Industrial Cybersecurity

Improving Security Through Access Control Policy Models

ADRIANO VALENZANO

The protection of a networked industrial control system (ICS) against cyberattacks and malicious behavior is a process that should be taken into account since the very beginning of the system's conception. This is true, in particular, for the design and verification of access control policies that build up the core of any protection scheme. The aim of this article is to assess the general framework and shed some light on the research activities concerning the analysis and verification of access policies in ICSs, which are currently being carried out in our laboratory.

Cyberattacks on ICSs and critical infrastructures have constantly gained popularity in the last years and received increased attention not only from the

Digital Object Identifier 10.1109/MIE.2014.2311313
Date of publication: 19 June 2014

1932-4529/14©2014IEEE

scientific community and technical interest groups but also from management and governmental authorities and even mass media.

There is a general consensus today that protecting ICSs against the malicious behavior and menaces carried out through cyberspace is no longer an option and that security has to be considered a basic requirement in all typical phases of an ICS's life cycle, including the design, development, deployment, operation, maintenance, and even disposal. This concept has been well assessed in large enterprises and mastered by those people who are responsible for managing and operating huge critical infrastructures such as those concerning power, gas, and water distribution; transportation systems; gas and oil production; and food production and distribution. In most of these cases, security-oriented standards and/or good practice guidelines have also been developed that are able to meet the specific requirements of the relevant application areas, and they are usually applied rigorously in planning and deploying new systems.

The situation, however, is a bit different when small and medium enterprises (SMEs) are considered, particularly in manufacturing and automation scenarios, where sometimes full awareness of potential risks and their unwanted consequences has yet to be reached. In fact, as far as we can say, in a nonnegligible number of SME cases, the protection against malicious attacks is not considered a mandatory requirement from the initial conception of a new ICS but rather is treated as a sort of add-on feature that can be introduced in the system at a later time, i.e., when the system is deployed and put to work. This assumption, however, is in clear contrast with the largely shared belief that the ICS's security should be dealt with not as a product but rather as a process and can lead to difficulty, if not impossibility, in effectively granting a satisfactory level of security for the target system.

Securing an ICS involves several different aspects, which range from the assessment of risks to the selection of proper security policies, adoption of countermeasures, management of security patches and updates, and detection and reaction to incidents. Every protection scheme, however, cannot abstract from the availability of suitable access control policies and mechanisms that are at the basis of the whole security framework. Access control alone is not sufficient to grant protection against malicious attacks; in fact, it is not able to prevent an attacker from impersonating a legitimate user and carrying out unwanted actions in his/her place. However, access policies and their enforcement are needed as a basic building block for implementing effective countermeasures in ICSs.

The Security Process

One of the main reasons for finding so many deep differences in the approach to ICS cybersecurity with respect to more conventional information technology (IT) systems is due to the much longer life of the former. ICSs are often designed and expected to last for decades, which is quite a long time with respect to the average life of many technologies used for their implementation (for instance, most types of central processing units and microcontrollers have a life span of only a few years). Many ICSs still in use today were originally conceived as insulated systems, i.e., not connected to any (external) communication network and/or to the Internet. The subsequent interconnection with other kinds of systems (e.g., office and enterprise networks) to take advantage of new appealing features, such as remote monitoring, control, and maintenance, is at the basis of their exposure to menaces that were not taken into account at all when those ICSs where originally designed.

In the last decade, this aspect has been pointed out several times in the scientific literature [1]–[5], and many solutions tailored to cope with the typical requirements of different application areas have also been proposed [6]–[10]. Moreover, the exposure of an ICS to threats that can be launched from inside its system has been recognized for several years [11] but attacks that originate from the outside and are carried out by leveraging remote connectivity are, surely and unfortunately, gaining more and more popularity. This is mainly due to the introduction in ICSs of hardware (h/w) and software (s/w) components, which are based on popular IT technologies developed and widely adopted in other application areas such as personal computing and even consumer electronics.

The progressive replacement of proprietary h/w and s/w elements with more conventional, inexpensive devices of widespread use and off-the-shelf availability, such as MS-Windows or Linux-based industrial personal computers (PCs), has brought a number of benefits from the points of view of functionality, performance, and cost. However, the same well-known security problems experienced by conventional IT-networked systems have been inherited. In the case of general-purpose IT systems, however, the reduced life span and rapid replacement of technologies have enabled the adoption of effective countermeasures and security approaches that can hardly be applied to ICSs [3]. On the one hand, in fact, some popular security solutions and common practices, such as the adoption of antivirus products and s/w patches and updates, are out of dispute in many ICSs because of their peculiar needs (e.g., real-time and availability requirements).

On the other hand, however, engineers often respect their basic principles and "never change a running system." This attitude is not due to a lack of flexibility in the engineers' understanding or will to introduce changes, but it is instead motivated by pragmatic reasons. Many ICSs have to work 24 h/d, 7 d/wk with total availability; even short stops can negatively affect the production and/or the quality of products, which unavoidably results in financial losses. A conservative approach is then more than understandable, in particular, in the absence of a suitable

The reduced life span and rapid replacement of technologies have enabled the adoption of effective countermeasures and security approaches that can hardly be applied to ICSs.

analysis of risks and a careful evaluation of costs (or losses) due to possible damages. The availability of a reliable cost/benefits estimate in these cases would likely drive engineers to introduce those changes that are needed to enhance the system's protection.

The practical consequence of this situation, however, is that most ICSs today still do not even include basic mechanisms that can be leveraged to protect them from cyberattacks or improve their security level.

There is a general consensus worldwide that the security of ICSs must be regarded and managed as a process rather than a product. This concept, for instance, has been totally recognized by main international standardization bodies such as the International Organization for Standardization/International Electrotechnical Commission (IEC) [12], National Institute of Standards and Technology [13], and International Society of Automation/IEC [14], which, in different documents (and although with different terminologies, scope, and approaches), agree that managing the security of industrial systems means taking care of the five logical and temporal steps depicted in Figure 1. The word *assets* used in the picture stands for "whatever has a value to the organization" and includes, of course, not only things but also people and the environment. Risk assessment in Figure 1 is a very critical step that takes into account the system to be protected as a whole and tries to evaluate the probability and consequences of unwanted events. To do this, sophisticated models of the system have to be used, and several kinds of analysis can be carried out. The results produced by risk assessment are then used to design and select suitable security countermeasures (also known as *security controls*) to be adopted in the system. The final validation step is fundamental in checking if the selected solutions are really able to satisfy the security requirements for protection and mitigate the risk of incidents by keeping it under a predefined and acceptable threshold. It is important to keep in mind, however, that the process in Figure 1 is iterative and can be triggered/retriggered by several different kinds of events during the whole life cycle of the system, including the design, implementation, operation, and maintenance phases. In practice, whenever a change occurs in the real system, for instance, because of upgrades, patches, or detected vulnerabilities, the process in Figure 1 has to be carried out with respect to the new system's configuration and iterated until the validation step considers the proposed solution as satisfactory.

