FlowCasting: A Dynamic Machine Learning based Dashboard for Bike-Sharing System Management

On the Description of Access Control Policies in Networked Industrial Systems

M. Cheminod, L. Durante, L. Seno and A. Valenzano
National Research Council of Italy
CNR–IEIIT, c.so Duca degli Abruzzi, 24, Torino, Italy
{manuel.cheminod, luca.durante, lucia.seno, adriano.valenzano}@ieiit.cnr.it

Abstract

The specification and verification of access control policies are fundamental steps in the process of securing industrial control systems and critical infrastructures.

The focus of this paper is on bridging the semantic gap between high-level access control policies specified in the Role-Based Access Control (RBAC) framework and the low-level security mechanisms actually implemented in the physical system.

Our approach is based on a novel kind of model, which includes two distinct views of the system, namely an RBAC-based specification and a low-level system description. The descriptive capabilities of the model are presented through an example derived from a real prototype plant for printed circuit boards re-manufacturing/demanufacturing.

1. Introduction

The protection of industrial distributed control systems (IDCS) and critical infrastructures against malicious attacks, made through their underlying communication networks and the Internet, is a topic that, besides receiving increasing attention in the last years, has aroused a lot of concern not only in the scientific community but also outside it, and has progressively involved management, decision-making and even government institutions in almost all developed countries.

From a technical point of view, there is a wide consensus, today, in considering the IDCS security as an iterative process rather than a product, consisting of a number of well-known and clearly defined steps [1]. One of the key elements in this process is the definition and implementation of suitable security policies (access policies, in particular) able to unambiguously specify *"who can do what on what object"* in the whole ICT infrastructure IDCS are based on.

Actually, as a matter of facts, a very large number of existing IDCS were not designed with their security in mind and, curiously enough, the same is also true for many new systems that have been recently developed from scratch. However, in a number of different situations (security-aware IDCS) the planning and specification of access policies usually occur at a high abstraction level, using techniques and formalisms, such as the Role Based Access Control (RBAC) [2], which enable reasoning about objects, operations and permissions without being forced to take into account a lot of details pertaining the actual system implementation. Unfortunately, there is another side of the coin, since in this way establishing a (correct) correspondence between the high-level policies and the actual protection mechanisms (also referred to as either controls or security controls) included into the system becomes a difficult and cumbersome task, in particular if it has to be performed by hand.

It is worth noting that this aspect is particularly important for those systems that were deployed with little or no attention to security, and still include few (and poor) low-level protection mechanisms, since acting on their configuration (i.e. changing the filtering rules of firewalls /switches, or the setting of passwords) is often the only viable alternative which is offered to security experts to enforce a predefined set of high-level access policies.

This paper builds on our previous experiences [3, 4, 5] gained in trying to bridge the semantic gap between the high level description of policies and their actual implementation in IDCS. Honestly speaking, the approach we present here is a sort of major departure with respect to our past research activities in this area, since it relies on a new kind of model that includes two different views of the system at the same time [6]. This new approach has been triggered by the lesson learned in some of the practical situations where our attempts to offer an adequate solution to this problem were only partially successful. The first view (*specification*) included in our innovative model enables the specification of policies according to the RBAC paradigm, taking advantage of the impressive amount of work already carried out in that framework and well consolidated in the literature, the second (complementary) view (*implementation*), on the other hand, focuses on the system implementation details such as its topology, device configurations, access control mechanisms and so on, that is on those low-level aspects which cannot be ignored, for instance, in checking the correctness of the policy implementation.

This work was partially supported by the CNR in the framework of the flagship project "Factory of the Future", GECKO "Generic Evolutionary Control Knowledge-based mOdule" subproject.

Our ultimate goal (not tackled in this paper) is the development of a new automated software tool for the analysis of coherency and correct implementation of policies in IDCS (*verification*), that could be able to overcome the limits experienced in our past work. The description capabilities of the model, presented in this paper through an example derived from a practical case study of interest, are at the basis of such an ongoing project.

The paper is then structured as follows: Sec. 2 briefly discusses some related works and recalls the limits met in our previous approaches. Sec. 3 describes a (small) real-world prototype plant for printed circuit boards (PCB) re-manufacturing/de-manufacturing we will use as example, Sec. 4 summarizes the features of the new model in an informal way. Sec. 5 shows how to use the model in the description of the considered system, while Sec. 6 gives an example of verification and, finally, Sec. 7 draws some conclusions.

2. Related works and previous experience

The modeling and analysis of access control policies in several domains, not particularly related to IDCS, have been extensively addressed in the scientific literature and a number of techniques have been proposed that are able to work at different high levels of abstraction. RBAC [2, 7], with its many flavor and extensions, is perhaps the most appealing and well-established solution of this kind also because of its commercial applications, for which a lot of work has been done and results, products and tools are now available. As other modeling frameworks, such as the discretionary [8] and mandatory [9, 10] access control approaches, however, RBAC was conceived to abstract from the actual implementation of systems, in order to let designers and administrators focus, in particular, on constraints on role authorization and operation execution independently on how this is actually achieved in the real system itself. Reasoning at a high level of abstraction has surely many advantages and makes the task of checking the policy coherency (i.e. the absence of conflicting policies) easier, manageable and even more elegant from a formal point of view. Unfortunately, a much weaker support is offered, in this case, to verifying the correct mapping of policies onto the low-level security mechanisms included in the system.

Several attempts to overcome this and other drawbacks were carried out in the past [11, 12, 13, 14, 15, 16, 17, 18, 19] but, unfortunately, nearly all of interesting proposals cannot abstract from the presence in the system of suitable software and/or hardware mechanisms that are able to enforce the correct policy implementation. Moreover, a high degree of homogeneity is usually assumed more or less explicitly for the network nodes [19].

In the past years we too tried to cope with such a practical limitation, essentially by enriching the RBAC model description with suitable information about the system architecture, types of devices and actions allowed/forbidden to the system users. Despite some partially positive and preliminary results, that were obtained by means of prototype analysis tools designed *ad-hoc*, however, we realized that a slightly different approach was needed because of the following main reasons:

- In some situations we found it quite difficult including in a pure RBAC model some configuration details (i.e. host accounts and physical locations of devices) that were important, instead, for the policy mapping analysis.
- Differences between RBAC roles/users and the system groups/user-names were awkward to be managed during the analysis, and the related sets hard to be kept constantly separated from the conceptual point of view.
- Access control policies and system configurations/settings are often managed and under the responsibility of different (and disjoint) teams of people, who are used to adopt different languages and procedures. A unified common formalism could surely help to overcome this issue.
- The inclusion in the model of a fine-grained description of the physical system is highly advisable when a significant amount of information (i.e. firewalls and switches filtering rules, device configurations, account settings and so on) has to be gathered from the system itself in a semiautomatic/automatic way.
- Checking the correctness of the system description, in particular for large and/or complex architectures, is easier if this view is kept separated in the security model.
- Industrial systems are often heterogeneous in their nature; moreover, the widespread adoption of h/w and s/w mechanisms able to enforce the high-level access policies is simply out of dispute.

