DOI: http://dx.doi.org/10.14236/ewic/ics-csr2014.6

Automated Analysis of Access Policies in Industrial Plants

Manuel Cheminod, Luca Durante, Lucia Seno, Adriano Valenzano
National Research Council of Italy – IEIIT
c.so Duca degli Abruzzi 24
I-10129 Torino
Italy
www.ieiit.cnr.it
{*manuel.cheminod, luca.durante, lucia.seno, adriano.valenzano*} *@ieiit.cnr.it*

Access control policies are a key element for designing and implementing secure industrial distributed systems. A new technique for checking the correct mapping of high-level access policies onto low-level access control mechanisms, which are included in the actual system, is presented in this paper by means of a case study. The proposed approach has been adopted for developing an automated analysis software tool, whose main characteristics are also discussed.

Keywords: Access control, RBAC, access policies, automated analysis, model checking

1. INTRODUCTION

Nowadays, industrial networked systems are exposed to the same security treats affecting general-purpose office/enterprise networks. In the former case, however, attacks can also have consequences on safety with dramatic damages to both the environment and human life.

Addressing the security of industrial networks efficiently is of paramount importance but, unfortunately, some peculiarities of industrial environments prevent the adoption of the popular *patch & fix* approach, which is frequently followed in many other application areas. In particular, on the one hand, authoritative and specific standard documents (ANSI/ISA (2009); ISO/IEC (2008); NIST (2010); Stouffer et al. (2008)) state that security is a process, which continuously affects the system since the early design phase till the end of its life cycle. On the other hand, some functional/performance requirements and the characteristics of industrial devices demand for special care in adopting security mechanisms, which are usually greedy for communication bandwidth and computing power, such as cryptography and sophisticated authentication techniques (Cheminod et al. (2013)).

Access control is a core element around which a secure system can be built and, to this purpose, the methodology based on (Cibrario Bertolotti et al. (2013)), presented in (Valenzano (2014)) and tested in the development of some relevant case studies (Cheminod et al. (2014)), enables modelling both the high level access control policies (*specification*), and the physical system with its low-level access mechanisms and configurations (*implementation*). An important step is then checking whether the *implementation* matches the *specification*. In fact, the real-world systems considered in this paper are not powerful enough to implement high level access control policies automatically by means of some special-purpose policy enforcement support, as assumed in most literature works (Hu et al. (2007); Masood et al. (2010); Hinrichs et al. (2013); Cau et al. (2013)). Actually, many industrial networked systems (and their devices) have been designed and implemented without security requirements in mind. Moreover, sophisticated techniques for granting confidentiality and/or authentication, such as cryptography, usually require much more CPU power and communication bandwidth than resources offered by many of these devices and systems. In the following we assume that only the simple and traditional low-level access control mechanisms are available in the system. This leads to the not trivial task of checking whether their actual configuration and global cooperation are able to satisfy the high level access control policy requirements correctly.

In particular, the following types of raw mechanisms are taken into account:

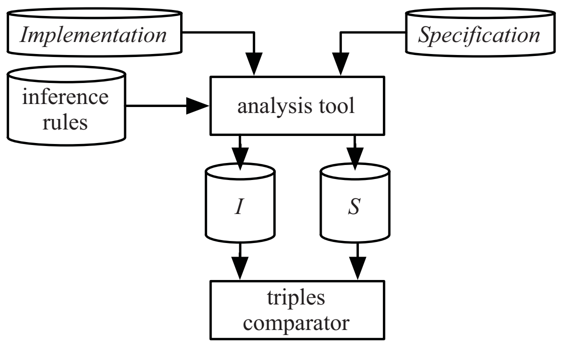


Figure 1: *Automated Analyzer Workflow*

- physical containment, such as rooms and cabinets possibly requiring some credential (key, password, badge) to be accessed;
- network reachability, managed through active network devices (e.g. firewalls, routers and switches) properly configured by means of suitable rules;
- local and remote access rights for system resources based, as usual, on *user* and *group* accounts and their associated *permissions* and/or *capabilities*.

Our system model includes a static description of the mechanisms listed above. The *implementation* view is then integrated with a specification of the effects of the mechanisms on the user behaviour (inference rules).

The block architecture of the whole analysis framework is depicted in Fig. 1. The sets of all the allowed users'actions (i.e. *who is allowed to do what to what*) consisting of triples (*user*, *operation*, *object*) are computed for both the specification and implementation views. These two sets ($\mathcal{S}$ and $\mathcal{I}$ respectively) are then compared, looking for discrepancies which represent mismatches between the high level access control policies and the actual system implementation. The sequence of all users' actions in the *implementation* can also be taken into account to simplify the identification of the causes of mismatch and to help fixing them.

As far as we know very few approaches, appeared in the literature, rely on similar assumptions (Nicol et al. (2008); Okhravi et al. (2009); Nicol et al. (2010)). Moreover, these techniques either deal with network devices only or assume a certain degree of homogeneity for system devices.

