Security Hardening of Industrial Control Systems through Attribute Based Access Control

Shwetha Gowdanakatte
Shwetha.Gowdanakatte@colostate.edu
Department of Systems Engineering,
Colorado State University
Fort Collins, Colorado, USA

Mahmoud Abdelgawad
m.abdelgawad@colostate.edu
Department of Computer Science,
Colorado State University
Fort Collins, Colorado, USA

Indrakshi Ray
indrakshi.ray@colostate.edu
Department of Computer Science,
Colorado State University
Fort Collins, Colorado, USA

ABSTRACT

Industrial Control Systems (ICS) form a part of nations' critical infrastructure. ICS comprises Programmable Logic Controllers (PLC) and other components. In an innovative and connected world, the vulnerabilities in ICS components can be exploited. Authentication and access control stand as the first level of defense for protecting ICS from cyberattacks. We demonstrate in our lab that PLC is prone to Denial of Service (DoS) attacks. Subsequently, an attribute-based access control mechanism is implemented to harden the security of PLC. The demonstration of the security hardened system showcases that PLC is no longer susceptible. Furthermore, the Coloured Petri Nets (CPN) is used to analyze the behavior of security hardened systems and provide formal assurance.

KEYWORDS

Attribute-Based Access Control (ABAC), Industrial Control Systems (ICS), Programmable Logic Controllers (PLC), Formal Verification, Coloured Petri Nets (CPN)

ACM Reference Format:
Shwetha Gowdanakatte, Mahmoud Abdelgawad, and Indrakshi Ray. 2023. Security Hardening of Industrial Control Systems through Attribute Based Access Control. In *Proceedings of the 9th Annual Industrial Control System Security Workshop (ICSS '23), December 4, 2023, Austin, Texas, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Industrial Control Systems (ICS) maintain critical infrastructure such as power plants, chemical plants, water treatment plants, railways, and transportation systems. Due to technological advancements, ICS connect to other Information Technology (IT) systems for remote monitoring, control, and data collection. Modern ICS support communication protocols over Ethernet and interact with remote web servers through the Internet. Such communications may cause cybersecurity attacks on ICS, such as the German steel mill attack [18], the Ukrainian power grid attack [1], and Stuxnet attack on the Iranian nuclear centrifuge [7].

Many of these problems are traced to flaws in the authentication and access control mechanisms. Vendors often forget to change default passwords, which can be compromised easily. There are also security design issues with authentication protocols in major ICS vendors such as Allen-Bradley, Siemens, Schneider Electric, and Automation Direct, which can lead to authentication bypass, password sniffing, password cracking, and password reset attacks [4]. Authentication and access control breach is often the first step in compromising the integrity or availability of the system, leading to catastrophic disasters.

ICS components have a long life. Consequently, it is impossible to change all the components in existing PLCs, or modify them. In this paper, we propose the use of a stronger access control mechanism to counter the effect of authentication vulnerabilities. This does not require changing the PLCs. We begin by demonstrating how authentication breach can cause a Denial of Service (DoS) attack in an existing PLC using our experimental setup. We then implement an attribute-based access control mechanism that verifies multiple attributes of the user to prevent unauthorized users from gaining access and launching the DoS attack. We verify this empirically. For more complex attacks, it may be impossible to verify systems using an experimental setup. Towards this end, we show how formal methods can be used for providing assurance about a system's behavior.

The access control mechanism that we use is NIST Next Generation Access Control (NGAC) [14]. NGAC is an attribute-based access control model suitable for situational monitoring application. ICS are event-based system and the policies may have to be changed on-the-fly, hence NIST NGAC is most suitable. This is because NIST NGAC support obligation policies through which access control configuration can be modified. We use the Coloured Petri Nets (CPN) [15] for formal analysis. CPN have an extensive tool support and through a modular approach they allow for analyzing complex systems. The CPN tool is used to verify and analyze the ABAC gateway functionality.

We proceed with this paper as follows. Section 2 describes PLC and their security vulnerabilities. It also illustrates the NIST NGAC used for PLC. Section 3 expresses the threat model and demonstrates a DoS attack on PLC. Section 4 explains how we design and implement an ABAC gateway to harden the security for PLC. Section 5 executes formal verification to exercise the PLC with and without the ABAC gateway. Section 6 briefs the related work, and section 7 concludes the paper and points to future work.

2 BACKGROUND

2.1 PLC and their Security Vulnerabilities

The generic architecture of an ICS is shown in Figure 1. ICS comprises Programmable Logic Controllers (PLC), Human Machine Interfaces (HMI), field devices (actuators and sensors), and optional Remote Monitoring and Control (RMC).

A PLC is an industrialized computer dedicated to monitoring and controlling a process. It generally consists of digital and analog modules, communication, software, and firmware module. An HMI

ICSS '23, December 4, 2023, Austin, Texas, USA
2023. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

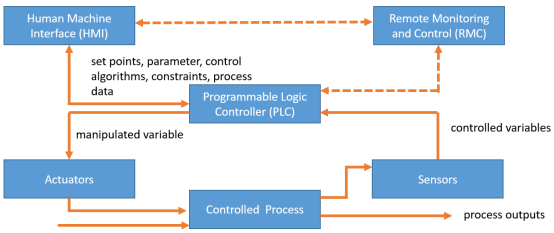

Figure 1: Generic Architecture of ICS

is a graphical user interface used for configuration settings, communicating parameters with the PLC, process monitoring, and event logging. The PLC receives configuration settings and commands for the controlled process from the HMI or the RMC. It receives feedback from sensors and controls the actuators by manipulating the control variables. It communicates the data and status of the controlled process to the HMI or the RMC. The typical life cycle of an ICS consists of design, build, commissioning, operation, and decommissioning. In addition to the hardware components, ICS vendors typically provide an engineering framework, which has engineering workstations that run application software for designing and developing control software, updating firmware and control software, and troubleshooting [12].

Examples of PLC include Siemens S7-1500 and Rockwell Compact Logix. The Siemens S7-1500 PLC uses a client-server model protocol, specifically, S7-Protocol version P3 designed over TCP/IP. Siemens provides a Totally Integration Automation (TIA) system as an engineering framework to work remotely with ICS devices. The TIA uses a handshake mechanism to establish a communication session with the PLC. This handshake mechanism utilizes the PLC firmware version to create a session key that is used in communication exchanges between TIA and PLC. Hence, all S7-1500 PLCs with the same firmware version can impersonate one another. This leads to various vulnerabilities in the P3 protocol, such as authentication bypass vulnerability (CVE-2019-10943) [20] and high-severity memory protection bypass vulnerability (CVE-2020-15782) [22].

Rockwell provides the so-called Studio 5000 as the engineering framework. All Rockwell PLCs use Common Industrial Protocol (CIP) for communications encapsulated in TCP/IP. Studio 5000 uses a handshake mechanism to establish a communication session with the PLC. Studio 5000 uses a hard coded key to establish a communication session with the PLC. Exchanging a hard coded key between the Studio 5000 and PLC leads to vulnerabilities, such as authentication bypass vulnerability (CVE-2021-22681) [23] and Denial-Of-Service vulnerability (CVE-2019-10952) [21].