An important aspect of this process is checking that the security policies adopted for the system, which are usually defined at a high level of abstraction, are correctly matched by the low-level protection mechanisms included in the system's implementation. This kind of verification cannot abstract from the availability of a suitable model for the system, as schematically shown in Figure 2. According to this approach, the high-level security policies are first checked against the system model, possibly by means of automated/semiautomated analysis s/w tools.

The results of the analysis are then used not only to update/correct/refine the policies but also to alter the model and low-level security mechanisms to obtain a perfect match with the policies themselves. Of course, the changes in the model have to be reflected in the

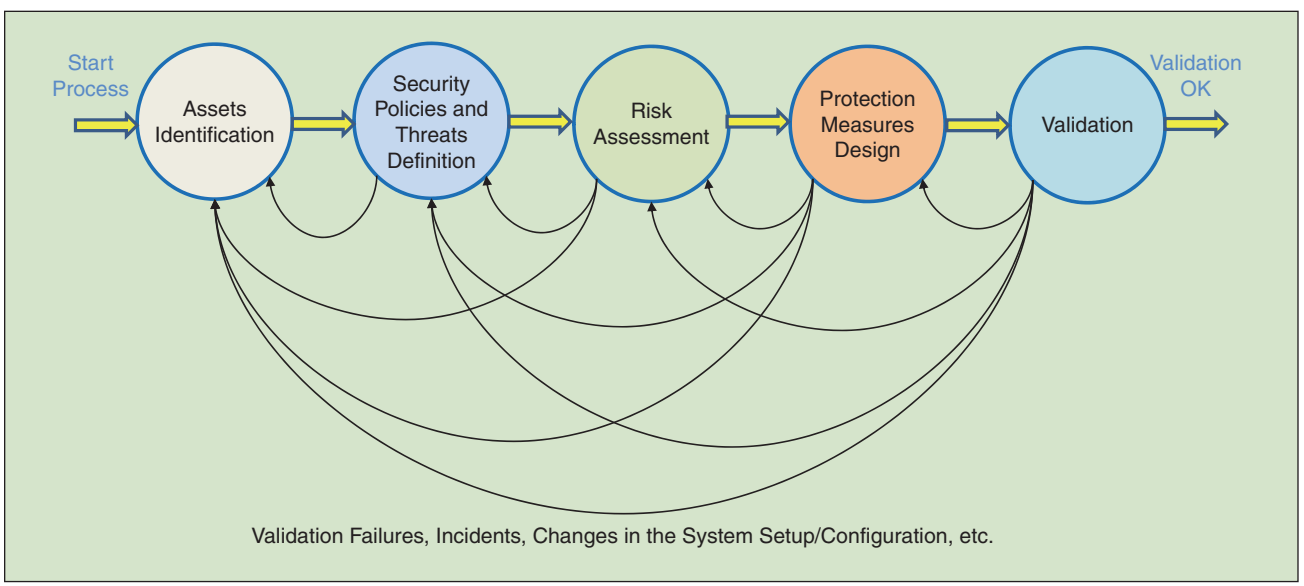


FIGURE 1 – The logical and temporal steps in securing an ICS.

real system and vice versa. In other words, the system and its model must always be kept perfectly aligned; this is another crucial aspect because it has to be implemented for the entire life of the system, particularly by introducing corresponding modifications to the model whenever anything is changed in the real system for whatever reason.

High-Level Access Policy Models for ICSs

As mentioned previously, the protection of an ICS involves several complementary aspects such as the analysis and evaluation of risks, the definition of security policies and strategies, and the design and deployment of countermeasures that are able to prevent and detect attacks and mitigate their effects on the controlled system. All these topics have been tackled extensively in the scientific literature in the past several years, and the interested reader can find an assessment of the state of the art in [3].

Access policies are a significant part of the security policies that can be adopted to protect ICSs. Protecting a system implies that all access to its resources is controlled and unauthorized access is filtered out by means of suitable mechanisms. The access policies define the high-level rules for regulating such an access-control process. It is worth repeating that access control is not sufficient on its own to grant the security of ICSs or assure their protection against malicious behaviors. However, it represents a fundamental element around which the whole security system should be built, in the same way that a reinforced door is not able on its own to prevent intrusions (for instance, if the attacker can steal or capture the key), but it is helpful in enforcing controlled access to a protected room. A popular example of exactly this is the Stuxnet attack against Siemens' programmable logic controllers (PLCs), which was carried out by impersonating a legitimate user despite the access control mechanisms that were operating.

A further issue concerns the design and development of more secure engineering s/w tools that are frequently used, for instance, to configure and manage embedded devices in ICSs. A recent research study [15] identified a number of potential security threats for both the tool users and developers that can affect such a class of s/w applications and proposed some guidelines to mitigate their unwanted exposure to risks. According to [15], access control is one of the ingredients that must be provided when an engineering tool performs accesses through the Internet, to download patches and updates, or is connected to the system devices. Consequently, the analysis of the access policy implementation also has to take into account the tool's interactions with the physical system.

The high-level security policies are first checked against the system model, possibly by means of automated/semiautomated analysis software tools.

In the past, several techniques were developed for specifying access policies at a high level of abstraction. Discretionary access control (DAC) [16], for instance, is based on the identities of the users and the explicit rules stating who is or is not allowed to execute what actions on what resources. The discretionary attribute is because users can be authorized to pass their rights to other users, while granting and revocation of rights are regulated by an administrative authority (i.e., the operating system administrator). Mandatory access control (MAC) instead assigns rights based on the rules issued by a central authority. A very common form of MAC is the multilevel security policy [17], [18], based on the classification (i.e., unclassified, confidential, secret, and top secret) of resources to be protected and subjects needing to access them.