Because of all these reasons we were convinced that an explicit view of the physical system and its low-level mechanisms had to be included in the security model.

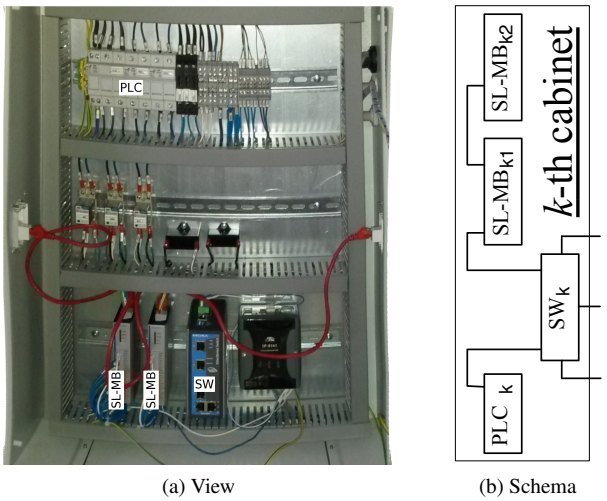
3. Plant

The modeled system considered in this paper is a plant for printed circuit boards (PCB) re/de-manufacturing, consisting of fifteen transport modules interconnecting the re-/de-manufacturing cells and the input/output buffers. Fig. 1 shows a part of the plant and the white box highlights one of the transport modules.

3.1 Cabinets

A cabinet $C_{k\in\{1,\ldots,15\}}$ (see Fig. 2a) is fastened to each transport module and contains equipment for the module

Figure 1: Plant view



(a) View (b) Schema

Figure 2: Cabinet

management and its interaction with its neighbour modules (see Fig. 2a, bottom part). The logical schema for devices and connections in C_k is shown in Fig. 2b. From top to bottom, C_k includes: two ModBus/TCP slaves (Moxa ioLogik E1241), namely $SL-MB_{k1}$ and $SL-MB_{k2}$, an industrial managed switch (Moxa EDS-405A), namely SW_k, and an industrial PC running a Soft-PLC application, namely PLC_k. Different commercial products (LinPAC-5141, Leonardo PC BOX, Moxa IA261-I and Moxa IA262-I) have been adopted to implement the industrial PC. Connections between devices are shown as solid lines connecting physical network ports.

3.1.1 ModBus/TCP slaves

ModBus/TCP slaves are provided with a web server allowing the device setup via an HTTP connection. Accesses to the web server can be protected by a user-defined password, but for our purposes we assume that the factory settings (i.e. default passwords) are initially left unchanged, as it often happens when just minimal changes are made to satisfy the plant functional requirements. We call $HTTPs_{SL-MB_{kn}}$ the web server running on ModBus/TCP slave n of cabinet C_k, and *admin* the activity enabled by such a server.

Moreover, each slave carries out its main task by interacting with a master through a network connection: we call $OPERs_{SL-MB_{kn}}$ the related listening daemon, and *oper* the provided functionality. Slave $SL-MB_{k2}$ has a daisy-chain connection to slave $SL-MB_{k1}$ which, in turn, is connected to SW_k.

3.1.2 Soft-PLCs

As mentioned before, control devices are industrial PCs, running some flavor of the Linux OS that allows users to login. Some devices are provided with keyboard/mouse/VGA interfaces, so that a physical console can also be connected but, in general, operations can be carried out remotely through a network connection. Control devices offer PLC functionalities by running a suitable software server, ISaGRAF in this case. In addition, an OPC-UA server is also installed on each PC. Summarizing, three main pieces of software are hosted on each industrial PC: a login module ($l\_m_{PLC_k}$) provided by the OS enables the *login* operation, the ISaGRAF server, called IGs_{PLC_k}, supports two distinct actions (*admin* for setup and configuration and *oper* for runtime operation), while the OPC-UA server, $OPCs_{PLC_k}$, allows *admin* and *oper* actions in its turn. Each PC is then connected to the switch SW_k.

3.1.3 Switches

Moxa EDS-405A industrial managed switches are equipped with five network ports. Configuration is allowed through a serial console, telnet or web browser connections and some basic security mechanisms are also provided. For sake of conciseness, in our example we do not include these details and we assume that no customization is made, so that each switch works just like a hub. In our plant configuration, two switch ports are used for connections to other devices in the same cabinet, whereas the remaining three ports are used for connections external to the cabinet.

3.2 Process-level network

A process level network connects each cabinet to the process-level control area, where configuration, supervision and management hosts are located.

In this paper we take into account only two cabinets besides the main devices of the process control area, as shown in the conceptual schema depicted in Fig. 3. The physical location of hosts pertaining to the two (control and process) levels, an aspect which is relevant to security, can be deduced through the plant map shown in Fig. 4, where R_x and d_y refer to a specific room and door, respectively.

To increase fault-tolerance, C_1 and C_2 are connected so as to build a physical ring by means of two of the ports

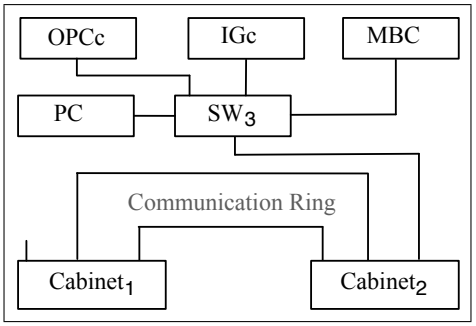


Figure 3: Plant schema

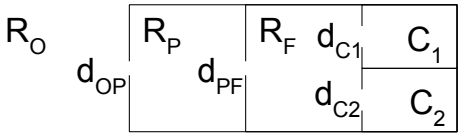


Figure 4: Plant map

of their internal switches devoted to external connections, while the third port allows connections between the cabinet and the process-level network devices. SW_3 adopts the same default configuration settings already mentioned for SW_1 and SW_2.

3.2.1 IGc, OPCc, MBC, PC

The upper part of Fig. 3 includes four hosts: *IGc* is a PC hosting the ISaGRAF client software, *OPCc* is a PC devoted to the OPC-UA client, *MBC* runs a ModBus/TCP controller while *PC* is a generic laptop. All hosts allow users to login and connect to the relevant resources through the process-level network, by means of their OS login modules (l_m_{IGc}, l_m_{OPCc}, l_m_{MBC} and l_m_{PC} respectively).