2.2 NIST NGAC for PLC

NGAC is a generic Attribute Based Access Control (ABAC) [14] architecture suitable for distributed systems and situational monitoring applications. The NGAC model consists of basic elements and relations. The basic elements include users, resources, operations, user attributes, resource attributes, and policy classes. NGAC expresses policy through assignment, association, prohibition, and obligation relations.

The relations connect basic elements in a hierarchy configuration. The privileges are derived from these relations for access control decisions. For the PLC, the policy in the NGAC form is given below.

- **Users** are the entities that request access to the resources.
- **Resources** are PLC components that need protection.
- **Environments** define the external conditions to users and resources needed for the access, such as location (IP address) and access time.
- **Operations** are actions a user can perform on the PLC.

Figure 2 illustrates the NGAC model for PLC as a graph [12]. The nodes shown by the user icon correspond to users. Solid boxes correspond to attributes, and ovals correspond to resources. The solid arrow edges represent assignment and the dotted edges denote association relation. The label on the edge denotes the operation name that can be performed by users having the attribute implied by one node of the association edge on resources having the attribute implied by the other node of the association edge. The environmental attributes and the obligation policies are not shown in Figure 2.

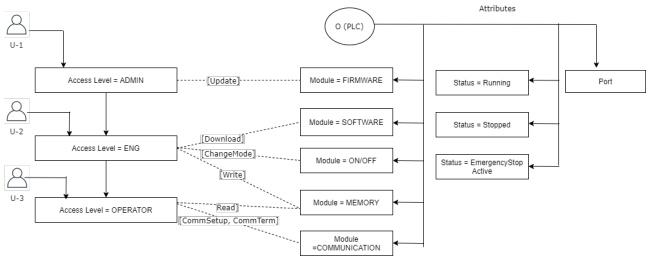

Figure 2: NGAC Model for PLC

The various attributes and operations are listed as follows:

PLC Attributes

(1) *Module* = {*Communication, Software, Memory, Firmware, Input/output*} represents a module of the PLC.
(2) *Status* = {*Stopped, Running, Emergency Stop Active* } represents current operational status of the PLC.
(3) *Port* = is of type string that represents the communication port of the PLC.

User Attributes

(1) *AccessLevel* = {*Operator, Engineer, Administrator*} represents access level of a requesting user.
(2) *DeviceID* is of type string. We use hard disk serial number of the authorized engineering workstation as *DeviceID* because it is relatively difficult to change the hard disk serial number of an attacker's device to match the authorized *DeviceID*.

Environmental Attributes

(1) *Time* is of type string that represents the time of access requested by the user.
(2) *Loc* is of type string that represents the location from where the user is trying to access the PLC.

Operations = { *CommSetup, Download, Update, ReadMem, WriteMem, ChangeMode, ResetMem, CommTerm* } represent operations performed on the PLC.

A policy in the NGAC module is a tuple defined as: $\langle\{userAttr\}, \{resourceAttr\}, \{envAttr\}, \{op\}\rangle$ where *userAttr*, *resourceAttr*, and *envAttr* denote the conditions on User Attributes, Resource Attributes, and Environment Attributes, respectively, and *op* signifies operations. This policy states that *op* is allowed only when the *userAttr*, *resourceAttr*, and *envAttr* are satisfied. If any of the conditions are false, the access is denied.

For instance, a Communication Setup Policy (*CommSetup*) is permitted provided the user has access level *Operator*, *Engineer*, and *Administrator* with device *"4c174602"* and the time of access is in the interval 7:00-16:00 EST. This policy is expressed as:

$$\langle\{User.AccessLevel \in \{Operator, Engineer, Administrator\} \wedge User.Device = \text{“4c174602”}\}, \{True\}, \{Env.Time = 700 - 16:00 EST \wedge Env.Loc = \text{“Org.local”}\}, \{CommSetup\}\rangle$$

The first component in the tuple gives conditions on the user attributes. There are no explicit conditions on the resource attribute, so the second is "true". The third component signifies condition over environmental attributes: location and time. The last component denotes the allowable operation.

3 ATTACK DEMONSTRATION

3.1 Threat Model

We adopted the threat model mentioned by Biham et al. [6]. The threat model assumptions are given below.

(1) The attacker has the knowledge to attack the targeted PLC.
(2) The attacker is an outsider and has no authorized access to the PLC or the engineering workstation.

3.1.1 Phase-1: Man-in-the-Middle attack – Interception of authenticated communication between the PLC and engineering workstation. The attacker performs the following operations.

(1) Obtains the PLC IP address through internal resources or external websites like *Shodan* [9]
(2) Intercepts the communication between the targeted PLC and the authenticated engineering workstation.
(3) Extracts the information on data required for establishing the communication with the PLC. This data depends on the specification of the PLCs. We need the firmware version for the Siemens PLCs and the sender's context for the Rockwell PLCs to generate the communication request packet.
(4) The attacker also extracts the function code and the related data of the operation.

3.1.2 Phase-2: Launching a DoS attack. The attacker establishes the communication with the targeted PLC to cause an availability attack as follows:

(1) Creates a communication request packet using the information obtained in Phase-1.
(2) Sends the communication request TCP packet to the targeted PLC's IP address and the TCP port.
(3) The PLC establishes the communication with the attacker's engineering workstation.
(4) The attacker then sends a crafted TCP packet with the modified function code and data to the TCP port of the PLC to cause a DoS attack.

3.2 Attack Demonstration

To demonstrate the DoS attack, we connected an engineering workstation directly to the Rockwell Compact Logix PLC over the Internet.

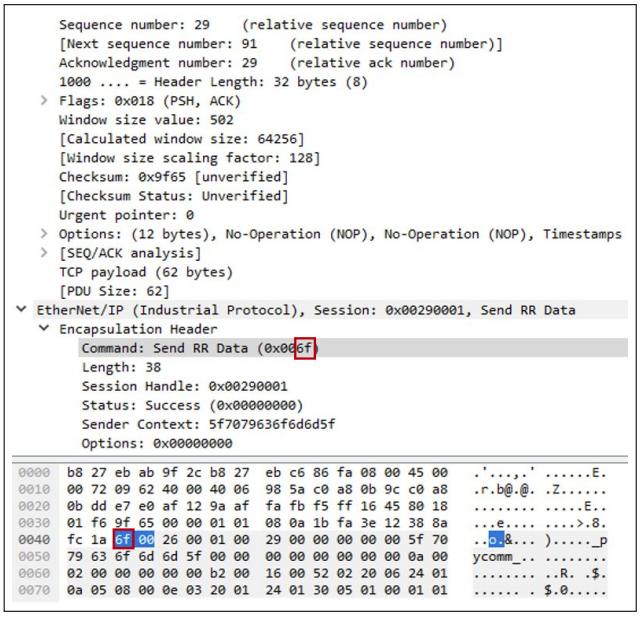

Figure 3: Wireshark Packet Intercept

3.2.1 Phase 1: Man-in-the-Middle attack: Interception of authenticated communication between the PLC and engineering workstation. We examined the communication between the authenticated engineering workstation and the PLC by capturing the network packets with Wireshark (a network packet analyzer [24]). Figure 3 shows an example request packet from the engineering workstation to the PLC intercepted by Wireshark. We extracted the 'Sender Context,' and the function code of the operation is performed from the Wireshark packet. The red boxes indicate the requesting the LED status code (6f).

