Empirical Study of PLC Authentication Protocols in Industrial Control Systems

Adeen Ayub
Department of Computer Science
Virginia Commonwealth University
Richmond, United States of America
ayuba2@vcu.edu

Hyunguk Yoo
Department of Computer Science
The University of New Orleans
New Orleans, United States of America
hyoo1@uno.edu

Irfan Ahmed
Department of Computer Science
Virginia Commonwealth University
Richmond, United States of America
iahmed3@vcu.edu

***Abstract*—Programmable logic controllers (PLCs) run a 'control logic' program that defines how to control a physical process such as a nuclear plant, power grid stations, and gas pipelines. Attackers target the control logic of a PLC to sabotage a physical process. Most PLCs employ password-based authentication mechanisms to prevent unauthorized remote access to control logic. This paper presents an empirical study on proprietary authentication mechanisms in five industry-scale PLCs to understand the security-design practices of four popular ICS vendors, i.e., Allen-Bradley, Schneider Electric, AutomationDirect, and Siemens. The empirical study determines whether the mechanisms are vulnerable by design and can be exploited. It reveals serious design issues and vulnerabilities in authentication mechanisms, including lack of nonce, small-sized encryption key, weak encryption scheme, and client-side authentication. The study further confirms the findings empirically by creating and testing their proof-of-concept exploits derived from MITRE ATT&CK knowledge base of adversary tactics and techniques. Unlike existing work, our study relies solely on network traffic examination and does not employ typical reverse-engineering of binary files (e.g., PLC firmware) to reveal the seriousness of design problems. Moreover, the study covers PLCs from different vendors to highlight an industry-wide issue of secure PLC authentication that needs to be addressed.**

***Index Terms*—PLC, industrial control systems, SCADA, password authentication, cyber-physical systems, critical infrastructure protection, CPS, stuxnet, TRISIS**

I. Introduction

In industrial control systems (ICS), programmable logic controllers (PLC) directly control and monitor physical processes such as the power grid, gas pipelines, and water treatment [1]. They run a control-logic program that defines how they should control a physical process. Attackers target control-logic in a PLC to sabotage a physical process [2]–[8]. For instance, Stuxnet [2] has infected the control logic of Siemens S7-300 PLCs to manipulate centrifuges' motor speed periodically from 1,410 Hz to 2 Hz to 1,064 Hz.

Control engineers utilize vendor-supplied engineering (programming) software to configure a PLC remotely. They can write a control logic in the software and then transfer (or download) it to a PLC. Similarly, they can acquire (or upload) the control logic from the PLC. Most PLCs employ password-based user authentication to protect control-logic from unauthorized access. When a control engineer attempts to access the control-logic in a PLC using an engineering software, the PLC requires password authentication and initiates an authentication protocol (typically proprietary).

This paper presents an empirical study on the security-design practices of authentication mechanisms by four popular ICS vendors i.e., Schneider Electric, Allen-Bradley, AutomationDirect, and Siemens. The study analyzes the authentication protocols of five industry-scale PLCs: 1) Schneider Electric Modicon M221, 2) Allen-Bradley MicroLogix 1100, 3) Allen-Bradley MicroLogix 1400, 4) AutomationDirect CLICK, and 5) Siemens S7-300. Similar to real-world ICS attacks such as TRITON [9] and the Ukraine power grid attack [10], our attack model assumes that an adversary has access to *Level-3* network of Purdue Model (i.e., control center network) [11], and can employ one or more of the following three capabilities to subvert a PLC authentication protocol and gain unauthorized access to a PLC: 1) *Eavesdropping* - read any messages between two communication parties, 2) *Fabrication* - initiate conversation with any other party and compose/send a new message, 3) *Interception* - intercept messages to block or modify them.

The empirical study is based solely on network traffic examination and reveals serious design issues and vulnerabilities in authentication protocols that are exploitable including lack of nonces, small-sized encryption key, weak encryption algorithm, and client-side local authentication The study confirms these findings empirically by creating and testing their proof-of-concept exploits derived from MITRE ATT&CK knowledge base of adversary tactics and techniques [12], including unauthorized password reset attack, authentication bypass, password cracking, and password sniffing attack. The study discusses the fundamental design-issues in the authentication mechanisms and further recommends countermeasures, including PLC memory protection, secure design of PLC authentication protocol, and control logic detection.

The rest of the paper is organized as follows. Section II provides the background. Sections III presents an overview of the study methodology, followed by a series of sections on PLC case studies on authentication protocols. Section IX gives recommendations to ICS vendors and operators to improve the security of the authentication protocols and ICS environments that run vulnerable PLCs. Section X presents the related work, followed by Section XI that concludes the study.

II. Background

Programmable logic controllers. PLCs are embedded devices programmed to automate and control physical processes such as assembly and gas pipelines [13]. They are equipped with control logic programs that define how an industrial process is controlled and monitored. The control logic programs are programmed and compiled by engineering software specific to the vendor. IEC61131-3 standard defines five languages to write a control logic: ladder logic, instruction list, functional block diagram, structured text, and sequential flow chart.

When a control logic program is created and compiled, it is written to the PLC's memory. This process is called *downloading*. On the other hand, reading a program from the PLC's memory is referred to as *uploading*. A PLC is usually equipped with communication interfaces such as RS-232 serial ports, Ethernet, and USB to communicate with the engineering software so that the control logic can be downloaded to or uploaded from a PLC. When the communication takes place over the network, it is in the form of request/response messages with the engineering software and PLC acting as a client and server, respectively. However, the exact communication protocol over the transport layer is proprietary and vendor-specific.

PLC authentication protocol. Most PLCs use password-based authentication for protection against unauthorized access. To authenticate with a legitimate user using an engineering software, they follow a protocol specific to the vendor. The sequence of steps and the messages exchanged for the authentication process determine the PLC authentication protocol.

III. Study Methodology

The purpose of this empirical study is to understand security-design practices, issues, and vulnerabilities in proprietary authentication protocols in industry-scale PLCs. These protocols are supposed to prevent unauthorized access to PLCs. However, this study shows that the PLC authentication mechanisms are inherently insecure and can be subverted by an adversary.

A. Adversary Model

Assumptions. Our adversary model assumes that an attacker has access to the Level-3 network of Purdue Model (i.e., control center network) [11]. This assumption is based on real-world ICS attacks (e.g., TRITON [9] and Ukraine Power grid attack [10]) that gain access to the control center via a typical IT attack vector (such as infected USB stick and social engineering attack). We assume that the attacker has access to PLCs and their respective engineering software along with a packet-sniffing tool such as Wireshark [14] to find vulnerabilities and prepare exploits before getting access to a target network.

After the Level-3 network access, the attacker can make use of libraries such as libpcap [15] to sniff the legitimate traffic and then communicate with a target PLC over the network to subvert the PLC authentication mechanism. We also assume that the attacker does not have to have skills to perform time-consuming binary reverse engineering of PLC firmware and control center applications to figure out pre-installed secret keys or cryptographic algorithms. The study shows that this information can be deduced from network traffic, making the attack vector more realistic to real world ICS environments.

Attacker's goal. An attacker's goal is to bypass the authentication mechanism of a password-protected PLC over the network to gain access. In this study, we assume that an attacker achieves this goal if any of the following tasks are accomplished without previously knowing the correct password: 1) gain plaintext password, 2) read control logic of a PLC, 3) modify control logic of a PLC, and 4) change the password.

Attacker capabilities. Our adversary model defines attacker's capabilities using the classic Dolev-Yao model [16] i.e., eavesdropping, fabrication, and interception of network messages. The attacker can employ one or more of these capabilities to achieve the goals mentioned earlier:

Eavesdropping - Read any messages between two communication parties.

Fabrication - Initiate conversation with any other party and compose/send a new message.

Interception - Intercept messages and block or modify/resend them. We utilize ARP poisoning for this.

B. Study Method

Our study method consists of a series of three steps to assess weaknesses in a PLC authentication protocol:

1) Understanding authentication protocol internals. The protocols are proprietary and their information is not publicly available. We explore the protocol internals by utilizing engineering programming software to generate traffic patterns during the authentication process, and performing differential analysis of network traffic to understand the message types and formats.

2) Identifying potential vulnerabilities. After understanding the protocol internals, we further explore the protocol internals with the goal of identifying potential weaknesses that make the protocols susceptible to subversion. After much deliberation, we identify eight vulnerabilities discussed in Section III-D.

3) Mapping an identified vulnerability to the MITRE ATT&CK framework. We create and test the proof-of-concept exploits derived from the MITRE ATT&CK knowledge base of adversary tactics and techniques to verify if the identified vulnerabilities are exploitable [12].

C. PLCs for the Study

Table I presents the PLCs and their engineering software used for the study. The PLCs are from four popular ICS vendors i.e., Schneider Electric, Allen-Bradley, AutomationDirect, and Siemens. Our choice of a PLC for the study is based on the following factors. It should 1) be a product of a popular ICS vendor, 2) not be obsolete; we acquired the PLCs in the year 2019, 3) have a similar cost; our PLCs cost

TABLE I
ICS VENDORS, PLCS, AND VENDOR-SUPPLIED ENGINEERING (PROGRAMMING) SOFTWARE USED IN THE STUDY

Vendors	PLCs	Engr. Software
Schneider Electric	Modicon M221	SoMachine Basic
Allen-Bradley	MicroLogix 1100 & 1400	RSLogix 500
AutomationDirect	CLICK	CLICK Software
Siemens	S7-300	SIMATIC STEP 7

between $1000 and 1500$, 4) have an Ethernet physical port, and 5) support IEC-61131-3 standard languages such as ladder-logic and instruction-list.

D. Study Findings

We present five case studies covering five PLCs in Sections IV, V, VI, VII, and VIII. The case studies reveal and discuss eight exploitable vulnerabilities (V) to understand the bad security-design practices of four popular ICS vendors for the proprietary authentication protocols, e.g., small key space, plaintext password, and lack of nonce in authentication messages. Table II shows the mapping between the vulnerabilities and MITRE ATT&CK (the general descriptions of the attacks are in Appendix). The ATT&CKs are also discussed in the case studies in the specific context of the authentication protocols. The vulnerabilities are described as follows:

V1 - Information disclosure. Clear text storage and transmission of sensitive information (e.g., password). It is found in CLICK, and both Micrologix 1100 and 1400 PLCs.

