolutionary fuzzers gradually learn the proper structure by referencing comments from submitted inputs.

In our research, we use a mutation-based fuzzing technique. Section 6 goes into great length about the process.

The Purdue Model is depicted graphically in the diagram below:

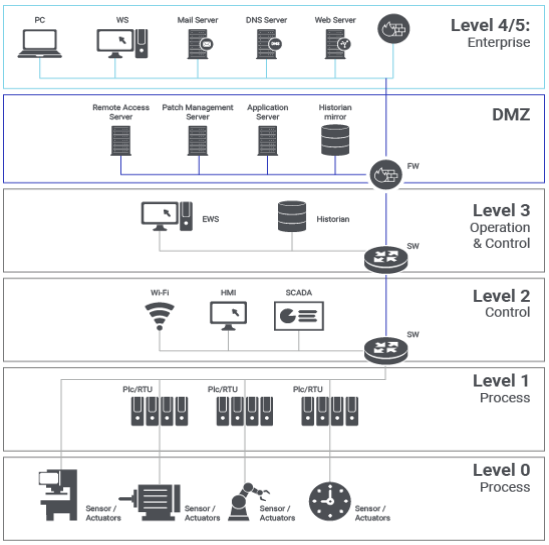

Fig. 1 Purdue Model for ICS Security

Source: Internet

3. LITERATURE SURVEY

Over the period, various studies have been carried out to understand the risk caused by the interconnectivity of various systems in industrial automation. Understanding the vulnerabilities and loopholes possessed by SCADA systems is vital to defending against hazardous situations.

M. Nawrocki et al (2020), in [1], studies and compares a wide range of internet traffic flowing through Internet Service Provider (ISP). This experiment is carried out for analysis of ICS traffic which can be helpful to study the packet flow and identify the weakness in the ICS network. In this experiment received data is

thoroughly classified into ICS and non-ICS packets. Then sanitization is applied to ICS packets to get all the significant information for the ICS domain. This helps in analyzing the headers and studying the data flow with more accuracy. This resulted in the discovery of previously undetected ICS devices as well as bringing awareness regarding potential attacks possible on the ICS network.

D. Upadhyay and S. Sampalli (2020) in [2], discuss different types of vulnerabilities in SCADA. Real incidents are used as a reference to study the impact of various actions performed by intruders as well as the authorized person. It covers a wide area of the study conducted in various aspects. Some of these concerned topics are software configuration, product configuration, network configurations, security devices, etc. There are various attack cases mentioned in the paper to explain the effect and presence of potential vulnerabilities in the SCADA systems. Apart from theoretical study, various recommendations are also provided to take basic steps in safeguarding the SCADA system.

IEC 62443 standards provide a foundation for cybersecurity in industrial automation and control systems. Hicham Lalaoui Hassani et al (2021), in [3], propose a new approach for risk evaluation of IIoT devices conforming to the IEC 62443 standards. This process is used to perform a risk evaluation of a system. The risk is calculated by taking into consideration how often attacks take place and how severe is the impact of an attack. This leads to the calculation of a score for each attack and determining the level of risk for the system. Therefore, it helps in detecting common vulnerabilities in industrial automation systems.

Ercan Nurcan Ylmaz et al (2018), in [4], describe an experiment carried out on Programmable Logic Controllers (PLCs) by performing a Denial of Service (DoS) attack to analyze the behavior of PLCs and simulate the real ICS environment. The study also emphasizes the harmful impacts of insiders, particularly in the event of cyber assaults against ICS, in evading system security measures and the discovery phase. The research concludes with the importance of network topology and regular monitoring of the devices in ICS. Therefore, it is observed that authorized access to PLCs and proper analysis of the device behavior with reporting can help prevent unsolicited situations.

Abraham Serhane et al (2018), in [5], also guides toward the vulnerabilities and threats possessed by Programmable Logic Controller-based systems. It provides an overview of the SCADA system, its components as well as the attack vectors concerning each component. It also sheds light on some possible reasons for the lack of security in SCADA systems and the possibilities of achieving a secure environment by mitigating the risks.