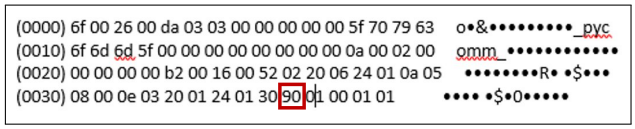

Figure 4: TCP Packets crafted with an invalid function code

3.2.2 Phase 2: Launching of DoS Attack. We launched a DoS attack by performing the following operations.

(1) Using the 'Sender Context' information and our prior knowledge of the working of Rockwell's communication protocol,

we created the communication request packet and established the communication with the PLC.
(2) After establishing the communication between the PLC and the engineering workstation, we sent the crafted TCP packets with an invalid function code, as shown in Figure 4. The red box indicates the change of the last byte from 05 to 90.
(3) Invalid function code caused a major recoverable fault on the PLC, which, in turn, changed the PLC status to "STOP mode".
(4) The PLC stopped the current process and disconnected from the network. Thus, sending a crafted TCP packet caused a DoS attack on the PLC.

4 SECURITY HARDENING

We designed and implemented an independent gateway to be positioned between the user's engineering workstation and the PLC. This gateway comprises an authentication module, a communication handler, a protocol analyzer, and an access control module. We named it the Attributed Based Access Control (ABAC) gateway.

Figure 5 shows the ABAC gateway architecture within dotted lines, where the ABAC components are colored pink. We use a Rockwell Compact Logix PLC for the implementation and testing of the ABAC gateway.

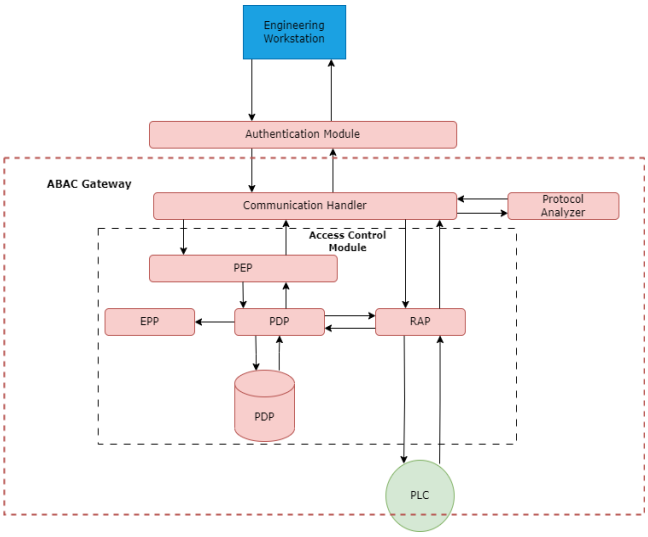

Figure 5: ABAC Gateway Architecture

The authentication and access control modules are the central parts of the ABAC gateway that prevent attackers from accessing the PLC. All authorized users with their user id (*UId*), the password (*Pwd*), the access level (*AccessLevel*), and the engineering workstations with their *DeviceId* must be pre-registered with the ABAC gateway in order to communicate with the PLC. The password is hashed with an MD5 cryptographic function and encrypted using Advanced Encryption Standard (AES). We use a randomly generated key called *Key* to encrypt the password with AES. The *UId*, the *Pwd*, and the *DeviceId* are stored in a database called *LoginDB*.

4.1 Authentication Module

The authentication module consists of a TCP server socket listening to incoming user requests. It is the primary function of the implementation. It is primarily responsible for the authentication process. It also handles the communication between the communication handler and the user. The user sends a PLC request packet.

The authentication module then requests users to provide their *UId* and the *Pwd* through the authentication process described below.

(1) The user sends the *UId* to the authentication module.
(2) The authentication module verifies the *UId* in the *LoginDB* to ensure that user is pre-registered.
(3) If the *UId* is found in the *LoginDB*, the authentication module sends the *Key* to the user to encrypt the password. If not, the authentication disconnects from the user's engineering workstation.
(4) The user responds with the encrypted *Pwd*.
(5) The authentication module decrypts the *Pwd* and validates.
(6) If the authentication is successful, the authentication module extracts the *DeviceID* forwards it along with the *PLCRequestPacket*, and *UId* to the communication handler.
(7) If the user fails to authenticate, the authentication module disconnects from the user's engineering workstation.

4.2 Protocol Analyzer

We implemented a Python parsing function to parse the PLC request packets. The parsing process is specific to the communication protocol used by the target PLC. We have implemented the protocol analyzer specific to Rockwell's Common Industrial Protocol (CIP) communication protocol.

The CIP is an object-oriented protocol. Each CIP object is a particular component that contains attributes (data), services (commands), connections, and behaviors (relationships between attribute values and services) [13].

An encapsulated CIP packet consists of the following: (i) *Command* (2 Bytes) - the encapsulated command code, (ii) *Length* (2 Bytes) - the length of the encapsulated data, (iii) *Session Handle* (4 Bytes) - the session identifier, and (iv) *Status* (4 Bytes) indicates whether the receiver successfully executed the request.

We used CIP and Wireshark captures to implement a CIP protocol analyzer that extracts the requested operation and other details needed for access control verification.

4.3 Communication Handler

We implemented the communication handler as a Python script. It functions as follows.

(1) Invokes the protocol analyzer to extract the *Operation* from the PLC packet.
(2) Extracts the IP address of the incoming request (*Source.IPaddress*), the *PLC.Port*, and the access time (*Env.AccessTime*) from the *PLCRequestPacket*.
(3) Forwards the *Operation*, the *UId*, the *DeviceID*, the *Source.IPaddress*, the *PLC.Port*, and the *Env.AccessTime* to the access control module to compute the decision,
(4) Receives the decision from the access control module. If the decision is to grant (*grant*), then it forwards the PLC request

packet to the PLC through the Resource Access Point (RAP). It also receives the PLC's response through the RAP and forwards it to the authentication module to send to the user.

(5) If the decision is to deny (*deny*), it forwards the decision to the authentication module. The authentication module disconnects from the user's engineering workstation.