V2 - Client-side authentication. There may be a case when the engineering software takes a password as user input and performs user authentication locally. Since the PLC does not perform authentication, it has to trust the remote communication party, which poses an exploitable vulnerability; the attacker can act like a trusted party to initiate the communication and read/write to the PLC memory. Both Micrologix 1100 and 1400 have this vulnerability.

V3 - Weak encryption scheme. An encryption scheme is weak if the key space is small, does not use nonces, or either individually encrypts each character or each portion of the message. Not having a strong encryption scheme allows an attacker to either decrypt the password in transit using the attacker's plaintext/ciphertext pairs or intercept an authorised password set/reset request. Siemens S7-300 and MicroLogix 1400 are vulnerable.

V4 - Small key space. The key space is too small (256 characters). Siemens S7-300 PLC is vulnerable.

V5 - Lack of nonces. The encryption scheme does not use nonces for encrypting the password in transit. Siemens S7-300 has this vulnerability.

V6 - Use of same keys in multiple sessions. Using the same secret keys for multiple sessions makes the analysis easier since the attacker can now see noticeable and repeated patterns in the encrypted messages along with having the ability to decrypt all messages once she figures out the key. Siemens S7-300 is vulnerable.

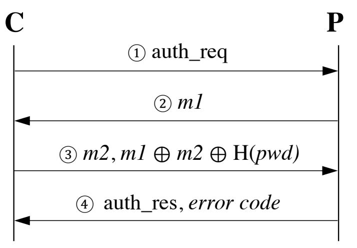


Fig. 1. Modicon M221 authentication protocol

V7 - Improper session management. Missing session management protocol allows an unauthorised attacker to perform privileged operations. Only CLICK PLC is vulnerable.

V8 - No write-protection. The PLC does not have the feature of password protection for write access. Only M221 is vulnerable.

IV. CASE STUDY I: MODICON M221

Schneider Electric Modicon M221 is a nano PLC made to automate machine processes. The engineering software used to program the controller is *SoMachine-Basic*. It supports ladder logic and instruction list programming languages to design control logic programs. SoMachine-Basic allows writing password-protected programs to the PLC, which can protect the controller from unauthorized access. Modicon M221 uses a proprietary protocol layer embedded in the Modbus protocol.

A. Authentication Protocol

Figure 1 describes the authentication process between Modicon M221 PLC and SoMachine-Basic engineering software. Note for all of the figures, we use `P` for the PLC, `C` for the legitimate engineering software, `C'` for the attacker's engineering software, and `M` for Man in the Middle. When a user tries to login, SoMachine-Basic sends a request message for a random one-byte mask (*m1*). In response, the PLC sends a value for *m1*. SoMachine-Basic then generates an arbitrary 1-byte value for another mask (*m2*). It then computes an XOR operation between *m1*, *m2*, and each byte of the SHA-256 hash of the password to mask the password hash. Next, SoMachine-Basic sends *m2* and the masked-hash to the PLC. On getting these values, the PLC unmasks the password hash by XORing it with *m1* and *m2* and then compares the unmasked-hash with the original password hash. The PLC responds with an error-code indicating either successful or failed authentication.

We determine the proprietary network protocol-layer of Modicon M221, encapsulated in Modbus protocol. Figure 15 (in appendix) shows the authentication messages and identifies the interesting fields in the proprietary layer. Figure 15(a) shows the byte pattern for *m1* request message. Figure 15(b) shows the packet containing the random one-byte value of *m1* always contains `0xfe` and `0x13` in its 9th and 10th index of the TCP payload (provided indexing starts at 0). Furthermore as seen in Figure 15(b), the value of *m2* precedes the masked password hash in the authentication request message that the engineering software sends to the PLC. Figure 15(c) and 15(d) show the message containing *m2* followed by the

TABLE II
ATTACK AND VULNERABILITY MAPPING FOR THE PLCS OF DIFFERENT VENDORS

MITRE Attack ID	Attack Name	Target PLCs				
		Modicon M221	S7-300/400	MicroLogix 1100	MicroLogix 1400	CLICK
T1555	Credentials from Password Stores	n/a	n/a	V1, V2	V1, V2	V1
T1040	Network sniffing	n/a	n/a	V1	V1	V1
T1098	Unauthorized Password Reset	V3, V4, V5, V8	n/a	V2, V5	V2, V5	n/a
T1562	Impair Defenses	n/a	n/a	V2	V2	V7
T1110.002	Password Cracking	n/a	V3, V4, V5, V6	n/a	n/a	n/a
T0830	Man in the middle	n/a	n/a	n/a	V3	n/a
T1562.002	Transmitted Data Manipulation	n/a	n/a	n/a	V3	n/a
T1499	Endpoint Denial of Service	n/a	n/a	n/a	V3	n/a

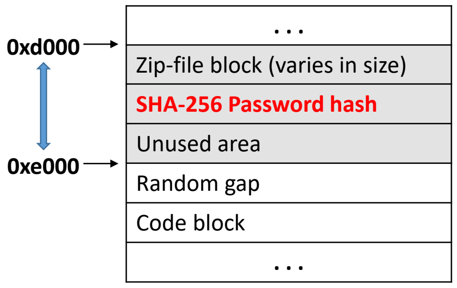


Fig. 2. Modicon M221 memory layout

masked password hash before and after getting manipulated respectively.

Protocol Vulnerabilities. M221 PLC utilizes XOR and two keys exchanged between the PLC and SoMachine-Basic to encrypt the password hash, which is a weak encryption scheme (V3). The key size used in the encryption scheme is small (V4) and the PLC does not have write protection (V8) because it allows an authorised remote user to write anywhere in its memory.

B. MITRE ATT&CK on Modicon M221 PLC

Unauthorised password reset (T1098). Because of the vulnerabilities mentioned above, the attacker can start a session with the PLC and send a write request to overwrite the password with her own.

Kalle et al.'s password reset attack [5] Initially, we study Kalle *et al.* [5]'s password attack on Modicon M221. The attack is challenging because the location of a 32-byte password hash is not fixed in the PLC memory. However, as illustrated in Figure 2, the password-hash always resides within the physical memory address-range of `0xd000` and `0xe000`, where `0xe000` is often an unused space and a zip-file of varying size starts from `0xd000`. The attack executes from an attacker's machine over the network in a series of following steps: 1) the attacker starts the authentication process and receives a one-byte mask (*m1*); 2) sends a write-message to the PLC to write the attacker's 32-byte password hash on the memory location `0xdfe0`(since `0xe000` - `0x20` = `0xdfe0`); 3) exchange the remaining authentication messages to send mask *m2* and masked-hash of the password; 4) If the authentication is not successful, the attacker performs the next iteration n wherein step 2, it moves the base-address of password hash by one-byte, i.e., `0xe000` + $n-1$.

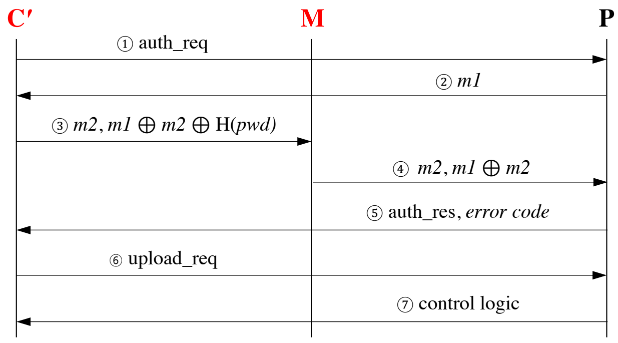


Fig. 3. Upload a control-logic into attacker's engineering software using zero-byte hash

Although this attack works successfully, it leaves a large footprint of failed authentication attempts in the network traffic. We further attempt to reduce this footprint by creating and evaluating an efficient password attack.

0x00ed (efficient) password reset attack Algorithm 1 (in Appendix) describes our efficient attack method to reset the Modicon M221 password. It utilizes the observation that the password hash is always located anywhere between the PLC memory address-range of `0xd000` and `0xe000` [5], shown in Figure 2. The attack overwrites this memory region (i.e., from `0xd000` to `0xe000`) with `0x00`, which replaces the password-hash with `0x00`. Note that this attack can use any homogeneous byte stream (i.e., all bytes have the same value); it does not have to be `0x00`. However, `0x00` can blend with benign traffic since the download/upload messages of Modicon M221 normally contain significant chunks of zero bytes.

The attacker initiates the authentication process with the PLC and sends zeros as a password hash to succeed. Specifically, the attack requests *m1* from the PLC and creates an arbitrarily chosen value for *m2*. It then performs a XOR operation using *m1*, *m2* and a 32-byte array of `0x00` to produce a 32-byte masked value representing a password hash. It sends the masked password hash to the PLC. In response, the PLC acknowledges the authentication attempt as successful.

Authentication via SoMachine-Basic. The attacker can utilize SoMachine-Basic to retrieve a control-logic from a target M221 PLC and then decompile it to a source code in ladder logic or instruction list. However, since the attacker overwrites the password-hash with zeros in the PLC, SoMachine-Basic cannot authenticate to the PLC. Our solution is to intercept the message containing an arbitrary masked hash-code and modify it with masked zeros. We utilize ARP-poisoning to test this approach.

Figure 3 illustrates a series of steps to execute this attack. The attacker starts the authentication process using SoMachine-Basic. When SoMachine-Basic asks for a password, the attacker provides a random sequence of ASCII characters. After SoMachine-Basic receives *m1* from M221 PLC, it sends a masked hash-code of the password to the PLC, which the attacker intercepts and modifies to a masked hash-code of zeros using *m1* (received earlier) and *m2* (acquired with the masked hash-code). The attacker then forwards the modified message to the PLC, resulting in successful authentication by SoMachine-Basic.