4 Model description

Our model consists of two parts: one, called *specification*, enumerating *"who can do what on what"* from the access control policy declaration point of view, the other, called *implementation*, concerning a similar description but based on the actual implementation of the system. In more detail, *specification* and *implementation* are defined as two sets of tuples $\mathcal{S}$ and $\mathcal{I}$:

$$\mathcal{S} = \{(\hat{u},\ \hat{\pi},\ \hat{\beta})\} \qquad \mathcal{I} = \{(\check{p},\ \check{\pi},\ \check{\beta})\} \tag{1}$$

where π and β indicate an action and an object respectively (different hat symbols distinguish between entities of the two sets), while u stands for *user* in the RBAC policy declaration and p stands for *player*, that is a physical person interacting with the system.

4.1 Specification

The definition of the high level security policies that regulate accesses to the resources of the system is specified with the support of the RBAC framework.

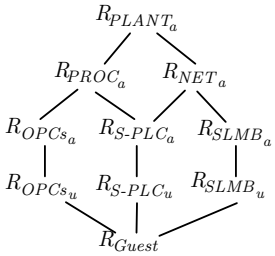


Figure 5: Role hierarchy

RBAC *ROLES*		
R_{PLANT_a}	$\geq$	R_{PROC_a}
R_{PLANT_a}	$\geq$	R_{NET_a}
R_{PROC_a}	$\geq$	R_{OPCs_a}
R_{PROC_a}	$\geq$	R_{PLC_a}
R_{NET_a}	$\geq$	R_{PLC_a}
R_{NET_a}	$\geq$	R_{SLMB_a}
R_{OPCs_a}	$\geq$	R_{OPCs_u}
R_{PLC_a}	$\geq$	R_{PLC_u}
R_{SLMB_a}	$\geq$	R_{SLMB_u}
R_{OPCs_u}	$\geq$	R_{Guest}
R_{PLC_u}	$\geq$	R_{Guest}
R_{SLMB_u}	$\geq$	R_{Guest}

Table 1: Role inheritance

This approach revolves around the concept of roles to which permissions are assigned. A permission binds an operation to an object. Users of the system are then assigned to roles, acquiring the related permissions. This approach allows for a compact description of policies as several different users can be assigned to a role (inheriting a specific set of permissions), without the burden of an explicit management of permissions done user by user.

Following the classification of the RBAC flavors of [2] we here consider *hierarchical RBAC* in which roles can be organized in a hierarchical fashion, allowing role inheritance to users and permission inheritance to roles.

The main goal in the definition of policies is to clearly state *"who can do what on what object"*. The *"what"* is specified by means of permissions, which are expressed as pairs combining an operation with an object, i.e.: *(admin,* $HTTPs_{SL-MB_{kn}}$*)* where *admin* and $HTTPs_{SL-MB_{kn}}$ are respectively an activity and one of the servers identified in Sec. 3.1. The *"who"* definition is provided by the specification of the roles and users in the system.

A proper description of roles cannot be provided without the description of their relationships, that is the role hierarchy. The inheritance relationship between roles for our sample plant is formalized in Tab. 1 while Fig. 5 shows a different representation of the same hierarchical structure. The graph in Fig. 5 can be read in two ways: permissions inheritance propagates from bottom to top while role membership propagates in the opposite direction.

We can now provide a description of the roles starting from the bottom of Fig. 5 with a generic guest role (R_{Guest}) which is the least powerful role in the hierarchy. Then, it is possible to identify roles whose responsibilities span over the different areas in the system: the cabinet networks and the process network. Roles R_{SLMB_x} and R_{PLC_x} are responsible for the low level operations in the production cycle. In particular, R_{SLMB_x} roles are focused on the ModBus/TCP slaves in the cabinets, with two levels of privileges, R_{SLMB_a} being more powerful (higher in the hierarchy) than R_{SLMB_u}. Similarly, the two roles R_{PLC_x} are related to the soft PLCs and their services. The two R_{OPCs_x} roles, instead, concern the process control area. Higher levels in the hierarchy reflect also different levels

USERS	*user assigment*	*ROLES*	*permissions assigment*	*PRMS*
P_{PLANT_a}		R_{PLANT_a}		
P_{PROC_a}		R_{PROC_a}		
P_{NET_a}		R_{NET_a}		
P_{OPCs_a}		R_{OPCs_a}		$(admin, OPCs_{PLC_1}), (admin, OPCs_{PLC_2})$
P_{PLC_a}		R_{PLC_a}		$(admin, IGs_{PLC_1}), (admin, IGs_{PLC_2})$
P_{SLMB_a}		R_{SLMB_a}		$(admin, HTTPs_{SL-MB_{11}}), (admin, HTTPs_{SL-MB_{21}}), (admin, HTTPs_{SL-MB_{12}}), (admin, HTTPs_{SL-MB_{22}})$
P_{PLC_u}		R_{PLC_u}		$(oper, IGs_{PLC_1}), (oper, IGs_{PLC_2})$ | $(login, l\_m_{PLC_1}),$
P_{OPCs_u}		R_{OPCs_u}		$(oper, OPCs_{PLC_1}), (oper, OPCs_{PLC_2})$ | $(login, l\_m_{PLC_2})$
P_{SLMB_u}		R_{SLMB_u}		$(oper, OPERs_{SL-MB_{11}}), (oper, OPERs_{SL-MB_{12}}), (oper, OPERs_{SL-MB_{21}}), (oper, OPERs_{SL-MB_{22}})$
P_{Guest}		R_{Guest}		

Table 2: RBAC assignments

of responsibilities. The R_{PROC_a} role is responsible for high level operations (management, configuration) in the process network and process control area and inherits from both R_{OPCs_a} and R_{PLC_a}. R_{NET_a}, instead, encompasses the activities related to the cabinets and thus inherits from R_{PLC_a} and R_{SLMB_a}.

A clear definition of roles and the related hierarchical structure enable us to provide an explicit permissions assignment (PA) as the one shown in Tab. 2, where each role ($ROLES$ column) is provided with the proper permissions ($PRMS$ column). Each permission is described by a pair consisting of an operation and an object already defined in Sec. 3. Tab. 2 shows that different roles are assigned permissions concerning their own areas of responsibilities: R_{PLC_u}, for instance, is given access to the *oper* action on the ISaGRAF SoftPLC server, while R_{OPCs_u} is given access to the OPC-UA server on the PLC. Since both roles need to operate locally on the PLCs, they are provided with the permission to perform the *login* operation on the PLCs' login modules ($l\_m_{PLC_i}$).

Actual *players* in the network are defined as users in the RBAC model. The left side of Tab. 2 reports the set of identified users together with the *user assignment* (UA) to roles, so that user P_{Guest} is assigned to role R_{Guest}, user P_{SLMB_u} to role R_{SLMB_u} and so on.