4.4 Access Control Module

We used a Python tool kit (i.e., *Vakt* library) to implement the access control module. This library supports NGAC-based ABAC.[16]. The *Vakt* library incorporates:

- *Vakt* Storage (Policy Information Point (PIP)): It stores the policies the user application adds. PIP provides 3 types of storage to store the policies; (i) Memory storage, (ii) SQL storage, and (iii) Mongo DB storage. We used SQL storage to create a policy database.
- *Vakt* Guard (Policy Decision Point (PDP)): PDP extracts policy information from PIP. Additionally, it communicates with the PLC through Resource Access Point (RAP) to extract the *PLC.Status*, and computes the decision.
- Policy Enforcement Point (PEP): The PEP is a user application that communicates with PDP through the "Vakt Inquiry" to check whether an access request is granted.
- Application Administration: A user application communicates with the PIP to add, modify, or update policies.

Figure 6 illustrates the "Communication Setup" policy implemented using *Vakt* library.

```
communication_setup = vakt.Policy(
    1,
    actions=[Eq('CommSetup')],
    subjects=[{'User.AccessLevel' : In('Operator', 'Engineer', 'Administrator'),
               'User.Device' : Eq("Equip21LOrgABC")}],
    context={'Env.AccessTime' : And(GreaterOrEqual(7.00), LessOrEqual(16.00)),
             'Env.AccessLoc' : Eq("OrgABC.local"),
             'PLC.Status' : Eq('Remote')},
    resources=[Eq('PLC')],
    effect=vakt.ALLOW_ACCESS
)
```

Figure 6: Example Policy Implementation

4.5 Resource Access Point

We implemented RAP as a Python script. It consists of a TCP client socket communicating with the PLC over a private network. The RAP

(1) Sends TCP/IP requests to the PLC to extract the information, such as the *PLC.Status* that is required for the access control verification.
(2) Receives the *PLCRequestPacket* from the communication handler and forwards it to the PLC.
(3) Forwards the PLC's response to the communication handler.

It is specific to the communication protocol used by the target PLC. For our experiment, we implemented the RAP based on CIP that is specific to the Rockwell PLCs [13].

4.6 Test Bed

We built a test bed that incorporates (i) an engineering workstation implemented on a Raspberry Pi3 micro-controller, (ii) the ABAC gateway implemented on a Raspberry Pi3, and (iii) Rockwell's Compact Logix as the target PLC. The engineering workstation communicates with the ABAC gateway through the Internet. The ABAC gateway communicates with the PLC through a private one-to-one Ethernet communication.

The Raspberry Pi3 is a single-board computer with Quad Core 1.2GHz Broadcom BCM2837 64bit CPU and 1GB RAM. It is suitable for implementing small scale industrial automation and test bed computers in the laboratory environment. The Raspberry Pi3 uses Unix-like proprietary Operation System (OS) called Raspberry Pi OS [25]. We used a small scale Rockwell's Compact Logix PLC, 1769-L18ER-BB1B CompactLogix 5370 with the firmware version 30. This PLC uses a Real-Time Operating System (RTOS) based on WinCE (Windows-based RTOS) [2].

4.7 Prevention of DoS Attack with ABAC Gateway

We demonstrate how the ABAC gateway prevents the DoS attack with two scenarios. We use the crafted TCP packets shown in Figure 4 to launch a DoS attack on the target PLC through the ABAC gateway. To send the crafted TCP packet, the user must first send a communication request (*CommSetup* operation) to establish the communication with the PLC.

4.7.1 Scenario 1 (Unauthenticated User). We sent the communication request with an unauthenticated user and the password in this use case. The ABAC gateway received the packet and disconnected from the engineering workstation.

4.7.2 Scenario 2 (Unauthorized Device). In this case, we assume an attacker successfully extracts the user ID and the password through brute force or other hacking techniques.

We sent the registration request packet with a valid user ID, password, and device ID. The ABAC validated the access control policies and disconnected the engineering workstation, as the access control policy failed on an unregistered device ID. Figure 7 represents the output from the ABAC gateway preventing the DoS attack from an unregistered device.

```
Connected by ('192.168.102.14', 36294)
Received from client : b'e\x00\x04\x00\x00\x00\x00\x00\x00\x00\x00\x00\x000.8.6   \x
00\x00\x00\x00\x00\x01\x00\x00\x00'
HexData 650004000000000000000000302e382e3620202000000000001000000
Commnad 6500
Command Final CommSetup
Returned Command CommSetup
Verifying Policy
Policy Failed on Device ID and Packets Rejected
```

Figure 7: ABAC: Policy fail on invalid DeviceID

5 FORMAL VERIFICATION

We used CPN to formally verify the ABAC gateway. First, we modeled CPN representing the PLC running without ABAC gateway and demonstrated the DoS attack. We then created CPN demonstrating PLC operating with ABAC gateway that prevented the DoS

attack. In both use cases, we demonstrate an attacker impersonating an engineering workstation and sending a crafted TCP packet to execute a DoS attack on the PLC.

The executable model comprises several CPNs representing the user, network, ABAC gateway, and PLC process. These nets are connected to form a CPN hierarchy. We express the CPN hierarchy using CPN block diagrams. These CPN block diagrams are not executable but describe the flow control of messages exchanged between the processes. The executable CPN is excluded due to lack of space. We use *CPNTools* [26] to generate a state-space model from the executable CPN. This is represented as a directed graph with states and transitions.

The user process represents either a legal user or an attacker. For the legal user, a TCP packet with a communication setup command (*COMM.SETUP*) is sent to PLC. In the attacker's case, a crafted TCP packet with a communication stop (*COMM.STOP*) command is sent to PLC to tackle a DoS attack and make PLC inoperable.

5.1 Use Case 1: PLC Running without ABAC Gateway

Figure 8 illustrates the CPN block diagram demonstrating the PLC running without the ABAC gateway. It includes three CPNs representing the user, network, and PLC. The solid arrows indicate messages sent from the user to PLC, while the dotted arrows express messages returned from PLC to the user. The user sends a request packet to initialize communication with PLC. The PLC receives the request and replies back with a challenge question. The user sends the challenge response, and the PLC confirms that the session is OK to be established. The user then sends a command to PLC to be executed and receives a response with PLC status.