C. Attack Evaluation

Experimental settings. We utilize Schneider Electric's Modicon M221 (firmware v1.5.1.0 and v1.6.0.1) and SoMachine-Basic (version 1.5 and version 1.6) to evaluate the attacks. SoMachine-Basic runs on Windows 7 while the attackers machine is a Ubuntu 16.04 VM. We use the Scapy packet manipulation tool to implement our attack scripts in Python.

Dataset. We utilize 52 different password-protected control-logic programs (including traffic light, Hot Water tank, and Temperature Control) for the evaluation.

Evaluation results. While executing the attacks, we overwrite the memory region from `0xd000`(where the `zip-file` block is mapped) to `0xe000`(the address after which the `code` block is mapped). We observe that overwriting this region (including `zip-file` block) does not crash the PLC or make it malfunction.

We evaluate and compare both Kalle *et al.* [5]'s attack and our 0x00ed (efficient) attack on 52 different control logic programs. Table IV (in appendix) shows their average values of run time, write requests, failed authentication attempts, payload size and attack success rate. It shows that 0x00ed attack is 150 times faster since it requires to perform the authentication process only once and has zero failed-authentication attempts. On the other hand, Kalle *et al.*'s attack has 2457 failed attempts and utilizes 2458 write requests.

V. Case Study II: Siemens S7-300

Siemens S7-300 is programmed with the help of SIMATIC STEP 7(TIA Portal). It supports different programming languages including Ladder Logic, Function Block Diagram, Statement List, standard language, sequential control, or status graph. Like most of other PLCS, S7-300 also comes with an option to set a password to disallow unauthorised access. However, it allows the users to opt for either only write protection or read/write protection. With write protection, the users can upload projects without being authenticated. But with the more strict read/write protection, the users must be authenticated in order to upload a project from or download it to the PLC.

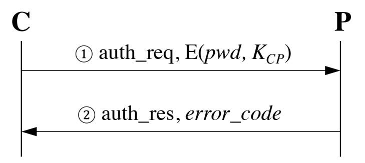


Fig. 4. Siemens S7-300 authentication protocol

A. Authentication Protocol

The authentication protocol of Siemens S7-300 PLC is described in Figure 4. The client sends an authentication request message with an encrypted password. User password is encrypted using a pre-shared key, K_{CP}. Then, the PLC responds with an error code; the value of which is zero if successful and other values for failed authentication.

Since the authentication protocol does not utilize nonce from both sides, it is vulnerable to replay attacks. Moreover, the encryption algorithm used for protecting user password is weak as described in Algorithm 2 (in appendix).

Encryption Algorithm. Algorithm 2 shows the weak encryption algorithm used in the authentication protocol. It takes as input an 8-byte password (ASCII) and one-byte key. Each character is substituted according to a substitution table, and then XORed with the key K for the first two characters. The remaining characters are XORed with K and E_{i-2}.

Protocol Vulnerabilities. Besides the lack of nonce (V5) which makes the authentication protocol vulnerable to replay attacks, it uses a very weak encryption algorithm (V3) which does not have sufficient confusion and diffusion layers. Thus, it is vulnerable to elementary known plaintext attack. If an attacker acquires just one plaintext/ciphertext pair, she can recover the secret key since $K = E_i \oplus N_i$ (for i ¡ 2). Since the substitution is conducted byte-by-byte (more like one-to-one substitution from an ASCII code to another code), it is trivial to reverse engineer the substitution table.

The PLC uses small key space (V4), i.e., just 8 bits which makes it susceptible to an exhaustive key search attack. Also, it uses the same key (V6) for the communication.

B. MITRE ATT&CK on S7-300 PLC

Password cracking (T1110.002). We present two attack scenarios that exploit primarily the vulnerabilities residing in the authentication protocol's weak encryption algorithm.

Attack scenario I: subverting write-protection This attack is applied to a PLC with the write-protection level. With write-protection, the PLC requires authentication only for write request messages allowing read request messages without authentication. Therefore, an attacker can read the PLC memory over the network that stores an encrypted password.

Since the attacker needs to compose and send a new read request message to a target PLC, the attack requires the capability of fabrication. Eavesdropping may be required too if the attacker has to figure out the IP address of targets on the fly.

After reading the SDB0 block from a target PLC, the attacker can extract the encrypted password and perform an exhaustive key search to find the encryption key, thereby acquiring the plaintext password.

Attack scenario II: subverting read/write-protection With read and write protection, the PLC requires authentication for both write and read requests. In this attack, the attacker's goal is to get the plaintext password. The attacker waits and eavesdrops the network until identifying an authentication request message transferred to a PLC. Once the attacker gets an authentication message, she can extract the encrypted password from the authentication message and decrypt it by an exhaustive key search.

C. Attack Evaluation

We performed our experiments on Siemens S7-300 (6ES7 315-2EH14-0AB0) on the latest firmware v3.2.8 and v3.2.17 and TIA Portal version v13, v15, and v16. We reverse engineered the substitution table (refer to Table V in appendix) used in the weak encryption algorithm and implemented our attacks in Python using the Snap7 library. The presented vulnerabilities and attacks were all confirmed successful and we responsibly reported the issues to the vendor. The vendor issued a security advisory including mitigation methods.

VI. Case Study III: MicroLogix 1100 and 1400 (Default)

We include two Allen-Bradley PLCs in our study from the MicroLogix series, i.e., 1100 and 1400. Allen-Bradley provides RSLogix 500 engineering software to configure and write a control-logic program for these PLCs. RSLogix 500 sets a password on a control-logic program before downloading it to a PLC, which is then used for user authentication. RSLogix 500 provides two password authentication options for MicroLogix 1400, i.e., 1) Default, and 2) Enhanced Password Security. The *Default* option has the same authentication protocol as the older firmware versions, whereas the *Enhanced Password Security* option has a different protocol. This section focuses on the *Default* option applicable to both MicroLogix 1100 and 1400 PLCs to cover current and past firmware versions while Section VII covers the *Enhanced Password Security* option.

RSLogix 500 allows a user to set 1) *Master Password*, 2) *Password*, and 3) *Subroutine Password*. Both *Master Password* and *Password* allow access to the PLC while *Subroutine Password* is to protect the unauthorized viewing of selected ladder logic files. When a password is set, all subsequent communications such as uploading or modifying the existing program on the PLC are allowed after successful login.

A. Authentication Protocol

Both MicroLogix 1100 and 1400 PLCs support the PCCC (Programmable Controller Communication Commands) network protocol. PCCC is a command/reply protocol and is transported over EtherNet/IP (ENIP), which is an adaption of Common Industrial Protocol (CIP). PCCC consists of Function Code (FNC) and PCCC Data. FNC for protected read and write is `0xa2` and `0xaa` respectively. The sub-fields of *PCCC Data* field for protected read and write function codes include *Byte Size* (the number of bytes to read/write), *File Number* (unique number identifying a file of control logic), *File Type* (which represents the file content), *Element Number* (the slot number of the PLC) and *Sub-element Number* (word offset in a file) [17].

In MicroLogix 1100 and 1400, the authentication occurs at the client-side (i.e., RSLogix 500). RSLogix 500 connects to the PLC and sends a read-request message for the password at the following fixed password location in the PLC memory: File number=`0x00`, File type=`0x00`, Element number=`0x0b`, and Subelement number=`0x00`). In response, the PLC sends a 10-byte original password-hash (if "encrypt password" option is enabled) or plaintext password. RSLogix 500 takes a password from a user-input and compares it with the original password to complete the authentication process. Note that the PLCs do not authenticate users (i.e., server-side authentication), allowing the attacker to read/write the PLC memory without authentication.

Protocol Vulnerabilities. We exploit two main vulnerabilities in this class of PLCs. First, the authentication gets done at the client side (V2). Secondly, the password gets transmitted in clear text if the user does not select "encrypt password" at the time of setting it (V1).

B. MITRE ATT&CK on MicroLogix 1100/1400 (Default)

We discuss three attack scenarios to show the subversion of client-side authentication, unauthorized resetting of PLC password, and sniffing password in transit.

Impair defenses (T1562). RSLogix enforces user-authentication locally at its end. Both MicroLogix 1100 and 1400 allow reading/writing to their memory without authentication. Since the PLCs do not perform the authentication, they have to trust the remote communication party such as RSLogix, which poses an exploitable vulnerability; the attacker can act like a trusted party to initiate the communication and read/write to the PLC memory.

To exploit this vulnerability, the attacker utilizes the PCCC protocol of MicroLogix family to read from or write to the PLC using the PCCC addressing, i.e., File number, File type, Element number, Subelement number, along with the read and write FNC codes.

Unauthorized password reset (T1098). The authentication protocol requires the password location to be fixed and hard-coded so that RSLogix can read the current password remotely for the authentication. However, the attacker can use this information to reset the password since the PLCs do not authenticate remote-clients.

We analyze the password location in both PLCs' memory and determine that the current password is present in three memory locations of Micrologix 1400, defined in PCCC addressing (File number, File type, Element number, Subelement number). The PCCC addresses are: 1) `00 00 00 00`, 2) `00 00 0b 00`, 3) `00 03 0b 00`. When a user sets a password, it is first written at location 3. For Micrologix 1100, the initial location is `00 02 0b 00`.

Determining the password location. We write a program, disable the "encrypt password" option to transfer the password in plaintext, set a password and download the program to the PLC. At the same time, we capture the traffic using Wireshark packet analyser. We find the packet that has the password string and check the PCCC addressing to identify the password address in the PLC's memory. Among the three password locations listed above, our network traffic shows that location (3) is where it is actually written when you set/reset the password.

Figure 9 (in appendix) shows part of the PCCC download messages for MicroLogix 1400 (Default). Figure 9(a) shows writing a new program without encrypting the password while Figure 9(b) shows with encrypted password.

Resetting the password. Figure 10 shows the crafted password reset message sent by the attacker to MicroLogix 1400 PLC. Note for MicroLogix 1100, only the File type would change in the crafted message. Since the password hash algorithm is unknown, the attacker downloads a program with her password to her PLC and captures the password hash from the network traffic. She then replaces the original hash with her password hash in the target PLC using our crafted write-message containing the attacker's hash and its memory location.