We also find a wide research area directed towards the security of protocols used in ICS. Various studies focus on the application of fuzzing techniques to identify vulnerabilities in the protocols.

Y. Yu et al (2020), in [6], introduces a state-driven network protocol fuzzer. It is developed after analyzing various available fuzzers like AFL and BooFuzz. The SGPFuzzer is a state-driven smart gray box protocol fuzzer that can comprehensively test stateful network protocols. SGPFuzzer in particular implements a learning system for protocol states and transitions that is automated as well as a recording mechanism that can learn and record a Fully automated state/state machine. In addition to this, SGPFuzzer comprises a series of clever mutation techniques that are piled on top of each other, This may be used to run the server under test (SUT) more efficiently in the more critical areas of the code.

H. Zhao et al (2019), in [7], propose a SeqFuzzer framework for protocol fuzzing from a deep learning perspective. It uses Long Short-Term Memory (LSTM) model for learning protocol structure and timing. This approach is a combination of deep learning and fuzzing technology. This paper contributes to proposing seq2seq-based fuzzing and being protocol independent it can find vulnerabilities in various ICS protocols. For proving the usability of their proposal, the authors have tested the Ether CAT

protocol and found various vulnerabilities. This paper illustrates how various technologies can be merged for better results.

Proprietary protocols are difficult to fuzz as not much information is available regarding the protocol. Thus, it becomes a need to develop a fuzzer according to the needs and configurations. M. Niedermaier et al (2017), in [8], introduce a framework called Propfuzz for proprietary protocol fuzzing. It is also capable of monitoring the controller. H. Gascon et al (2015), in [9], as well describes one such fuzzer for proprietary ICS protocols using black-box fuzzing.

Z. Luo et al (2020), in [10], proposes a generation-based fuzzing approach using a coverage-guided technique. For evaluating the results of this experiment authors have built Peach* on the foundation of Peach Fuzzer. This technique has shown a significant improvement in vulnerability detection compared to other fuzzers when tested on various ICS protocols like Modbus, DNP3, etc.

Modbus, being an open-source and widely used protocol has been a target of various malicious activities. Thus, there has been significant research done to identify various vulnerabilities in this protocol.

Kevin Lamshöft and Jana Dittmann (2020) in [11], introduce an evaluation framework for identifying hidden messages in ICS network protocols. This framework is tested on Modbus TCP protocol as it is a popular one used in ICS. The research approach uses pattern-based taxonomy for network information hiding proposed previously. This framework tests and identifies network hiding patterns. Thus, it helps in discovering various ways attacks can take place.

Fovino et al (2009), in [12], proposes a more secure version of the Modbus protocol which provides integrity, authentication, non-repudiation, and anti-replay services. The experiment is performed on the power-plant testbed. Results show that this version of Modbus provides good security without any significant overhead.

Christopher Parian et al (2020), in [13], focuses on performing attacks on Modbus protocol and finding vulnerabilities. The two major attacks performed are the Man-in-the-middle attack and malware infection over a master. Scapy is used to exploit the weaknesses of Modbus over TCP/IP. The result turned out to be master falling prey to the malware and attacks being performed successfully. Thus, indicating the vulnerabilities in Modbus protocol. B. Chen et al (2015), in [14], also emphasizes the Modbus protocol. It presents a study of the cyber-physical test-bed framework and is a simulation of a real-time system to find the vulnerabilities. The results constitute the impact various attacks on the Modbus protocol show on the whole system.

As Modbus is an open-source protocol lot of research work is conducted on it. But there are also various proprietary protocols being used in the industry today which tend to possess the ability to be a vulnerable entry point if exploited tactfully.

4. SYSTEM ARCHITECTURE

This system is a combination of two sub-units. The first one is a network packet sniffer and the second is a protocol fuzzer. Figure 2 explains the architecture of this system.