By combining all information coming from the user assignment, role assignment and role hierarchy, all permissions provided to the different players can be explicitly enumerated, still in a completely implementation-independent manner. The result of this enumeration is the *specification* set $\mathcal{S}$.

4.2 Implementation

In order to obtain tuples of set $\mathcal{I}$, we need both a static description of the system (devices and their physical locations, interconnections and configurations, services available on each host, etc.) and some rules able to specify, given a player state (physical location, owned credentials, etc.), the actions he/she can perform while interacting with the system as well as the possible effects of each operation on the player state itself.

4.2.1 System static description

The static description of the system has to include the elements introduced in Sec. 3 enriched with the addition of further details about the operations a player can execute such as, for example, pre-conditions that need to be satisfied in the player state before an action can actually be undertaken and post-conditions, that is, the effects of actions on the player state.

Implementation details are necessary in order to define, for example, whether a particular service can be accessed remotely or not and the gaining of access to local resources and services after a successful login operation on a given network host.

In an RBAC model objects and operations belong to disjoint sets, so that the designer can build permissions by selecting any possible (op, ob) pair. Typical industrial systems, however, usually include a lot of special objects where only ad hoc operations are meaningful, and this would lead to have an excessive number of forbidden pairs, if the two sets were considered as two disjoint entities. To cope with this aspect, we introduce the concept of *extended object*, B, by specifying the meaningful operations for each object and binding them to the object itself. This choice has, obviously, no impact on the computation of all possible meaningful tuples admitted by the system. In detail, we consider two main categories of extended objects: rooms (2) and resources on hosts (3).

$$\check{B}_r ::= \left\langle \check{\beta}_r, \left\{ \check{\Pi}_r \right\} \right\rangle \tag{2}$$

$$\check{B}_h ::= \left\langle \check{\beta}_h, \left\{ \check{\Pi}_h \right\}, \{(\check{n}, \check{g})\} \right\rangle \tag{3}$$

In both (2) and (3), $\check{\beta}$ is the object unique identifier, and $\{\check{\Pi}\}$ is the set of operations, enriched with implementation details, that are allowed on $\check{\beta}$. The possibly empty set $\{(\check{n}, \check{g})\}$ of (3) stands for the accounts associated to the resource, (e.g. accounts defined for a file server or a DBMS). For each account, $\check{n}$ and $\check{g}$ represent a username and a group respectively; we assume that usernames and groups are linked through a many-to-many relation. In particular, since $\check{n}$ and $\check{g}$ are not uniquely defined within the whole system, we will use notation $\check{\beta}_h{:}\check{n}$ to address an account on a specific resource. The structure of operations $\check{\Pi}$ will be described later.

Then the system model $\mathcal{D}$ can be defined as:

$$\mathcal{D} = \left(H, \left\{ \check{B}_r \right\}, LBS\right) \tag{4}$$

where H is the set of hosts $\{h\}$ (note that we call *host* any physical device able to provide services, e.g. a PLC, but also networking devices such as switches etc.), $\{\check{B}_r\}$ is the set of rooms as defined in (2), while *LBS* is the set of physical connections $\{lbs\}$ between the elements of H. The structure of elements in (4) is defined in Tab. 3 (in the following, the notation [.] is used for optional parameters):

- h is a *host* with unique identifier id_h, $\check{\beta}_r$ is the unique identifier of the room where the host is located, $\{pp\}$ is the set of the host physical ports, while $\{\check{B}_h\}$ is a set of extended objects belonging to the host and $\{fr\}$ are the filtering rules, in the case the host provides packets forwarding between its interfaces.
- pp is a physical port with unique identifier id_{pp}. A set of data link addresses *dla* can be associated to the port, each *dla* being bound to a set of network addresses $\{na\}$. The notation $[w,[c],\{\check{\beta}_r\}]$ is used to take into account possible wireless interfaces: w means that wireless communications are supported, $[c]$ is the credential which is possibly required to connect to the interface and $\{\check{\beta}_r\}$ is the set of rooms where the interface is accessible.
- Filtering rules $\{fr\}$ have a general form based on the elements listed in the lower part of Tab. 3.
- lbs is a bidirectional link between two physical ports.

We will not discuss the filtering rules in more detail because, for the purpose of this paper, it is enough mentioning that they are evaluated so as to check the remote availability of resources based on network reachability computations. In particular, we rely on the approach presented in [20] for the relevant computations.

Operations in the model need to be enriched with implementation information. Room operations $\check{\Pi}_r$, for instance, have the following form:

$$\check{\Pi}_r ::= \langle \check{\pi}_r, \{\langle d, \{c\}\rangle\}\rangle \tag{5}$$

where $\check{\pi}_r$ is the operation name (e.g. *enter*) whereas the set $\{\langle d,\{c\}\rangle\}$ describes all the doors that allow entering the room. Set $\{c\}$ contains the credentials needed to open door d. The adoption of a set is useful, in this case, to model situations where each player has his/her own credential to open a given door, besides circumstances where just one credential is shared among all players. Moreover,

h	::=	$\langle id_h, \check{\beta}_r, \{pp\}, \{\check{B}_h\}, \{fr\}\rangle$		
pp	::=	$\langle id_{pp}, \{\langle dla, \{na\}, [w,[c],\{\check{\beta}_r\}]\rangle\}\rangle$		
fr	::=	$\langle id_{pp_s}, id_{pp_d}, dla_s, na_s, pn_s, dla_d, na_d, pn_d, pr, act\rangle$		
	id_{pp_s}	source physical port	dla_d	dest data link addr
	id_{pp_d}	dest physical port	na_d	dest network addr
	dla_s	source data link addr	pn_d	dest port number
	na_s	source network addr	pr	protocol (TCP,UDP,...)
	pn_s	source port number	act	action (*allow* or *deny*)
lbs	::=	(id'_{pp}, id''_{pp})		

Table 3: Elements of $\mathcal{D}$

pre	::=	$\langle phy\_acc\ [c]\rangle$
		$\mid \langle loc\_acc\ \check{\beta}'_h{:}\check{n}'[c]\rangle$
		$\mid \langle loc\_acc\ \check{\beta}'_h{:}\check{g}'[c]\rangle$
		$\mid \langle rem\_acc\ port\ [c]\rangle$
port	::=	$dla \mid lp$
lp	::=	$\langle id_{lp}, [pn], na, [pr]\rangle$
post	::=	$[\check{\beta}''_h{:}\check{n}'']$

Table 4: Pre-conditions and post-conditions f

different credentials can also be assigned to the same door, depending on the direction it is actually crossed.