This paper presents an example of automated specification and analysis making use of a software tool which is based on the framework mentioned above. The tool integrated graphical user interface (GUI), in particular, helps the security engineer in developing both the model *specification* and *implementation* views, and provides support for the analysis and evaluation of the results.

The paper is organized as follows: Section 2 introduces the example plant, while Section 3 discusses the model development by means of the s/w tool interface. Section 4 presents the access policy analysis for the case study and, finally, Section 5 draws some conclusions.

2. CASE STUDY OVERVIEW

The case study presented in this paper is extracted from a real-world scenario of a fully-automated plant for the production of sweets and chocolates. In particular, the core of the plant is a production line which transforms raw materials into final products after several processing steps. Different steps are performed, as usual in this kind of environments, by different automation "cells", each one specialized in carrying out a particular sequence of operations. From an architectural point of view, however, cells are very similar as they are based on the same types of devices and technical solutions, so for our purposes we will focus on two main representative kinds of cell only.

Fig. 2 shows the basic architecture of the plant, as it is displayed by the graphical interface of our automated analysis software tool. White boxes in Fig. 2 are physically-delimited areas, such as rooms or cabinets.

The `LOBBY` rectangle on the left side of the picture represents the entrance to the plant area, and is accessible to all the system users (employees, operators, maintainers, guests and so on). A door connects the `LOBBY` area to the adjacent `OFFICE` zone, where several networked workstations are located and available for use by the company employees (of course all rooms have one or more doors but, for the sake of simplicity, the picture shows only those elements which are of interest for this paper).

The bottom half of Fig. 2 describes two different automation cells. Each cell consists of three rooms called `SUPERVISION`, `CONTROL` and `FIELD` respectively. Actually, devices belonging to the same automation cell are spread over three distinct areas, which are kept logically (and sometimes physically) distinct to take into account differences in functionality and usage. For instance, sensors and actuators are located in the control and field areas: in particular, the conventional choice of placing those devices devoted to I/O functions in the field

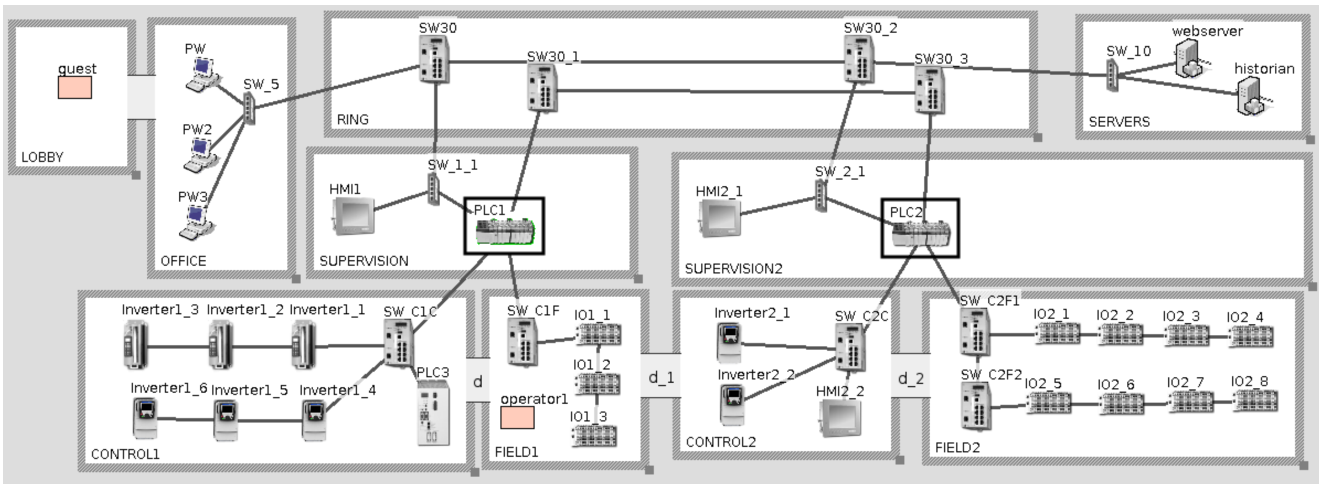


Figure 2: *Case study*

area has been adopted in this case, while other equipment (such as variable frequency drivers) is installed in the control zone. Operations of control and field devices are monitored and managed in the `SUPERVISION` zone, where Programmable Logic Controllers (PLCs), Human Machine Interfaces (HMIs) and special-purpose workstations can be found.

Some PLCs are actually the control core elements of cells (rectangular boxes in the `SUPERVISION1` and `SUPERVISION2` rooms) and their four main network connections are needed to enable the correct execution of their tasks. In fact, a cell "supervisor" PLC can access devices in the control, field and supervision areas through three different networks (control, field and supervision network respectively). Moreover, a fourth link (process network) enables each supervisor node to exchange data with its peers for synchronization and coordination purposes. For reliability and fault-tolerance reasons, the process network topology is ring-shaped and includes switches and traffic control devices appearing in the `RING` area in Fig. 2. All the cells in the plant are connected to the process network infrastructure. Also the supervision network has a ring topology, however this is not shown in the picture. Moreover, it is worth noting that supervisor PLCs are the only kind of devices which are connected to all the (four) networks included in the plant.