Network sniffing (T1040). The authentication or password set/reset protocol does not encrypt the password in transit, allowing the attacker to eavesdrop the password in plaintext or hash-code if the "encrypt password" option is enabled.

Credentials from password stores (T1555). If the project downloaded on the PLC does not have "encrypt password" enabled, the attacker can read the plaintext password remotely in two ways. 1) By simply connecting to the PLC through her engineering software(since the PLC sends the password as a result of read requests generated by the engineering software after it goes online), 2) By sending a crafted read request for the locations(mentioned before) which contain the password.

C. Attack Evaluation

Experimental settings. We evaluate the attacks on MicroLogix 1400 Series B (firmware version 15.000 and version 21.006), MicroLogix 1100 Series B (firmware version 16.000) and RSLogix 500 (version 9.05.01 and version 12.00.01). RSLogix 500 v9.05.01 and RSLogix 500 v12.00.01 run on Windows 7 VM and Windows 10 VM, respectively, while the attacks run on Ubuntu 16.04 VM.

Datasets. We utilize 27 password-protected control-logic programs for MicroLogix 1100 and 1400. These programs target different physical processes such as traffic light, conveyor belt, elevator, etc.

Evaluation result. Our evaluation achieves reading and writing to the PLCs without authentication and resetting their passwords in about 7 to 8 seconds. We also notice that increasing the data size to be read from or written to the PLCs do not significantly affect the run-time of the program. We perform the attacks on MicroLogix 1400 Series B (firmware version 15.000 and 21.006) and confirm that the latest firmware version is vulnerable if the default option is selected. However, the vendor has patched these vulnerabilities in the latest firmware by adding another controller type for MicroLogix 1400 with "Enhanced Password Security."

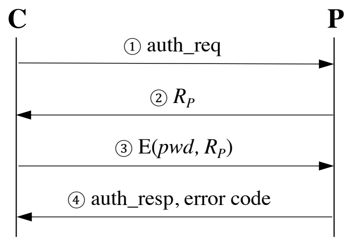


Fig. 5. MicroLogix 1400 (Enhanced Password Security) authentication protocol

VII. Case Study IV: MicroLogix 1400 (Enhanced Password Security)

A. Authentication Protocol

For MicroLogix 1400 with enhanced password security, the authentication gets done at the server-side unlike its default security option. Figure 5 shows the authentication process that takes place when the engineering software (RSLogix 500) attempts to connect to the PLC. First, the engineering software sends an authentication request (*auth_req*) which prompts the PLC to respond with a random number (R_P). Then the engineering software sends the encrypted password which we assume is an encryption function of the password, using R_P as an encryption key. We figure this out by launching a man-in-the-middle attack and manipulating the random number to one taken from an old network traffic. We observe that the third message (③) is the same for the same password and the same R_P, which means a client nonce is not used. It changes only when 1) the value of R_P changes 2) the value of password changes. The PLC responds with *error_code* the value of which is `0x00` for successful authentication and otherwise for failed authentication.

Set/reset password protocol. The protocol for setting a password is similar to the authentication process. Figure 11 (in appendix) shows the protocol for writing a password to the PLC. First, the engineering software initiates an authentication process by sending `msg` ① (refer to Figure 12 in appendix for the actual PCCC messages transferred). The PLC responds with a 20-byte random number (R_P). Then the engineering software sends a 40-byte response, the first 20 bytes of which is an encryption function of old password, using R_P as an encryption key. The next 20 bytes are an encryption function of new password, using R_P as an encryption key. We figured out this information by launching a MITM attack and updating the value of R_P being sent from the PLC to the engineering software with one taken from a previous network traffic. We realized that the 40-byte response was the same as the one observed in the previous traffic for the same password. We

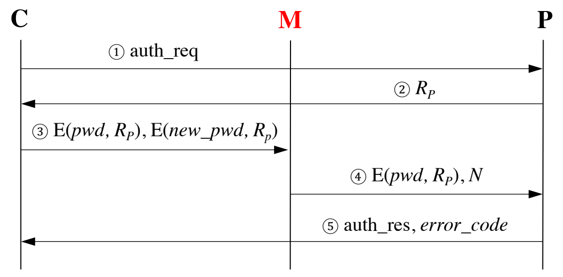

C
M
P
① auth_req
② R_P
③ E(pwd, R_P), E(new_pwd, R_p)
④ E(pwd, R_P), N
⑤ auth_res, $error\_code$

Fig. 6. MicroLogix 1400 (Enhanced Password Security): DoS attack

then kept the same R_P value and changed the value of the 40-byte response from the engineering software. We realized that the PLC did not let us update the password. So we again updated R_P to one taken from a previous network traffic and changed the value of the password we were trying to set from the engineering software. We observed that the second chunk of 20 bytes in the 40-byte response changed according to the new value of password.

Protocol Vulnerabilities. Despite the fact that a server nonce is used, this PLC has a weak encryption scheme for setting/resetting the password (V3).

B. MITRE ATT&CK on MicroLogix 1400 (Enhanced Password Security)

With a server nonce involved, an attacker cannot simply replay an authentication code previously captured. To attack MicroLogix's enhanced password security protocol, we assume an attacker with all the capabilities of our adversary model (i.e., eavesdropping, fabrication, and interception). The attacker's goal is to modify the password of a target PLC by interfering with the password set/reset protocol.

Man in the middle (T0830). The attacker's goal is to be able to overwrite the authorised password reset request with her own value. So, she poisons the ARP cache of the RSLogix software and MicroLogic 1400 PLC and intercepts the network traffic in between.

Transmitted Data Manipulation (T1565.002). After taking the role of man-in-the-middle, the attacker is able to manipulate the data in transit between RSLogix 500 and the PLC to launch her Denial of Serivce attack.

Endpoint Denial of Service (T1499). In this attack, the attacker makes a PLC inaccessible by overwriting the original password of a PLC with a garbage value. Figure 6 shows the attack sequence. The attacker sits at a man-in-the-middle position between MicroLogix 1400 and RSLogix 500, waiting for a field engineer to set/reset the password of the PLC. The attacker sniffs the client's authentication request (①) and the PLC's response (②). Then, she intercepts the third message (③) containing the 20-byte authentication code (i.e., *E(pwd,R_P)*) and the 20-byte (encrypted) new password (i.e., *E(new_pwd, R_P)*), and replaces the new password (the second 20-byte) with a random number N (④). Since the authentication code is correct, the PLC will process the password reset request and overwrite the current password with the output derived from the decryption function of N. Since N is a random number, the decrypted output will be a garbage value which would not represent an ASCII string known to the user(or attacker). This version of PLC does not provide a factory-reset feature for clearing its memory. Hence, this attack makes the PLC permanently unusable (even the attacker cannot access it).

C. Attack Evaluation

We performed our experiments on RSLogix 500 (version 12.00.01) and MicroLogix 1400 (firmware version 21.006) for evaluating the attacks. The engineering software runs on Windows 10 while the attack scripts run on Ubuntu 16.04 VM. We use Scapy as our packet manipulation tool. After evaluating the protocol on different control logic programs we performed the attack and confirm that it was successful since it did not allow us to login with the password set originally.

VIII. Case Study V: CLICK PLC

AutomationDirect's CLICK PLC is programmed using CLICK programming software in ladder logic. The PLC also has the option of protecting the read/write operation from unauthorized users. The engineering software is used to download a control logic program with a password into the PLC. Subsequent read/write operations can be done after successful authentication only. Unlike other PLCs in our study, CLICK PLC uses User Datagram Protocol (UDP) for the transport layer.

A. Authentication Protocol

Figure 7 shows the protocol for authenticating from the CLICK programming software to the PLC.

First, the engineering software sends the password to the PLC in clear text. The initial response to the authentication request is the same regardless of the correctness of the entered password. The engineering software follows with requesting *a1* to which the PLC responds with an *error_code*. The value of *error_code* is `0x0000` if the authentication is successful and otherwise for failed authentication. The value of *error_code* is followed by the entered password in the same message. If the authentication is successful the communication with the PLC proceeds and the engineering software can perform upload/download operations.

We derive information about the protocol fields via differential analysis of the network traffic generated by different control logic programs for upload/download and authentication operations. To summarize, the first four bytes(`0x4b4f5000`) in the UDP payload of all messages exchanged between the PLC and the engineering software are proprietary and stay constant. The fifth and sixth bytes identify the transaction ID the value of which is initiated by the engineering software and increments with each packet sent to the PLC. The response to a particular request packet can be determined by matching the transaction IDs. The next two (seventh and eighth) bytes are some sort of a checksum of the payload. We, however, did not reverse engineer the algorithm of the checksum. The ninth and tenth bytes contain the length of the payload. Finally, the rest

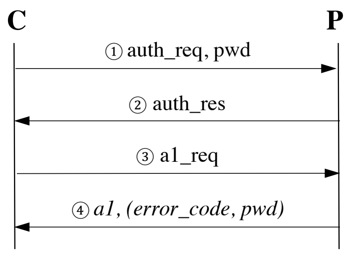


Fig. 7. CLICK authentication protocol

of the bytes are reserved for the actual data that gets transferred which we now refer to as CLICK data. We also determine that the first two bytes in the CLICK data remain constant for all the messages. For upload and download messages, the third byte in the CLICK data is `0x65`. We also find out that the fourth byte in the CLICK data is `0x04` for upload messages and `0x05` for download messages.

Protocol Vulnerabilities. Our attacks exploit two vulnerabilities in the PLC. The first one, Information Disclosure (V1), exists because of two reasons. 1) The password gets transmitted in clear text to the PLC, and 2) The PLC stores sensitive information (e.g., last entered password) in credential stores thereby allowing remote attackers to read the information. The second vulnerability exists because the PLC does not authenticate users per session (V7).

B. MITRE ATT&CK on CLICK PLC

Network sniffing (T1040). The first attack scenario requires only the eavesdropping capability in which the attacker can passively sniff the network traffic. Since the password gets transmitted in clear text, an attacker can eavesdrop the network traffic going from the engineering software to the PLC and acquire the password at the time of authentication as well as downloading a password-protected program. In Appendix, Figure 13(a) shows the write request message containing the password in clear text at the time of downloading the program while Figure 13 (b) shows the message being sent from the engineering software to the PLC at the time of authentication.

