No Attacks Are Available: Securing the OpenPLC and Related Systems

Wael Alsabbagh[*], Chaerin Kim[*], Peter Langendörfer[*]

Abstract: The use of Programmable Logic Controllers (PLCs) expands in industrial domains, which makes ensuring the security of Industrial Control Systems (ICSs) become paramount. The OpenPLC project, the first open-source initiative, provides flexible and cost-effective PLC solutions to build up affordable test-beds, as well as conduct experiments and academic researches. This project has wildly grown in the last few years, thus it is essential to address the most emerging security challenges it encounters. This paper introduces a new OpenPLC architecture, called OpenPLC Aqua, provided with a set of security solutions designed specifically to overcome the vulnerabilities that the current OpenPLC versions are prone to. The new OpenPLC architecture includes four security features: 1) user credentials encryption, securing the *Webserver*, Whitelisting and secure SSL/TLS communication channel. The OpenPLC Aqua software was tested against several attack scenarios that were feasible against the old OpenPLC versions. Our experimental results showed our enhanced OpenPLC software is secure and resistant against several attack scenarios e.g., authentication, injection, Man-in-the- Middle and replay attacks. The OpenPLC Aqua is publicly available and a proof of concept demo is also published with this paper.

Keywords: OpenPLC, Security Solutions, Mitigation Solutions, Industrial Control Systems.

1 Introduction

Programmable Logic Controllers (PLCs) play a crucial role in monitoring and controlling various industrial processes and applications e.g., manufacturing, energy management, transportation, etc [ALS2]. Real hardware PLCs are pricey and sometimes are unaffordable for scientists to design small experimental labs, and conduct their experiments or academic researches. The OpenPLC [ALV1] project provides an appropriate alternative solution for implementing low cost PLCs and Industrial Control System (ICS) environment. Its open-source and collaborative nature offers a unique platform for developers, researchers, and even industrial experts to contribute, innovate, and test new PLC based solutions and technologies. With a user-friendly interface, extensive documentation, and online community forums, the OpenPLC allows users to learn, exchange ideas, distribute PLC codes and compatibilize it with a wide range of hardware platforms, including popular Single Board Computers (SBCs) such as Raspberry Pi and Arduino [ALV2]. However, since the use of the OpenPLC has

[*]IHP – Leibniz-Institut für innovative Mikroelektronik, Frankfurt (Oder), Germany, (alsabbagh, chaerin, Langendoerfer)@ihp-microelectronics.com.

significantly grown in the last few years few years, it is essential to ensure the integrity, availability, and confidentiality of the project. Like other ICS components, the OpenPLC is susceptible to a range of security issues that can compromise the operation of OpenPLC based systems and disrupt the physical processes they control. These security issues arise from various factors e.g., software design flaws, insecure communication protocols, weak authentication protocols, and others.

One of the primary security concerns in OpenPLC is the potential for unauthorized access to the OpenPLC configuration [ALS1]. If an attacker gains access, he can maliciously modify the logic controlling industrial processes, potentially leading to physical damages, production disruptions, and safety hazards. Another security challenge in the OpenPLC is the presence of software vulnerabilities. In our former paper [ALS1], we showed that the OpenPLC still has some design flaws that allow adversaries to steal ready-to-execute programs from the *Webserver*, and conduct successful control logic injection attacks. Furthermore, the communication channels used by the OpenPLC systems are also prone to security risks. The absence of encryption exposes sensitive data e.g., user-credentials [ALS1], control commands [ALS1, ALV2, ALS3], configurations [ALS1, ALS3], etc. to attackers who are capable of intercepting, manipulating, and compromising the integrity of the data transmitted over the network. Therefore, we introduce in this paper a more secure OpenPLC software that is integrated with robust security solutions to enhance the overall security posture and trustworthiness of the OpenPLC project. First, we implemented an Advanced Encryption Standard (AES) algorithm to encrypt the user credentials i.e., the *username* and *password*. The outputs of the AES algorithm are then encoded from binary to ASCII, using the Base64[1] algorithm before they are finally stored in the *openplc.db* database. This managed successfully to close the vulnerability we found in our former paper [ALS1] and attackers are no more able to neither sniff the user credentials from the network nor to steal them from the *openplc.db*. Furthermore, the *Webserver* in the OpenPLC Aqua is accessible only by users with root permissions. This prevents external adversaries from accessing ST file copies of the programs that the user uploaded into the OpenPLC in the past [ALS1]. We also implemented a whitelisting approach that allows only pre-approved or trusted users to upload a new ST file into the OpenPLC. This approach aims at detecting any malicious attempt to manipulate the running program by an unauthorized user. Finally, the OpenPLC Aqua secures the data transmitted between the client (user) and server (OpenPLC) over the internet by using SSL/TLS (Secure Sockets Layer/Transport Layer Security) protocols. This ensures that the information remains confidential and cannot be intercepted or read by malicious attackers i.e., it helps to prevent eavesdropping and data tampering.