The `SERVERS` room, appearing in the upper right corner of the plant layout, is devoted to accommodate several host computers for shared services (e.g: web servers, historians, mail servers and so on) and can be accessed, in principle, from both the office and cell environments.

Finally, it is worth noting that the plant view, offered by the automated tool GUI, includes both the plant physical topology (rooms, cabinets, doors) and the communication infrastructure layout (i.e. cables and devices).

3. MODELLING PHASE

Our analysis framework is based on the development of a twofold system model (Valenzano (2014)) that can be specified by means of a suitable modelling language. The language provides a means to describe devices, rooms and their connections, network connections and so on. The description by hand of a complex system, however, is generally a rather difficult and error-prone process. For this reason, a GUI application (partially still under development) enables the user to graphically describe a whole system in details.

Some features of the tool GUI have already been mentioned in describing the plant layout of Fig. 2.

Most industrial systems we have dealt with in the past, were based on a restricted number of different device types. Of course this also depends, to some extent at least, on several factors such as the total number of devices building up the system, the complexity/cost for managing devices from different vendors and so on. Generally speaking, the introduction of a new type of device in an industrial environment is not a simple step, in particular when the complete interoperability with the pre-existing environment has to be granted. To simplify the designer's task, our tool offers the possibility of specifying "template" objects, so that basic blocks in the system can be modelled with devices which are representative for a whole category of equipment. For instance, the model for the case study in Fig. 2

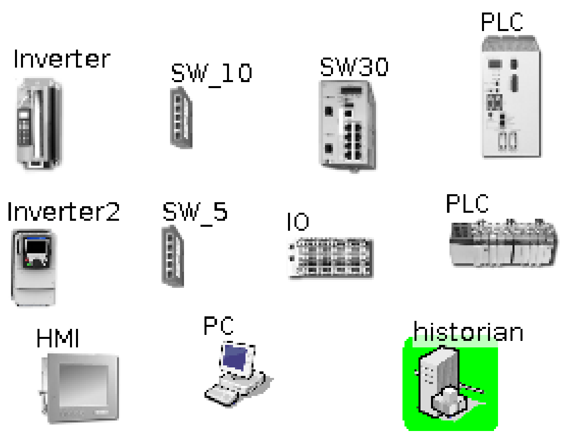


Figure 3: *Device templates*

is based on the small set of objects shown in Fig. 3. Then, the tool allows the designer to build a system by connecting complex, pre-specified blocks. Also room objects, not shown in Fig. 3, are derived from a template, which is simpler than the host one because a room can defined by means of few elements. For instance, in the following textual specification:

```
room  <CONTROL2,<enter,<d_1,{}>,<d_2,{}>>>
room  <FIELD1,<enter,<d,{}>,<d_1,{}>>>
```

names (`CONTROL2` and `FIELD1`) identify the corresponding room objects in Fig. 2. Both rooms have two doors (`d_1`, `d_2`, and `d`, `d_2` respectively). This also implicitly means that `d_1` connects `CONTROL2` and `FIELD1`, whereas `d` is shared by `CONTROL1` and `FIELD1` and, finally, `d_2` is placed between `CONTROL2` and `FIELD2` (see Fig. 2). Each empty field {}, which appears in the specification immediately after a door name, states that no particular credential (key, badge etc.) is needed to cross the related object (in the opposite case the required credential should be specified there). During the analysis phase, the position of each user in the plant is considered, as it affects the operations he/she is able to carry out on devices. For this reason, also the physical placement of devices in the system is stored in the model description.

As discussed in (Valenzano (2014)), the system model includes two different views called specification and implementation respectively. The analysis tool enables the designer to build both of them starting from scratch if needed.

3.1. Implementation view

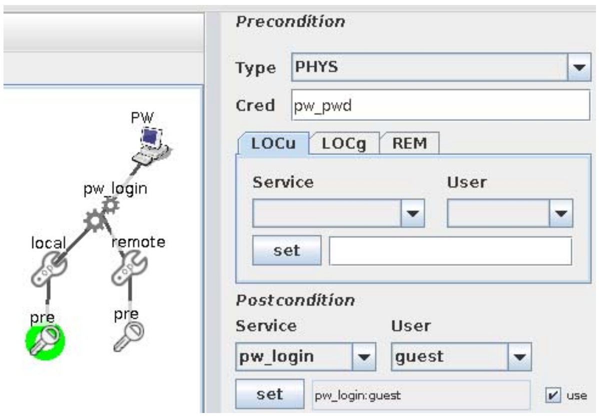
The implementation view of the model is based on a number of object types (i.e. hosts, rooms, doors, network connections, services and so on) and their relationships. The description of a host, for instance, includes physical characteristic (network interfaces, room where the host is located) and services the host provides. As already mentioned in Section 1, the implementation view is needed to compute the user behaviour in the system. Users are elements of the model and have to be described as well. The typical user state description, in this case, includes his/her unique identifier, the set of assigned credentials (passwords, badges, certificates and so on), the set of login operations he/she has already performed successfully, and the room where he/she is currently located. It is worth noting that the last two pieces of information can change dynamically as the user interacts with the system (i.e. by logging on some host, by crossing a door, etc.).

