Attribute Based Access Control Model for Protecting Programmable Logic Controllers

Shwetha Gowdanakatte
Shwetha.Gowdanakatte@colostate.edu
Colorado State University
Fort Collins, Colorado, USA

Indrakshi Ray
Indrakshi.Ray@colostate.edu
Colorado State University
Fort Collins, Colorado, USA

Siv Hilde Houmb
siv.houmb@ntnu.no
Norwegian University of Science and Technology
Oslo, Norway

ABSTRACT

Industrial Control Systems (ICS) were traditionally designed as stand-alone systems and isolated from Internet Technology (IT) networks. With the advancement in communication technology, the attack surface has increased; vulnerabilities in ICS components such as Programmable Logic Controllers (PLC), and Human Machine Interfaces (HMI) can now be accessed and exploited. Authentication and access control form the first level of defense for protecting ICS from attacks. Unfortunately, vulnerabilities stemming from improper authentication and access control are very common. We focus our attention to investigate these vulnerabilities, specifically those centered around PLCs, and demonstrate how the use of Attribute-Based Access Control (ABAC) helps protect against them and make ICS more resilient to attacks. We design an ABAC model for PLC, show how it can be enforced, analyze the resulting system and demonstrate their resilience against some sample vulnerabilities.

CCS CONCEPTS

• **Security and privacy → Multi-factor authentication**; **Access control**; **Authorization**; **Denial-of-service attacks**; **Multi-factor authentication**; **Access control**.

KEYWORDS

Cybersecurity , Role Based Access Control (RBAC), Attribute-Based Access Control (ABAC), Industrial Control Systems (ICS), Supervisory Control And Data Acquisition (SCADA), Human Machine Interface (HMI), Programmable Logic Controller (PLC).

ACM Reference Format:
Shwetha Gowdanakatte, Indrakshi Ray, and Siv Hilde Houmb. 2022. Attribute Based Access Control Model for Protecting Programmable Logic Controllers. In *Proceedings of the 2022 ACM Workshop on Secure and Trustworthy Cyber-Physical Systems (SaT-CPS '22), April 27, 2022, Baltimore, MD, USA.* , 10 pages. https://doi.org/10.1145/3510547.3517926

SaT-CPS '22, April 27, 2022, Baltimore, MD, USA

ACM ISBN 978-1-4503-9229-7/22/04…$15.00
https://doi.org/10.1145/3510547.3517926

1 INTRODUCTION

Industrial Control Systems (ICS) consist of control systems and associated instrumentation needed to operate or automate an industrial process. ICS typically include Human Machine Interfaces (HMI), Programmable Logic Controllers (PLC), field devices (sensors, actuators), network systems, and communication protocols. ICS often form nations' critical infrastructure and therefore must be protected from security breaches.

Initial ICS were isolated from Information Technology (IT) networks and therefore less vulnerable to security threats. The modern-day ICS can connect to other IT systems for remote monitoring, control, and data collection due to technological advancements in the PLC and the HMI design. The current day PLC and HMI can support communication protocols over Ethernet. They also facilitate remote web servers for remote communications. While the design of PLC and HMI has evolved with respect to functionality and communication, security issues have been largely ignored leading to attacks such as the Stuxnet attack on the Iranian nuclear centrifuge [6]. The availability of tools, such as search engines like "Shodan"[14] help locate ICS systems connected to the Internet, serve to aid attackers. Cyberattacks on an ICS can lead to severe consequences. Examples include blocked or delayed information, unauthorized software modifications, utility disruption, facility terrorism, and interference with the safe operation of systems.

Many of these problems can be traced to flaws in the authentication and access control mechanisms implemented in the ICS components. The vendors often forget to change default passwords which can be compromised through guessing or brute force attack. Additionally a recent study shows there are 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 [1].

Access control is typically achieved through Role-Based Access Control (RBAC) [13] mechanism, which provides authorized access to authenticated users based on their role. Although RBAC offers some protection, the set of roles varies across organizations and is unsuitable for protecting resources managed by different stakeholders. Moreover, RBAC employs static policies where access is contingent only on the role. In an event driven complex system such as ICS, many operations depend on the status of the process being performed. Safety critical operations must be performed only under certain status and at certain environmental conditions. Therefore, only role based access is not sufficient. More fine-grained policies based on user, resource, and environment attributes can offer improved security.

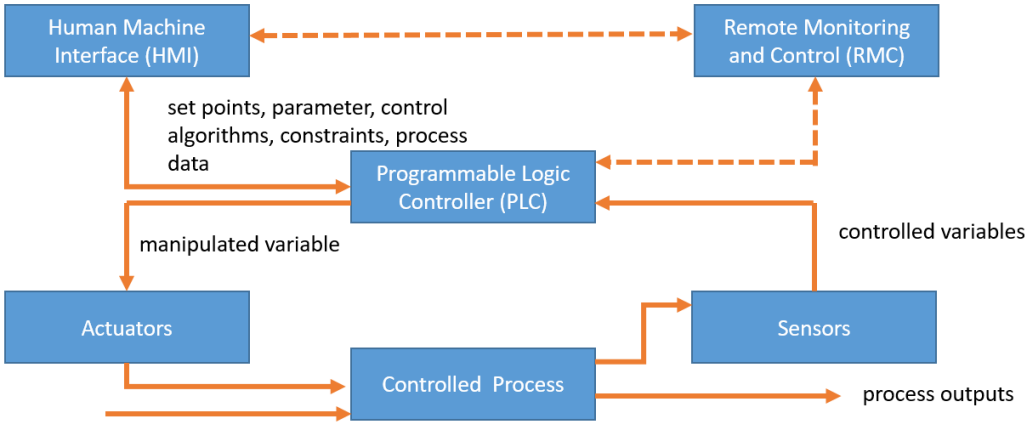

Figure 1: Generic Architecture of ICS

We begin by discussing PLC security features and vulnerabilities related to authentication and access control. We then demonstrate how we can improve the security posture of PLCs by strengthening the authentication mechanism and advocate the use of ABAC model for access control. ABAC is more flexible, more dynamic, and offers more fine-grained access control compared to RBAC [3]. In ABAC, the user attributes, resource attributes, and environmental attributes are combined to define complex policies, making ABAC a more efficient access control method. The nature of these attributes may be static or dynamic. There are two standardized efforts with respect to ABAC models. One is the eXtensible Access Control Markup Language (XACML) model and the other is NIST Next Generation Access Control (NGAC) [7]. We chose NGAC because it allows for ease of policy management and for its ability to model dynamic policies – policies that can be altered while they are deployed – through the concept of obligations. We formulate an ABAC model for use with PLCs and also provide an enforcement architecture. We then provide a security analysis of our approach and demonstrate how it protects against some recently discovered vulnerabilities.