The rest of the paper is structured as follows: Section 2 provides related works, while Section 3 introduces the OpenPLC Aqua architecture. In Section 4 we show our experimental results and conclude this paper in Section 5. Please note that our experiments and the open-source code of OpenPLC Aqua are publicly available as Section 6 shows.

[1] https://docs.python.org/3/library/base64.htm

2 Related Work

The OpenPLC project was first released by Alves [ALV1] in 2014, and consists of Editor, *RunTime*, and Human Machine Interface (HMI) Builder see figure 1.

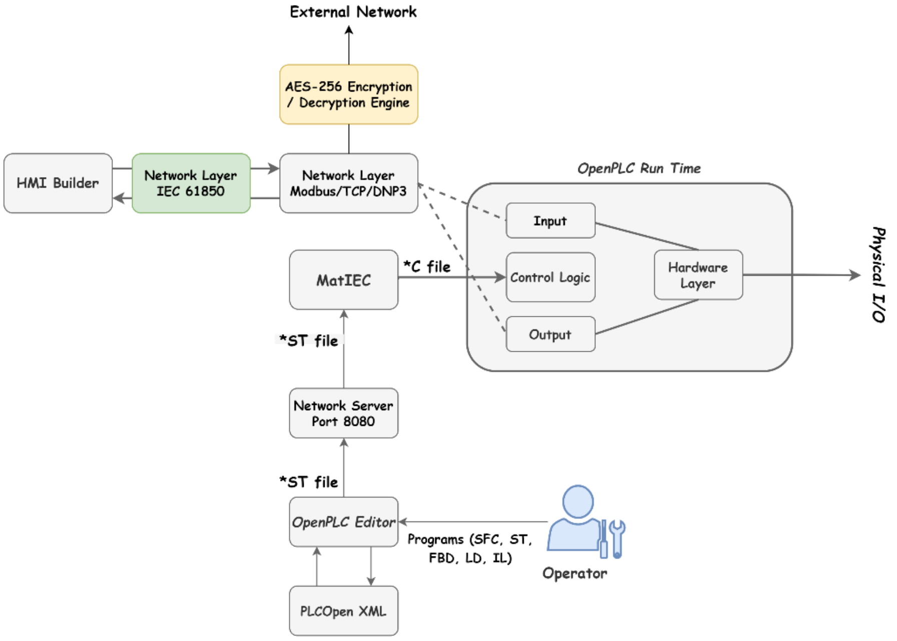


Fig. 1: OpenPLC Internal Architectures (boxes highlighted in gray color indicates the original OpenPLC; in yellow color the features added in OpenPLC NEO, in green color the features added in OpenPLC61850

The development environment that is used to create programs is called Editor. This Editor supports program-development using several programming languages identified by IEC 61131-3 e.g., Function Block Diagram (FBD), Ladder Diagram (LD), Structured Text (ST), Instruction List (IL), and Sequential Function Chart (SFC). Based on IEC 61131-3, the programs are saved as XML files. Once the program is created, the built-in module in Editor compiles all programs into an ST file. This ST file is then utilized in the OpenPLC *RunTime* for the execution of control logics. The communication protocols supported by OpenPLC are Modbus and Distributed Network Protocol 3 (DNP3) using the default ports 502 and 20000 respectively.