To give the reader a flavour of the tool descriptive capabilities, let us now focus on three main representative object types in the plant, that is personal workstations, PLCs, and HMI devices.

Personal Workstations (PW)
Standard, commercial devices are adopted in the system to enable access to several functionalities by different categories of users. For instance, conventional personal computers (PCs) are configured to let users log into the system and gain access to the PC local resources through the operating system services.

The model implementation view corresponding to this situation is shown in Fig. 4. In the picture `PW` provides a `pw_login` service, which can be invoked through two distinct operations labelled `local` and `remote` respectively. This takes into account the common condition where the log in operation can be carried out either locally (physical access to the `PW` console) or remotely (via some available network connection). In our model, each possible operation is associated to a precondition, which must be evaluated as true in order to enable the execution of the operation itself. For instance, Fig. 4 shows that the type of precondition for the local login operation is `PHYS`, that is the user needs gaining physical access to the `PW` console (i.e. he/she must be in the same room where the host is placed) to log in. Moreover, a credential (a password) is needed to complete the operation successfully (`pw_pwd` in the upper right part of Fig. 4).

Operations also have related postconditions, describing the effect of the operation execution. For instance, a successful login in Fig. 4 enables the user to access the operating system services with username `guest` and associated rights. It is worth noting that, in order to keep the example simple, only one user is shown in the picture, but the modelling framework allows for the detailed specification of different user and group accounts associated with the service.



Precondition

Type: PHYS
Cred: pw_pwd

Service	User

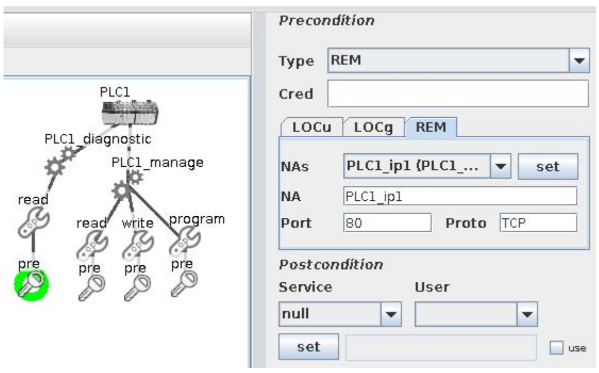
set

Postcondition

Service	User
pw_login	guest

set pw_login:guest ☑ use

Figure 4: *Modeling of PW services*



Precondition

Type: REM
Cred:

NAs: PLC1_ip1 (PLC1_... set
NA: PLC1_ip1
Port: 80 Proto: TCP

Postcondition

Service	User
null	

set ☐ use

Figure 5: *Modeling of PLC services*

Programmable Logic Controllers

PLCs are another interesting case of device modelling. PLCs, in fact, are becoming more and more powerful and capable of providing several kinds of services, and many alternatives can be found on the market with characteristics and performance ranging in a very broad interval. For the purpose of this paper, however, we do not take into account differences between the low-level functionalities of such devices, instead we focus on their high-level capabilities and available security features. As already mentioned, a PLC is used in our plant as the basic cell controller. Services offered by this kind of device include configuration, administration and diagnostic functionalities.

Fig. 5 shows an example of diagnostic service description (`PLC1_diagnostic`). The service can be invoked only remotely through the IP address of one of the PLC network interfaces. Most PLCs are able to offer diagnostic information via a simple embedded web server, which can be easily accessed by issuing requests to a suitable TCP protocol port. No credential is required, in this case, to perform the operation, as shown in Fig. 5 where the field "Cred" has been left empty.