The network sniffer is used to record the packets sent and received between the HMI machine and the target PLC. To begin the communication, the user must provide the source and destination IP addresses as input. Once the startup procedure is complete, the HMI machine may query the PLC for the required data. The sniffer intercepts and saves data delivered and received over this network in a pcap file. A single packet has several parameter values, such as packet type (request or reply), register offset, data to be saved, and so on. The protocol packet structure and these parameters are then utilized in the fuzz portion to convey altered data.

The fuzzing section conducts duties such as sending faulty packets to observe the target PLC's activity. The preceding process's data is utilized to build packets containing random

information, which are subsequently delivered over the existing network. The target PLC then reacts to the queries given by the HMI machine. This answer is evaluated to identify any vulnerabilities in the supplied protocol.

5. TEST ENVIRONMENT

The testing environment consisted of the following hardware and software components:

1. PLC: We used a generic Programmable Logic Controller which supports proprietary protocols.
2. HMI Computer: We converted a desktop with Windows 10 OS installed into an HMI. For this experiment, we installed all the required software applications on this machine.
3. Switch: A 24-port switch with category 5 ethernet cables was used. The switch creates a communication channel between HMI and PLC.
4. Software applications: We used various tools for information gathering purposes like VT Scada, Codesys, and Wireshark 6.2. Visual Studio Code 2019 was used as IDE for programming.
5. Programming language: This application was completely coded using the Python3 programming language

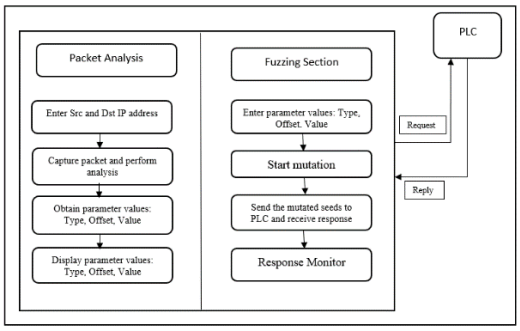

Fig. 2 System Architecture

6. METHODOLOGY

This experiment was conducted in two stages. We created a network packet sniffer in the first phase to monitor the traffic between the Human Machine Interface (HMI) and the Programmable Logic Controller (PLC). The second phase involved communicating with the Programmable Logic Controller (PLC) directly via the plugin running on a desktop without the use of proprietary software.

To begin, we utilized a simulation tool called VT SCADA to learn how HMI is used to program PLC. Using this tool, basic programs such as an ON/OFF switch and temperature control were constructed. These programs were used to investigate how HMI requests are handled and output is shown on the screen. The next step was to build a packet sniffer and investigate the information exchange in real-time.

Scapy, a Python tool, was used to create the packet sniffer. The sniffer was used to examine the packet and determine its fundamental structure. As a client, the computer transmits the initial packet to establish the connection. PLC, as a server, answers the plugin's request. This startup procedure is only performed once. We retrieve the data from the registers after a solid connection is established.

Then, we created a plugin to communicate with the PLC without any help from proprietary or licensed software. This is the fuzzing phase. The information collected from the sniffer packets is given to the fuzzer application. This is known as the seed input. The fuzzer software creates numerous altered packets, or seeds, and sends them to the target PLC. Manipulation is carried out in many fields of the package. This is the seed generation process.

6.1 Seed Generation Process

The fuzzer provides seed to the target system or code. The seeds in this scenario are the packets sent over the network. And, to generate a variety of such seeds with legitimate but random inputs, we used packet manipulations over many fields. Before proceeding, consider the numerous packet fields that may be altered to interrupt communi-cation and execute attacks. We concentrated on sequence number manipulation, data mani-pulation, and register value manipulation in our research. Thus, the actions of reading data from a certain memory region and writing data to a specified place were both properly accomplished.

The packet structure and accompanying information for numerous proprietary protocols are not publicly available. Thus, constructing a packet sniffer aided in breaking down and compre-hending the fundamental structure of these proprietary protocols to undertake fuzz testing.