In 2017, Alves [ALV3] introduced an enhanced version of project, called OpenPLC Neo, by adding an AES Encryption Layer placed between the Network Layer and the external network (see figure 1, the yellow box). This AES Encryption Layer encrypts all the messages sent from the OpenPLC to an external client e.g., user, HMI, etc. by using a symmetric key and then forwards the appropriate ciphertext to the external network. In opposite, it decrypts the messages it receives by using a symmetric key provided by the user, and sends it then to the network layer for further processing. The OpenPLC Neo enables a secure end-to-end encrypted channel between the PLC and the user or HMI, without requiring any external hardware to encrypt the data. However, the project relies totally on the secrecy and integrity of the encryption keys. If the keys are not properly managed, such as being weak, compromised, or improperly stored, it can weaken the overall security of the AES encryption and expose OpenPLC Neo to cyber-attacks. Roomi [ROO1], in 2021, introduced the OpenPLC61850 that was also an enhancement of the OpenPLC project (see figure 1, the green box). The new version supports the International Electrotechnical Commission (IEC) 61850 protocols[2], including IEC 61131-3 standard [TIEG] for programming the PLC logics and uses MatIEC compiler [DESO] for the compilation of the programmed logics. In a follow up work Roomi [ROO2] evaluated the design of the OpenPLC61850 under network-based attacks e.g., false data and command injections. Their attack scenarios managed successfully to manipulate different Circuit Breaker (CB) that are communicated to the OpenPLC. However, Roomi did not introduce particular security solutions to protect the OpenPLC61850 against cyber-attacks and recommended only to include the security measures proposed in [HUSS] as a part of forthcoming research.

3 OpenPLC Aqua Security Features

Figure 2 depicts the internal architecture of our OpenPLC Aqua software. As can be seen from the figure, four additional security features are integrated in this version: 1) AES-128 Encryption Algorithm, 2) locked *Webserver*, 3) Whitelisting function and 4) SSL/TLS communication channels. In the following we elaborate our security features in more detail.

3.1 AES-128 Encryption Algorithm

The encryption process that we implemented in the OpenPLC Aqua aims at encrypting the user credentials i.e., username and password, by applying a customized AES-128 encryption algorithm as depicted in Figure 3. The plaintext messages, here the *username* and *password*, are divided in chunks of 128 bits each. If the original message size is not multiple of 128, the last block is padded with random bits to form a full 128-bit block. The Cipher Block Chaining (*CBC*) is used then to process the 128-bit blocks. At this step a random Initialization Vector (*IV*) is provided as the starting of pseudo-block.

[2] libIEC61850. https://webstore.iec.ch/searchform&q=61850.