Similarly, $\check{\Pi}_h$ is used to describe operations a player can perform on resources on hosts and has the following form:

$$\check{\Pi}_h ::= \langle \check{\pi}_h, \{f\}\rangle \tag{6}$$

where $\check{\pi}_h$ is the operation name (e.g. *login*, *oper*, *admin*) while $\{f\}$ describes both the preconditions and possible effects of $\check{\pi}_h$ on the resource involved in the action or (possibly) on other resources. Tab. 4 shows the $f ::= pre, post$ syntax, where four different and mutually exclusive kinds of preconditions can be specified. Their semantics is, informally, the following:

- *phy_acc* $[c]$ means that a player must have physical access to host h and own credential c (if specified) in order to be able to perform $\check{\pi}_h$.
- *loc_acc* $\check{\beta}'_h{:}\,\check{n}'\ [c]$ and *loc_acc* $\check{\beta}'_h{:}\,\check{g}'\ [c]$ state that, in order to perform $\check{\pi}_h$ on $\check{\beta}_h$, a player must already be active (e.g., logged) on some object $\check{\beta}'_h$, by means of either the player's username $\check{n}'$ or the group $\check{g}'$ he/she belongs to. Note that, if h and h' are the host where $\check{\beta}_h$ and $\check{\beta}'_h$ are located respectively, in general $h' \neq h$, that is the preconditions may involve resources on different hosts. Moreover, when specified, credential c must be owned by the player.
- *rem_acc port* $[c]$ means that the operation can be carried out by a player through a remote connection at either the network or data link level (lp is a logical port whose unique identifier is id_{lp} and optional port number (e.g. 8080) pn). The player must own credential c when specified.

The element *post* in Tab. 4 takes into account the possible effects of the operation: in particular $\check{\beta}''_h{:}\,\check{n}''$, if specified, means that, by performing the operation, the player obtains a local access with username $\check{n}''$ to object $\check{\beta}''_h$. Note that, in general, $\check{\beta}''_h$ may be running on a host $h'' \neq h$.

4.2.2 Player state

The set $\mathcal{I}$ of all actions that each player can perform in the system can be computed by letting the player move around the system rooms, carrying out all possible operations $\check{\pi}$ on objects $\check{\beta}$. To this purpose, the player state $\mathcal{T}_{\check{p}}$ is defined as

$$\mathcal{T}_{\check{p}} = \left(LA, \check{\beta}_r, C_{\check{p}}\right) \tag{7}$$

In (7), $LA ::= \{(\check{\beta}_h, \check{n})\}$ is the set of local accesses already gained by player $\check{p}$, $\check{\beta}_r$ is the current position (room) of $\check{p}$ and $C_{\check{p}} ::= \{c\}$ is the set of credentials he/she owns.

4.2.3 Inference rules

Each player state $\mathcal{T}_{\check{p}}$ is combined with the system description $\mathcal{D}$ by means of a framework of *inference rules*. These rules describe how to match the capabilities of the player with the preconditions defined for the operations on the objects. A successful match corresponds to an action that can be performed by a player. For brevity reasons we omit the formal description of the inferences rules, and just mention how preconditions are evaluated: a player can enter a room if he/she is in an adjacent room and owns the required credential c; a physical access is satisfied if the player can reach the same room where the target host is located (and owns c, of course); a local access is satisfied if the player has acquired a proper access level on the object (and owns c); a remote access is satisfied if the player owns c and has access to a host that is allowed to communicate with the target object.

The key to a complete analysis of the player's actions is the evaluation of the operation's *post condition*: an operation performed by a player can, in fact, increase his/her capabilities by letting him/her reach a new room or gain a new access on a host, thus enabling *sequences* of actions. The explicit enumeration of all the actions performed by all the players leads to the *implementation* set $\mathcal{I}$.

5 Example

In the following the model described in Sec. 4 is used to describe the example plant of Sec. 3.

5.1 Plant model

The description of the set $\{\check{B}_r\}$ of extended object rooms for the considered plant, according to our model, results as in Fig. 6.1, where $c_{d_{XY}}$ are credentials a player must own to go from room R_X to room R_Y crossing door d_{XY} (see Fig. 4) and $c_{d_{C_k}}$ are credentials needed to open cabinet C_k.

$$\left\{\begin{array}{l} \langle R_O, \{\langle enter, \{\langle d_{OP}, \{c_{d_{OP}}\}\rangle\}\rangle\}\rangle, \\ \langle R_P, \{\langle enter, \{\langle d_{OP}, \{c_{d_{OP}}\}\rangle, \langle d_{PF}, \{c_{d_{PF}}\}\rangle\}\rangle\}\rangle, \\ \langle R_F, \{\langle enter, \{\langle d_{PF}, \{c_{d_{PF}}\}\rangle\}\rangle, \left\langle close, \left\{\begin{array}{l}\langle d_{C_1}, \varnothing\rangle, \\ \langle d_{C_2}, \varnothing\rangle\end{array}\right\}\right\rangle\}\rangle, \\ \langle C_1, \{\langle open, \{\langle d_{C_1}, \{c_{d_{C_1}}\}\rangle\}\rangle\}\rangle, \\ \langle C_2, \{\langle open, \{\langle d_{C_2}, \{c_{d_{C_2}}\}\rangle\}\rangle\}\rangle \end{array}\right\} \quad (1)$$

Figure 6: $\{\check{B}_r\}$ definition for the plant

The other elements of $\mathcal{D}$ and, in particular, the set of hosts H with their associated resources $\{\check{B}_h\}$ are described in the following.

As described in Sec. 3, host PLC_1 has three associated resources: a login module, an ISaGRAF server and an OPC-UA server. The login module, supporting the *login* operation, is modeled according to Fig. 7.1 and we assume there are two predefined accounts, namely a simple user and an administrator. A player can perform the *login* operation on $l\_m_{PLC_1}$ provided that he/she has physical access to cabinet C_1 and the required credentials as either user or administrator (in this case he/she gains the corresponding local access on $l\_m_{PLC_1}$). Alternatively, the player can log in remotely if he/she has user credentials. Moreover, a player who is already logged in as user can elevate his/her local privileges to administrator provided that he/she has the administrator credentials, while a player already logged in as administrator can login as simple user without any additional credential.

Analogously, the ISaGRAF server is modeled as in Fig. 7.2. In this case the *admin* operation can be performed by a player only if he/she is already logged on $l\_m_{PLC_1}$ as administrator, while *oper* can be executed by both users and administrators. Moreover, both *oper* and *admin* can be accessed remotely.

The OPC server does not have any associated account and its description is shown in Fig. 7.3. The *admin* and *oper* operations can be accessed remotely only, and *admin* can be executed by a player only if he/she owns the OPC server administrator credentials, while *oper* is allowed to players with either user or administrator credentials.