2 BACKGROUND

2.1 Industrial Control System (ICS)

The generic architecture of an ICS is shown in Fig. 1. It incorporates Programmable Logic Controllers (PLC), Human Machine Interfaces (HMI), field devices such as actuators and sensors, and optional Remote Monitoring and Control (RMC). A PLC is an industrialized computer dedicated to monitor and control a process. It generally consists of digital and analog modules, communication module, software module, and firmware module. An HMI 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.

2.2 Authentication and Access Control in ICS

Authentication Generally, most PLCs provide two step authentication for accessing, software, firmware, and communication modules. In the first step, the user is authenticated to get online access to the PLC. In the second step, the connected user will be authenticated once more for performing specific operations.

Access Control In general, PLCs provide Discretionary Access Control (DAC) for its CPU access. HMIs and remote monitoring components provide password based authentication and Role Based Access Control (RBAC) for accessing different features such as configuration settings and remote operations.

2.3 PLC Security Features and Vulnerabilities

We now discuss security aspects for two example PLCs: Siemens S7-1500 PLC and Rockwell Compact Logix PLC.

2.3.1 S7-1500: Security Features and Vulnerabilities. S7-1500 PLC is a new generation of PLC manufactured by Siemens. It has two step password based authentication and DAC for authorization. The engineering framework is referred to as the Totally Integration Automation (TIA) system.

Message Protection S7-1500 PLC uses S7-Protocol version P3, which is designed over TCP/IP with the client-server model [5]. Some cryptographic protection in the P3 protocol includes a key exchange protocol that the PLC and the TIA portal use to establish a secret shared key, which is called the *session key*, The key exchange protocol accomplishes session key establishment with a four-step process described as follows. (i) The TIA portal first sends a request message *Hello* to the PLC to initiate a new session. (ii) The PLC responds with a message that contains the PLC firmware version and a challenge text which we refer to as *ServerSessionChallenge*. (iii) The TIA portal receives the PLC response. It determines the

PLC public key from the firmware version and creates the *SessionKey* using a randomly generated key called *KeyDerivationKey* and the *ServerSessionChallenge*. It sends the *SessionKey* to the PLC, encrypting it with the PLC public key. (iv) The PLC decrypts the *SessionKey* with its private key corresponding to the firmware version and sends *OK* response to the TIA Portal, thus establishing the communication with the TIA portal. P3 protocol also provides a message integrity protection algorithm that calculates a Message Authentication Code (MAC) value based on the session key and the message type.

Access Control and Authentication Vulnerabilities The P3 protocol key exchange uses one-way group authentication. All S7-1500 PLCs with the same firmware have the same public key. Therefore the PLCs sharing the same firmware version can impersonate each other. If the private key of any of these PLCs' is extracted by analyzing the firmware, then the public-private key exchange is compromised. Additionally, The TIA portal verifies the integrity of a message from the PLC, but the PLC does not verify the integrity of a message from the TIA portal. Therefore the PLC does not ensure that the currently communicating TIA portal is the same that successfully communicated an earlier session. This leads to the following vulnerabilities in the P3 protocol.

- CVE-2019-10943:*Authentication bypass vulnerability*: An attacker can exploit this vulnerability to send crafted TCP packets directly to TCP port 102 to modify the running code. The binary code is modified without changing the source code, causing an integrity attack on the targeted PLC. No authentication is required to exploit this vulnerability [9].
- CVE-2020-15782:*High-severity memory protection bypass vulnerability*: An attacker can exploit this vulnerability to write directly to a memory location on the targeted PLC, causing the violation of memory protection [11]. The attacker with network access to TCP port 102 could write code and arbitrary data to the protected memory areas, or read sensitive data to launch further attacks.

2.3.2 Compact Logix PLC: Security Features and Vulnerabilities. Compact Logix PLC is manufactured by Allen-Bradley/Rockwell. It has two step password based authentication for accessing software modules, and role based access control for its CPU access. The Studio 5000 is the engineering framework. All Allen-Bradley/Rockwell PLCs incorporate Common Industrial Protocol (CIP) encapsulated in TCP/IP for communications.

Message Protection CIP provides (i) Data integrity through TLS Hash-based Message Authentication Code (HMAC). (ii) Confidentiality through data encryption. (iii) Secure data transport through TLS (RFC 5246) and DTLS.

Access Control and Authentication Vulnerabilities The following vulnerabilities were discovered in Compact Logix PLCs.

- CVE-2021-22681:Authentication bypass vulnerability: Studio 5000 Logix designer uses a hard coded key to verify Rockwell Logix PLCs communicating with Studio 5000 software. Authentication bypass vulnerability allows an attacker to discover hard-coded cryptographic key shared between the PLCs and studio 5000. The attacker can use this hard-coded key to impersonate Studio 5000 application to connect with Rockwell PLCs, causing an integrity attack.[12].
- CVE-2019-10952: Denial-Of-Service vulnerability: This vulnerability in the PLC web server allows an authenticated user to send crafted HTTP or HTTPS packets directly to the web server port to force the web server to become unreachable state, leading to DoS Attack [10].

2.4 Attribute Based Access Control (ABAC)

ABAC is the most generalized access control model where access to resources is contingent upon properties of the user, resource, and environment which are referred to as attributes. There are two standardization efforts for ABAC, namely, eXtensible Access Control Markup Language XACML [7] and NIST Next Generation Access Control (NGAC) [7]. XACML is developed for collaborative environments, such as for data sharing across different organizational domains. XACML provides a policy architecture and a policy specification language. XACML is very expressive and can express different types of policies as rules which can be combined using algorithms.

NIST NGAC expresses policies using relations. Two important relations are *assignment* and *association*. Attributes of users and resources are represented in the form of containers. Users/resources are *assigned* to containers if they possess the corresponding attributes. Containers can be organized in the form of a hierarchy through the *assignment relation*, when the presence of one implies that of the other. *Association relations* are between user attributes and resource attributes. These relations are labeled with operations. This implies that users possessing attributes indicated by association relation can perform labeled operations on resources having the attributes associated with the association relation. NGAC also supports obligation policies which are operations that may be performed before or after access. We chose NGAC over XACML because through obligation dynamic policies can be supported. Moreover, policy management is better supported in NGAC.