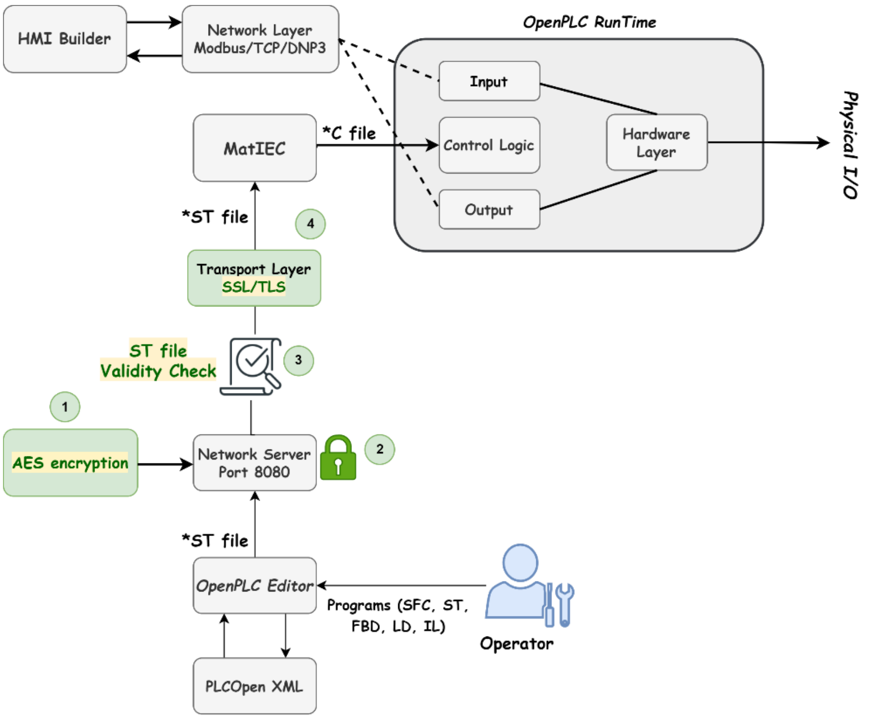


Fig. 2: OpenPLC Aqua Internal Architecture Diagram

In this work, we used a function called *os.urandom*[3] that is provided by python library to generate two random 16 bytes. One is used as a secret key (*K*) and the other as an *IV*. Both *K* and *IV* are stored in the *Webserver*. For the *password*, we used the *IV* to encrypt the initial block while the *K* is used to encrypt the entire blocks. Whereas in the *username*, we used the *K* and *IV* in a reverse way. The *K* here is utilized as an initialization vector to encrypt the initial block, while the *IV* is used as a key to encrypt the entire blocks. Once the ciphertext of both *username* and *password* are calculated, the ciphertexts are then encoded with the help of the Base64 encoding algorithm. The results of the encoding process are finally presented in ASCII format and stored in the *openplc.db* database see figure 4.

3.2 Securing the Webserver

Since the *Webserver*, which contains sensitive data and information about the entire project e.g., user credentials, user programs, etc., was the hugest security gap in the

[3] https://docs.python.org/3/library/random.html

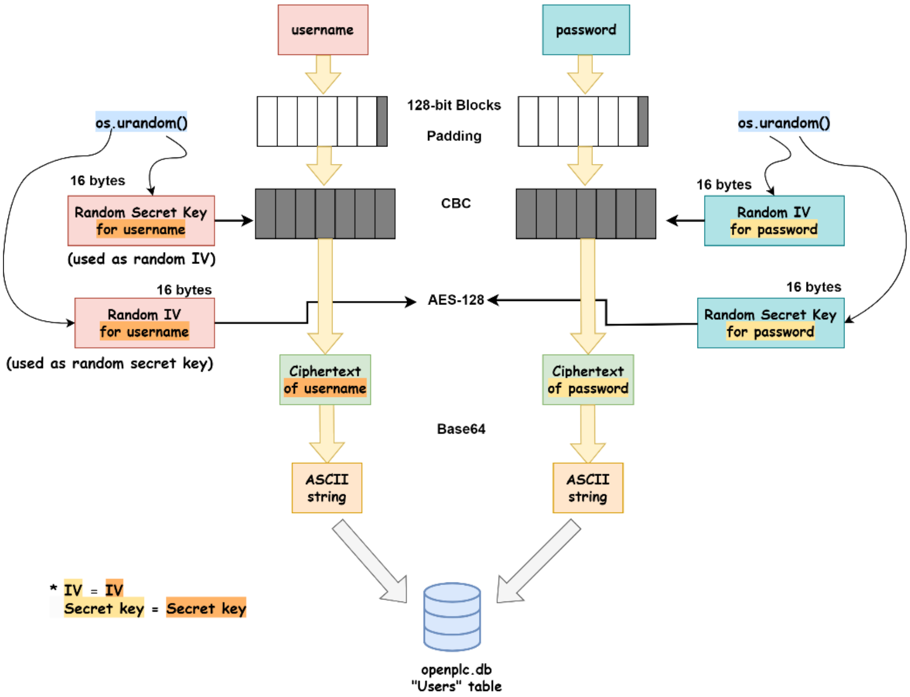
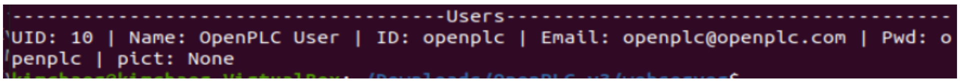


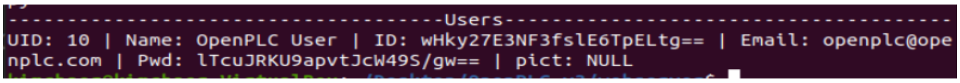
Fig. 3: The AES-128 Encryption Algorithm used in OpenPLC Aqua