Credentials from password stores (T1555). The second case of attack requires only the fabrication capability. We found that the third message of the authentication process (③ *a1_req*), which prompts the PLC to send the *error_code* along with the entered password, is the same for different sessions and control logic programs. Surprisingly, one can directly send this *a1_req* message to a PLC skipping the first request/response messages (① and ②), and the PLC responds with the password in the first request message (①) of the most recent authentication. That means an attacker can send an *a1_req* request to the PLC and determine the last entered password and whether it was correct or not by checking the value of *error_code*. Figure 14 (in appendix) shows the request and response messages captured during the attack.

Impair defenses (T1562). We found from our network analysis of the protocol that the PLC does not authenticate a user per session. If a user in one protocol session is authenticated to a PLC, a global-state indicating successful authentication in the PLC considers all the requests as authenticated while the initially authenticated session is alive. This vulnerability allows an attacker with fabrication capability to read or modify the control logic of a PLC. The simplest way to exploit this vulnerability is using an attacker's engineering software at the time a legitimate user is logged in. The attacker can engineer the PLC until the legitimate user disconnects from the PLC. Initially, this vulnerability allowed us to access the PLC only until the legitimate user is connected to the PLC, but we found a way to retain the access even after the legitimate user disconnects. Once the attacker exploits the mentioned vulnerability and gains access to the PLC, she sends a specially crafted write-request packet to the PLC that essentially changes the PLC's protection-level to no-protection, allowing the attacker to read/write the PLC without authentication.

C. Attack Evaluation

We evaluate our attacks on the latest firmware of CLICK PLC (v2.60) and the latest programming software (v2.60). The programming software runs on Windows 10 VM while the attacker scripts run on Ubuntu 16.04 VM. Finally, we implement all of our attacks using Python and/or Scapy. We ran our attacks for different control logic programs and found that our attacks worked successfully for all of the programs. We have made a responsible disclosure of the vulnerabilities and the vendor has patched the vulnerabilities.

IX. Discussion on Authentication Protocols

Table III (in Appendix) presents a comparison of the identified vulnerabilities in the PLCs of our study (discussed in Section III-D), along with the affected PLC firmware versions. It depicts that the vulnerabilities exist across multiple vendors showing evidence of the bad-design practices in ICS industry for the proprietary PLC authentication-protocols.

Fundamental Design Issues. The act of authentication is proving an assertion of identity. It often involves a pair of user ID and password in typical IT domains. We identify three fundamental design issues in the authentication protocols that are common in the PLCs of our study, representing a general trend in PLC design across several ICS vendors.

Single user authentication. The PLCs utilize only a single password without identification data (such as username), essentially authenticating a single user group sharing the password. Thus, a PLC considers the communicating user as an authorized entity if the user knows the correct password. It opens a security hole since fine-grained access-control cannot be applied with a single user setting.

One-way authentication. The PLC authentication protocols support only the client authentication based on a user password. The PLCs as a server do not authenticate client applications such as engineering software. Thus, the client-only authentication allows an attacker to utilize a rogue PLC to disguise control center services. (see appendix for more information about rogue PLCs).

Unprotected write. Some PLCs like Modicon M221 support read-protection only and allow write-operations without authentication, allowing users to recover the PLCs by overwriting PLC configurations if they forget the password. However, if a PLC enforces both read and write protection and the users forget the PLC password, there is no ordinary way to recover the PLC, causing the loss of remote access to the PLC.

Furthermore, read-protection in PLCs can be considered more critical than write-protection in some situations. For instance, if the control logic of a PLC is changed in an unauthorized way, the modifications could be detected [18]–[20]. However, suppose the attacker reads the control logic without tampering with the existing PLC settings. In that case, it is stealthy and may help to craft a tailored cyber-weapon in further attack stage [2]).

Mitigation. The fundamental solution would be completely redesigning the protocols, but this would incur a high cost and may have backward compatibility issues [21]. Moreover, ICS devices are usually not patched on time and have a very long life-cycle compared to common IT devices [22]. Therefore, we should expect that insecure legacy devices will keep participating in a real-world ICS environment for a long time. In this regard, network detection can be seamlessly integrated into the existing ICS setting. In particular, control logic detection [6], [23] and verification [24]–[33] can be utilized to alleviate current situation. Partitioning the memory space and enforcing memory access control [34] can also prevent some password modification attacks. For example, the password reset attacks on Modicon M221 PLC overwrite some memory regions with an attacker's password hash. If the memory region is configured as read-only, then the attacks would fail.

Other suggestions include employing standard cryptography methods such as digital signatures (for messages such as control logic manipulation, password reset, etc.), increasing the key length to 256 bits (for long term protection against brute-force attacks) [35], using network monitoring tools like Snort [36], ArpAlert [37] and ArpWatchNG [38] (for detection and prevention of the attack involving MITM). But perhaps the best solution would be to prevent direct access to the SCADA devices by means of having a Demilitarized Zone (DMZ) that separates the Information technology (IT) domain from Operational Technology networks [39]–[41].

X. Related work

While the existing literature presents a wide-range of recent ICS attacks [4], [5], [23], [42]–[58], this section covers the PLC authentication attacks that are closest to our study. The current work focuses on demonstrating one particular attack using one vulnerability in a PLC [59]–[61]. On the other hand, we systematically study PLC authentication protocols of four major vendors' PLCs using the MITRE ATT&CK framework and identify several vulnerabilities and attack vectors to understand the design-issues of the proprietary protocols.

Wardak *et al.* [59] investigated access control vulnerabilities focusing mainly on password based access control in Siemens S7-400 PLC. They showed how passwords are stored in the PLC memory, how they can be sniffed and cracked etc and carried out attacks on ICS setup such as replay, PLC memory corruption, unauthorised password updating.

Similarly, Sandruwan *et al.* [60] present how an attacker can bypass login to change the register values of the PLC by simply injecting a worm in one of the PCs connected to the PLC network. They also propose a direct attack on the authentication mechanism of Siemens PLCs. They suggest replaying the authentication request messages containing the original password hash in order to successfully authenticate as an attacker. They also suggest using a password dictionary to generate a list of hashes and then comparing with the one captured in the network traffic. If a match is found, the attacker can use the corresponding plaintext word to authenticate herself with the PLC.

Grandgenette *et al.* [61] discover authentication bypass vulnerabilities in RSLogix 5000 software and ControlLogix 5573 PLC by examining the Common Industrial Protocol in the network traffic and reverse engineering the RSLogix 5000 software using IDAPro. They discover that the authentication is based on a challenge response mechanism and the 2058 bit RSA key used to decrypt the challenge is hard coded into RSLogix 5000 software. They reverse engineered the RSLogix 5000 software and figured out the entire authentication process and were able to replicate it using their own code in order to communicate with the PLC. Our work relies solely on network traffic examination to identify the authentication vulnerabilities unlike this one which also uses tools like IDAPro to reverse engineer the engineering software binary file.

XI. Conclusion

This paper presented an empirical study of the authentication mechanisms in five PLCs of four ICS vendors: Schneider Electric's Modicon M221, Allen-Bradley's MicroLogix 1100, and MicroLogix 1400, AutomationDirect's CLICK PLC, and Siemens' S7-300. It identified eight vulnerabilities in password authentication protocols and evaluated them using proof-of-concept exploits derived from MITRE ATT&CK knowledge base of adversary tactics and techniques. The study utilized network traffic examination to explore the internals of proprietary protocols and showed evidence of significant weaknesses in the ICS industry's design practices across four vendors. The sole reliance on network traffic for exploit development makes our study practical for real-world scenarios. It further provided recommendations to improve the authentication protocols based on the study findings.

Acknowledgement

We would like to thank Marina Krotofil and our anonymous reviewers for sharing their expert knowledge and for providing valuable (technical) suggestions.

This work was supported, in part, by the Virginia Commonwealth Cyber Initiative, an investment in the advancement of cyber R&D, innovation, and workforce development. For more information, visit www.cyberinitiative.org.

References

[1] N. Kush, E. Foo, E. Ahmed, I. Ahmed, and A. Clark, "Gap analysis of intrusion detection in smart grids," in *2nd International Cyber Resilience Conference (ICRC 2011)*, 2011, pp. 38–46.

[2] N. Falliere, L. O. Murchu, and E. Chien, "W32.stuxnet dossier," *White paper, Symantec Corp., Security Response*, vol. 5, no. 6, p. 29, 2011.

[3] I. Ahmed, V. Roussev, W. Johnson, S. Senthivel, and S. Sudhakaran, "A SCADA System Testbed for Cybersecurity and Forensic Research and Pedagogy," in *Proceedings of the 2nd Annual Industrial Control System Security Workshop (ICSS)*, 2016.

[4] H. Yoo and I. Ahmed, "Control Logic Injection Attacks on Industrial Control Systems," in *IFIP International Conference on ICT Systems Security and Privacy Protection*. Springer, 2019, pp. 33–48.

[5] S. Kalle, N. Ameen, H. Yoo, and I. Ahmed, "CLIK on PLCs! Attacking control logic with decompilation and virtual PLC," in *Binary Analysis Research (BAR) Workshop, Network and Distributed System Security Symposium (NDSS)*, 2019.

[6] H. Yoo, S. Kalle, J. Smith, and I. Ahmed, "Overshadow Plc to Detect Remote Control-Logic Injection Attacks," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2019, pp. 109–132.

[7] S. Qasim, A. Ayub, J. Johnson, and I. Ahmed, "Attacking the IEC-61131 Logic Engine in Programmable Logic Controllers in Industrial Control Systems," in *Critical Infrastructure Protection XV*, J. Staggs and S. Shenoi, Eds. Cham: Springer International Publishing, 2021.

[8] S. Bhatia, S. Behal, and I. Ahmed, "Distributed Denial of Service Attacks and Defense Mechanisms: Current Landscape and Future Directions," in *Versatile Cybersecurity*, vol. 72. Cham: Springer International Publishing, 2018.