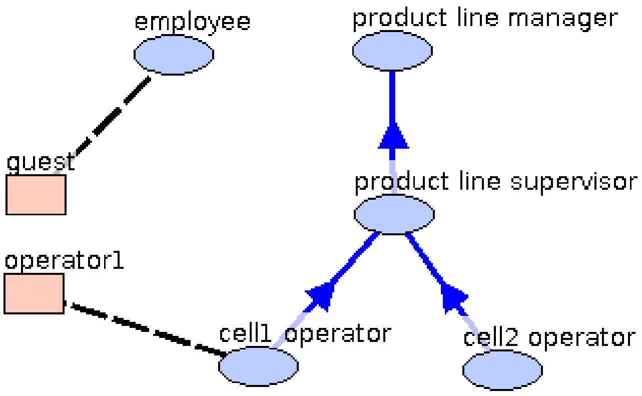


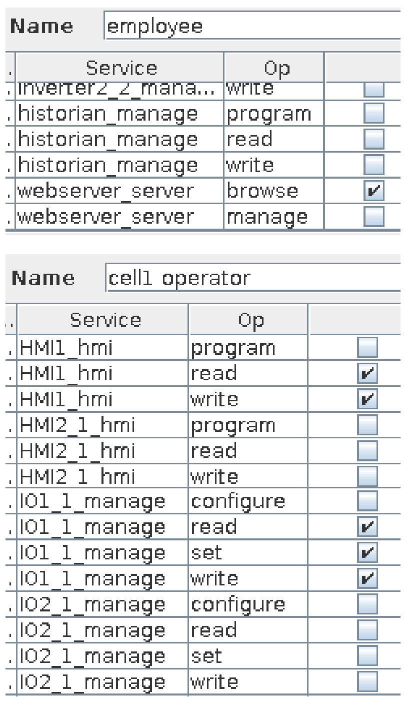
Figure 6: *Role hierarchy*

The `read` and `write` operations of the service `PLC1_manage`, appearing in the left part of the picture, are specified following the same approach as before, even though with some slight differences concerning the port and communication protocol involved in this case (the correct values for these parameters usually depend on the software application used to access the PLC management service).

Finally, the `program` operation, also shown in the picture, allows the user to upload part-programs to the device and has a similar description, which foresees a password knowledge for invoking the action. It is worth noting that, even today, most PLCs available on the market, do not provide a more sophisticated authentication support, but rely on the knowledge of simple passwords. Nevertheless, our tool is able to model more advanced security mechanisms when they are included in the devices of interest.

Human Machine Interfaces

HMIs are equipments used all over the plant to help operators keeping track of production data, reading/setting parameter values, changing operating modes for devices and so on. Different types of HMIs are adopted to this purpose, ranging from very simple products (basically raw displays) used to show read-only values, to complex and powerful devices equipped with network connections, dynamic interfaces and support for security. Also in this case, we are particularly interested in the last aspect and our modelling approach is able to take into account different security capabilities as needed. In the following we consider HMIs which are able to offer management services at two different levels: at the field level the user can physically interact with the touch screen to read/write some basic values from/to the field devices while, at the supervision level, the HMI services can be configured/reconfigured remotely, provided that the user owns the proper credentials.



Name employee

Service	Op	
inverter2_2_mana...	write	☐
historian_manage	program	☐
historian_manage	read	☐
historian_manage	write	☐
webserver_server	browse	✓
webserver_server	manage	☐

Name cell1 operator

Service	Op	
HMI1_hmi	program	☐
HMI1_hmi	read	✓
HMI1_hmi	write	✓
HMI2_1_hmi	program	☐
HMI2_1_hmi	read	☐
HMI2_1_hmi	write	☐
IO1_1_manage	configure	☐
IO1_1_manage	read	✓
IO1_1_manage	set	✓
IO1_1_manage	write	✓
IO2_1_manage	configure	☐
IO2_1_manage	read	☐
IO2_1_manage	set	☐
IO2_1_manage	write	☐

Figure 7: *Assignment of permissions to roles*

3.2. Specification view

As mentioned before, the complete system model also includes a specification view, that is the description of "who can do what to what" at a high abstraction level (access policy level). In (Cheminod et al. (2014)) we discussed how an RBAC-based approach (Sandhu et al. (1996); ANSI INCITS 359-2012 (2012)) can help in achieving this goal. In this paper we do not provide explicit RBAC syntax and definitions. However, the tool screenshots in Fig. 6, Fig. 7, and Tab. 3.2 contain the RBAC definitions of the specification view. In particular, Fig. 6 is based on the RBAC $ROLES$, *roles hierarchy* RH, and *users assignment to roles* US. Fig. 7, instead, shows *permissions* $PRMS$ (i.e. pairs $\langle$*object, operation*$\rangle$) and *permissions assignment to roles* PA. All PAs are then listed in Tab. 3.2. As a first step, roles have to be identified for all the users who have to carry out some task in the system. Moreover, suitable permissions have to be assigned to roles, taking advantage of the role hierarchy that supports permission inheritance in a straightforward and powerful way. Two distinct logical domains can be identified in the plant case study, concerning typical office and automation field activities respectively. The simple schema shown in Fig. 6 (where roles are represented by ovals and users by squares, respectively) allows the definition of both the main roles involved in the design and their hierarchical organization. For instance, the generic `employee` role takes into account those users needing to

Role	Object	Operation
cell1 operator	HMI1_hmi	read/write
	inverter1_1_manage	read/write
	IO1_1_manage	read/write
	IO1_1_manage	set
	PLC1_diagnostic	read
cell2 operator	HMI2_1_hmi	read/write
	inverter2_1_manage	read/write
	IO2_1_manage	read/write
	IO2_1_manage	set
	PLC2_diagnostic	read
product line supervisor	PLC1_manage	read/write
	PLC2_manage	read/write
product line manager	PLC1_manage	program
	PLC2_manage	program
	historian_manage	program
	historian_manage	read/write
employee	webserver_server	browse

Table 1: *Permission assignment*

access the office workstation and web server data for performing their tasks. The upper part of Fig. 7 shows how permissions can be assigned to the `employee` role, by selecting the proper checkboxes on the right.