OpenPLC as our investigations showed in [ALS1], the OpenPLC Aqua limits the access to the *Webserver* to only legitimate users with root permission. In addition, we also noticed that the OpenPLC keeps copies of all uploaded programs in the *Webserver*. These copies are ready-to-execute ST files and attackers who are still capable of accessing the *Webserver* can retrieve the programs and use them to inject the OpenPLC e.g., replace the currently running program with an old one. The problem in the OpenPLC software is that those copies are irremovable. Meaning that even if the user deletes the programs from the user-dashboard, their corresponding copies in the *Webserver* are not removed [ALS1]. To avoid abusing those copies, we improved the OpenPLC Agua security by adding a script that removes any copy from the *Webserver* once the user deletes its corresponding program from the user-dashboard.



```
-------------------------------------Users-------------------------------------
UID: 10 | Name: OpenPLC User | ID: openplc | Email: openplc@openplc.com | Pwd: o
penplc | pict: None
```

Fig. 4a: The *username* and *password* before the encryption as they are stored in the *openplc.db*



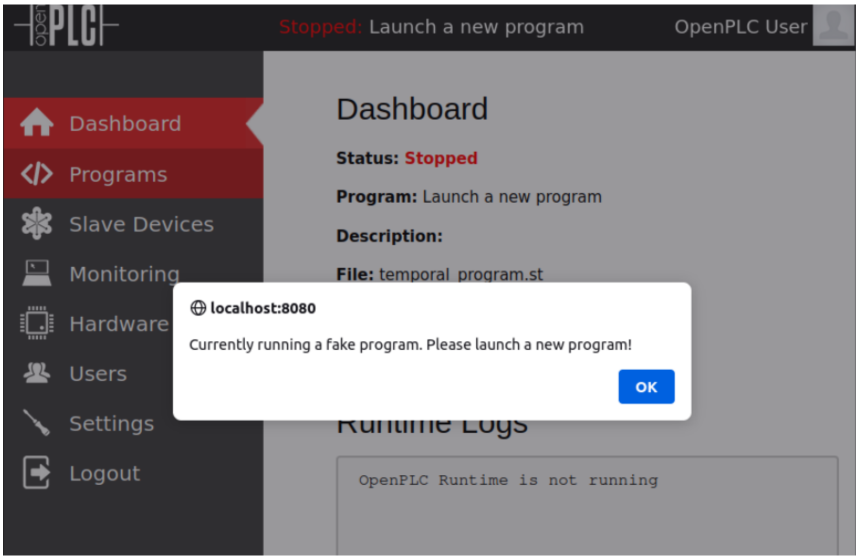
```
------------------------------------Users-------------------------------------
UID: 10 | Name: OpenPLC User | ID: wHky27E3NF3fslE6TpELtg== | Email: openplc@ope
nplc.com | Pwd: lTcuJRKU9apvtJcW49S/gw== | pict: NULL
```

Fig. 4b: The *username* and *password* after the encryption as they are stored in the *openplc.db*

Fig. 4: The representation of the *username* and *password* in both OpenPLC software versions. (a) in the OpenPLC, (b) in the OpenPLC Aqua

However, removing the copies from the *Webserver* has limitations. The OpenPLC starts launching any program (already uploaded), by first reading the name of the program from the *Active_Program* index, and recalling the corresponding ST copy file in the *Webserver* the *Active_Program* indicates. Since the OpenPLC Aqua deletes a copy together when the user removes its original program from the user-dashboard list, the name of the copy still remains in the '*Active_Program'* index. Meaning that, the *Active_Program* indicates to a missing copy (removed) in the *Webserver*. This causes an error in launching the software itself.

To overcome this challenge, we configure the OpenPLC Aqua with a blank program (fake program) that assists the software with launching itself. The objective of this program is not to run on hardware controllers, so the software keeps giving an alert to the user to launch a new proper program see figure 5. It is worth mentioning that we designed this blank program for a very specific use i.e., to launch the software itself. Therefore, if any user attempts to start the blank program in the OpenPLC, the session will be immediately closed. This adds more security to the software as it prevents any attempt to abuse the blank program.



Status: Stopped

Program: Launch a new program

Description:

File: temporal_program.st

localhost:8080

Currently running a fake program. Please launch a new program!

```
OpenPLC Runtime is not running
```

Fig. 5: OpenPLC Aqua alarms the user to launch a new program

3.3 Whitelisting Approach

In the OpenPLC Aqua, we added a function that verifies the ST file content, as well as the IP address each time a new ST file is uploaded into the OpenPLC. This bans any attempt to maliciously upload an ST file from suspicious users see figure 6.