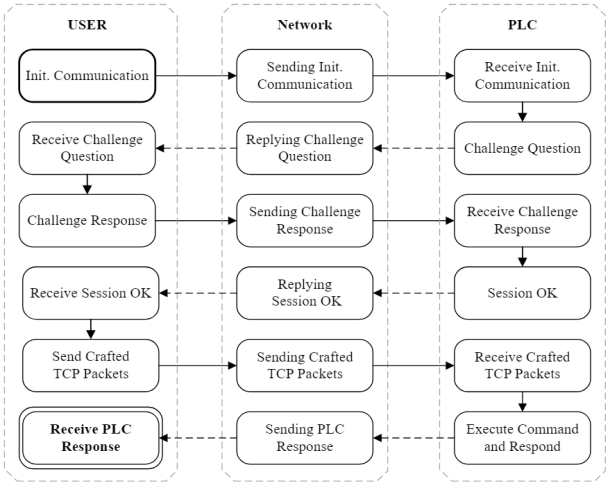

Figure 8: PLC Running without ABAC Gateway

Table 1 shows the testcases that express the description of the TCP packets, input values of TCP packets (i.e., tokens), expected output, and actual output reporting the PLC status. These testcases are designed as authorized users, unauthorized users, and attackers. We assume that these users successfully responded to the challenge question, and the attackers exploited the authentication and also successfully responded to the challenge question. These users and attackers have various IP addresses denoting different computers. The first two testcases demonstrate a legal user, an administrator (ADMIN) listed in the DAC access control list (ACL), having discretion of executing the *COMM.SETUP* and *COMM.STOP* command on the PLC. The third testcase demonstrates a legal user, a registered user (USER) listed in the DAC access control list (ACL) but does not have the privilege to execute *COMM.STOP* command. The fourth testcase demonstrates an attacker (ATTACKER-1) who impersonates USER to execute *COMM.STOP* command. The fifth testcase demonstrates an attacker (ATTACKER-2) who impersonates ADMIN to execute *COMM.STOP* command. We expected that ATTACKER-2 would be able to perform a DoS attack.

We applied these testcases to the executable CPN of Use Case 1 and the state-space generated. The testcases TC#1 and TC#2 passed successfully, showing that the ADMIN user can perform *COMM.SETUP* and *COMM.STOP* operation on the PLC. The testcases TC#3 and TC#4 failed, and the PLC rejected the TCP packet. The reason is that TC#3 representing USER that does not have the privilege to execute *COMM.STOP*. Hence, DAC blocked the TCP request packet. TC#4 represents ATTACKER-1, who impersonated USER and obtained the same access discretion. Thus, DAC also blocked its TCP request packet. TC#5 had successfully passed. The ATTACKER-2 impersonated the ADMIN user, gained its discretion, and stopped the PLC.

As shown in Table 2, the state space is fully connected, and all states are reachable. It also shows that the state space is acyclic since the number of states and transitions of the state space and SCC graph are the same. A home state (i.e., home marking) is a state that is reachable from all other states. A dead state (i.e., dead marking) is a state that has no enabled transitions. Dead transition never occurs; it is disabled in all reachable states.

We use *CPNtools* built-in functions, `ListDeadMarking(fu())` and `PredAllNodes(fu())`, to investigate these states and transitions. Table 2 shows that the state `1653` is a home state. Verifying this state indicates the CPN state when the PLC process receives the TCP packet. It means that the five testcases (tokens) were processed and reached the PLC, but not all need to be continuous. The dead states `1378` and `1745` indicate the CPN terminated when the TCP packet was rejected by the PLC process, for the USER sent TCP packet to stop the PLC, and ATTACKER-1 impersonated USER and sent the same packet to stop the PLC. The two dead transitions express that the network process did not send the PLC status to the user process when a TCP packet to stop PLC was rejected. An instance of the dead transition is `NetworkTCPPackets'Send_PLC_Status_to_User 2`. The analysis of Use Case 1 summarizes that DAC is not robust, and the attacker can typically impersonate a legal user and execute attacks on PLC.

5.2 Use Case 2: PLC Operates with ABAC Gateway

We added executable CPN nets representing the ABAC gateway processes, including the authentication, communication handler, and NGAC process. These CPN nets are positioned between the network and the PLC process.

TC#	Description	Input (Token)	Expected Output	Actual Output	Testcase Status
1	ADMIN: A legal user listed in DAC-ACL and has privilege to setup PLC communication	{IP = {srcIPAddr = "10.255.10.7", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", CIP = {Command = Comm.Setup, SessionHandle = "established"}}}	PLC Status Running	PLC Status Running	Passed
2	ADMIN: A legal user listed in DAC-ACL privilege to stop PLC communication	{IP = {srcIPAddr = "10.255.10.7", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	PLC Status Stopped	Passed
3	USER: A legal user listed in DAC-ACL but does not have privilege to stop PLC communication	{IP = {srcIPAddr = "10.255.10.23", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	TCP Packet Rejected	Failed
4	ATTACKER-1: An attacker impersonates USER to stop PLC communication	{IP = {srcIPAddr = "13.255.255.1", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	TCP Packet Rejected	Failed
5	ATTACKER-2: An attacker impersonates ADMIN to setup PLC communication	{IP = {srcIPAddr = "13.255.255.3", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	PLC Status Stopped	Passed

Table 1: Testcases demonstrate legal access and attacks for Use Case 1

State Space		SCC Graph		Status
#State	#Transition	#State	#Transition	Full
1820	5733	1820	5733	
Home State		Dead State		#Dead Transitions
[1653]		[1378,1745]		2

Table 2: State-Space Analysis for Use Case 1

Figure 9 illustrates the CPN block diagram demonstrating the PLC utilizing the ABAC gateway.

In addition to the terminal where a user receives a PLC response, two places (unauthenticated and decision deny), colored red in Figure 9, are also terminals. The unauthenticated place corresponds to the case when a user fails to authenticate, and decision deny place corresponds to where policy decision point (PDP) denies the access request.

We use the same testcases presented in Table 1. In addition, we added extra TCP attributes to the tokens, including userID, encrypted password, deviceID, and access time. These attributes are required for the authentication process and NGAC policy validation.

Table 3 shows 6 testcases that verify the authentication and authorization processes of the ABAC gateway. We added a new testcase (TC#4) expressing an attacker (ATTACKER-1) impersonating a legal user (USER) but not having the same deviceID. USER's deviceID is `"SR567"`, whereas ATTACKER-1's deviceID is `"SR999"`. In the testcase TC#5, the same attacker (ATTACKER-1) crafted an TCP packet changing the deviceID to be as exact as USER, `"SR567"`. The testcase TC#6 presents the ATTACKER-2 impersonating the ADMIN user with the decrypted password and crafted deviceID as `EncPWD = "PW123"` and `deviceID = "SR123"`. Again, we expected that ATTACKER-2 would be able to perform a DoS attack.

Repeating the analysis process, we applied these testcases to the executable CPN of Use Case 2 and generated the state space. The testcases TC#1 and TC#2 passed successfully, showing that the ADMIN user was authenticated and authorized to perform *COMM.SETUP* and *COMM.STOP*. The testcase TC#3 failed. The reason is the USER attributes (UserID = `"ADMIN-01"`, DeviceID = `"SR123"`), resource attributes (PLC Module = `COMM`, Port = `44818`), environment attributes (IP = `"10.255.10.23"`, AccessTime = `"13:25EST"`), and Operation (`STOP`), does not fulfill the PLC-STOP policy. Hence, the NGAC-PDP decision was denied access, and the NGAC-PEP disconnected from the user.