Role-based access control (RBAC) [19], [20] appeared more recently but gained consensus rapidly as a better alternative to DAC and MAC in a large number of practical applications. In fact, RBAC maps access policies on the structure of organizations in a more natural way than DAC/MAC by adopting an underlying model based on roles and objects rather than on users and objects. A role is usually a job function or title (e.g., operator, maintainer, or plant engineer) with associated activities and duties. Actually, in most real-world situations,

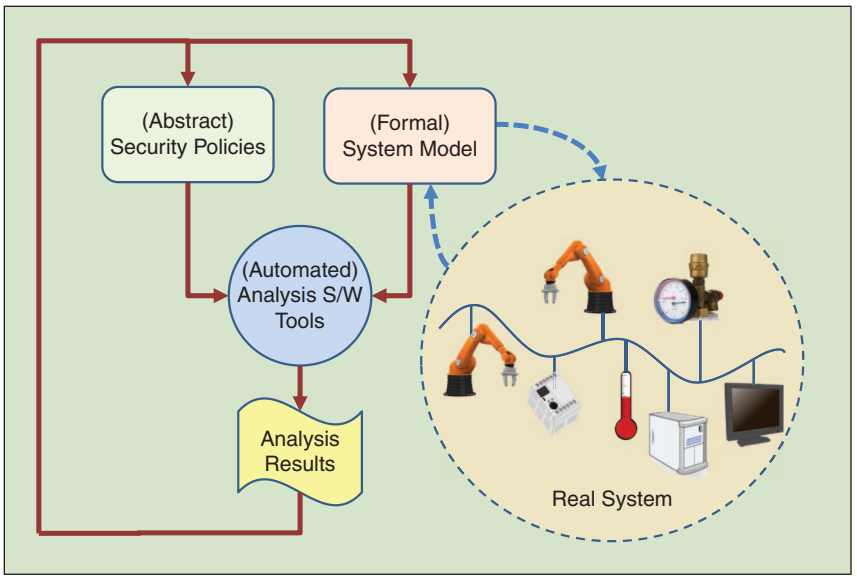


FIGURE 2 – The security policy verification process.

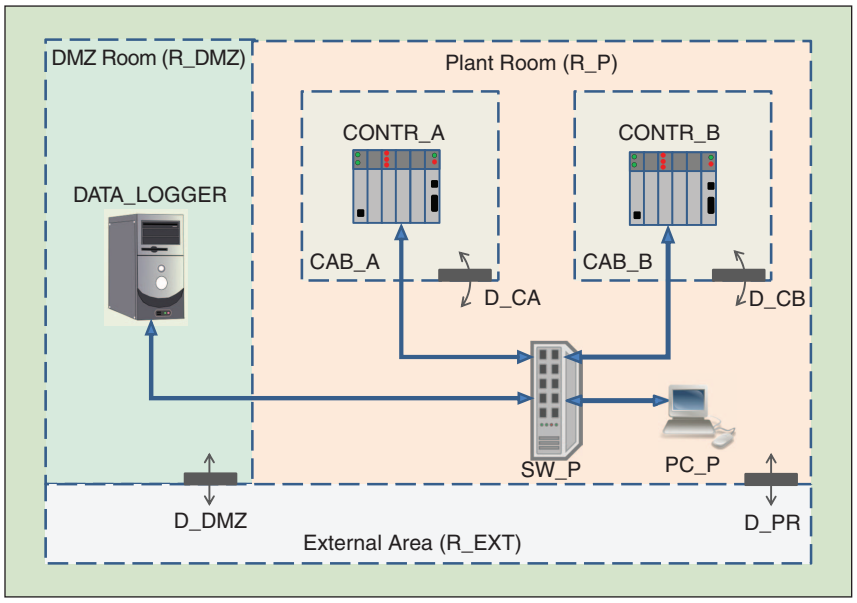


FIGURE 3 – A simple plant example.

knowing the organizational responsibilities of a user is much more significant than knowing the user's identity. In RBAC, the access policies are defined by assigning users and permissions to roles. Of course, RBAC also includes objects that are acted on by roles and operations that can be executed on objects. Since its acceptance as an industry consensus standard in 2004, the RBAC approach has been successfully adopted in a number of technical areas that are typical of the information technology domain, such as those dealing with operating systems or database management systems. However, several RBAC concepts have also been included in the definition of major security-related standards and, in particular, in the IEC 62351 suite concerning the information infrastructure of power systems. IEC 62351 part 8 [21], in fact, builds on the RBAC users-roles-permissions authorization concept and specifies several kinds of elements (i.e., extensions to the data models for implementing RBAC, the format of credentials, the transmissions of roles over the network, and so on) that are needed to enhance the cybersecurity of communications in the management of power industry information infrastructures. Finally, extensions of RBAC have also been studied and developed to cope with the access control needs and features of smart grids [22].

For example, let us consider the simple plant sketched in Figure 3, representing two cooperating machines whose controllers (CONTR_A and CONTR_B) are enclosed into two separate cabinets (CAB_A and CAB_B) and connected through an Ethernet switch (SW_P) to their control network (CN). The switch also enables connecting the CN to a host (DATA_LOGGER) placed in a demilitarized zone (DMZ). Finally, a conventional computer is connected to SW_P to enable remote actions on CONTR_A or CONTR_B and DATA_LOGGER. The CN area and DMZ are disjointed from a logical point of view, and they are also located in distinct rooms. The room doors (D_DMZ and D_PR) and cabinet doors (D_CA and D_CB) are also shown in Figure 3. Moreover, to keep the example as simple as possible, other devices and connections (in particular, from the DMZ to the enterprise network and the Internet) are not considered and are not shown in the figure.

A simple RBAC model definition for the system can include, for instance, operator and maintainer roles distinct for A and B (assuming that machines A and B are of significantly different types) and a plant supervisor role. A operators are allowed to start/stop the machine control service on machine A (plc_A) only, while B operators can do the same with machine B (plc_B control service). A maintainers, in addition, are permitted to configure plc_A, and the same is possible for B maintainers with respect to B. Finally, supervisors can perform any possible action on any machine and, in particular, start and stop the data logging service on DATA_LOGGER.

Table 1, besides summarizing the elements relevant to RBAC that are included in the plant (i.e., objects, operations, and roles), also shows the assignment of permissions to roles. The definition of the users and

TABLE 1- THE RBAC OBJECTS, OPERATIONS, AND ROLES FOR THE EXAMPLE PLANT.

OBJECTS	OPERATIONS	ROLES				
		A_operator	A_maintainer	B_operator	B_maintainer	Supervisor
plc_A	Start	Y	Y	N	N	Y
	Stop	Y	Y	N	N	Y
	Configure	N	Y	N	N	Y
plc_B	Start	N	N	Y	Y	Y
	Stop	N	N	Y	Y	Y
	Configure	N	N	N	Y	Y
data_logging	Start	N	N	N	N	Y
	Stop	N	N	N	N	Y

their assignment to roles, as listed in Table 2, complete the RBAC description. The rightmost column in Table 2 shows all the actions allowed to the plant users by means of triples (user, operation, and object): the set of all triples in the column completely describes "who can do what on what" and is the RBAC view of the system in Figure 3. By assumption, actions not included in the set of Table 2 are forbidden in the modeled system.