People having production-related responsibilities, are assigned to a different hierarchy of roles, partly reflecting the structure of the physical system. `cell1 operator` and `cell2 operator`, at the bottom of the hierarchy depicted in Fig. 6, are responsible for routine operations of the cell machines: this involves checking/adjusting basic parameter values and monitoring the cell behavior through one or more HMIs. The two roles differ only for their associated cells, and the lower part of Fig. 7 shows an example of permission assignment to `cell1 operator`.

The `product line supervisor` role is placed at an intermediate level in the hierarchy. Supervisors, in fact, are responsible for the correct operation of the whole line and, for this reason, are assigned more powerful permissions, involving the coordination and cooperation of the activities of different cells (PLCs).

At the highest level in the hierarchy, users belonging to `product line manager` role should be able to reconfigure and upload configurations and programs to the line programmable devices, in order to quickly adapt to dynamic requirements and changes of the whole production.

The assignment of permissions to roles for our running example is summarised in Tab. 3.2. For the sake of conciseness, only one instance is shown in the table for all devices configured in the same way.

On the contrary, rooms have not been taken into account at the specification level since we consider them as low level security mechanisms, and thus neither listed in Tab. 3.2, and consequently nor mentioned in set $\mathcal{S}$ for any user. Set $\mathcal{S}$ is shown in Tab. 4.2.

The specification model view can then be completed by describing the assignment of actual users to roles.

Roles are abstract entities for the system, while users correspond to physical persons. The tool GUI enables the description of users in the same (specification) view adopted for roles identification, as highlighted by the square boxes labelled `guest` and `operator1` in Fig. 7. Dashed lines appearing in the picture represent binding operations and, in particular, the assignment of user `guest` to the `employee` role and of user `operator1` to cell operators.

The definition of users also puts into correspondence a subset of objects in the specification and implementation views, since users are a key element of the system model for analyzing access policies. Users defined through the application window in Fig. 7 are automatically added to the view already introduced in Fig. 2, and the designer can set their initial position by simply dragging the relevant items to the selected locations. For instance, Fig. 2 shows that user `guest` is initially in the `LOBBY` area, while `operator_1` is in the field area of cell 1 (`FIELD1`). In a similar way, the designer can assign the proper set of credentials to each user (`pw_pwd` to `guest`, and no credential to `operator1`). This assignment, together with the initially empty set of acquired local accesses (successful logins), complete the user's state information.

4. ANALYSIS PHASE

The automated tool stores the system model and its details, using an internal compact format which is not directly ready for the analysis (for instance, it also contains information about the graphical layout). After a pre-processing step, a more convenient representation of the specification and implementation views is extracted and fed to the analysis module, which computes the two sets of triples $\mathcal{S}$ and $\mathcal{I}$. In practice, this representation describes the system by means of the formal language presented in (Cibrario Bertolotti et al. (2013)). The following fragment is an example of formal description for host `PLC1`, which is located in the `SUPERVISION1` room, is equipped with four physical ports and offers two services, `PLC1_manage` and `PLC1_diagnostic`, as sketched in Fig. 5:

```
host <PLC1, supervision,
    <PLC1_pp_1, <PLC1_mac1, PLC1_ip1>>,
    <PLC1_pp_2, <PLC1_mac2, PLC1_ip2>>,
    <PLC1_pp_3, <PLC1_mac3, PLC1_ip3>>,
    <PLC1_pp_4, <PLC1_mac4, PLC1_ip4>>;
         ,
<PLC1_diagnostic,
   <read, <rem_access <PLC1_ip1,'80','TCP'> {}>>
  , {} > // no account defined on object PLC1_diagnostic
 ,
<PLC1_manage,
   <read, <rem_access <PLC1_ip1, '8888', 'TCP'> {}>> ,
   <write, <rem_access <PLC1_ip1, '9999', 'TCP'> {}>> ,
   <program, <rem_access <PLC1_ip1> plc_sw>>
   , {} > // no account defined on object PLC1_manage
 , {} >; // no filtering rules available for PLC1
```

The `read` operation is accessible through a network connection (`rem_acc` in the formal description) to the TCP address `PLC1_ip1`, port `80`. No credential is required to do this, whereas the operation `program` for service `PLC1_manage` requires the credential `plc_sw`.

4.1. Implementation check

The analysis module computes the whole set of $\mathcal{I}$ triples, by using suitable *inference rules* which enable deducing what a user can do, starting from his/her initial state. The inference rule evaluation involves checking several conditions (e.g. required credential owned/not owned by the user, user login status for relevant hosts, network connectivity between the user and remote services). It is worth noting that, in our case, the network connectivity takes into account both the physical and logical topology, and the latter depends on the filtering rules defined for each network device.