The testcase TC#4 represents the scenario where ATTACKER-1 impersonated USER and obtained its credentials. This testcase failed with an unauthenticated output because the ATTACKER-1 has a different deviceID. USER has deviceID = `"SR567"` and ATTACKER-1 in TC#4 has deviceID = `"SR999"`. In testcase TC#5, although ATTACKER-1 crafted a TCP packet changing the deviceID to be as exact as USER, it failed with an unauthorized output because the ATTACKER-1 has a different IP address, where the NGAC policy fails. TC#6 also failed with the authorization process. The ATTACKER-2 impersonated the ADMIN user, gained credentials and deviceID, and passed the authentication process. However, the NGAC-PDP denied access, and the NGAC-PEP disconnected from the user because the ATTACKER-2 attributes do not fulfill the PLC-STOP policy. The ATTACKER-2 has a different IP address than the ADMIN uses.

Table 4 reports that the state space is acyclic and fully connected. It shows that the state `683` is a home state, representing the CPN state where the authentication process receives the user credentials and deviceID. The dead states `722` and `723` indicate that for testcase TC#4, the access request packet ends at the unauthenticated state, and the CPN is terminated. The dead states `3156,3157,3158,3159,3160,3161,3162,3163,3164,3165,3166`, and `3167` indicate that, for testcases TC#3, TC#5 and TC#6, the access request packet passed the authentication process but terminated at the NGAC-PDP where the access request is denied and disconnected. The 27 dead transitions indicate the region where the communication handler, NGAC policy points, and PLC were not enabled. It is because the authentication process terminates the CPN execution for the testcase TC#4, which represents unauthenticated users. The analysis of Use Case 2 shows that although the credentials had been decrypted, the attacker had been blocked from accessing the PLC in two places, the authentication and NGAC process. The Use Cases 1 and 2 analysis implies that incorporating authentication and ABAC-NGAC is typically more robust than the challenge question authentication and DAC for PLC protection.

TC#	Description	Input (Token)	Expected Output	Actual Output	Testcase Status
1	ADMIN: a legal user assigned in NGAC Comm.Setup policy	{IP = {srcIPAddr = "10.255.10.7", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", UserID = "ADMIN-01", EncPWD = "PWD123", DeviceID = "SR123", AccessTime = "13:10EST", CIP = {Command = Comm.Setup, SessionHandle = "established"}}}	PLC Status Running	PLC Status Running	Passed
2	ADMIN: a legal user assigned in NGAC Comm.Stop policy	{IP = {srcIPAddr = "10.255.10.7", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", UserID = "ADMIN-01", EncPWD = "PWD123", DeviceID = "SR123", AccessTime = "13:15EST", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	PLC Status Stopped	Passed
3	USER: a legal user has not assigned in NGAC Comm.Setup policy	{IP = {srcIPAddr = "10.255.10.23", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", UserID = "USER-01", EncPWD = "PWD567", DeviceID = "SR567", AccessTime = "13:25EST", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	Access Denied & User Disconnected	Failed
4	ATTACKER-1: an attacker impersonates USER to stop PLC communication	{IP = {srcIPAddr = "13.255.255.1", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", UserID = "USER-01", EncPWD = "PWD567", DeviceID = "SR999", AccessTime = "13:35EST", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	Authentication fails & User Disconnected	Failed
5	ATTACKER-1: an attacker impersonates USER to stop PLC communication	{IP = {srcIPAddr = "13.255.255.1", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", UserID = "USER-01", EncPWD = "PWD567", DeviceID = "SR567", AccessTime = "13:40EST", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	Access Denied & User Disconnected	Failed
6	ATTACKER-2: an attacker impersonates ADMIN to setup PLC communication	{IP = {srcIPAddr = "13.255.255.3", dstIPAddr = "129.10.1.3"}, TCP = {srcPort = "5357", dstPort = "44818", UserID = "ADMIN-01", EncPWD = "PWD123", DeviceID = "SR123", AccessTime = "13:45EST", CIP = {Command = Comm.Stop, SessionHandle = "established"}}}	PLC Status Stopped	Access denied & User Disconnected	Failed

Table 3: Testcases demonstrate legal access and attacks for Use Case 2

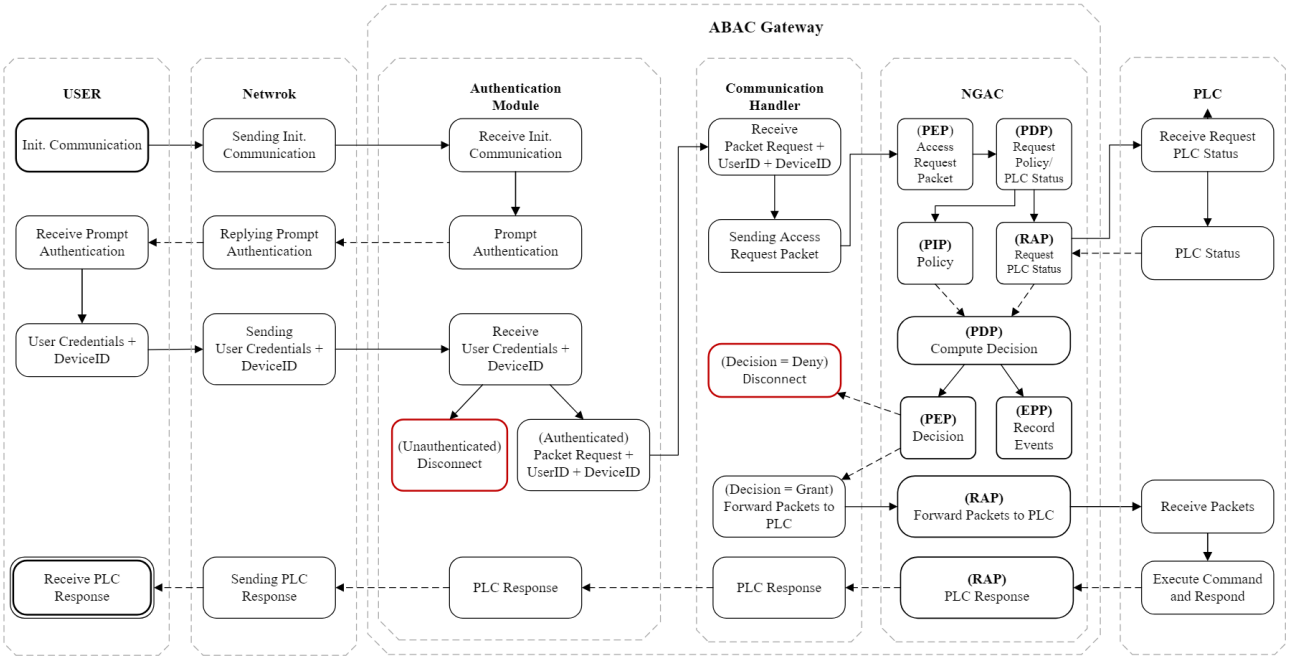

Figure 9: PLC Operates with ABAC Gateway

5.3 Security Properties Verification

We consider certain states to verify security properties (authentication and authorization property). We verify states (i.e., desired states) where a user is checked for registration and an access request is validated. We use reachability analysis to verify that directed paths start from the initial states to desired states.