The high abstraction level adopted in RBAC and similar techniques facilitates the specification of access policies with a high degree of independence from the actual system implementation; moreover, the verification of the coherency of policies in a given set can be carried out in efficient and formally elegant ways. Unfortunately, the same high-level description frameworks are of little help in checking whether policies are correctly implemented in a real system. Unavoidably, the solutions that appeared in the literature to overcome this limitation suffer from two types of drawbacks: they either assume that the underlying system is able to guarantee the policy enforcement [23]–[25] or rely on h/w and s/w extension mechanisms [26]–[28] to grant some kind of enforcement nevertheless. For many reasons, however, this approach is largely infeasible for ICSs because ICSs are often very heterogeneous in nature, their special h/w and s/w components can hardly be changed significantly and, last but not least, they frequently include very poor security control mechanisms.

Very few techniques [30]–[32] and s/w tools [29], [33]–[35] that appear in the literature enable some correctness check of policy implementation in real networked systems without assuming the availability of suitable enforcement mechanisms. In particular, in [33]–[35], we tried to enrich a system description, mostly based on RBAC, with elements and information that are able to also take into account aspects concerning its actual implementation such as the network topology, services installed on the different nodes, and so on. That experience, however, was successful only in part, and the lessons learned in a number of case studies eventually led us to adopt a different solution, as explained in the section "A Twofold Model for Checking Policies." The main reasons that drove this change in the approach are:

- In a number of practical situations, we found it quite difficult to include in a high-level (RBAC) model some important configuration details, such as, for instance, host accounts and physical locations of devices, that are more relevant for the policy mapping analysis.
- In real-world situations, different teams of people are often responsible for the definition of high-level policies and the system configuration/setting. Moreover, they are frequently familiar with different languages and procedures (i.e., operators must not be allowed to reprogram machines A and B versus user x should not be able to log on to host y and upload files to PLC z).
- A disjointed view of the actual system implementation details (i.e., firewalls and switches filtering rules, device configurations, account settings, and so on) can help in gathering the needed information from the system itself in a semiautomatic/automatic way.
- Industrial systems are heterogeneous in their nature, and enforcing high-level policies by altering the h/w and s/w architecture of network nodes with mechanisms introduced ad hoc is definitely not feasible.

The high abstraction level adopted in RBAC and similar techniques facilitates the specification of access policies with a high degree of independence from the actual system implementation.

A Twofold Model for Checking Policies

Some of the limitations listed in the previous section can be overcome, at least in part, by adopting a new kind

TABLE 2- THE RBAC USERS AND THEIR ASSIGNMENT TO ROLES.

ROLE	USER	OPERATION ALLOWED TO USER
A_operator	Al	(Al, start, plc_A) (Al, stop, plc_A)
A_maintainer	Alfred	(Alfred, start, plc_A) (Alfred, stop, plc_A) (Alfred, configure, plc_A)
	Arnold	(Arnold, start, plc_A) (Arnold, stop, plc_A) (Arnold, configure, plc_A)
B_operator	Brian	(Brian, start, plc_B) (Brian, stop, plc_B)
	Basil	(Basil, start, plc_B) (Basil, stop, plc_B)
B_maintainer	Bert	(Bert, start, plc_B) (Bert, stop, plc_B) (Bert, configure, plc_B)
Supervisor	Paul	(Paul, start, plc_A) (Paul, stop, plc_A) (Paul, configure, plc_A) (Paul, start, plc_B) (Paul, stop, plc_B) (Paul, configure, plc_B) (Paul, start, data_logging) (Paul, stop, data_logging)

Different teams of people are often responsible for the definition of high-level policies and the system configuration/setting.

of model that includes two logically separate views of the system concerning the high-level policies and the low-level mechanisms, respectively. For our purpose, low-level mechanisms means all those physical (h/w and s/w) system settings, such as user accounts, passwords, and switch and host port configurations that have to be carefully set up to support the desired access-control scheme. Recently, we focused on the development of a twofold model [36], [37], which, besides describing policies according to the RBAC framework, is also able to capture the fine-grained implementation details of the system such as firewall rules, configurations of both industrial/automation devices (i.e., PLCs, numerical controllers, intelligent sensors/actuators) and traffic control equipment (i.e., switches, routers, and access points), physical locations (e.g., rooms, cabinets) of devices, and credentials (i.e., physical keys, badges, passwords, and certificates) needed to access system resources and invoke operations on them. A formal and complete description of this approach can be found in [36].

The high-level view of the system (we call it the *specification*) is based on hierarchical RBAC [20] and, in practice, allows for describing the access policies by means of the same kind of triples shown in the right column of Table 2. Moreover, the hierarchical organization of roles enables the inheritance of roles to users and of permissions to roles. For instance, in our example, A maintainers inherit rights from A operators, and the same happens for the B roles. Supervisors, on the other hand, inherit permissions from both A and B roles (multiple inheritances).

Figure 4 shows how the twofold model can be used to analyze correctness of the policy implementation from a conceptual perspective. One difficult point in this case could be the ability to put into correspondence the two separate views of the system in Figure 4, which are frequently under the responsibility of different teams of designers and administrators. While the sets of users in RBAC and in the real system consist of very similar groups of people/agents and can be kept aligned with moderate effort, the objects, operations, and rights might not be a one-to-one correspondence in the two views. In particular, in the real system case, access rights are often specified in terms of accounts/