3 RELATED WORK

Duka et al. [2] proposes authenticated data exchanges through Message Authentication Codes (MAC) between the ICS components such as PLCs and HMIs. The control software within the PLCs and HMIs are constructed with MAC so that the data exchanged between the user and the PLC is verified for authentication and the integrity. This provides better security against Man-in-the-middle attack, but does not address how the Man-in-the-middle attack can be prevented from a rogue PLC. This also requires a design change in the PLC so that it can construct and verify the MAC. Implementation becomes vendor specific and does not provide centralized solution for complex ICS system. Additionally this work does not address password based attacks described in Section 1.

Many industries and organizations are moving towards adopting the RBAC mechanism [13] for ICS. RBAC is used to restrict ICS user privileges based on their roles. Major PLC and HMI manufacturers are incorporating RBAC to their PLC and HMI security features. Rockwell's factory talk security software provides RBAC for accessing PLC and HMI [4]. Honeywell ACS labs [8] proposes a two-layered RBAC for ICS, which provides better security than single-layered RBAC.

M. Onori et al. [16] implemented ABAC for ICS applications. It formulates access policies with user attributes, resource attributes, and environmental attributes, providing complex security policies and authentication. It does not address the security problems at the component level, such as PLC and HMI vulnerabilities. It does not address authentication issues. Also, this paper incorporates XACML standard which does not support dynamic policies.

We implement ABAC as a gateway module between the engineering workstation and the ICS. The ICS is not directly accessible from the engineering workstation. The ABAC gateway module verifies each incoming request from engineering workstation for it's integrity, authenticity and access rights, thus preventing the attack from a rogue workstation. Also, it addresses the component level vulnerabilities that are related to access control by enforcing policies for different operations and encapsulating IP addresses of different ICS components in the ABAC gateway module. Additionally, our method incorporates NGAC standard which supports dynamic policies. Dynamic policies are more suitable for event driven systems such as ICS.

4 APPLICATION OF ABAC FOR ICS

We first present our threat model and then our solution using NIST NGAC ABAC model.

4.1 Threat Model and Example Attack

We make the following assumptions about our threat model based on the work carried out by Bhiam et al. [5] and the vulnerabilities discussed in Sections 2.3.1 ans 2.3.2. We assume that the attacker is an outsider who has no authorized access to the PLC or engineering workstation. The attacker has access to the PLC through the Internet. The attacker can impersonate an authenticated engineering workstation. The attacker can capture and modify the TCP packets communicated between the engineering workstation and the PLC.

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

(1) Obtains the PLC IP address either through internal resources or external websites like Shodan [14].
(2) Intercepts the communication between the targeted PLC and the authenticated engineering workstation.
(3) Extracts the critical information from the packet required to establish the communication and generate a crafted TCP packet. For example, the TCP packets between a Siemens S7-1500 PLC and the TIA portal, contain the PLC firmware information in a plain text. All the S7-1500 PLCs with same firmware version have same public key. The attacker can extract the firmware version from the TCP packet to get the public key. Using this public key, the attacker can successfully establish the communication with the targeted PLC [5]. Similarly, the Rockwell PLCs contain hard-coded cryptographic key in their communication packet. The attacker can extract the hard-coded cryptographic key from the TCP packet and mimic the authenticated Studio 5000 to establish the communication with targeted Rockwell PLC [12].
(4) Creates crafted TCP packets for the targeted PLC.
(5) The attacker can also extract other confidential information at this stage from captured TCP packets.

4.1.2 Man-in-the-Middle Attack - Phase-2: In Phase-2, the attacker establishes the communication with the targeted PLC to cause an integrity or availability attack.

(1) The attacker creates and sends a communication request TCP packet to the targeted PLC's IP address and TCP port.
(2) The PLC responds with a challenge question.
(3) The attacker responds with a challenge response encrypting it with the cryptographic key that was extracted in phase-1.
(4) The PLC decrypts the challenge response with the private key and establishes the communication with the attacker's engineering workstation.
(5) The attacker sends crafted TCP packets to the TCP port of the PLC to cause an integrity or availability attack.

In a similar manner, attackers can exploit the vulnerabilities discussed in Sections 2.3.1 and 2.3.2 to launch attacks on PLC.

4.2 NIST NGAC for ICS

Our NGAC model consists of the following entities.

Users are the entities that require access to resources.
Resources are objects that need protection. We will consider PLC as a target resource for our implementation.
Environments define the conditions on factors external to users and resources needed for the access.
Operations are actions that a user is allowed to perform on the target PLC.

The NIST NGAC model, shown in Fig. 2, is drawn in the form of a graph. The nodes correspond to users (shown by user icon), attributes (shown as solid boxes), and resources (shown as ovals). The solid arrow edges denote assignment relation. The dotted edges signify association relation. The label on the association edge indicates the name of the operation that can be performed by users having the attribute that is indicated by one node of the association edge on resources having the attribute that is indicated by the other node of the association edge. The environmental attributes and the obligation policies are not shown in the figure. The various attributes and operations are listed below:

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

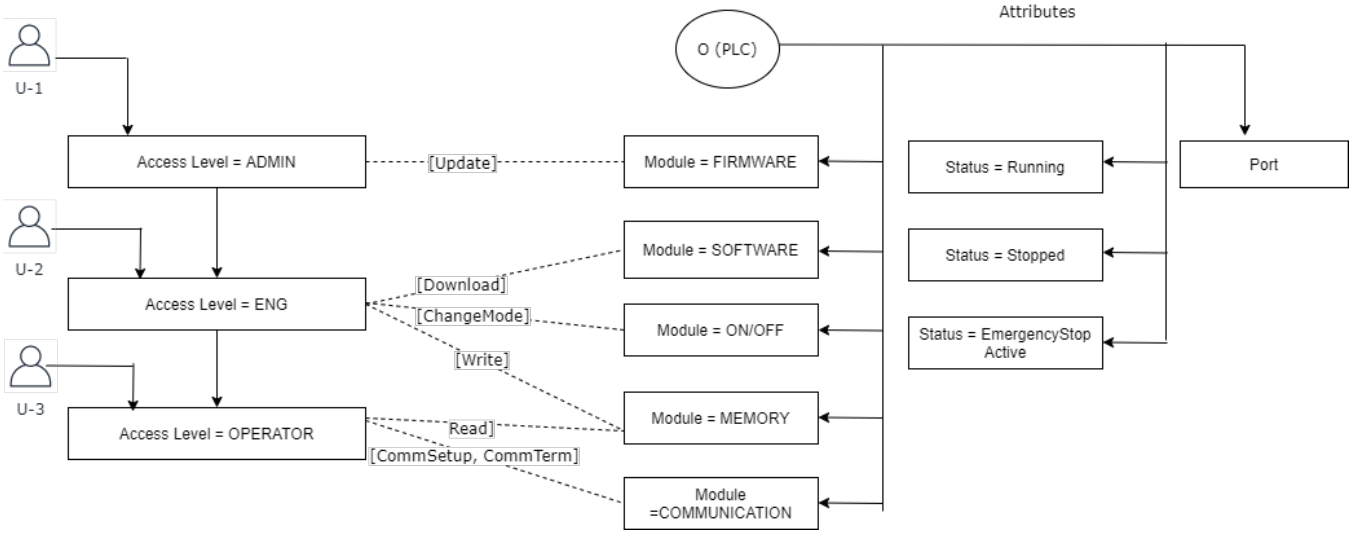

Figure 2: NGAC ABAC Model for PLC

(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.