The Soft-PLC in cabinet C_2 is modeled in the same way as PLC_1, so that the ISaGRAF and OPC server descriptions are exactly the same as for PLC_1. In this case, however, *login* can be executed only remotely, since $l\_m_{PLC_2}$ is not equipped with a console.

According to the model, all slave objects share the same description. The server allowing the setup of slave $SL-MB_{kn}$, is modeled as in Fig. 8.1. In detail, the single available operation (*admin*) is remotely accessible through the logical port number 8080, provided that a player knows the default password, c_{def}. The process enabling the slave standard operation, on the other hand, is described as in Fig. 8.2. The *oper* action is also remotely accessible through the logical port number 532 and does not require any specific credential.

All switches are described in the same way; the set of associated object resources is an empty set, while the filtering rules specify that every switch just works as a hub.

Finally, all *login* operations for hosts at the process level require physical access to the relevant host and always cause the player to gain local access on the host itself. Moreover, credential $c_{u_{IGc}}$ is needed to access l_m_{IGc}, $c_{u_{OPCc}}$ for l_m_{OPCc}, $c_{u_{MBC}}$ for l_m_{MBC}, while no credential is necessary for performing *login* on l_m_{PC}.

5.2 Players initial state

To keep our example as simple as possible, we assume that exactly one player is defined for each role specified in the RBAC policies declaration. Moreover, we assume

$$\langle l\_m_{PLC_1},\quad \left\{ \left\langle login, \left\{ \begin{array}{l} \langle phy\_acc\ c_{u_{SP_a}}\ l\_m_{PLC_1}{:}u_{SP_a}\rangle, \\ \langle phy\_acc\ c_{u_{SP_u}}\ l\_m_{PLC_1}{:}u_{SP_u}\rangle, \\ \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{SP_a}} l\_m_{PLC_1}{:}u_{SP_a}\rangle, \\ \langle loc\_acc\ l\_m_{PLC_1}{:}u_{SP_a}\ l\_m_{PLC_1}{:}u_{SP_u}\rangle, \\ \langle rem\_acc\ \langle lp\_SSH_{PLC_1}, 22, IP_{PLC_1}, TCP\rangle\ c_{u_{SP_u}}\ l\_m_{PLC_1}{:}u_{SP_u}\rangle \end{array} \right\} \right\rangle \right\}, \left\{ \begin{array}{l} (u_{SP_u}, SP_u), \\ (u_{SP_a}, SP_a) \end{array} \right\} \rangle \tag{1}$$

$$\langle IGs_{PLC_1},\quad \left\{ \begin{array}{l} \left\langle admin, \left\{ \begin{array}{l} \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{IG_a}} IGs_{PLC_1}{:}u_{IG_a}\rangle, \\ \langle rem\_acc\ \langle lp\_IGs_{PLC_1}, IGs\_pn, IP_{PLC_1}, TCP\rangle\ c_{u_{IG_a}} IGs_{PLC_1}{:}u_{IG_a}\rangle \end{array} \right\} \right\rangle, \\ \left\langle oper, \left\{ \begin{array}{l} \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{IG_u}} IGs_{PLC_1}{:}u_{IG_u}\rangle, \\ \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{IG_a}} IGs_{PLC_1}{:}u_{IG_a}\rangle, \\ \langle rem\_acc\ \langle lp\_IGs_{PLC_1}, IGs\_pn, IP_{PLC_1}, TCP\rangle\ c_{u_{IG_u}} IGs_{PLC_1}{:}u_{IG_u}\rangle, \\ \langle rem\_acc\ \langle lp\_IGs_{PLC_1}, IGs\_pn, IP_{PLC_1}, TCP\rangle\ c_{u_{IG_a}} IGs_{PLC_1}{:}u_{IG_a}\rangle \end{array} \right\} \right\rangle \end{array} \right\}, \left\{ \begin{array}{l} (u_{IG_u}, IG_u), \\ (u_{IG_a}, IG_a) \end{array} \right\} \rangle \tag{2}$$

$$\langle OPCs_{PLC_1},\quad \left\{ \begin{array}{l} \left\langle admin, \left\{ \begin{array}{l} \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{OPC_a}} OPCs_{PLC_1}{:}u_{OPC_a}\rangle, \\ \langle rem\_acc\ \langle lp\_OPCs_{PLC_1}, OPCs\_pn, IP_{PLC_1}, TCP\rangle\ c_{u_{OPC_a}}\rangle \end{array} \right\} \right\rangle, \\ \left\langle oper, \left\{ \begin{array}{l} \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{OPC_u}}\rangle, \\ \langle loc\_acc\ l\_m_{PLC_1}{:}SP_u\ c_{u_{OPC_a}}\rangle, \\ \langle rem\_acc\ \langle lp\_OPCs_{PLC_1}, OPCs\_pn, IP_{PLC_1}, TCP\rangle\ c_{u_{OPC_u}}\rangle, \\ \langle rem\_acc\ \langle lp\_OPCs_{PLC_1}, OPCs\_pn, IP_{PLC_1}, TCP\rangle\ c_{u_{OPC_a}}\rangle \end{array} \right\} \right\rangle \end{array} \right\}, \varnothing\rangle \tag{3}$$

Figure 7: $\left\{\check{B}_h\right\}$ definition for PLC_1

that all players are in R_O (i.e. in the external area) at the beginning, and the sets of their local accesses are initially empty:

$$\mathcal{T}_{\check{p}} = (\varnothing, R_O, C_{\check{p}})$$

The set of credentials $C_{\check{p}}$ assigned to players in the actual implementation of the access control policies is summarized in Tab. 5. All players have credentials to cross d_{OP} as well as d_{PF}, and know the default password to access the HTTP server on the ModBus/TCP slaves, whereas only subsets of them owns credentials to open, for example, cabinets C_1 and C_2 or administrator credentials for PLC_1 and PLC_2.