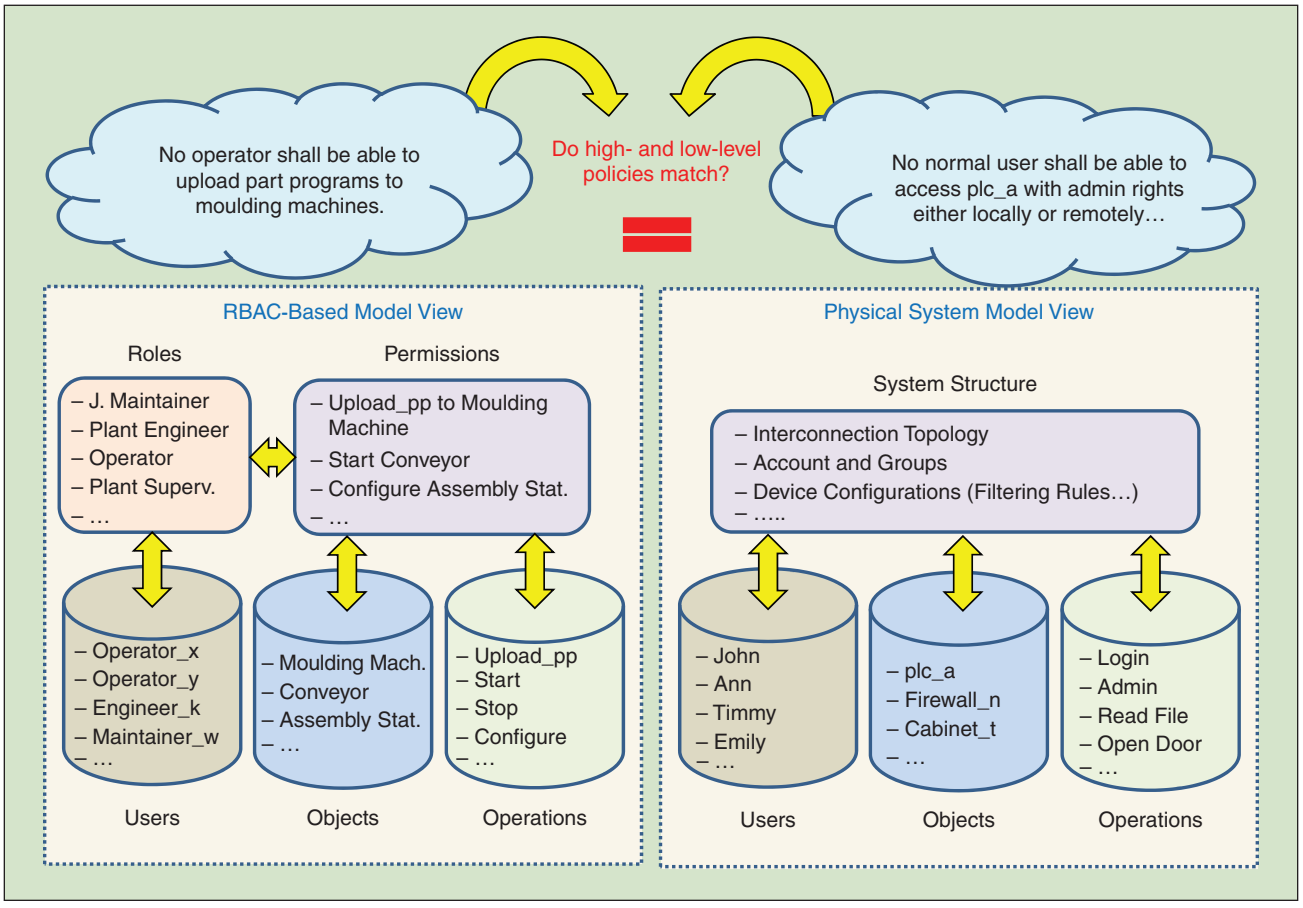


FIGURE 4 – A twofold model for the analysis of security policies.

usernames and groups, so a user can acquire different rights depending on the credentials (e.g., username, physical key, and badge) he/she owns at a given moment. The actual system implementation, moreover, usually contains more (and possibly different) objects than the RBAC specification counterpart. The inclusion of this kind of information in the model, however, is fundamental to automatically verify whether policies such as "no operator can upload part-programs to moulding machines" are correctly implemented, for instance, by correctly configuring the available h/w and s/w settings included in system and/or introducing additional protection countermeasures. Performing this verification could mean being able to check that any unprivileged user is unable to either physically access the control panel of a PLC or to log in remotely to the PLC with administration rights (of course, this depends on how the moulding machine is actually implemented as it may consist, in principle, of a large set of options concerning its h/w and s/w components that cannot be taken into account directly in the specification view of the system). We call *implementation* the model part concerning the description of the physical system. In its turn, the implementation consists of two basic parts: a detailed (static) description of the system, including its building elements, and a set of rules that are used to compute all the possible actions that each user can perform on each object in the system.

Physical System Description

The physical system description is based on three main sets, which are hosts, rooms, and links. Hosts represent the nodes in the network and include not only computers or special equipment such as PLCs, but also h/w traffic control devices such as switches and firewalls. For instance, DATA_LOGGER, CONTR_A, CONTR_B, SW_C, and PC_P are hosts in the system of Figure 3. Rooms are special objects keeping track of the actual locations of hosts and are used to model areas and cabinets (e.g., R_DMZ, R_P, R_EXT, CAB_A, and CAB_B). Rooms are also important in modeling wireless communications

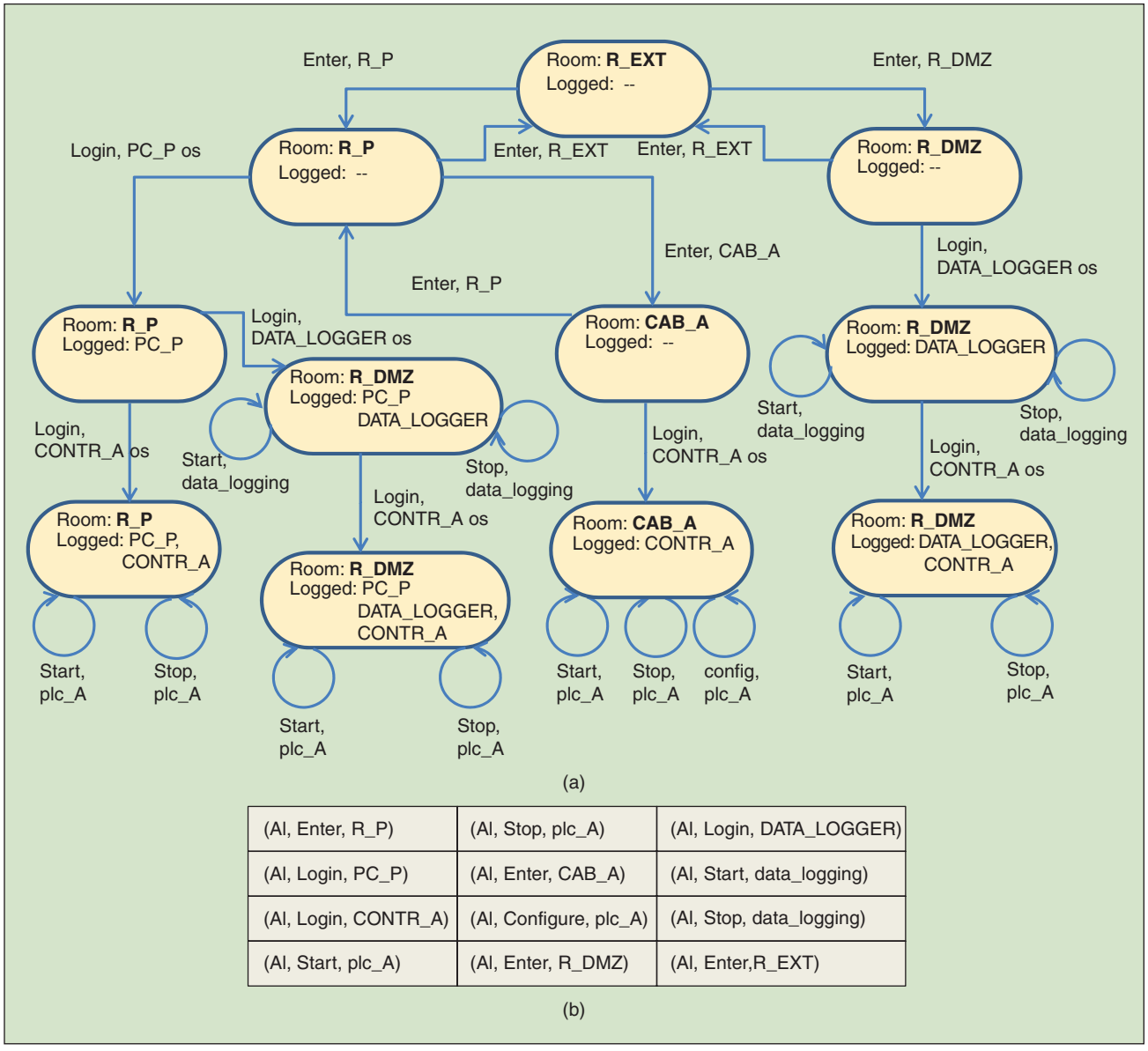
A disjointed view of the actual system implementation details can help in gathering the needed information from the system itself in a semiautomatic/automatic way.