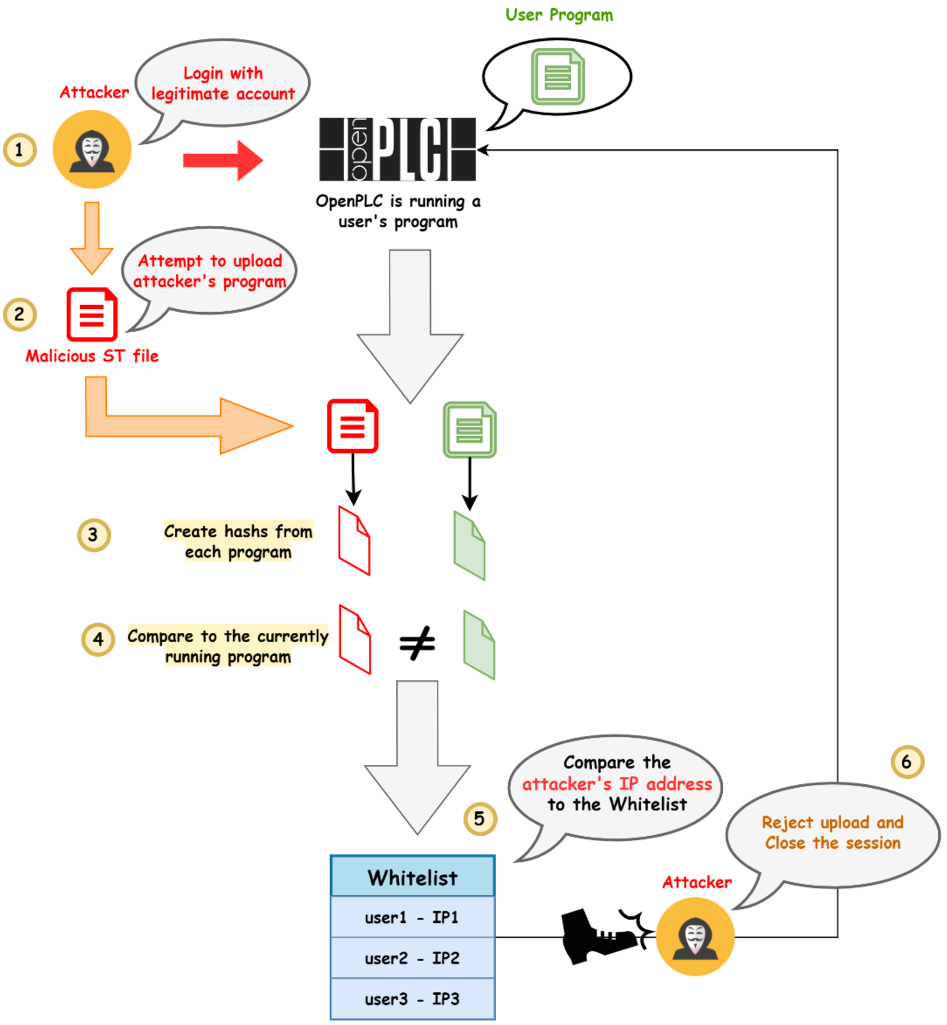

User Program
Attacker
Login with legitimate account
1
OpenPLC is running a user's program
Attempt to upload attacker's program
2
Malicious ST file
3
Create hashs from each program
4
Compare to the currently running program
≠
5
Compare the attacker's IP address to the Whitelist
6
Reject upload and Close the session
Attacker
Whitelist
user1 - IP1
user2 - IP2
user3 - IP3

Fig. 6: Whitelisting approach in OpenPLC Aqua

At the beginning, if a trusted user creates a new account using the user-dashboard, a pair of his IP address and username will be automatically added to the whitelist. By assuming that only the default-account is registered in the user-dashboard, and the user of the default-account attempts to upload a different program than the OpenPLC is currently running; the OpenPLC Aqua creates two hashes from both: the old program and the new one. After that, it compares the hashes of the programs and only if they are not matched, it checks the IP address of the user who attempts to upload. If his IP address is not listed in the whitelist, it then rejects the upload process, closes the session and the suspicious user is automatically logged out from the session. Otherwise, i.e., the IP address of the user is listed in the whitelist, the OpenPLC approves the upload process and the new program is then ready to start. This approach generates a solid security advantage to our project. For instance, even if the attacker could somehow access the OpenPLC with legitimate user-credentials and accounts, he would not be able to upload his malicious program.