We used the plugin to send a request suggesting one of the procedures listed below. The PLC decodes the request using the parameter values in the packet's header. The reaction from the PLC was watched, and the findings were obtained.

6.2 PLC memory structure

PLC has its memory structure briefly bifurcated into three types:

(i) System Memory: The user program's addressing components, such as addressing inputs, outputs, timers, counters, and bit memory, can all be stored in system memory.

(ii) Task memory: During program execution, some user program components are stored in task memory, which is volatile storage. Some components of the user project are copied by the CPU from program memory into task memory.

(iii) Program memory: User programs, data, and configuration are stored in non-volatile program memory. Consequently, a program will first be stored in the program memory section when it is downloaded to the CPU.

The packet manipulation process was done for four major operations:

1) Read Single Register

2) Read Multiple Registers

3) Write Single Register

4) Write Multiple Registers

For all the operations mentioned above, response time was monitored. The next section discusses the observations made during this experiment.

7. RESULT

The results primarily focus on the response time required by the PLC to perform the action requested by the HMI machine.

As discussed in the methodology, we monitored the response time for every operation performed. Figure 3 shows the graphical representation of the results. As observed during the process, the operation of reading multiple registers performed on task memory has less response time compared to system memory and program memory. Task memory also shows less response time on average.

We performed the above-mentioned four operations in a set of 50 attempts. Figure 4 shows the graphical representation of successful and unsuccessful attacks. It was observed that not all were successfully implemented. Operations like reading single register values and writing single register values are implemented successfully on every attempt. But that is not the case with reading and writing multiple registers.

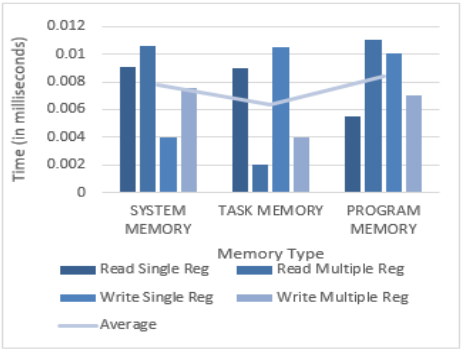

Fig. 3 Graphical representation of response time

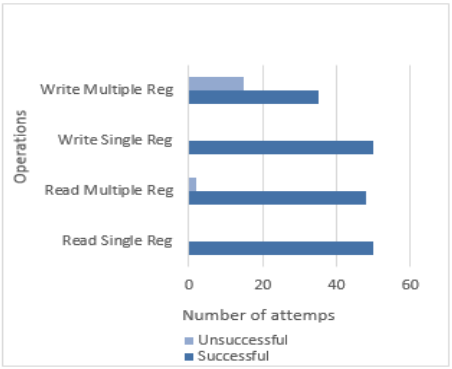

Fig. 4 Graphical representation of success rate of each operation

8. CONCLUSION AND FUTURE SCOPE

This research has led us to understand how important proprietary protocol security is. Although these protocols were invented a long time ago when cyber security was not a big concern, in today's world they still hold the position of useful communication modes and need to be secured by all means. Various such protocols might not be well known today and are not in the competition as others but still, they possess as a threat to the device which supports them. As cyber-attacks don't require physical proximity to the network, target systems have become more vulnerable than before.

Using this approach, we can identify possible vulnerabilities in ICS protocols as well as various attacks, which can be performed on these protocols.

In future work, we would like to focus on enhancing the mutation techniques to obtain more varied and concrete results. The formation of mutation seeds largely depends on the information known before the process. Thus, having more knowledge about the protocol can be leveraged to test the protocol in all dimensions. We can create a generalized platform to learn proprietary protocols, and protocol learning capability and acquire maximum coverage.