and establishing where a wireless link is accessible in the system. Each host is equipped with a set of communication ports, is associated with a room object, and is also bound to a set of filtering rules specifying details about the physical and logical interconnections between the host communication ports, the protocols adopted, and the condition (allowed/forbidden) for each existing port interconnection path. The description of the system in Figure 3, for instance, specifies that host SW_P is equipped with ten physical ports, is placed in room R_P, and is configured so as to enable bidirectional communications only between those ports connected to CONTR_A, CONTR_B, DATA_LOGGER, and PC_P, respectively.

Each host in the model also includes a set of resources representing services and functionalities installed on the host itself and potentially accessible to users in the system. In Figure 3, CONTR_A and CONTR_B could be implemented as some kind of industrial PC running a soft PLC application. The resources in this case could be the operating system login service, allowing users to log in to the node, and the soft PLC task that provides the PLC functionalities. In the example, another s/w service, like the one offered by an OPC-UA server, is also installed on CONTR_A and CONTR_B to enable data logging. Resources for SW_P, instead, include the login service for the switch and a configuration service made available for its management. Besides the conventional login functionality, DATA_LOGGER supports a suitable service (e.g., OPC-UA client) to collect data remotely from both CONTR_A and CONTR_B. No special service is installed on PC_P except its operating system login module, as it is conceived simply as a remote console for accessing the other nodes in the plant.

The objects have an associated set of operations that can be executed on them. A room operation consists of an action and a set of credentials (e.g., physical key, badge, and access code) that users must own to perform the action itself. For instance, the enter operation is associated to rooms R_DMZ and R_P in Figure 3, and users must own credentials cr_{D_DMZ} and $cr_{D\_PR,}$ respectively, to perform it. The same is true for the open operation (called *enter* again in the following) and credentials cr_{CAB_A} and cr_{CAB_B} concerning cabinets CAB_A and CAB_B.

The operations affecting host resources have a more complex structure since each action is associated to a set of pre- and postconditions. Preconditions describe the requirements that a user needs to meet to be able to perform the relevant action. The model includes three different types of possible preconditions called *physical_access*, *local_access*, and *remote_access*, which, respectively, take into account the ability of accessing the resource physically (i.e., a direct access to the host console), locally (a successful login already performed on the host), or remotely (a successful login through the network from a remote host where the user is already logged in). Each type of precondition includes a (possibly empty) set of required credentials taking into account the ability of the users to access the resource (e.g., password knowledge) and acquire rights pertaining to a given user account/group bound to the resource itself. In Figure 3, CONTR_A and CONTR_B accept remote connections from PC_P and DATA_LOGGER (but outgoing connection requests from the controllers are blocked) so that their control operations (i.e., the soft PLC applications) can be started/stopped by logging into the system with credentials



(AI, Enter, R_P)	(AI, Stop, plc_A)	(AI, Login, DATA_LOGGER)
(AI, Login, PC_P)	(AI, Enter, CAB_A)	(AI, Start, data_logging)
(AI, Login, CONTR_A)	(AI, Configure, plc_A)	(AI, Stop, data_logging)
(AI, Start, plc_A)	(AI, Enter, R_DMZ)	(AI, Enter,R_EXT)

(b)

FIGURE 5 – (a) The partial LTS construction for the example plant. (b) All the actions on the objects that are permitted to AI in the form of triples obtained through the LTS construction.

c_{CONTR_A} and c_{CONTR_B}, respectively (remote access precondition). Another way to achieve the same result is by logging in through the controller embedded console with credential c_{CC_A}/c_{CC_B} (physical access precondition) after opening the relevant cabinet. The PLC configuration operation, instead, can be performed only by accessing the console with c_{CC_A}/c_{CC_B}. Finally, the data service running on A and B (OPC-UA server) cannot be started or stopped manually and is managed directly by the node operating system. The data logging client on DATA_LOGGER can be started/stopped by each user logged either locally or remotely on that host (DATA_LOGGER accepts incoming connections from PC_P, but PC_P in its turn does not accept incoming connections) and only a single (administration) account is provided by the relevant login service, which requires the credentials c_{DLOG}. The configuration of SW_P can be performed by logging remotely on that node from either PC_P or DATA_LOGGER with the credential c_{SW_P}, while the login service of PC_P is configured with a single account requiring the c_{PC_P} credential.

Postconditions take into account the effect of operations performed on the host resources. In particular, the effect of a successful login operation through the service offered by a given host allows the performing user to gain access to the resources located on that host and the rights assigned by the operating system to the user/group involved in the login operation.

User's Actions

The set of all possible actions that a user can perform in the physical system can be computed by constructing a labeled transition system (LTS). This can be done by letting each user move from one room to the other and carry out all

the operations he/she is enabled to perform on resources located on hosts in the rooms. By basic assumption, in our model, the actions of one user cannot affect the behavior of the others, so the implementation view, which consists of the set of all allowed (user, object, and operation) triples, can be obtained by combining the independent contributions of the different users.

The LTS construction is based on the definition of the user's state and a set of rules (inference rules) stating how the user's capabilities can be put into correspondence with the preconditions required to perform operations on resources. For instance, the rule for crossing a door *d* and moving from room *x* to room *y* (enter operation) specifies that, to perform the action, rooms *x* and *y* must be adjacent and separated by door *d*, the current position of the user must be in room *x* and, if a credential (i.e., a physical key) is needed to open the door, the user must also own it. The effect of the action execution is that the user moves from *x* to *y*, thus gaining physical access to the hosts in the destination room. A complete and formal discussion of all the inference rules in the model can be found in [36].