[9] "Attackers Deploy New ICS Attack Framework "TRITON" and Cause Operational Disruption to Critical Infrastructure," https://www.fireeye.com/blog/threat-research/2017/12/attackers-deploy-new-ics-attack-framework-triton.html, [Online; accessed 12-April-2021].

[10] R. M Lee, M. J Assante, and T. Conway, "Analysis of the cyber attack on the Ukrainian power grid," SANS Industrial Control Systems, Tech. Rep., 2016.

[11] I. Ahmed, S. Obermeier, M. Naedele, and G. G. Richard III, "SCADA Systems: Challenges for Forensic Investigators," *Computer*, vol. 45, no. 12, pp. 44–51, 2012.

[12] "MITRE ATT&CK," https://attack.mitre.org/, 2020.

[13] I. Ahmed, S. Obermeier, S. Sudhakaran, and V. Roussev, "Programmable Logic Controller Forensics," *IEEE Security Privacy*, vol. 15, no. 6, pp. 18–24, November 2017.

[14] "Wireshark tool," https://www.wireshark.org/, 2020.

[15] "TCPDUMP & LIBPCAP," https://www.tcpdump.org/, [Online; accessed 15-April-2021].

[16] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[17] S. Senthivel, I. Ahmed, and V. Roussev, "SCADA network forensics of the PCCC protocol," *Digital Investigation*, vol. 22, pp. S57–S65, 2017.

[18] D. Hadžiosmanović, R. Sommer, E. Zambon, and P. H. Hartel, "Through the Eye of the PLC: Semantic Security Monitoring for Industrial Processes," in *Proceedings of the 30th Annual Computer Security Applications Conference (ACSAC)*, 2014.

[19] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A Trusted Safety Verifier for Process Controller Code." in *Proceeding of the 21st Network and Distributed System Security Symposium (NDSS)*, 2014.

[20] Z. Sun, B. Feng, L. Lu, and S. Jha, "OAT: Attesting operation integrity of embedded devices," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 1433–1449.

[21] A. Bolshev, J. Larsen, M. Krotofil, and R. Wightman, "A Rising Tide: Design Exploits in Industrial Control Systems," in *10th USENIX Workshop on Offensive Technologies (WOOT 16)*. Austin, TX: USENIX Association, Aug. 2016.

[22] K. Wang, G. Cretu, and S. J. Stolfo, "Anomalous Payload-based Worm Detection and Signature Generation," in *Proceedings of the 8th International Conference on Recent Advances in Intrusion Detection*, ser. RAID'05, 2006.

[23] S. Senthivel, S. Dhungana, H. Yoo, I. Ahmed, and V. Roussev, "Denial of engineering operations attacks in industrial control systems," in *Proceedings of the Eighth ACM Conference on Data and Application Security and Privacy*, 2018, pp. 319–329.

[24] H. Wan, G. Chen, X. Song, and M. Gu, "Formalization and verification of PLC timers in Coq," in *2009 33rd Annual IEEE International Computer Software and Applications Conference*, vol. 1. IEEE, 2009, pp. 315–323.

[25] M. Zhou, F. He, M. Gu, and X. Song, "Translation-based model checking for PLC programs," in *2009 33rd Annual IEEE International Computer Software and Applications Conference*, vol. 1. IEEE, 2009, pp. 553–562.

[26] S. E. McLaughlin, S. A. Zonouz, D. J. Pohly, and P. D. McDaniel, "A Trusted Safety Verifier for Process Controller Code." in *NDSS*, vol. 14, 2014.

[27] S. O. Biha, "A formal semantics of PLC programs in Coq," in *2011 IEEE 35th Annual Computer Software and Applications Conference*. IEEE, 2011, pp. 118–127.

[28] R. Wang, Y. Guan, L. Luo, X. Song, and J. Zhang, "Formal modelling of PLC systems by BIP components," in *2013 IEEE 37th Annual Computer Software and Applications Conference*. IEEE, 2013, pp. 512–518.

[29] L. Garcia, S. Mitsch, and A. Platzer, "HyPLC: Hybrid programmable logic controller program translation for verification," in *Proceedings of the 10th ACM/IEEE International Conference on Cyber-Physical Systems*, 2019, pp. 47–56.

[30] M. Zhang, C.-Y. Chen, B.-C. Kao, Y. Qamsane, Y. Shao, Y. Lin, E. Shi, S. Mohan, K. Barton, J. Moyne *et al.*, "Towards automated safety vetting of PLC code in real-world plants," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 522–538.

[31] H. Carlsson, B. Svensson, F. Danielsson, and B. Lennartson, "Methods for reliable simulation-based PLC code verification," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 2, pp. 267–278, 2012.

[32] B. F. Adiego, D. Darvas, E. B. Viñuela, J.-C. Tournier, S. Bliudze, J. O. Blech, and V. M. G. Suárez, "Applying model checking to industrial-sized PLC programs," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 6, pp. 1400–1410, 2015.

[33] S. Zonouz, J. Rrushi, and S. McLaughlin, "Detecting industrial control malware using automated plc code analytics," *IEEE Security & Privacy*, vol. 12, no. 6, pp. 40–47, 2014.

[34] "Securing Real-Time Microcontroller Systems through Customized Memory View Switching., author=Kim, Chung Hwan and Kim, Taegyu and Choi, Hongjun and Gu, Zhongshu and Lee, Byoungyoung and Zhang, Xiangyu and Xu, Dongyan," in *NDSS*, 2018.

[35] "Cryptography Length Recommendation," https://www.keylength.com/en/3/, 2020, [Online; accessed 15-March-2021].

[36] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[37] "Arpalert," https://www.arpalert.org/arpalert.html, [Online; accessed 15-March-2021].

[38] "arpwatch," https://en.wikipedia.org/wiki/Arpwatch, [Online; accessed 15-March-2021].

[39] A. Hassanzadeh, S. Modi, and S. Mulchandani, "Towards effective security control assignment in the Industrial Internet of Things," in *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, 2015, pp. 795–800.

[40] M. Almgren, P. Andersson, G. Björkman, M. Ekstedt, J. Hallberg, S. Nadjm-Tehrani, and E. Westring, "RICS-el: Building a National Testbed for Research and Training on SCADA Security (Short Paper)," in *Critical Information Infrastructures Security*, E. Luiijf, I. Žutautaitė, and B. M. Hämmerli, Eds. Cham: Springer International Publishing, 2019, pp. 219–225.

[41] "DMZ," https://www.fortinet.com/resources/cyberglossary/what-is-dmz, [Online; accessed 4-April-2021].

[42] M. Krotofil, K. Kursawe, and D. Gollmann, *Securing Industrial Control Systems*. Cham: Springer International Publishing, 2019, pp. 3–27.

[43] A. Bolshev, J. Larsen, M. Krotofil, and R. Wightman, "A Rising Tide: Design Exploits in Industrial Control Systems," in *Proceedings of the 10th USENIX Conference on Offensive Technologies*, ser. WOOT'16. USA: USENIX Association, 2016, p. 178–188.

[44] A. Erba, R. Taormina, S. Galelli, M. Pogliani, M. Carminati, S. Zanero, and N. O. Tippenhauer, "Constrained Concealment Attacks against Reconstruction-Based Anomaly Detectors in Industrial Control Systems," in *Annual Computer Security Applications Conference*, ser. ACSAC '20, New York, NY, USA, 2020, p. 480–495.

[45] E. López-Morales, C. Rubio-Medrano, A. Doupé, Y. Shoshitaishvili, R. Wang, T. Bao, and G.-J. Ahn, "HoneyPLC: A Next-Generation Honeypot for Industrial Control Systems," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '20, New York, NY, USA, 2020, p. 279–291.

[46] S. E. McLaughlin, "On Dynamic Malware Payloads Aimed at Programmable Logic Controllers." in *HotSec*, 2011.

[47] S. McLaughlin and P. McDaniel, "SABOT: specification-based payload generation for programmable logic controllers," in *Proceedings of the 2012 ACM conference on Computer and communications security*, 2012, pp. 439–449.

[48] A. Abbasi and M. Hashemi, "Ghost in the plc designing an undetectable programmable logic controller rootkit via pin control attack," *Black Hat Europe*, vol. 2016, pp. 1–35, 2016.

[49] L. Garcia, F. Brasser, M. H. Cintuglu, A.-R. Sadeghi, O. A. Mohammed, and S. A. Zonouz, "Hey, My Malware Knows Physics! Attacking PLCs with Physical Model Aware Rootkit." in *NDSS*, 2017.

[50] B. Lim, D. Chen, Y. An, Z. Kalbarczyk, and R. Iyer, "Attack induced common-mode failures on PLC-based safety system in a nuclear power plant: practical experience report," in *2017 IEEE 22nd Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE, 2017, pp. 205–210.

[51] N. Govil, A. Agrawal, and N. O. Tippenhauer, "On ladder logic bombs in industrial control systems," in *Computer Security*. Springer, 2017, pp. 110–126.

[52] S. McLaughlin and S. Zonouz, "Controller-aware false data injection against programmable logic controllers," in *2014 IEEE International Conference on Smart Grid Communications (SmartGridComm)*. IEEE, 2014, pp. 848–853.

[53] D. Beresford, "Exploiting siemens simatic s7 plcs," *Black Hat USA*, vol. 16, no. 2, pp. 723–733, 2011.

[54] J. Klick, S. Lau, D. Marzin, J.-O. Malchow, and V. Roth, "Internet-facing PLCs-a new back orifice," *Blackhat USA*, pp. 22–26, 2015.

[55] R. Spenneberg, M. Brüggemann, and H. Schwartke, "Plc-blaster: A worm living solely in the plc," *Black Hat Asia*, vol. 16, pp. 1–16, 2016.

[56] S. A. Qasim, J. Lopez, and I. Ahmed, "Automated Reconstruction of Control Logic for Programmable Logic Controller Forensics," in *Information Security*, Z. Lin, C. Papamanthou, and M. Polychronakis, Eds. Cham: Springer International Publishing, 2019, pp. 402–422.

[57] S. A. Qasim, J. M. Smith, and I. Ahmed, "Control Logic Forensics Framework using Built-in Decompiler of Engineering Software in Industrial Control Systems," *Forensic Science International: Digital Investigation*, vol. 33, p. 301013, 2020.