$\check{p}$	*credentials*										
	$c_{dOP}, c_{dPF}, c_{def}$	c_{dC_1}, c_{dC_2}	c_{uSP_u}	c_{uSP_a}, c_{uIG_a}	c_{uIG_u}	c_{uOPC_u}	c_{uOPC_a}	c_{uIG_c}	c_{uOPC_c}	c_{uMBC}	c_{uSLMB_a}
P_{PLANT_a}	•	•		•			•	•	•	•	•
P_{PROC_a}	•	•		•			•	•	•		
P_{NET_a}	•	•		•				•		•	•
P_{OPCs_a}	•						•		•		
P_{PLC_a}	•	•		•				•			
P_{SLMB_u}	•									•	
P_{SLMB_a}	•	•								•	•
P_{PLC_u}	•	•	•		•			•			
P_{OPCs_u}	•					•			•		
P_{Guest}	•										

Table 5: Players credentials

6. Verification

Tab. 6 describes the $\mathcal{S}$ and $\mathcal{I}$ sets for the plant of Fig. 3. The two leftmost columns show all the meaningful pairs (op, ob), one for each row. Black symbols are used to indicate tuples $(\hat{u}, \hat{\pi}, \hat{\beta}) \in \mathcal{S}$, whereas white symbols are used for tuples $(\check{p}, \check{\pi}, \check{\beta}) \in \mathcal{I}$. The upper half of the table describes elements of $\mathcal{I}$ that do not have any counterpart in $\mathcal{S}$. In practice, tuples in this part take into account system operations and objects that are not involved in the specification of the RBAC policies, since they only concern the access control policies implementation in the real system. The lower half of the table mainly includes (op, ob) pairs that can be found in both $\mathcal{S}$ and $\mathcal{I}$. Here we expect the relevant portions of $\mathcal{S}$ and $\mathcal{I}$ to be exactly the same. Flaws in the implementation would reflect in cells with either a single • or a single ○, meaning, respectively, that the configuration prevents a player from performing an allowed operation or that the operation is allowed by the system but forbidden by the specified policies.

From Tab. 6 we observe that the overall situation is not completely satisfactory in the system, since the lower half of the table also includes some cells containing only one symbol. These discrepancies are highlighted with special symbols ♦, ♧, ♤, ♡.

The ♧ symbol means that the physical system allows all players to execute the *admin* operation on the HTTP server of the ModBus/TCP slaves, even if the RBAC policies explicitly authorize only players P_{NET_a}, P_{PLANT_a} and P_{SLMB_a} to do so. Of course, this is a violation according to the kind of verification of our interest. The anomaly is clearly due to the fact that *admin* can be remotely invoked through a network connection by any player who owns c_{def} (see Fig. 8.1). As a consequence, anybody who can login on a host connected to the slaves (e.g. P_{SLMB_u}) can also perform *admin*. In order to fix this problem we can change the default password to c_{SLMB} and assign it to the designated players (P_{NET_a}, P_{PLANT_a} and P_{SLMB_a}) only. Similarly, the anomaly denoted by symbol ♤, shows how players who can gain a local access to resources on hosts *MBC IGc* or *PC* (e.g. P_{OPCs_u}) can execute *oper* on the slaves. This is fixed by introducing a firewall between

$$\langle HTTPs_{SL-MB_{kn}}, \quad \left\{ \ \langle admin, \left\{ \langle rem\_acc \ \langle lp_{HTTPs_{SL-MB_{kn}}}, 8080, IP_{SL-MB_{kn}}, TCP \rangle \ c_{def} \rangle \right\} \rangle \ \right\}, \varnothing \rangle \tag{1}$$

$$\langle OPERs_{SL-MB_{kn}}, \quad \left\{ \ \langle oper, \left\{ \langle rem\_acc \ \langle lp_{OPERs_{SL-MB_{kn}}}, 532, IP_{SL-MB_{kn}}, TCP \rangle \right\} \rangle \ \right\}, \varnothing \rangle \tag{2}$$

Figure 8: $\left\{\check{B}_h\right\}$ definition for $SL-MB_{kn}$

switches SW_2 and SW_3 to allow requests from the process level to slaves only if the origin is the host MBC.

Cells marked with the ♡ symbol show that players P_{PLC_u}, P_{PLC_a} and P_{PROC_a}, can perform *oper* on the slaves, since they can gain local access on PLC_1 and PLC_2. Actually, this is not a real issue since Soft PLC users and roles inheriting from them are usually allowed to normally operate on slaves. Rather, this suggests reconsidering the related RBAC policies and adding the relevant tuples in $\mathcal{S}$ (causing a • to appear in all cells marked with ♡).

Finally, cells marked ♦ indicate that the real system does not allow the corresponding players to login on l_m_{PLC2}. This flaw directly derives from the l_m_{PLC2} description, which specifies that only players owning the $c_{u_{SP_u}}$ credentials can login there. Players owning the administrator credentials can then "upgrade" their local access. Unfortunately, P_{PLANT_a}, P_{PROC_a} and P_{NET_a} own administrator credentials only and, as a consequence, are unable to login. We fix this problem assigning them both $c_{u_{SP_u}}$ and $c_{u_{SP_a}}$ credentials.

This brief comparison has put into evidence some possible discrepancies between *specification* and *implementation* parts of the model and offered some insights on the possible ways to fix the detected problems. It is worth reminding that no anomaly analysis method has been discussed as this falls outside the scope of this paper.

7. Conclusions

The design and verification of access control policies in IDCS demand for new techniques able to cope with a number of characteristics, that are peculiar to most situations where a physical process is interfaced and controlled through a distributed infrastructure. The difficulty or even impossibility to modify the architecture of the special-purpose hardware and software frequently adopted in industrial systems, their real-time requirements and their long lifespan (i.e. decades) with respect to that of many IT technologies (i.e., few years), in fact, make the existing appealing solutions, developed for other application fields, hard to apply to industrial systems.

The goal of this work is exploiting the advantages of the well known RBAC framework for high level policies declaration as well as taking into account the low-level security mechanisms, typical of industrial systems (that, although very poor, are often the only elements that can be used to implement the policies) in order to verify the correctness of policies implementation. To the best of our knowledge, several other approaches appeared in the literature have dealt very extensively with the former aspect, but no attempt has been carried out, till today, to include also the description of the physical system in the model as we have done in our proposal. This approach, in fact, does not directly reduce the efforts needed for defining and managing access policies but can help in the validation and verification process.

The model presented in this paper is based on the idea of considering the two views of the system, specification and implementation, as separated entities and to verify the correctness of policies implementation by comparing the results of the two descriptions. What we have presented in this paper is an informal definition of the two parts of the model as well as a real system modeling example in order to prove that the model is suitable for policies declaration as well as for the description of low-level implementation details in industrial systems.

Clearly this is just the first necessary step of our work, which is aimed at the development of an automated analysis tool able to help with policies declaration and to perform the verification analysis. Ongoing activities will include the definition of analysis algorithms for verifying whether the specification part of the model correctly matches the implementation part and, finally, the design and development of an automated analyzer prototype.

References

[1] M. Cheminod, L. Durante, and A. Valenzano, "Review of Security Issues in Industrial Networks", *IEEE Trans. Ind. Informat.*, vol. 9, no. 1, pp. 277–293, 2013.

[2] ANSI INCITS 359-2012, *Role Based Access Control*, 2012.

[3] M. Cheminod, I. Cibrario Bertolotti, L. Durante, and A. Valenzano, "Automatic Analysis of Security Policies in Industrial Networks", in *Proc. of the 8th IEEE Int. Wksp. on Factory Communication Systems*, WFCS, 2010, pp. 109–118.