In general, each state of the LTS should consist of both the current state of the user and an updated description of the physical system. The proposed approach, however, makes use of a static system description, so that its inclusion in the LTS state can be avoided to keep the computation simpler.

In Figure 3, let us assume that all uses are initially in the external area (R_EXT) and the set of credentials (cr_{D_DMZ}, cr_{D_RP}, cr_{CAB_A}, c_{CONTR_A}, c_{CC_A}, c_{DLOG}, and c_{PC_P}) is assigned to both operators and maintainers of machine A, while the set (cr_{D_RP}, cr_{CAB_B}, c_{CONTR_B}, c_{CC_B}, c_{DLOG}, and c_{PC_P}) is granted to B maintainers. B operators are given the same set of maintainers' credentials with the exception of cr_{CAB_B}. Plant supervisors, however, are given all the credentials existing in the system.

Figure 5 shows a possible construction of a partial LTS concerning user *Al*, where, for the sake of conciseness, the constant set of the user's credentials has been omitted for each state in the picture. Please be warned that the structure of the LTS shown in the picture has only an illustrative purpose: it is not minimal, complete, or the best one, and our construction algorithms actually build the LTS in a different and more efficient way. This aspect is generally crucial in approaches based on model checking where the number of states of the LTS must be kept as small as possible and the computation time is a critical issue for the applicability of the analysis techniques to systems of significant size.

Figure 5(b) shows all the actions on the objects that are permitted to *Al*, in the form of triples obtained through the LTS construction. The overall implementation view of the model, which is the set of all allowed triples for all users, can then be obtained by simply combining the different users' contributions that are independent by assumption.

TABLE 3 - THE OPERATIONS PERMITTED TO THE USERS IN THE EXAMPLE PLANT.

	OBJECT	OPERATION	USER						
			Al	Alfred	Arnold	Brian	Basil	Bert	Paul
Specification + Implementation	plc_A	Start							
		Stop							
		Configure							
	plc_B	Start							
		Stop							
		Configure							
	data_logging	Start							
		Stop							
Implementation Only	R_P	Enter							
	R_DMZ	Enter							
	R_EXT	Enter							
	CAB_A	Enter							
	CAB_B	Enter							
	DATA_LOGGER login service	Login							
	PC_P login service	Login							
	CONTR_A login service	Login							
	CONTR_B login service	Login							
	SW_P login service	Login							
	SW_P configuration service	Configure							

Policy Mapping Verification

As mentioned before, the mapping of policies onto access mechanisms can be checked by comparing the triples in the specification and implementation views of the model. Table 3 summarizes the situation for the plant in Figure 3, where allowed triples in the two model views are shown in

The set of all possible actions that a user can perform in the physical system can be computed by constructing a labeled transition system.

different colors, which are green for the specification and yellow for the implementation, respectively. Each purple cell corresponds to an action on the object in the cell row that is not allowed to the user in the cell column in both the specification and implementation views. The upper part of the table concerns objects appearing in both the high- and low-level descriptions, while the lower part deals with the objects not included in the RBAC specification. By comparing triples in the upper part of Table 3, we can see that some RBAC policies have their correct counterparts in the implementation: this occurs for the cells colored in both green and yellow. However, some situations (only yellow cells) exist where the involved operations are possible in the actual system, even though they are not allowed by the RBAC policies. For instance, operator *Al* can configure plc_A because of the wrong assignment of credentials cr_{CAB_A} and c_{CC_A}, which enable him to open cabinet A and log in to the controller. This problem can be easily fixed by removing either cr_{CAB_A}, or c_{CC_A} (but removing both is better) from the credential set assigned to the A operators. Similarly, all operators and maintainers are allowed to start/stop the data-logging process. This occurs because A operators and maintainers are enabled to enter R_DMZ and log into DATA_LOGGER; moreover, all users are allowed to log in to PC_P and remotely connect to the data logger host in R_DMZ. An obvious fix is to not assign credential c_{DLOG} to all maintainers and operators or, alternatively, to configure DATA_LOGGER to also reject connection requests coming from PC_P and restricting the access to room R_DMZ by removing c_{D_DMZ} from the credential set assigned to A operators and maintainers.

The detection and correction of these unwanted situations can appear a trivial task, which can be easily carried out by hand, when simple examples such as that in Figure 3 are considered. Unfortunately, real systems are much larger and more complex, and an exhaustive check of the correct policy implementation can hardly be performed without the aid of suitable modeling techniques such as those presented in this article.

Conclusions

The constant increase of menaces carried out through cyberspace pushes toward the adoption of efficient defensive measures for securing critical infrastructures and ICSs. Access control, although not able to grant security on its own, is a fundamental element for building any protection scheme for ICSs. Risk mitigation, in fact, also implies a reduction of the probability of unwanted events, but this can be obtained only with the support of efficient access control techniques.

Access control policies, however, are often defined at a high level of abstraction, which makes them independent from their actual implementation in the physical system, but, consequently, the correct mapping of policies onto the low-level access mechanisms (i.e., configurations and settings) of the system devices is a difficult and often error-prone task. The twofold model presented in this article is a tentative though appealing approach to bridge the semantic gap between the abstract definition of access policies for ICSs and their correct mapping onto the low-level access control mechanisms provided by the actual system. This aspect is particularly important in ICSs, where the policy implementation cannot rely on suitable h/w and/or s/w extensions and support able to grant some kind of policy enforcement. Most ICSs today are still very heterogeneous in nature, include special-purpose devices and equipment, and make use of a very small number of poor low-level protection mechanisms. Exploiting the configuration of such simple elements and their moderate capabilities is often the only viable way for designers to make their access schemes, which are conceived at a relatively high abstraction level, work as expected in the real system.

As in most solutions based on model-checking techniques, the explosion of the state space (i.e., the number of states in the LTS) is a crucial issue for the feasibility of the proposed approach, and this depends, in our case, on the number of rooms, nodes, users, and possible actions. Some assumptions on which our model is based (such as the independence of user behaviors, the assignment of permissions to roles instead of users, and the static configuration of the physical system, which cannot be changed by the users' actions) can be exploited to construct the LTS in a quasi-minimized form. Although performance figures are not available yet, our initial experiments are encouraging from this point of view as systems consisting of a few hundred nodes and several roles can be analyzed in a reasonable amount of time with ordinary computing resources.