[58] M. H. Rais, Y. Li, and I. Ahmed, "Spatiotemporal G-code Modeling for Secure FDM-based 3D Printing," in *Proceedings of the ACM/IEEE twelfth International Conference on Cyber-Physical Systems*, ser. ICCPS '21. New York, NY, USA: Association for Computing Machinery, 2021.

[59] H. Wardak, S. Zhioua, and A. Almulhem, "Plc access control: a security analysis," in *Industrial Control Systems Security (WCICSS), 2016 World Congress on IEEE, pp. 1–6*, 2016.

[60] G. P. H. Sandaruwan, P. S. Ranaweera, and V. A. Oleshchuk, "PLC security and critical infrastructure protection," in *Proc. 2013 IEEE 8th Int. Conf. on Industrial and Information Systems, pp. 81-85*, 2013.

[61] R. Grandgenett, W. Mahoney, and R. Gandhi, "Authentication bypass and remote escalated I/O command attacks," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, 2015, pp. 1–7.

[62] E. Biham, S. Bitan, A. Carmel, A. Dankner, U. Malin, and A. Woo, "Rogue7: Rogue Engineering-Station attacks on S7 Simatic PLCs," in *Black Hat USA*, 2019.

APPENDIX

A. MITRE ATT&CK for Exploitation

We have created several exploits of these vulnerabilities and evaluated them successfully on their respective PLCs. Figure 8 illustrates our attack model where the attacker is in control center and can communicate with PLCs at field sites. Table II summarizes and compares eight attack instances on target PLCs. The attacks are obtained from MITRE ATT&CK knowledge base of adversary tactics and techniques [12]. They are discussed as follows under the given ICS context.

T1555 - Credentials from password stores. The attacker can send a crafted request to read the cache containing the last entered password.

T1040 - Network sniffing. The attacker can sniff the network traffic at the time of authentication and/or downloading a project and obtain credentials in clear text.

T1098 - Unauthorised Password Reset. The attacker sends a crafted request to reset the password with her own.

T1562 - Impair Defenses. The attacker comes up with a technique to impair preventive security controls such as authentication.

T1110.002 - Password Cracking. The attacker exploits the vulnerabilities(mentioned ahead) to crack the password if she is able to sniff the network traffic.

T0830 - Man in the middle (MITM). The attacker sits between the machine running the Engineering Software and the PLC by poisoning the ARP cache of the two machines to manipulate data.

T1565.002 - Transmitted data manipulation. The MITM attacker sniffs and manipulates the data in transit(including password hash) at the time of downloading a password protected project.

T1499 - Endpoint Denial of service. The MITM manipulates the fields containing the password hash and makes the PLC inaccessible exploiting a lack of factory reset feature.

B. Rogue PLCs

A rouge PLC can be a PLC compromised remotely by an attacker or a PLC that an attacker slips into a target environment. Specifically, Siemens S7-1500 PLCs have a built-in private key for authentication. However, the engineering software cannot detect a rogue PLC since all PLCs of the same model and firmware version share the same private key [62]. The attacker can also reverse engineer the PLC firmware to extract the private key to use it in other PLCs. Moreover, a rogue PLC can also be a software-based implementation (virtual PLC). Kalle *et al.* [5] utilize a virtual PLC to hide an infected control logic in a target PLC from a legitimate engineering software in real-time.

C. Vulnerability Disclosure

We responsibly disclosed all vulnerabilities to respective vendors by August 2020. Siemens, Allen-Bradley, and Schneider Electric released firmware patches and issued CVEs within three months. AutomationDirect released the patches in February, 2021.

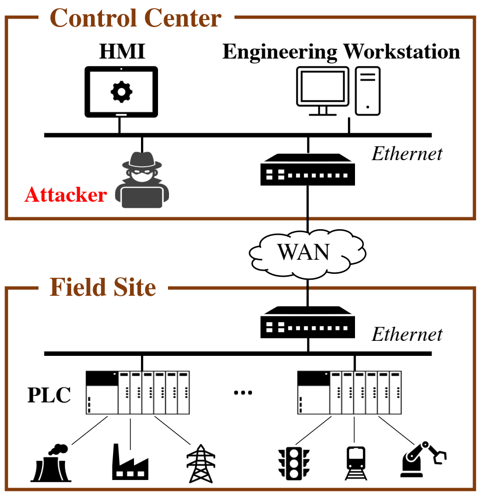


Fig. 8. Attack model illustration

Algorithm 1 Pseudocode for Password Reset Attack

```
1: $zero \leftarrow$ 128 byte array of 0x00
2: $startAddress \leftarrow$ 0xd000
3: $endAddress \leftarrow$ 0xe000
4: $offset \leftarrow startAddress$
5: $maxSize \leftarrow$ 128 // maximum payload length of M221
6: while $offset \neq endAddress$ do
7:     Send a write request(addr:$offset$, size:$maxSize$, payload:$zero$)
8:     $offset \leftarrow offset + maxSize$
9: end while
10: $m1 \leftarrow$ Request $m1$ from PLC
11: $m2 \leftarrow$ Random number between 0-255
12: $hashSize \leftarrow$ 32 // SHA-256
13: for $i$ = 0 to $hashSize$-1 do
14:     $maskedHash[i] \leftarrow m1 \oplus m2$
15: end for
16: Send an authentication request($m2$,$maskedHash$) to PLC
```

Algorithm 2 Pseudocode of the weak encryption algorithm

```
Input: password ($P_0 ... P_7$), $K$ (where $K$ is one-byte secret key)
Output: encrypted_password ($E_0 ... E_7$)
1: for $i$ = 0 to 7 do
2:     $N_i$ = Substitute($P_i$)
3:     if $i$ ¡ 2 then
4:         $E_i = K \oplus N_i$
5:     else
6:         $E_i = K \oplus E_{i-2} \oplus N_i$
7:     end if
8: end for
```

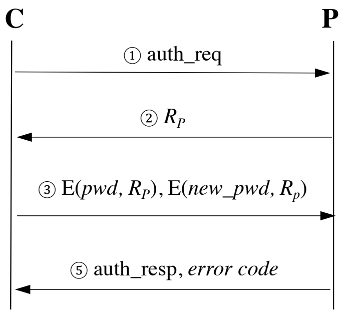
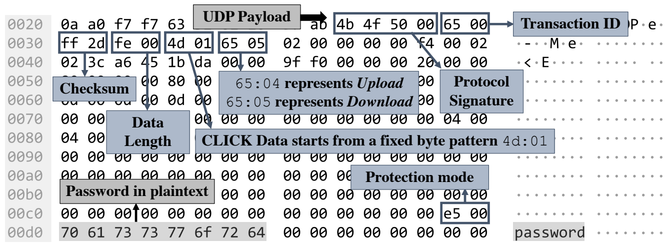
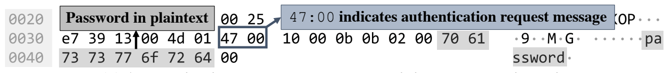


Fig. 11. MicroLogix 1400 (Enhanced Password Security): Password set/reset protocol



```
0020  0a a0 f7 f7 63          ab 4b 4f 50 00 65 00          P·e·
0030  ff 2d fe 00 4d 01 65 05  02 00 00 00 00 f4 00 02    ·-··M·e· ········
0040  02 3c a6 45 1b da 00 00  9f f0 00 00 00 20 00 00    ·<·E···· ········
0050           00 80 00                          00       ········ ········
0060           00 0d 00                          00       ········ ········
0070  00 00          00 00 00  00 00 00 00 00 00 04 00    ········ ········
0080  04 00                                               ········ ········
0090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00    ········ ········
00a0  00 00 00 00 00 00 00 00  00 00 00                   ········ ········
00b0                    00 00  00 00 00 00 00 00 00 00    ········ ········
00c0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 e5 00    ········ ········
00d0  70 61 73 73 77 6f 72 64  00 00 00 00 00 00 00 00    password ········
```

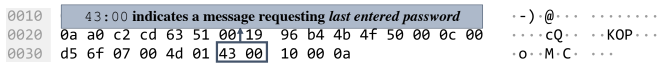
(a) Download message containing password in plaintext



```
0020                          00 25                               KOP···
0030  e7 39 13 00 4d 01 47 00  10 00 0b 0b 02 00 70 61    ·9··M·G· ······pa
0040  73 73 77 6f 72 64 00                                ssword·
```

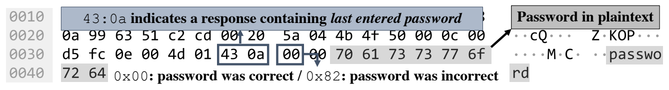
(b) Authentication request message containing password in plaintext

Fig. 13. CLICK password exposure-1 (eavesdropping)



```
0010                                                      ··-)·@··· ········
0020  0a a0 c2 cd 63 51 00 19  96 b4 4b 4f 50 00 0c 00    ····cQ·· ··KOP···
0030  d5 6f 07 00 4d 01 43 00  10 00 0a                   ·o··M·C· ···
```

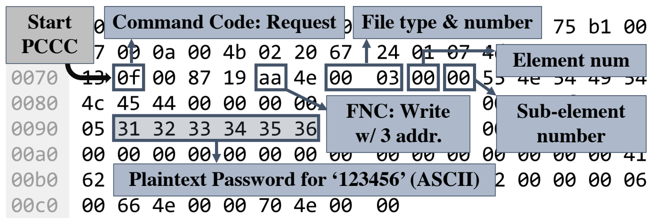
(a) Message requesting last entered password


43:0a indicates a response containing last entered password
Password in plaintext
0x00: password was correct / 0x82: password was incorrect

```
0010
0020  0a 99 63 51 c2 cd 00 20  5a 04 4b 4f 50 00 0c 00    ··cQ···  Z·KOP···
0030  d5 fc 0e 00 4d 01 43 0a  00 00 70 61 73 73 77 6f    ····M·C· ··passwo
0040  72 64                                               rd
```