[4] I. Cibrario Bertolotti, L. Durante, T. Hu, and A. Valenzano, "A Unified Class Model for Checking Security Policies in ICT Infrastructures", in *Proc. of the IEEE First AESS European Conf. on Satellite Telecommunications*, ESTEL, 2012, pp. 1–6.

[5] I. Cibrario Bertolotti, L. Durante, T. Hu, and A. Valenzano, "A Model for the Analysis of Security Policies in Industrial Networks", in *Proc. of the 1st Int. Symp. for ICS & SCADA Cyber Security Research*, ICS-CSR, 2013, pp. 66–77.

[6] I. Cibrario Bertolotti, L. Durante, and A. Valenzano, "A Twofold Model For The Description of Access Policies in Industrial Systems, CNR-IEIIT Technical Report (in review)", 2013, pp. 1–20.

	π	β	users/players									
			P_{Guest}	P_{NET_a}	P_{OPCs_a}	P_{OPCs_u}	P_{PLANT_a}	P_{PROC_a}	P_{SLMB_a}	P_{SLMB_u}	P_{PLC_a}	P_{PLC_u}
System only	*open*	C_1, C_2		○			○	○	○		○	○
	close	R_F		○			○	○	○		○	○
	enter	R_F, R_O, R_P	○	○	○	○	○	○	○	○	○	○
	login	l_m_{IGc}		○			○	○			○	○
	login	l_m_{MBC}		○			○		○	○		
	login	l_m_{OPCc}			○	○	○	○				
	login	l_m_{PC}	○	○	○	○	○	○	○	○	○	○
System & RBAC	*admin*	$HTTPs_{SL-MB_{11}}$	♧	○ ●	♧	♧	○ ●	♧	○ ●	♧	♧	♧
	admin	$HTTPs_{SL-MB_{12}}$	♧	○ ●	♧	♧	○ ●	♧	○ ●	♧	♧	♧
	admin	$HTTPs_{SL-MB_{21}}$	♧	○ ●	♧	♧	○ ●	♧	○ ●	♧	♧	♧
	admin	$HTTPs_{SL-MB_{22}}$	♧	○ ●	♧	♧	○ ●	♧	○ ●	♧	♧	♧
	admin	IGs_{PLC_1}		○ ●			○ ●	○ ●			○ ●	
	oper	IGs_{PLC_1}		○ ●			○ ●	○ ●			○ ●	○ ●
	admin	IGs_{PLC_2}		○ ●			○ ●	○ ●			○ ●	
	oper	IGs_{PLC_2}		○ ●			○ ●	○ ●			○ ●	○ ●
	admin	$OPCs_{PLC_1}$			○ ●		○ ●	○ ●				
	oper	$OPCs_{PLC_1}$			○ ●	○ ●	○ ●	○ ●				
	admin	$OPCs_{PLC_2}$			○ ●		○ ●	○ ●				
	oper	$OPCs_{PLC_2}$			○ ●	○ ●	○ ●	○ ●				
	oper	$OPERs_{SL-MB_{11}}$	♤	○ ●	♤	♤	○ ●	♡	○ ●	○ ●	♡	♡
	oper	$OPERs_{SL-MB_{12}}$	♤	○ ●	♤	♤	○ ●	♡	○ ●	○ ●	♡	♡
	oper	$OPERs_{SL-MB_{21}}$	♤	○ ●	♤	♤	○ ●	♡	○ ●	○ ●	♡	♡
	oper	$OPERs_{SL-MB_{22}}$	♤	○ ●	♤	♤	○ ●	♡	○ ●	○ ●	♡	♡
	login	l_m_{PLC2}		○ ●			○ ●	○ ●			○ ●	○ ●
	login	l_m_{PLC2}		♦			♦	♦			♦	○ ●

Table 6: $\mathcal{S}$ and $\mathcal{I}$

[7] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-Based Access Control Models", *IEEE Computer*, vol. 29, no. 2, pp. 38–47, 1996.

[8] M. Harrison, W. Ruzzo, and U. J.D., "Protection in Operating Systems", *Communications of the ACM*, vol. 19, no. 8, pp. 461–471, 1976.

[9] L. LaPadula and D. Bell, "Secure Computer Systems: A Mathematical Model, Technical Report ESD-TR-278, vol. 2, The Mitre Corp., Bedford, MA", 1973.

[10] K. Biba, "Integrity Considerations for Secure Computer Systems, Technical Report TR-3153, The Mitre Corp., Bedford, MA", 1977.

[11] V. C. Hu, E. Martin, J. Hwang, and T. Xie, "Conformance Checking of Access Control Policies Specified in XACML", in *Proc. of the 31st IEEE Annual Int. Computer Software and Applications Conf.*, volume 2 of *COMPSAC*, 2007, pp. 275–280.

[12] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and Change-Impact Analysis of Access-Control Policies", in *Proc. of the 27th ACM Int. Conf. on Software Engineering*, ICSE, 2005, pp. 196–205.

[13] M. Masood, A. Ghafoor, and A. Mathur, "Conformance Testing of Temporal Role-Based Access Control Systems", *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 2, pp. 144–158, 2010.

[14] G. Faden, "RBAC in UNIX Administration", in *Proc. of the 4th ACM Wksp. on Role-based access control*, RBAC, 1999, pp. 95–101.

[15] T. L. Hinrichs, D. Martinoia, W. C. Garrison, A. J. Lee, A. Panebianco, and L. Zuck, "Application-Sensitive Access Control Evaluation Using Parameterized Expressiveness", in *Proc. of the 26th IEEE Symp. on Computer Security Foundations*, CSF, 2013, pp. 145–160.

[16] A. Cau, H. Janicke, and B. Moszkowski, "Verification and enforcement of access control policies", *Formal Methods in System Design*, , pp. 1–43, 2013.

[17] D. M. Nicol, W. H. Sanders, S. Singh, and M. Seri, "Usable Global Network Access Policy for Process Control Systems", *IEEE Security Privacy*, vol. 6, no. 6, pp. 30–36, 2008.

[18] D. M. Nicol, W. H. Sanders, M. Seri, and S. Singh, "Experiences Validating the Access Policy Tool in Industrial Settings", in *Proc. of the 43rd IEEE Hawaii Int. Conf. on System Sciences*, HICSS, 2010, pp. 1–8.

[19] H. Okhravi, R. H. Kagin, and D. M. Nicol, "PolicyGlobe: A Framework for Integrating Network and Operating System Security Policies", in *Proc. of the 2nd ACM Wksp. on Assurable and usable security configuration (SafeConfig)*, SafeConfig, 2009, pp. 53–62.

[20] A. Liu and A. Khakpour, "Quantifying and Verifying Reachability for Access Controlled Networks", *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 551–565, 2013.