4.3 Policy Formalization

Each policy is expressed as a tuple as follows:
$\langle\{userAttr\}, \{resourceAttr\}, \{envAttr\}, \{op\}\rangle$ where *userAttr*, *resourceAttr*, and *envAttr* denote the conditions on User Attributes, Resource Attributes, and Environment Attributes respectively, and *op* signifies operations. The above 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.

4.3.1 Communication Setup Policy. *CommSetup* operation is permitted provided the user has access level *Operator*, *Engineer*, or *Administrator* with device *"4c174602"* and the time of access is in the interval 7:00-16:00 EST. The first component in the tuple gives condition on the user attributes. As there are no explicit condition on the resource attribute, the second component is "true". The third component signifies condition over environmental attributes which in this case are location and time. The last component denotes the allowable operation. This is formalized as follows.

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

4.3.2 Memory Write Policy. *WriteMem* operation is permitted provided the user has access level *Engineer*, or *Administrator* with the PLC status is *Stopped*. This is formalized as follows.

$$\langle\{User.AccessLevel \in \{Engineer, Administrator\}, \{PLC.Status = Stopped\}, \{True\}, \{WriteMem\}\rangle$$

4.3.3 Firmware Update Policy. The operation *Update* is permitted provided the user has access level *Administrator* and the operating mode of PLC is *Stopped*. This is formalized as follows.

$$\langle\{User.AccessLevel \in \{Administrator\}, \{PLC.Status = Stopped\}, \{True\}, \{Update\}\rangle$$

4.4 Access Control Architecture

ABAC is implemented as an independent gateway module between the *Enterprise network* and the PLC on an embedded controller. It consists of an authentication module, a communication handler, and an access control module. An overview of the ABAC gateway architecture is shown within dotted lines in Fig. 3, where the ABAC components are colored pink.

Engineering Workstation The user communicates with the PLC through an engineering workstation, shown as a blue box in Fig. 3.

Authentication Module authenticates the identity of the user requesting access to the PLC through a valid user id (*UId*) and password (*Pwd*). When a user requests access to the PLC, the authentication module prompts the user to enter user id (*UId*) and password (*Pwd*). Additionally, it also captures the hard disk serial number of the device (*DeviceID*) from which the user attempts to login. The authentication module will be implemented in the ABAC gateway and consists of a login database to store the details of user identity. The schema of the login database, which we refer as *LoginDB*, is as follows in keeping with the current standards.
$LoginDB = \langle UId, Pwd, LoginCreateTime, LastModTime, PwdExpTime\rangle$ where *UId, Pwd, LoginCreateTime, LastModTime, PwdExpTime* denote user id, password, time at which the login was created, last time the password was modified, and the password expiration time. The authentication module encrypts *UId*, *Pwd* and *DeviceID*.

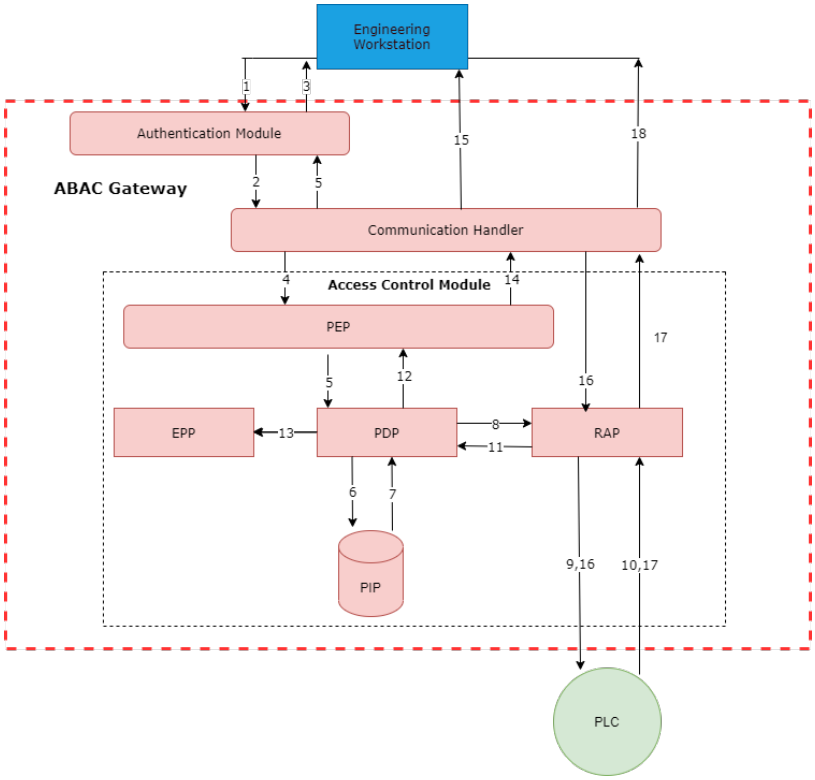

Figure 3: Communication between an engineering work station and a PLC through ABAC gateway

Communication Handler handles all communications between the *Enterprise network* and the ABAC module. It consists of communication settings such as port forwarding, Network Address Translation (NAT), and other communication features. The communication handler extracts the following components from an incoming connection using a packet sniff function [15]. It attaches the date and time, calculates the checksum, and packs it into a consolidated packet, which we refer to as *AccessRequestPacket*, the structure of which is given as follows:
AccessRequestPacket = ⟨ *Src.Port*, *Dst.Port*, *Src.IPAddr*, *Dst.IPAddr*, *Length*, *DataPacket*, *DateTime*, *Checksum* ⟩ where: (i) *Src.Port* and *Dst.Port* denote source and destination ports that are extracted from the TCP layer. The PLC attribute *Port* is extracted from the *Dst.Port* component. (ii) *Src.IPAddr* and *Dst.IPAddr* denote source and destination IP addresses that are extracted from the IPv4 layer. The source IP address is extracted from the *Src.IPAddr* component. (iii) *Length* and encrypted *DataPacket* are extracted from the TCP. The *DataPacket* consists of the requested *Operation*, PLC attribute *Module* and the related data. (iv) *DateTime* denotes the date and time of the incoming packet. The communication handler records the date and time when the incoming packet is received and adds to the *AccessRequestPacket*. The environmental attribute *DateTime* is extracted from *DateTime* component. (v) *Checksum* is calculated and added at the end of the request packet for later verification.
Access Control Module is described below.