Beginning with the user's initial state, the analysis module keeps track of the available operations, by building a graph structure (called Labelled Transition System - LTS) in which nodes represent the possible user states, and edges the action(s) (each one in the form of an <object, operation> pair) that can be performed in the current state. A triple (<user, object, operation>) belonging to $\mathcal{I}$ is simply obtained by adding the user to the action pair. When the user can perform more operations which do not affect his/her current state, the same edge is labelled with all pairs corresponding to the relevant actions. Conversely, since some operations are able to modify the user state (i.e. entering a room changes the user physical location, a successful login operation gives the user new rights) the LTS construction algorithm iterates over any new created state, building the graph as usual.

A very simple example of LTS concerning user `guest` is shown in Fig. 8. The user initial state is painted gray in picture: `guest` begins his/her activity when he/she is in the `LOBBY` area. From the lobby he/she

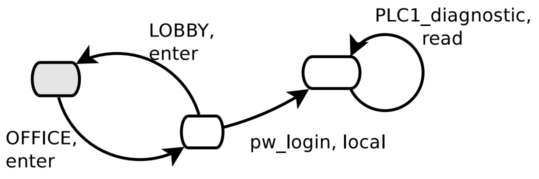


Figure 8: *guest LTS subset*

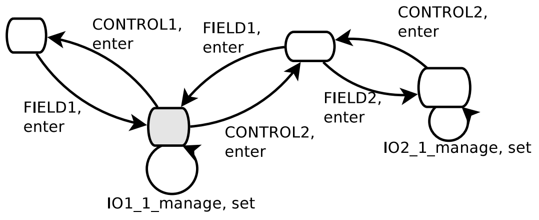


Figure 9: *operator1 LTS subset*

can reach the `OFFICE` room and carry out the `local` operation on the `pw_login` object.

Another simple example is depicted in Fig. 9, where the `operator1` user, starting in room `FIELD1`, is able to reach the `CONTROL2` room and carry out the `enter` action.

A selected subset of $\mathcal{I}$ elements, involving users `operator1` and `guest`, is shown in Tab. 4.1.

User	Object	Operation
operator1	IO1_1_manage	set
operator1	IO2_1_manage	set
guest	historian_manage	read/write
guest	inverter1_1_manage	read/write
guest	inverter2_1_manage	read/write
guest	IO1_1_manage	read/write
guest	IO2_1_manage	read/write
guest	pw_login	local
guest	PLC1_diagnostic	read
guest	PLC2_diagnostic	read
guest	PLC1_manage	read/write
guest	PLC2_manage	read/write
guest	webserver_server	browse

Table 2: `operator1`*'s and* `guest`*'s sets* $\mathcal{I}$

4.2. Specification check

The computation of the set $\mathcal{S}$ is simpler than $\mathcal{I}$, since only the user-to-role and permission-to-role assignment relationships have to be considered (see Fig. 6, Fig. 7, and Tab. 3.2), besides the role hierarchy and related permission inheritance. For instance, Tab. 4.2 shows some triples obtained for users `guest` and `operator1`.

User	Object	Operation
operator1	PLC1_diagnostic	read
operator1	inverter1_1_manage	read/write
operator1	HMI1_hmi	read/write
operator1	IO1_1_manage	set
operator1	IO1_1_manage	read/write
guest	webserver_server	browse

Table 3: `operator1`*'s and* `guest`*'s sets* $\mathcal{S}$

4.3. Comparison and evaluation

The sets $\mathcal{I}$ and $\mathcal{S}$ are then compared and discrepancies flagged as possible *flaws* in either the device configuration or specification design. In our simple example

- user `guest` should be allowed to access the web server only (see Tab. 4.2), but Tab. 4.1 shows that he/she is enabled to carry out several operations on objects in the server room and production areas.
- By design requirement, user `operator1` should be allowed to access only objects placed within his/her cells boundaries (`FIELD1` and `CONTROL1`). However, the analysis found out that he/she can also perform some action in `FIELD2` (operation `set` on object `IO2_1_manage`). By contrast, he/she cannot carry out other operations that are foreseen in the plant specification (e.g. `read` operation on object `PLC1_diagnostic` and others - see Tab. 4.2).

The comparison of Tables 4.1 and 4.2 is enough to discover flaws but, in order to understand the causes of these problems, a deeper inspection of the LTS structure would be needed. In particular, Fig. 9 shows that `operator1` can do the `set` operation on object `IO2_1_manage` of cell2, which is forbidden by the plant specifications. By following the `operator1` steps in the LTS, we could see that this happens after he leaves room `FIELD1` and enters the `CONTROL2` and `FIELD2` areas.

Similarly, the reasons that prevent `operator1` from performing the `read` operation on object `PLC1_diagnostic`, are less evident; in fact, object `PLC1_diagnostic` can be accessed only remotely.

This means that `operator1` must already be logged on some plant device connected to `PLC1` through a network path. (Un)fortunately, `operator1` is in the `FIELD1` room, where only the `CONTROL1`, `CONTROL2`, and `FIELD2` areas are reachable. In none of them a device is present, which allows `operator1` to log in.