CPNTools provides automatic generation tools (predecessors and successors) that generate directed paths from an initial state to a particular state and from this state to terminal states. CPNTools also has a built-in reachability function, such as `Reachable(`s_i`,`s_j`)`, used to verify these paths. We use these tools and functions for security property verification.

State Space		SCC Graph		Status
#State	#Transition	#State	#Transition	Full
4876	18057	4876	18057	
Home State [683]		Dead State [722,723,3156,3157,3158,3159, 3160,3161,3162,3163,3164,3165, 3166,3167]		#Dead Transitions 27

Table 4: State-Space Analysis for Use Case 2

The authentication property is defined as verifying the identity of a user, process, or device as a prerequisite to allowing access to a system's resources [19]. In the case of PLC operating without the ABAC gateway, a user identity is represented as userID and challenge response. The combination of userID, encrypted password, and deviceID is a user identification to access the PLC utilizing the ABAC gateway. The authorization property verifies that a requested action or service is granted for a specific resource [19]. The requested action (operation to be performed on the PLC) is verified by DAC when the PLC is running without the ABAC gateway, whereas it is verified by the NGAC when it operates with the ABAC gateway.

We generate paths reaching the home state reported in Use Case 1, Table 2 (`1653`), and from home state reaching the dead states (`1378` and `1745`). The generated paths are described as a sequence of state numbers. For instance, a path from the initial state (state#1) to the desired state (state#1378) is generated as `[1,2,4,8,15,27,45,71,95,129,170,222,282,352,378, 460,551,652,760,876,996,1122,1248,1378]`. For all cases, displaying the description of these sequence state numbers and verifying them shows that the user (either legal user or attacker) has bypassed the authentication process. We investigated the tokens' values of the terminal states of these paths. It turns out that the tokens of the testcase TC#1, TC#2, and TC#5, which represent the ADMIN user and ATTACKER-2, have reached the PLC executes command and response. It indicates that not only has an attacker bypassed the authentication process but also the authorization process and the PLC execute command place is reachable.

The reachability of home state (`683`) and dead states (`722` and `723`) reported in Use Case 2, Table 4, generates paths that show a user (either legal user or attacker) bypasses the authentication process only if the deviceID is valid; otherwise the user fails to bypass the authentication process. However, the paths generated for the dead states (`3156`,`3157`,`3158`,`3159`,`3160`,`3161`,`3162`,`3163`, `3164`,`3165`,`3166` and `3167`) reach the authorization process. In particular, they reach the states described as `[PEP,PDP, RAP, PLCStatus,RAP,PDP,PIP,PDP,PEP,Disconnect]`. We investigated the tokens' values of the terminal states of these paths. The tokens of all testcases TC#3, TC#4, TC#5, and TC#6, which represent unauthorized legal users and attackers, were terminated in the authorization process. Only tokens of the testcases TC#1 and TC#2 were terminated in the user process receiving the PLC response. It indicates that the unauthorized user and attacker failed to reach the PLC to execute the command and respond place.

This reachability verification summarizes that for the DAC access control mechanism, an attacker impersonating a legal user can only execute privileged operations to PLC that are authorized by the legal user. However, there is no guarantee that the attacker cannot impersonate an administrator user with the full privilege to perform a DoS attack. It is because once the user, who might be an attacker, bypasses the authentication process, the DAC does not verify the characteristics of this user to grant access to the PLC; instead, it only restricts the privilege based on the user's permission. On the other hand, the ABAC gateway verifies the access based on NGAC policy comprising user attributes, environment, resource, and operation attributes to grant and deny access to the PLC.

6 RELATED WORK

Duka *et al.* [8] propose authenticated data exchanges through Message Authentication Codes (MAC) between the ICS components, such as PLC and HMI, by constructing the MAC in the software. The control software within the PLC and HMI are constructed with MAC to verify the data exchanged between the user and the PLC for authentication and integrity. This does not address how a rogue PLC can prevent the man-in-the-middle attack. The approach requires a design change in the PLC so that it can construct and verify the MAC. Implementation becomes vendor-specific and does not provide a centralized solution for complex ICS systems. Additionally, this work does not address password-based attacks.

Gowdanakatte *et al.* [12] introduce an ABAC architecture to protect ICS against vulnerabilities that arise from improper authentication and access control. The authors use NIST NGAC to model the controls of ICS users accessing the PLC components. They formalized the NGAC security model, including PLC Attributes, User Attributes, Environmental Attributes, and Operations that users are privileged to execute on the PLC.

The NGAC formalization covers three policies: Communication Setup Policy, Memory Write Policy, and Firmware Update Policy. The architecture is proposed as an independent gateway module between an Enterprise network and PLC on an embedded controller. The authors also provided security analysis with sequential diagrams. This work is promising for developing an isolated access control module implanted into any ICS to strengthen the system authorization and elevate its resiliency. However, the ABAC gateway implemented requires formal verification and experimental evidence to evaluate its reliability.

RBAC mechanism is becoming the de-facto access control mechanism for ICS [11], [3], and [17]. RBAC provides more coarse-grained access control compared to ABAC. RBAC is used to restrict ICS user privileges based on their roles. Major PLC and HMI manufacturers are incorporating RBAC into their PLC and HMI security features. Rockwell's factory talk security software provides RBAC for accessing PLC and HMI [3]. Honeywell ACS labs [17] proposes a two-layered RBAC for ICS, which provides better security than a single-layered RBAC.

Esiner *et al.* [10] design and implement a message authentication scheme using lightweight hash-chaining-based cryptography. The authors focus on the performance and analyze the latency of security checking as the message authentication scheme is fast and flexible, targeting smart grid systems specifically. To ensure compatibility, the solution was implemented as a bump-in-the-wire (BITW) to apply to various deployment settings and communication models.

The analysis demonstrates that the message authentication scheme achieved an end-to-end communication latency below 2 milliseconds in an intra-substation-alike setting. Although our focus is authorization, this work would be practical as an alternative to the existing authentication mechanism of ICS to lower the security checking latency.

The literature on PLC security considers using CPN for PLC communication protocols and PLC program validation. Nonetheless, the research of the PLC security field shall consider formal verification for access control used to protect PLC resources, which is our consideration in this paper.

Bhurke *et al.* [5] review formal analysis methods used for security analysis of the communication protocols used in the PLC domain. They begin by describing various attacks of PLC, including DoS, an-in-the-Middle (MITM) attacks, and Replay Attacks. Then, they explore formal methods used for the analysis of PLC communication protocols such as Symbolic Model Verifier (SMV), Isabelle/HOL (Higher Order Logic), Promela model checker (SPIN), Vienna Development Method (VDM), and Coloured Petri Nets (CPN).