[Policy Enforcement Point (PEP)]: PEP receives access request from the user, authenticates the packet by performing initial verification of checksum, and forwards it to the PDP for decision making.
[Policy Decision Point (PDP)]: PDP on receiving the request, computes the decision based on the policy stored in the PIP and returns a grant or deny decision to the PEP. It also generates an event log and forwards it to EPP for record-keeping.
[Policy Information Point (PIP)]: PIP stores the policy information for accessing critical resources. The PIP incorporates databases for user attributes, access policies, and geo-ip database for mapping IP addresses with locations.
[Resource Access Point (RAP)]: The PDP and EPP modules, and the communication handler communicate with the PLC through the RAP module.
[Event Processing Point (EPP)]: The EPP stores all the events received by the PDP.

The RAP is implemented as a software library and will be configured in the PLC software. The PLC communicates with the ABAC module through the RAP.

The PEP module consists of databases for user attributes, access policies, and geo-ip database for mapping IP addresses with corresponding locations. The schema of these databases are given below.

AttributesDB = ⟨*UId*, *DeviceID*, *AccessLevel*⟩ where *UId, DeviceID, AccessLevel* denote user id, device id, and the access level of the corresponding user.
PolicyDB = ⟨*Operation*, *Policy*⟩ where *Operation, Policy* denote PLC operation and the corresponding policy.
GeoIPDB = ⟨*IPAddress*, *Loc*⟩ where *IPAddress, Loc* denote IP address and the corresponding location. Location can be any geographical location or the organization domain address.

The EPP module consists of a database for event logs referred to as *EventDB* whose schema is given below.
EventDB = ⟨*UId*, *Operation*, *Src.IPAddr*, *Decision*, *DateTime*⟩ where *UId, Operation, Src.IPAddr, Decision, DateTime* denote user id, requested operation, source IP address, decision made, and the date and time of the access. *Decision* value is either *Grant* or *Deny*.

4.5 Communication Protocol between the Engineering Work Station and the PLC

The PLC IP address is encapsulated in the ABAC module with port forwarding rules. The user accesses the PLC through the ABAC gateway. The user access request is encapsulated in a TCP packet, which we refer to as *RequestPacket*. Also, in our protocol we denote the encrypted data as *Enc.X* where *X* is the unencrypted data. The *RequestPacket* is as follows:
RequestPacket = ⟨ *Header*, *Src.Port*, *Dst.Port*, *Length*, *Enc.DataPacket*, *Checksum* ⟩ where *Header* denotes the TCP protocol header, *Src.Port* denotes the source port, *Dst.Port* denotes the destination (PLC) port, *Length* denotes length of the *Enc.DataPacket*, *Enc.DataPacket* denotes the encrypted user request to the PLC consisting of *Operation*, the PLC attribute *Module*, and the related data, and *Checksum* that is used for the verification of the packet.

In order to perform *Download*, *ReadMem*, *WriteMem*, *ChangeMode* and *CommTerm* operations, the first step is to establish communication between the engineering workstation and the PLC through the ABAC gateway. This is done through performing *CommSetup* operation. The access policy for *CommSetup* operation is defined in Section 4.3. The protocol sequence appears below.

[Step 1: Engineering Workstation {*Enc.UId, Enc.Pwd, RequestPacket***} → Auth Module]** The engineering workstation on initiating the communication with the PLC will be directed to the authentication module that prompts the user to enter her credentials. The authentication module captures *Enc.UId*, *Enc.Pwd* and the *Enc.DeviceID* from which the user logs in.
[Step 2: Auth Module {*UId, RequestPacket, DeviceID***}→ Comm Handler]** Authentication module forwards {*UId*, *RequestPacket*, *DeviceID*} to the communication handler on valid authentication.
[Step 3: Auth Module {*Deny***}→ Engineering Workstation]** Authentication module denies the connection from the engineering workstation on invalid authentication and disconnects.
[Step 4: Comm Handler {*UId, AccessRequestPacket, DeviceID***}→ PEP]** Communication handler creates the *AccessRequestPacket* from *RequestPacket*, and forwards it to the PEP along with the *UId* and *DeviceID*.
[Step 5: PEP {*UId,AccessRequestPacket,DeviceID***} → PDP]** PEP forwards the *UId*, *AccessRequestPacket*, and *DeviceID* to the PDP for decision making.
[Step 6: PDP {*UId, Operation, Src.IPAddr***} → PIP]** PDP extracts *Operation*, *PLC.Module* and *Src.IPAddr* from the *AccessRequestPacket*. The *Operation* for this case is *CommSetup*. It communicates with the PIP to extract the *Policy* for *CommSetup*, *User.AccessLevel*, *Loc* and *User.DeviceID*.
[Step 7: PIP {*Policy, User.AccessLevel, Loc, User.DeviceID***} → PDP]** PIP returns the *Policy* for *CommSetup*, *User.AccessLevel*, *Location*, and *User.DeviceID* to the PDP.
[Step 8: PDP{*request PLC.Status***} → RAP]** PDP communicates with the RAP to get the current status of the PLC.
[Step 9: RAP {*request PLC.Status***} → PLC]** RAP communicates with the PLC to extract the current status.
[Step 10: PLC {*PLC.Status***} → RAP]** PLC responds to the RAP with its current status.
[Step 11: RAP{*PLC.Status***}→ PDP]** RAP responds to the PDP with the PLC status.
[Step 12: PDP {*Decision***} → PEP]** PDP computes the decision and forwards it to the PEP.
[Step 13: PDP {*Event***} → EPP]** PDP records an *Event* with the EPP that consists of *UId*, *Operation*, *Src.IPAddr*, *Decision*, and *DateTime*.
[Step 14: PEP {*Decision* **→ Comm Handler]** PEP forwards the decision to the communication handler. If the *Decision* is *Deny*, then the communication handler disconnects from the engineering workstation.
[Step 15: Comm handler {*RequestPacket* **→ RAP]** If the *Decision* is *Grant*, then the communication handlers forwards *RequestPacket* to the RAP.
[Step 16: RAP {*RequestPacket* **} → PLC]** RAP forwards *RequestPacket* to the PLC to perform *CommSetup* operation.
[Step 17: PLC {*ResponsePacket***} → Communication Handler]** PLC accepts the connection from the engineering workstation and establishes the communication. It sends response to the communication handler through RAP. The format of the *ResponsePacket* is similar to the *RequestPacket* and consists of the same fields. The *Enc.DataPacket* component consists of response from the PLC to the user.
[Step 18: Communication Handler {*ResponsePacket***} → Engineering Workstation]**: The communication handler forwards *ResponsePacket* to the engineering workstation.