9. REFERENCES

M. Nawrocki, T. C. Schmidt, M. Wahlisch, Uncovering Vulnerable Industrial Control ¨ Systems from the Internet Core, Proceedings of 17th IEEE/IFIP Network Operations and Management Symposium (NOMS) (2020).

D. Upadhyay and S. Sampalli, SCADA (Supervisory Control and Data Acquisition) systems: Vulnerability assessment and security recommendations, Computers & Security, Elsevier (2020).

Hicham Lalaoui Hassani, Ayoub Bahnasse, Eric Martin, Christian Rolanda, Omar Bouattane, Mohammed El Mehdi Diouri, Vulnerability and security risk assessment in an IIoT environment in compliance with standard IEC 62443, The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), Procedia Computer Science (191) (2021) 33-40.

Ercan Nurcan Ylmaz, Bünyamin Ciylan, Serkan Gönen, Erhan Sindiren, Gökçe Karacayılmaz, Cyber security in industrial control systems: Analysis of DoS attacks against PLCs and the insider effect, 6th International Istanbul Smart Grids and Cities Congress and Fair (ICSG), IEEE (2018).

Abraham Serhane, Mohamad Raad, Raad Raad,Willy Susilo, Programmable Logic Controller based systems (PLC-BS): vulnerabilities and threats, SN Applied Sciences, Springer Nature (2019).

Y. Yu, Z. Chen, S. Gan and X. Wang, SGPFuzzer: A State-Driven Smart Graybox Protocol Fuzzer for Network Protocol Implementations, in IEEE Access, vol. 8 (2020) 198668-198678, doi: 10.1109/ACCESS.2020.3025037.

H. Zhao, Z. Li, H. Wei, J. Shi and Y. Huang, SeqFuzzer: An Industrial Protocol Fuzzing Framework from a Deep Learning Perspective, 12th IEEE Conference on Software Testing, Validation and Verification (ICST) (2019) 59-67, doi: 10.1109/ICST.2019.00016.

M. Niedermaier, F. Fischer and A. von Bodisco, PropFuzz — An IT-security fuzzing framework for proprietary ICS protocols, International Conference on Applied Electronics (AE) (2017) 1-4.

Gascon, H., Wressnegger, C., Yamaguchi, F., Arp, D., Rieck, K. PULSAR: Stateful Black-Box Fuzzing of Proprietary Network Protocols. In: Thuraisingham, B., Wang, X., Yegneswaran, V. (eds) Security and Privacy in Communication Networks. SecureComm, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol 164. Springer (2015) Cham. Available: https://doi.org/10.1007/978-3-319-28865-9_18.

Z. Luo, F. Zuo, Y. Shen, X. Jiao, W. Chang and Y. Jiang, ICS Protocol Fuzzing: Coverage Guided Packet Crack and Generation, 57th ACM/IEEE Design Automation Conference (DAC) (2020) 1-6.

Kevin Lamshöft, Jana Dittmann, Assessment of Hidden Channel Attacks: Targetting Modbus/TCP, IFAC –Papers OnLine, 53(2) (2020) 11100-11107. Available:

https://doi.org/10.1016/j.ifacol.2020.12.258.

Fovino, I.N., Carcano, A., Masera, M., Trombetta, A. Design and Implementation of a Secure Modbus Protocol, In: Palmer, C., Shenoi, S. (eds) Critical Infrastructure Protection III. ICCIP. IFIP Advances in Information and Communication Technology (311) Berlin, Heidelberg, Springer (2009). Available: https://doi.org/10.1007/978-3-642-04798-5_6

Christopher Parian, Terry Guldimann, Sajal Bhatia, Fooling the Master: Exploiting Weaknesses in the Modbus Protocol, Procedia Computer Science (171) (2020) 2453-2458. Available: https://doi.org/10.1016/j.procs.2020.04.265.

B. Chen, N. Pattanaik, A. Goulart, K. L. Butler-purry and D. Kundur, Implementing attacks for Modbus/TCP protocol in a real-time cyber-physical system test bed, *IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)* (2015) 1-6.