3.4 SSL/TLS Communication Channel

Our investigations conducted on the last version of OpenPLC in [ALS1] proved that the software is prone to replay attacks. We showed that adversaries with appropriate attacking tools can reproduce certain traffic captures (HTTP packets) from previous communication sessions over the Internet to access and make malicious changes in target OpenPLC based systems. SSL/TLS (Secure Sockets Layer/Transport Layer Security) is a communication approach that provides an appropriate secure channels over the internet. It establishes an encrypted connection between a client (user) and server (OpenPLC). This encryption ensures that the data transmitted between them remains confidential and cannot be easily intercepted or revealed by unauthorized parties as the case in the existing OpenPLC software. Therefore, to introduce confidentiality, integrity, and end-point authentication in the OpenPLC Aqua software, we implemented the SSL/TLS handshake approach depicted in figure 7.

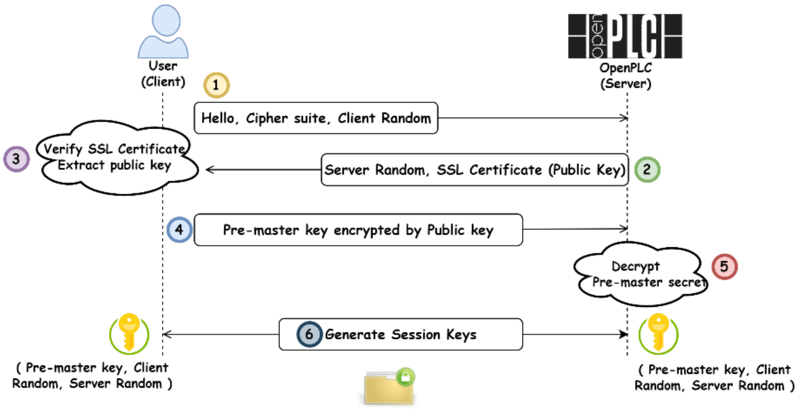

User
(Client)
OpenPLC
(Server)
1
Hello, Cipher suite, Client Random
3
Verify SSL Certificate
Extract public key
Server Random, SSL Certificate (Public Key)
2
4
Pre-master key encrypted by Public key
Decrypt
Pre-master secret
5
6
Generate Session Keys
(Pre-master key, Client Random, Server Random)
(Pre-master key, Client Random, Server Random)

Fig. 7: SSL/TLS communication process approach

First 1) the client (user) sends a *Hello* message informing the server (OpenPLC) that he wants to establish a secure channel between the both parties and provides the OpenPLC with his cipher suites and the compatible SSL/TLS version. Then 2) the OpenPLC responds back to the user by sending its random, the chosen cipher suite, and the SSL certificate. After that 3) the client verifies the SSL certificate with the authority and extracts a public key from it. The client then 4) generates an encrypted pre-master key by using the extracted server public key. Afterwards, the OpenPLC 5) decrypts the transmitted pre-master key to verify the extracted key. Once the verification is finished, both client and server 6) create the same session key from the random they got from each other and the pre-master key. From now on, all data transmitted between both client and server is encrypted and decrypted using the master key.

4 Experimental Results

To assess our developed OpenPLC software, we conducted five different attack scenarios on four OpenPLC versions: OpenPLCV3, OpenPLC Neo, OpenPLC61850 and OpenPLC Aqua. All Our experimental results are listed in table 1.

Attack Scenario	OpenPLC V3	OpenPLC Neo	OpenPLC61 850	OpenPLC Aqua
Authentication Attack	✓	✗	✓	✗
Man-in-the-Middle Attack	✓	✗	✓	✗
Control Logic Injection Attack	✓	✓	✓	✗
Replay Attack	✓	✗	✓	✗
Access Attack	✓	✓	✓	✗

Tab. 1: The success of various attacks against different OpenPLC software versions

As we noticed from the table, our OpenPLC Aqua is secure against all the attacks we conducted and show more resistance compared to the previous OpenPLC software.