4.6 *WriteMem* Operation through ABAC Gateway

WriteMem operation is used to write to a memory location of the PLC. The policy for *WriteMem* is defined in Section 4.3. The protocol for executing the operation is given below.

[Step 1] The prerequisite for performing any operation on the PLC is to perform *CommSetup* operation as in Section 4.5.
[Step 2] The engineering workstation requests *WriteMem* operation through the ABAC gateway.
[Step 3] The user attributes (*UID* and *User.DeviceID*) are captured through *CommSetup* operation.

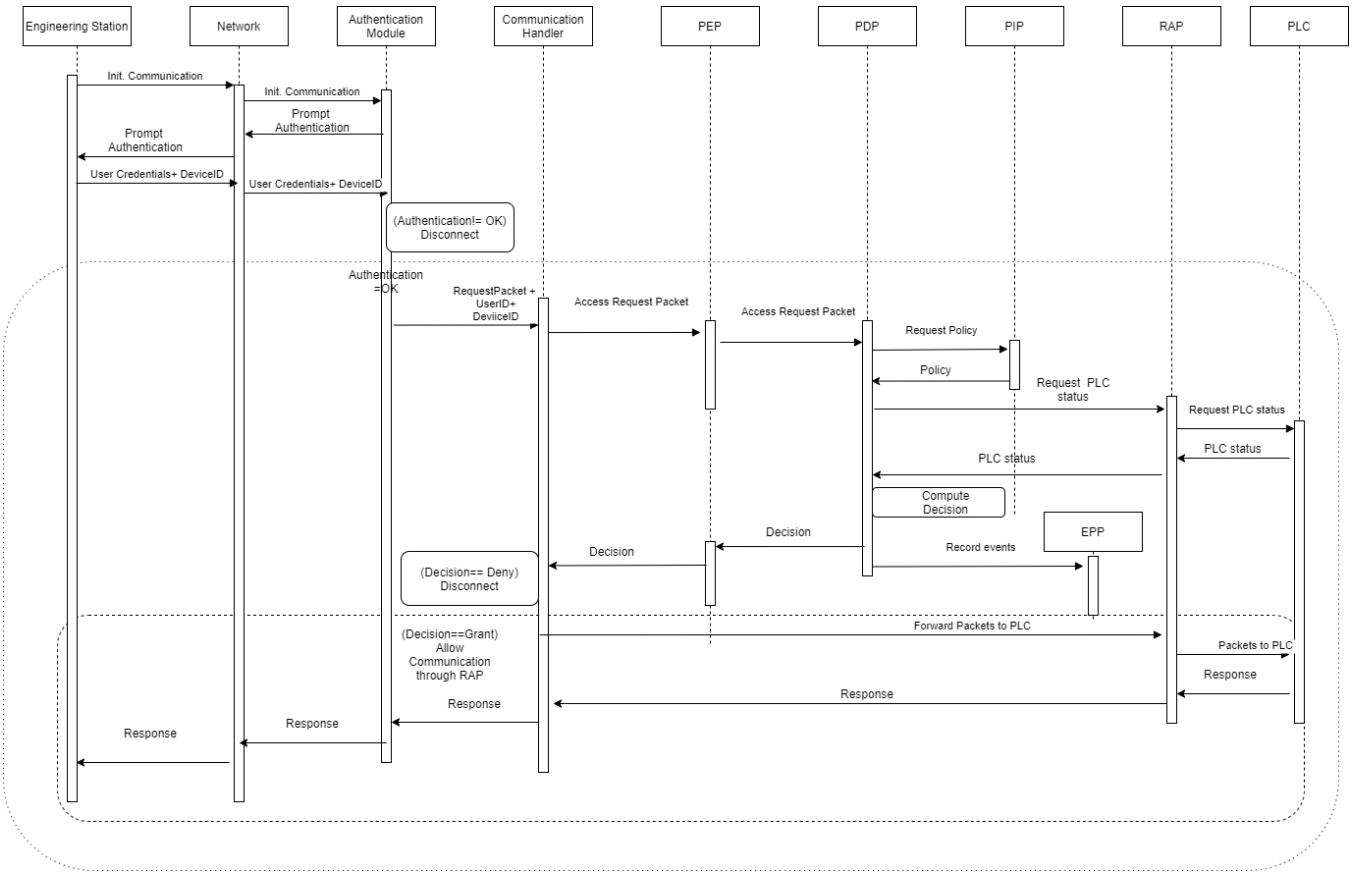

Figure 4: Communication Setup Protocol

[Step 4] The ABAC gateway received the *WriteMem* operation and validates access policy 4.3 for *WriteMem* operation for the currently logged in user and the current *PLC.Status*.
[Step 5] If the decision is to *Grant*, the ABAC gateway forwards the *WriteMem* operation to the PLC. The PLC performs the operations and sends response to the engineering workstation through the ABAC gateway.
[Step 6] If the decision is to *Deny*, the ABAC gateway disconnects from the engineering workstation.

5 SECURITY ANALYSIS

The communication between the various components is shown in Fig. 5. We assume that our ABAC module is trusted and tamper resistant. We also assume that the communication between the ABAC gateway and the PLC is protected.

We now discuss how an attack exploiting the vulnerabilities related to access control or authorization discussed in Sections 2.3.1 and 2.3.2 can be prevented by our ABAC module. The sequence diagram for attack prevention is as shown in Fig. 6. The PLC is not directly accessible to the attacker as its IP address is encapsulated in the ABAC gateway with port forwarding rules. The attacker has to go through the ABAC gateway to perform an attack on the PLC.

The attacker can either use the engineering framework such as the TIA portal or Studio 5000 from a rogue engineering workstation or custom application impersonating the engineering framework. We refer to the user interface used by the attacker as *AttackInterface*. This attack can be prevented by the ABAC gateway at multiple levels as described below.