The authors focus on CPN to analyze a client-server communication model utilized upon Highway Addressable Remote Transducer over IP (HART-IP) protocol. They use CPN to model the HART Protocol to assess the protocol's overall behavior in the absence of threats.

Schné *et al.* [27] use CPN for PLC program validation. The PLC operations are generally simulated by CPN models, and the result values are compared to the desired ones. The authors formally define the PLC input and output values that express the color sets of the CPN model. They illustrate the approach by using an oil tank warning system as an example. The system initialization and operation states are defined, and validation states are described. The validation points to an undesired state (acknowledge instead of a warning), highlighting a safety risk. The authors will address program error localization in future work.

7 CONCLUSION

Authentication breaches in ICS can have devastating consequences. ICS devices have a long life, and it may be impossible to patch all the vulnerabilities. Towards this end, we showed how a NIST NGAC attribute-based access control model can protect against authentication vulnerabilities in PLC by verifying some additional user and device attributes.

In our prototype, we observed no significant latency in the PLC performance due to the ABAC controls. We also proved the security hardening of the PLC formally using CPN.

We are currently investigating the use of NIST NGAC for the security hardening of other devices in an ICS environment. We expect implementation challenges of complex ICS systems with security controls. In the future, we plan to analyze the latency, performance, and throughput of the ICS due to the incorporation of the ABAC module and see how this can be improved.

ACKNOWLEDGMENTS

This work was supported in part by funding from NIST under Award Number 60NANB23D152, NSF under Award Number CNS 1822118, ARL, Statnett, AMI, NewPush, and Cyber Risk Research.

REFERENCES

[1] Tejasvi Alladi, Vinay Chamola, and Sherali Zeadally. 2020. Industrial Control Systems: Cyberattack trends and countermeasures. *Computer Communications* 155 (2020), 1–8. https://doi.org/10.1016/j.comcom.2020.03.007
[2] Allen-Bradley. 2013. 1769 CompactLogix Controllers User Manual. https://literature.rockwellautomation.com/. Accessed: 2023.
[3] Rockwell Automation. 2021. FactoryTalk Security System Configuration Guide. Accessed: 2023.
[4] Adeen Ayub, Hyunguk Yoo, and Irfan Ahmed. 2021. Empirical study of PLC authentication protocols in industrial control systems. In *Proceedings of the IEEE Security and Privacy Workshops (SP)*. IEEE, San Francisco, CA, USA, 383–397.
[5] Anish Uday Bhurke and Faruk Kazi. 2021. Methods of Formal Analysis for ICS Protocols and HART-IP CPN modelling. In *Proceedings of the Asian Conference on Innovation in Technology (ASIANCON)*. IEEE, PUNE, India, 1–7.
[6] Eli Biham, Sara Bitan, Aviad Carmel, Alon Dankner, Uriel Malin, and Avishai Wool. 2019. Rogue7: Rogue engineering-station attacks on S7 simatic PLCs. *Black Hat USA* 2019 (2019).
[7] Thomas M Chen and Saeed Abu-Nimeh. 2011. Lessons from stuxnet. *Computer* 44, 4 (2011), 91–93.
[8] Adrian-Vasile Duka, Béla Genge, and Piroska Haller. 2018. Enabling authenticated data exchanges in industrial control systems. In *Proceedings of the 6th International Symposium on Digital Forensic and Security (ISDFS)*. IEEE, Antalya, Turkey, 1–5.
[9] Shodan Search Engine. 2023. https://www.shodan.io/. Accessed: 2023.
[10] Ertem Esiner, Daisuke Mashima, Binbin Chen, Zbigniew Kalbarczyk, and David Nicol. 2019. F-pro: a fast and flexible provenance-aware message authentication scheme for smart grid. In *International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)*. IEEE, Beijing, China, 1–7.
[11] Santiago Figueroa-Lorenzo, Javier Añorga, and Saioa Arrizabalaga. 2019. A role-based access control model in modbus SCADA systems. A centralized model approach. *Sensors* 19, 20 (2019), 4455.
[12] Shwetha Gowdanakatte, Indrakshi Ray, and Siv Hilde Houmb. 2022. Attribute Based Access Control Model for Protecting Programmable Logic Controllers. In *Proceedings of the ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (CODASPY)*. ACM, Baltimore, MD, USA, 47–56.
[13] ACROMAG Incorprated. 2019. Introduction To Ethernet/IP. https://www.acromag.com". Accessed: 2023.
[14] American National Standards Institute. 2020. *Information Technology - Next Generation Access Control (NGAC)*. Information Technology. NIST, New York, NY. Accessed: 2023.
[15] Kurt Jensen and Lars M. Kristensen. 2009. *CPN ML Programming*. Springer Berlin Heidelberg, Berlin, Heidelberg, 43–77.
[16] Kolotaev. 2021. Attribute Based Access Control. https://github.com/kolotaev/. Accessed: 2023.
[17] Honeywell ACS Labs. 2014. RBAC Driven Least Privilege Architecture For Control Systems. Accessed: 2023.
[18] Robert M Lee, Michael J Assante, and Tim Conway. 2014. German steel mill cyber attack. *Industrial Control Systems* 30, 62 (2014), 1–15.
[19] Michael Nieles, Kelley Dempsey, Victoria Yan Pillitteri, et al. 2017. An introduction to information security. *NIST special publication* 800, 12 (2017), 101.
[20] National Institute of Standards and Technology (NIST). 2019. https://nvd.nist.gov/vuln/detail/CVE-2019-10943/. Accessed: 2023.
[21] National Institute of Standards and Technology (NIST). 2019. https://nvd.nist.gov/vuln/detail/CVE-2019-10952. Accessed: 2023.
[22] National Institute of Standards and Technology (NIST). 2021. https://nvd.nist.gov/vuln/detail/CVE-2020-15782/. Accessed: 2023.
[23] National Institute of Standards and Technology (NIST). 2021. https:https://nvd.nist.gov/vuln/detail/CVE-2021-22681. Accessed: 2023.
[24] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. 2006. *Wireshark & Ethereal network protocol analyzer toolkit*. Elsevier, Rockland, MA, USA.
[25] Raspberry Pi. 2023. Raspberry Pi. https://www.raspberrypi.com/documentation/. Accessed: 2023.
[26] Anne Vinter Ratzer, Lisa Wells, Henry Michael Lassen, Mads Laursen, Jacob Frank Qvortrup, Martin Stig Stissing, Michael Westergaard, Søren Christensen, and Kurt Jensen. 2003. CPN tools for editing, simulating, and analysing coloured Petri Nets. In *Proceedings of the International conference on application and theory of Petri Nets (ICATPN)*. Springer, Eindhoven, Netherlands, 450–462.
[27] Tamás Schné and Tibor Holczinger. 2013. Coloured Petri Net based PLC program validation with a fast simulation method. In *Proceedings of the International Conference on Process Control (PC)*. IEEE, Strbske Pleso, Slovakia, 179–184.