5 Conclusion

This work defined an alternative design to secure OpenPLC based systems and SCADA networks by introducing and integrating four security features in the new OpenPLC Aqua software. The new software protects the OpenPLC and its systems from different attacks that were feasible in the last software versions. For instance, encrypting and encoding the user credentials prevent adversaries from performing authentication attacks. Furthermore, the OpenPLC Aqua locks the *Webserver* allowing only legitimate users with root permissions to access. In addition, the ST file copy is removed from the *Webserver* once the user deletes its corresponding program from the user-dashboard. We also secure all the communication session between the user and OpenPLC by implementing SSL/TLS protocols. Finally, to ensure that only authorized and trusted users can upload/change the program in the OpenPLC, we introduced a whitelisting approach which prevents malicious infections. The experimental results proved that OpenPLC Aqua was able to significantly enhance the security of OpenPLC project by preventing different attack scenarios.

6 Demo and Open-Source Code

Please refer to [ALS4] for the demo of our experiments, and the GitHub repository [ALS5] for detailed information of OpenPLC Aqua.

Bibliography

[ALS1] Alsabbagh, W.; Kim, Ch.; Langendörfer, P. Good Night and Good Luck: A Control Logic Attack on OpenPLC. IECON 2023- 49th Annual Conference of the IEEE Industrial Electronics Society, Singapore, Singapore, 2023, pp. 1-8, doi: 10.1109/IECON51785.2023.10312570.

[ALS2] Alsabbagh, W.; Langendörfer, P. A Flashback on Control Logic Injection Attacks against Programmable Logic Controllers. Automation 2022, 3, 596–621. [Online]. Available: https://doi.org/10.3390/automation3040030.

[ALS3] Alsabbagh, W.; Amogbonjaye, S.; Urrego, D.; Langendörfer, P. A Stealthy False Command Injection Attack on Modbus based SCADA Systems. 2023 IEEE 20th Consumer Communications & Networking Conference (CCNC), Las Vegas, NV, USA, 2023, pp. 1-9, doi: 10.1109/CCNC51644.2023.10059804.

[ALV1] Alves, Th.; Buratto, M.; De Souza, F. M.; Rodrigues, T. V. OpenPLC: An open source alternative to automation. IEEE Global Humanitarian Technology Conference (GHTC 2014), San Jose, CA, USA, 2014, pp. 585-589, doi: 10.1109/GHTC.2014.6970342.

[ALV2] Alves, Th.; Morris, Th. OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research, Computers & Security, Volume 78, 2018, Pages 364-379, ISSN 0167-4048, https://doi.org/10.1016/j.cose.2018.07.007.

[ALV3] Alves, Th.; Morris, Th.; Yoo, S. Securing SCADA Applications Using OpenPLC With End-To- End Encryption. (2017) In Proceedings of the 3rd Annual Industrial Control System Security Workshop (ICSS 2017). Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/3174776.3174777.

[ROO1] Roomi, M. M.; Ong, W. S.; Mashima, D.; Hussain S. M. S. OpenPLC61850: An IEC 61850 compatible OpenPLC for Smart Grid Research, SoftwareX, vol. 17, Jan. 2022, doi: 10.1016/j.softx.2021.100917.

[TIEG] Tiegelkamp, M.; John, K. IEC 61131-3: Programming Industrial Automation Systems; Springer: Berlin/Heidelberg, Germany, 2001; Volume VI, p. 376.

[DESO] De Sousa, M.; Carvalho, A. An IEC 61131-3 compiler for the MatPLC. EFTA 2003. 2003 IEEE Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.03TH8696), Lisbon, Portugal, 2003, pp. 485-490 vol.1, doi: 10.1109/ETFA.2003.1247746.

[HUSS] Hussain, S. M. S.; Ustun, T. S.; Kalam, A. A Review of IEC 62351 Security Mechanisms for IEC 61850 Message Exchanges. In IEEE Transactions on Industrial Informatics, vol. 16, no. 9, pp. 5643-5654, Sept. 2020, doi: 10.1109/TII.2019.2956734.

[ROO2] Roomi, M. M.; Ong, W. S.; Hussain, S. M. S.; Mashima, D. IEC 61850 Compatible OpenPLC for Cyber Attack Case Studies on Smart Substation Systems. in IEEE Access, vol. 10, pp. 9164-9173, 2022, doi: 10.1109/ACCESS.2022.3144027.

[ALS4] https://www.youtube.com/watch?v=knVTQfUdNfU.

[ALS5] https://github.com/rnrn0909/OpenPLC-Aqua.git.