[Step 1: AttackInterface {*Enc.UId, Enc.Pwd, RequestPacket***} → Auth Module]** In order to perform an attack, the attacker sends *RequestPacket* to perform *CommSetup* operation as the initial step, and he or she will be prompted to enter valid credentials.
[Step 2: Auth Module {*UId, RequestPacket, Enc. DeviceID***}→ Comm handler]** If the attacker successfully hacks through authentication module by brute force attack, then authentication module captures *Enc. DeviceID* and and forwards *UId*, *RequestPacket*, and *DeviceID* to the communication handler.
[Step 3: Auth Module {*Deny***}]** If the attacker fails to enter valid credentials, then the authentication module rejects the *RequestPacket* and disconnects, thus preventing the attack.
[Step 4: Comm Handler {*UId,AccessRequestPacket, DeviceID***}→ PEP]** The communication handler creates the *AccessRequestPacket* from *RequestPacket*, and forwards it to the PEP along with the *UId*, and *DeviceID*.

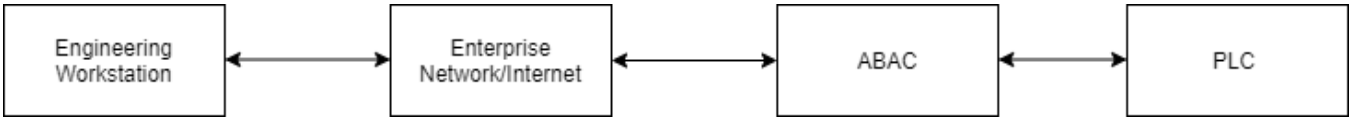

Figure 5: Communication Architecture

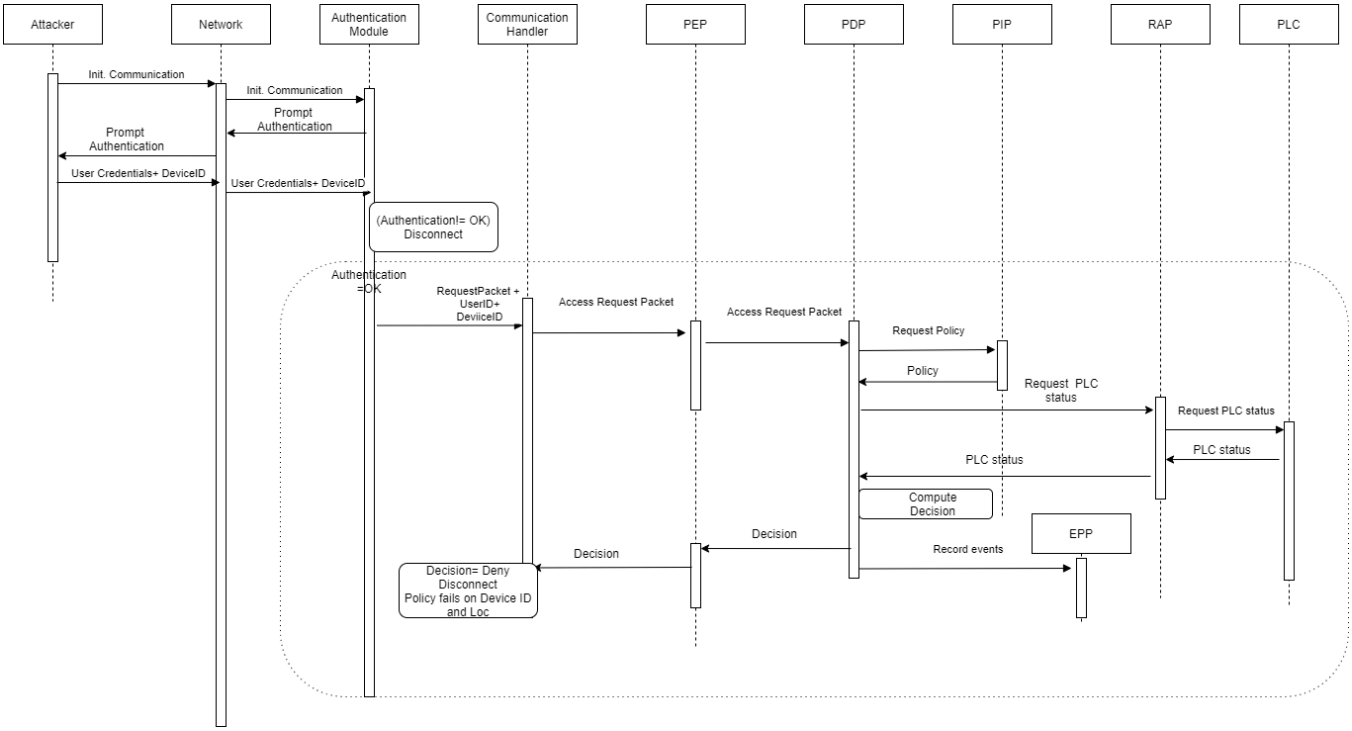

Figure 6: Attack prevention by ABAC Module

[Step 5: PEP {*UId, AccessRequestPacket, DeviceID***} → PDP]** The PEP forwards the *UId,AccessRequestPacket*, and *DeviceID* to PDP for decision making.

[Step 6: PDP {*UId, Operation, Src.IPAddr***} → PIP]** The PDP extracts *Operation*, *PLC.Module* and *Src.IPAddr* from the *AccessRequestPacket*. The *Operation* for this case is *CommSetup*. It communicates with the PIP to extract the *Policy* for *CommSetup*, *User.AccessLevel*, *Loc*, and *User.DeviceID*.

[Step 7: PIP {*Policy, User.AccessLevel, Location, DeviceID***} → PDP** The PIP returns the *Policy* for *CommSetup*, *User.AccessLevel*,*User.DeviceID*, and *Loc* to the PDP.

[Step 8: PDP {*request PLC.Status***} → RAP]** The PDP communicates with the RAP to get the current status of the PLC.

[Step 9: RAP {*request PLC.Status***} → PLC]** The RAP communicates with the PLC to extract the current status.

[Step 10: PLC {*PLC.Status***} → RAP]** The PLC responds to the RAP with its current operating mode.

[Step 11: RAP {*PLC.Status***}→ PDP]** The RAP responds to the PDP with the current operating mode of the PLC.

[Step 12: {*Decision***} → PEP]** PDP computes decision based on the access policy and makes *Deny* decision, as the conditions for approval fails because:

- *User.DeviceID* is unique and registered with the authorized organization or a manufacturing facility. We use hard disk serial number as *User.DeviceID*. The attackers *DeviceID* does not match as long as the device is not stolen or the attack is not performed by a rogue or disgruntled employee.
- *Event.Loc* is unique to the organization. The request has to go through company domain address (for example, *"OrgABC.local"*) as defined in the access policy in 4.3. The remote attacker cannot possibly perform the attack from *"OrgABC.local"* as long as the attack is not performed by a rogue or disgruntled employee who is still employed.

Thus the attack is prevented at Step 12 by the PDP module.