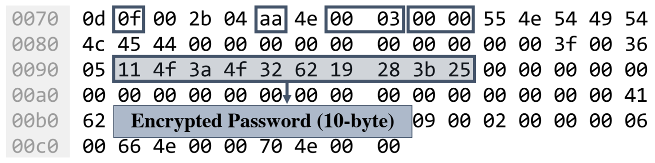
(b) Response message containing last entered password

Fig. 14. CLICK password exposure-2 (fabrication)



```
                                                 75 b1 00
      7 00 0a 00 4b 02 20 67  24 01 07 4
0070  13 0f 00 87 19 aa 4e 00  03 00 00 55 4e 54 49 54
0080  4c 45 44 00 00 00 00                0
0090  05 31 32 33 34 35 36                0
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 41
00b0  62                                2 00 00 00 06
00c0  00 66 4e 00 00 70 4e 00  00
```

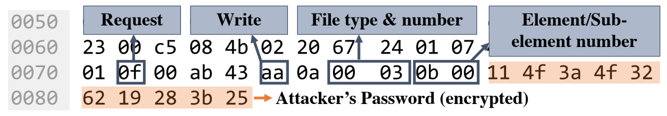
(a) PCCC download message with *plaintext* password



```
0070  0d 0f 00 2b 04 aa 4e 00  03 00 00 55 4e 54 49 54
0080  4c 45 44 00 00 00 00 00  00 00 00 00 00 3f 00 36
0090  05 11 4f 3a 4f 32 62 19  28 3b 25 00 00 00 00 00
00a0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 41
00b0  62                       09 00 02 00 00 00 06
00c0  00 66 4e 00 00 70 4e 00  00
```

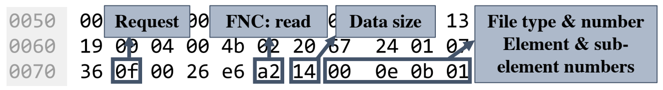
(b) PCCC download message with *encrypted* password

Fig. 9. MicroLogix 1400 (Default): PCCC messages for downloading a password-protected PLC program



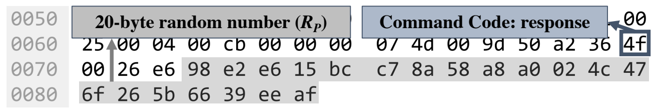
```
0050
0060  23 00 c5 08 4b 02 20 67  24 01 07
0070  01 0f 00 ab 43 aa 0a 00  03 0b 00 11 4f 3a 4f 32
0080  62 19 28 3b 25
```

Fig. 10. MicroLogix 1400 (Default): password reset attack



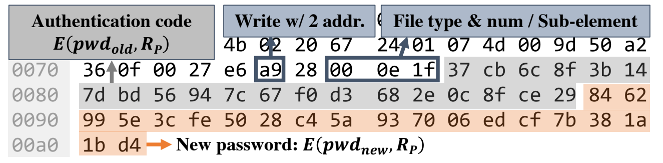
```
0050  00          00            13
0060  19 00 04 00 4b 02 20 67  24 01 07
0070  36 0f 00 26 e6 a2 14 00  0e 0b 01
```

(a) Authentication request (msg ①)



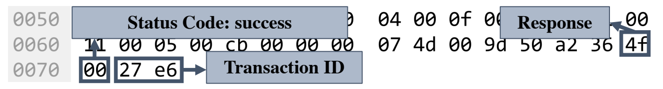
```
0050                                                00
0060  25 00 04 00 cb 00 00 00  07 4d 00 9d 50 a2 36 4f
0070  00 26 e6 98 e2 e6 15 bc  c7 8a 58 a8 a0 02 4c 47
0080  6f 26 5b 66 39 ee af
```

(b) Response with a random number (msg ②)


Authentication code $E(pwd_{old}, R_P)$
Write w/ 2 addr.
File type & num / Sub-element
New password: $E(pwd_{new}, R_P)$

```
            4b 02 20 67  24 01 07 4d 00 9d 50 a2
0070  36 0f 00 27 e6 a9 28 00  0e 1f 37 cb 6c 8f 3b 14
0080  7d bd 56 94 7c 67 f0 d3  68 2e 0c 8f ce 29 84 62
0090  99 5e 3c fe 50 28 c4 5a  93 70 06 ed cf 7b 38 1a
00a0  1b d4
```

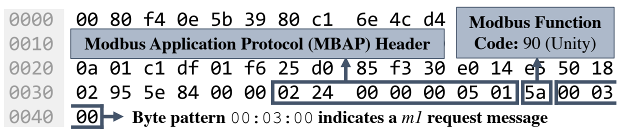
(c) Send authentication code with new password (msg ③)



```
0050                           04 00 0f 00             00
0060  11 00 05 00 cb 00 00 00  07 4d 00 9d 50 a2 36 4f
0070  00 27 e6
```

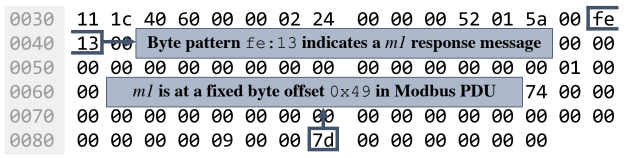
(d) Response with authentication result (msg ④)

Fig. 12. MicroLogix 1400 (Enhanced Password Security): PCCC messages for password reset


Modbus Application Protocol (MBAP) Header
Modbus Function Code: 90 (Unity)
Byte pattern 00:03:00 indicates a m1 request message

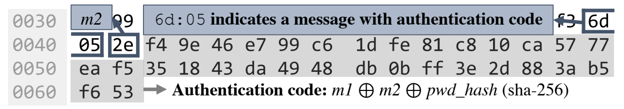
```
0000  00 80 f4 0e 5b 39 80 c1  6e 4c d4
0010
0020  0a 01 c1 df 01 f6 25 d0  85 f3 30 e0 14 e5 50 18
0030  02 95 5e 84 00 00 02 24  00 00 00 05 01 5a 00 03
0040  00
```

(a) *m1* request message (msg ①)


Byte pattern fe:13 indicates a m1 response message
m1 is at a fixed byte offset 0x49 in Modbus PDU

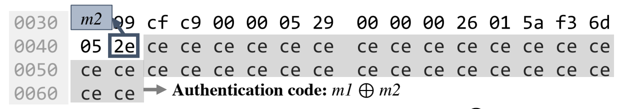
```
0030  11 1c 40 60 00 00 02 24  00 00 00 52 01 5a 00 fe
0040  13 00                                      00 00
0050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 01 00
0060  00                                         74 00 00
0070  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
0080  00 00 00 00 09 00 00 7d  00 00 00 00 00 00
```

(b) *m1* response message (msg ②)


m2
6d:05 indicates a message with authentication code
Authentication code: m1 ⊕ m2 ⊕ pwd_hash (sha-256)

```
0030     99                                      f3 6d
0040  05 2e f4 9e 46 e7 99 c6  1d fe 81 c8 10 ca 57 77
0050  ea f5 35 18 43 da 49 48  db 0b ff 3e 2d 88 3a b5
0060  f6 53
```

(c) Authentication code from Attacker’s engineering software (msg ③)



```
0030     99 cf c9 00 00 05 29  00 00 00 26 01 5a f3 6d
0040  05 2e ce ce ce ce ce ce  ce ce ce ce ce ce ce ce
0050  ce ce ce ce ce ce ce ce  ce ce ce ce ce ce ce ce
0060  ce ce
```

(d) Authentication code to PLC (msg ④)

Fig. 15. Authentication with zero-bytes hash

TABLE III
VULNERABILITY COMPARISON IN THE PLCS OF DIFFERENT VENDORS

Vul ID	Vulnerabilty	Target PLCs' firmware versions affected				
		M221	**S7-300/400**	**MicroLogix 1100**	**MicroLogix 1400**	**CLICK**
V1	Information Disclosure	n/a	n/a	Ver <= 16.0	Ver <= 21.2	Ver 2.6
V2	Client side authentication	n/a	n/a	Ver <= 16.0	Ver <= 21.1	n/a
V3	Weak encryption algorithm	Ver <1.6.2	All versions	n/a	Ver 21.6	n/a
V4	Small key space	Ver <1.6.2	All versions	n/a	n/a	n/a
V5	Lack of nonces	n/a	All versions	n/a	n/a	n/a
V6	Use of same keys	n/a	All versions	n/a	n/a	n/a
V7	Improper session management	n/a	n/a	n/a	n/a	Ver 2.6
V8	No write protection	Ver <= 1.6.2	n/a	n/a	n/a	n/a

TABLE IV
COMPARISON BETWEEN TWO PASSWORD RESET ATTACKS ON MODICON M221

Attack type	Run time /sec	Write requests	Payload size	Failed auth. attempts	Attack success rate
0x00ed (efficient) attack	0.06571	32	128	0	100%
Kalle *et al.* [5]	9.93	2458	32	2457	100%

TABLE V
ENCODING METHOD USED IN SIEMENS S7-300 ENCRYPTION ALGORITHM

Character	Encoded (Hex)	Character	Encoded (hex)	Character	Encoded (Hex)	Character	Encoded (Hex)	Character	Encoded (Hex)	Character	Encoded (Hex)
space	10	@	70	‘	50	0	0	P	60	p	40
!	11	A	71	a	51	1	1	Q	61	q	41
”	12	B	72	b	52	2	2	R	62	r	42
#	13	C	73	c	53	3	3	S	63	s	43
$	14	D	74	d	54	4	4	T	64	t	44
%	15	E	75	e	55	5	5	U	65	u	45
&	16	F	76	f	56	6	6	V	66	v	46
’	17	G	77	g	57	7	7	W	67	w	47
(	18	H	78	h	58	8	8	X	68	x	48
)	19	I	79	i	59	9	9	Y	69	y	49
*	1a	J	7a	j	5a	:	a	Z	6a	z	4a
+	1b	K	7b	k	5b	;	b	[	6b	{	4b
,	1c	L	7c	l	5c	<	c	\	6c	—	4c
-	1d	M	7d	m	5d	=	d	]	6d	}	4d
.	1e	N	7e	n	5e	>	e	ˆ	6e	∼	4e
/	1f	O	7f	o	5f	?	f	_	6f		