Problems concerning user `guest`, instead, are due to the assignment of credential `pw_pwd`. `guest` is allowed to perform the login operation on the `PW` office workstation. Gaining access to the workstation, which is connected to the system supervision network, lets him/her leverage all those remote accessible services that do not require a password knowledge.

In our intention, the result analysis should help the designer in identifying possible fixes for the detected flaws, applying them to the model and running the analysis step again for the verification.

In our simple example, possible fixes for flaws concerning the `guest` user include the reconfiguration of sensible services (a password should be required for their invocation) and the introduction of changes to the network topology (such as the deployment of a firewall to logically separate the office and production areas). The `operator1` case, instead, could be tackled by implementing some form of physical access control between rooms in the production areas. This can be modelled as (new) credentials needed to traverse the doors in the system model. An alternative solution is enclosing sensible devices (e.g. PLCs) in cabinets protected by physical access control mechanism. Again, this can be modelled with our tool by adding small “rooms” in the system, representing cabinets, and setting specific access credentials for them.

5. CONCLUSIONS AND FUTURE WORK

In this work we have shown how some security-related aspects of a real system can be modelled and studied with the support of an automated software application. The proposed approach offers several advantages. For instance, it enables the designer to work with templates as opposed to being forced to write the specification of complex devices over and over. This is particularly useful in industrial environments where the same kind of devices are frequently deployed in different places. Another advantage is that the tool graphical interface allows the user to build a complex system model without any particular knowledge of the underlying formalism adopted by the analysis module. In this way, the complexity and discomfort of using a formal language is hidden by an easy-to-use application. Moreover, the user can design the specification and implementation views of the model at the same time, and the tool keeps track of their mutual relationships automatically.

The industrial plant scenario presented in this paper is purposely simplified, and admittedly the discrepancies, which have been identified, are rather straightforward. However, our goal is demonstrating the feasibility of our formal analysis approach and its potential usefulness for industrial environments.

Future work will include further improvements of the tool in two main directions, that is usability and integration. The template-based solution, already implemented in the tool, is a first step to achieve user friendliness; the next planned step concerns the ability of building templates of cells rather than of devices.

When the integration of modules shown in Fig. 1 is considered, a main goal is letting the designer compose his/her own optimized system by selecting different modelling tools as well as differently optimized analysis modules as needed.

A post-analysis module is also being studied, which focuses on the evaluation of results obtained by comparing the S and I sets. The basic idea, in this case, is taking as much advantage as possible of the information already included in the LTSs.

REFERENCES

ANSI INCITS 359-2012 (2012) *Role based access control*.

ANSI/ISA (2009) *Security for industrial automation and control systems: establishing an industrial automation and control systems security program*.

Cau, A., Janicke, H., and Moszkowski, B. (2013) Verification and enforcement of access control policies. *Formal Methods Syst. Design*, 43 (3). 450–492.

Cheminod, M. et al. (2014) On the description of access control policies in networked industrial systems. In: *Proc. of the 10th IEEE Int. Wksp. on Factory Communication Systems (WFCS)*, 1–10.

Cheminod, M., Durante, L., and Valenzano, A. (2013) Review of security issues in industrial networks. *IEEE Trans. Ind. Informat.*, 9 (1). 277–293.

Bertolotti, I. C. et al. (2013) 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)*, 66–77.

Bertolotti, I. C. et al. (2013, Dec.) *A twofold model for the description of access policies in industrial systems*. Torino, Italy: CNR-IEIIT, Tech. Rep. IEIIT-CENG-13-01.

Hinrichs, T. L. et al. (2013) Application-sensitive access control evaluation using parameterized expressiveness. In: *Proc. of the 26th IEEE Symp. on Computer Security Foundations, CSF*, 145–160.

Hu, V. C. et al. (2007) 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*, 275–280.

ISO/IEC (2008) *Information technology - Security techniques - Information security risk management*.

Masood, A., Ghafoor, A., and Mathur, A. (2010) Conformance testing of temporal role-based access control systems. *IEEE Trans. Dependable Secure Comput.*, 7 (2). 144–158.

Nicol, D. M. et al. (2008) Usable global network access policy for process control systems. *IEEE Security Privacy,* 6 (6). 30–36.

Nicol, D. M. et al. (2010) Experiences validating the access policy tool in industrial settings. In: *Proc. of the 43rd IEEE Hawaii Int. Conf. on System Sciences (HICSS)*, 1–8.

NIST SP 800-37 rev. 1 (2010) *Guide for applying the risk management framework to federal information systems - A security life cycle approach*.

Okhravi, H., Kagin, R. H., and Nicol, D. M. (2009) 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)*, 53–62.

Sandhu, R. S. et al. (1996) Role-based access control models. *IEEE Computer,* 29 (2). 38–47.

Stouffer, K., Falco, J., and Scarfone, K. (2008) *Guide to industrial control systems (ICS) security*. NIST SP 800-82.

Valenzano, A. (2014) Access control policy models for improving the security of industrial control systems. *IEEE Ind. Electron. Mag.*, 8 (2). 27–39.