[Step 13: PDP {*Event***} → EPP]** The PDP records an *Event* with the EPP that consists of *UId*, *Operation*, *Src.IPAddr*, *Decision*, and *Time*.

[Step 14: PEP {*Decision* → **Comm Handler] The PEP forwards the decision to the communication handler. If the *Decision* is *Deny*, then the communication handler disconnects from the *AttackerInterface*.

Therefore, the attacker fails to establish communication with the targeted PLC and cannot send the crafted TCP packets to perform the intended attack. To summarize, an intended attack can be prevented at two stages: Step 3 by the authentication module and Step 12 by the access control module.

Now we describe how various forms of attacks are prevented by our ABAC module.

[Protection against integrity attacks] Without the ABAC gateway, the attacker can directly send a crafted *RequestPacket* packet from a rogue engineering workstation or another application to the PLC to cause an integrity attack, such as those caused by exploiting vulnerabilities CVE-2019-10943, CVE-2020-15782, and CVE-2021-22681 discussed in Sections 2.3.1 and 2.3.2. For example, an integrity attack is prevented by the ABAC module by enforcing memory write policy *WriteMem* described in Section 4.3, restricting the access to only authorized workstations through unique device ID and only under certain conditions.

[Protection against repudiation attacks] The ABAC gateway records every request with *UId*, *Src.IPAddr*, *DateTime*, *Loc* in *EventDB*, hence providing protection against repudiation attacks.

[Protection against availability attacks (DoS)] The attacker cannot perform a DoS attack directly on the PLC web server by exploiting the vulnerabilities such as CVE-2019-10952 discussed in Section 2.3.2. The ABAC validates each *RequestPacket* and the *User.DeviceID* against access policies and the PLC status before forwarding it to the PLC.

[Protection against reply attacks]: The *Communication Handler* module of the ABAC gateway captures the *Date and Time* for each incoming *RequestPacket* at every session to ensure that *RequestPacket* is not duplicated from the previous session, thus providing the protection against reply attacks.

[Protection against confidentiality attacks] Even though *RequestPacket* is visible for an attacker, *User.DeviceID*, *User.AccessLevel* and access policies are not available to the user.

[Protection against privilege elevation] *User.AccessLevel* is stored in an embedded database in the ABAC gateway module. It is local to the ABAC gateway and is not accessible to the external user, thus the user cannot change his or her access privilege.

6 CONCLUSION

Security of ICS is challenging. Industries like oil and gas, power, and other manufacturing industries are constantly under attack due to lack of sophisticated security mechanisms in ICS. This paper investigates vulnerabilities of some existing PLCs. We propose a solution that strengthens the authentication and access control mechanism. We demonstrate how ABAC can be implemented for protecting the PLC. Specifically, we demonstrate how the NIST NGAC model can be used for protecting the PLC. We show how such policies can be specified and enforced. In future, we plan to explore the use of ABAC for other ICS components. We also plan to develop a test bed simulating an oil and natural gas setting. This will help us evaluate the efficacy of our approach and also the performance penalty incurred because of enforcing ABAC.

ACKNOWLEDGEMENTS

The work was supported in part by funding from NSF under Award Numbers CNS 1822118, NIST, Cyber Risk Research, Statnett, AMI, and ARL.

REFERENCES

[1] A. Adeen, Y. Hyunguk, and A. Irfan. 2021. Empirical Study of PLC Authentication Protocols in Industrial Control Systems. 383–397. https://doi.org/10.1109/SPW53761.2021.00058 Last accessed 12 December 2021.

[2] D. Adrian-Vasile, G. Béla, and H. Piroska. 2018. Enabling authenticated data exchanges in industrial control systems. 1–5. https://doi.org/10.1109/ISDFS.2018.8355337 Last accessed 12 December 2021.

[3] M. Aftab, Z. Qin, S. Zakria, S. Ali, Pirah, and J. Khan. 2018. The Evaluation and Comparative Analysis of Role Based Access Control and Attribute Based Access Control Model. In *International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*. 35–39. https://doi.org/10.1109/ICCWAMTIP.2018.8632578

[4] Rockwell Automation. 2021. FactoryTalk Security System Configuration Guide. https://literature.rockwellautomation.com/idc/groups/literature/documents/qs/ftsec-qs001_-en-e.pdf. Last accessed 21 November 2021.

[5] E. Biham, S. Bitan, A. Carmel, A. Dankner, U. Malin, and A. Wool. 2019. Rogue Engineering Station Attacks on Simatic S7 PLCs. https://i.blackhat.com/USA-19/Thursday/us-19-Bitan-Rogue7-Rogue-Engineering-Station-Attacks-On-S7-Simatic-PLCs.pdf/. Last accessed 5 July 2021.

[6] T. M. Chen and S. Abu-Nimeh. 2011. Lessons from Stuxnet. *Computer* 44, 4 (2011), 91–93. https://doi.org/10.1109/MC.2011.115

[7] D. Ferraiolo, R. Chandramouli, D. Kuhn., and V. Hu. 2016. Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC). In *ACM International Workshop*. 13–24. https://doi.org/10.1145/2875491.2875496

[8] Honeywell ACS Labs. 2014. RBAC Driven Least Privilege Architecture For Control Systems. https://www.osti.gov/servlets/purl/1124080/. *Technical Report* (2014). Last accessed 20 June 2021.

[9] NIST. 2019. https://nvd.nist.gov/vuln/detail/CVE-2019-10943/. Last accessed 1 July 2021.

[10] NIST. 2019. https://nvd.nist.gov/vuln/detail/CVE-2019-10952. Last accessed 1 Jan 2022.

[11] NIST. 2021. https://nvd.nist.gov/vuln/detail/CVE-2020-15782/. Last accessed 1 July 2021.

[12] NIST. 2021. https:https://nvd.nist.gov/vuln/detail/CVE-2021-22681. Last accessed 1 Jan 2022.

[13] F. Santiago, A. Javier, and A. Saioba. 2019. A Role-Based Access Control Model in Modbus SCADA Systems. A Centralized Model Approach, Vol. 19. https://doi.org/10.3390/s19204455

[14] Shodan. [n. d.]. https://www.shodan.io/. Last accessed 23 July 2021.

[15] Unknown. 2018. Packet Sniffing in Python. https://www.uv.mx/personal/angelperez/files/2018/10/sniffers_texto.pdf/. Last accessed 15 July 2021.

[16] E. Yalcinkaya, A. Maffei, and M. Onori. 2017. Application of Attribute Based Access Control Model for Industrial Control Systems. *International Journal of Computer Network and Information Security* 9 (2017), 12–21. https://doi.org/10.5815/ijcnis.2017.02.02