However, while the LTS construction and analysis can be almost completely automatized, a significant part of the system description still has to be carried out by hand with the aid of graphical editing supports. For this reason, we are investigating the possibility of extracting (part of) the needed information directly from the device configuration files and databases, possibly in an automatic/semiautomatic way. A second area that needs significant manual intervention (and expertise) is the evaluation of results produced by the analysis to fix unwanted situations. We are conscious that much work still has to be done to make this task easier and more intuitive.

Future work will primarily focus on the development of efficient LTS construction and analysis algorithms and on the removal of some constraints

currently adopted in the model such as the assumption of users' independence. The availability of an automated s/w tool prototype, currently under development, will also enable the analysis of a number of case studies concerning real-world systems and a more effective evaluation of the benefits and disadvantages of the proposed approach in practical situations.

Acknowledgment

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

Biography

Adriano Valenzano (adriano.valenzano@ieiit.cnr.it) is a director of research with the National Research Council of Italy. He is currently with the Istituto di Elettronica e di Ingegneria dell'Informazione e delle Telecomunicazioni, Torino, Italy, where he is responsible for research concerning industrial computer networks and systems. He is a Senior Member of the IEEE and vice president of the Piedmont chapter of the Italian National Association for Automation. He is the recipient of the 2013 ABB and IEEE Industrial Electronics Society Lifetime Contribution to Factory Automation Award for meritorious technical contribution and technical leadership. He has authored/coauthored more than 200 refereed journal and conference papers in the area of computer engineering. Since 2007, he has served as an associate editor for *IEEE Transactions on Industrial Informatics.*

References

[1] D. Dzung, M. Naedele, T. P. Von Hoff, and M. Crevatin, "Security for industrial communication systems," *Proc. IEEE*, vol. 93, no. 6, pp. 1152–1177, 2005.

[2] A. Creery and E. J. Byres, "Industrial cybersecurity for power system and SCADA networks," in *Proc. IEEE IAS Annual Petroleum and Chemical Industry Conf.*, Sept. 12–14, 2005, pp. 303–309.

[3] 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.

[4] L. Pièrre-Cambacédès, M. Tritschler, and G. N. Ericsson, "Cybersecurity myths on power control systems: 21 misconceptions and false beliefs," *IEEE Trans. Power Delivery.*, vol. 26, no. 1, pp. 161–172, 2011.

[5] P. Radmand, A. Talevski, S. Petersen, and S. Carlsen, "Taxonomy of wireless sensor network security attacks in the oil and gas industries," in *Proc. 24th IEEE Int. Conf. Advanced Information Networking Applications (AINA)*, Perth, Apr. 20–23, 2010, pp. 949–957.

[6] S. C. Patel, G. D. Bhatt, and J. H. Graham, "Improving the cybersecurity of SCADA communication networks," *Commun. ACM*, vol. 52, no. 7, pp. 139–142, 2009.

[7] F. M. Cleveland, "Cybersecurity issues for advanced metering infrastructure (AMI)," in *Proc. IEEE Power Energy Society General Meeting—Conversion and Delivery of Electrical Energy in the 21st Century*, Pittsburgh, PA, July 20–24, 2008, pp. 1–5.

[8] J. Liu, Y. Xiao, S. Li, W. Liang, and C. L. P. Chen, "Security and privacy issues in smart grids," *IEEE Commun. Surv. Tutorials*, vol. 14, no. 4, pp. 981–997, 2012.

[9] A. A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, and S. Sastry. (2009, 22–24 July). "Challenges for securing cyberphysical systems," in *Proc. Workshop Future Directions Cyber-Physical Systems Security*. [Online]. Available: http://cimic.rutgers.edu/positionPapers/cps-security-challenges-Cardenas.pdf

[10] G. N. Ericsson, "Cybersecurity and power system communication—Essential parts of a smart grid infrastructure," *IEEE Trans. Power Delivery*, vol. 25, no. 3, pp. 1501–1507, 2010.

[11] A. Treytl, T. Sauter, and C. Schwaiger, "Security measures in automation systems—A practice-oriented approach," in *Proc. 10th IEEE Int. Conf. Emerging Technologies Factory Automation (ETFA), Catania*, Sept. 18–21, 2005, pp. 847–855.

[12] *Information Technology—Security Techniques—Information Security Management Systems—Requirements*, ISO/IEC Std. 27 001, 2005.

[13] *Recommended Security Controls for Federal Information Systems and Organizations,* NIST SP 800-53 rev. 3, 2010.

[14] *Industrial Communication Networks—Network and System Security—Part 1-1: Terminology, Concepts and Models,* IEC/TS 62443-1-1, 2009.

[15] A. Hristova, S. Obermeier, and R. Schlegel, "Secure design of engineering software tools in industrial automation and control systems," in *Proc. 11th IEEE Int. Conf. Industrial Informatics (INDIN)*, Bochum, July 2013, pp. 695–700.

[16] M. H. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM,* vol. 19, no. 8, pp. 461–471, 1976.

[17] D. E. Bell and L. J. LaPadula, "Secure computer systems: Mathematical foundations," Tech. Rep. ESD-TR-278, The Mitre Corporation, Bedford, MA, 1973.

[18] K. J. Biba, "Integrity considerations for secure computer systems," Tech. Rep. TR-3153, The Mitre Corporation, Bedford, MA, 1977.

[19] 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.

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

[21] *Power Systems Management and Associated Information Exchange—Data and Communications Security—Part 8: Role-Based Access Control,* IEC/TS 62351-8, 2011.

[22] D. Rosic, U. Novak, and S. Vukmirovic, "Role-based access control model supporting regional division in smart grid system," in *Proc. 5th IEEE Int. Conf. Computational Intelligence, Communication Systems Networks (CICSyN)*, Madrid, June 6–7, 2013, pp. 197–201.

[23] V. C. Hu, E. Martin, J. Hwang, and T. Xie, "Conformance checking of access control policies specified in XACML," in *Proc. 31st IEEE Annual Int. Computer Software Applications Conf. (COMPSAC)*, Beijing, July 23–27, 2007, pp. 275–280.

[24] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz, "Verification and change-impact analysis of access-control policies," in *Proc. 27th ACM Int. Conf. Software Engineering (ICSE)*, St. Louis, May 15–21, 2005, pp. 196–205.

[25] 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.

[26] G. Faden, "RBAC in UNIX administration," in *Proc. 4th ACM Workshop Role-Based Access Control*, Fairfax, VA, Oct. 28–29, 1999, pp. 95–101.

[27] 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. 26th IEEE Computer Security Foundations Symp. (CSF)*, New Orleans, LA, June 26–28, 2013, pp. 145–160.

[28] A. Cau, H. Janicke, and B. Moszkowski, "Verification and enforcement of access control policies," *Formal Meth. Syst. Des.*, vol. 43, no. 3, pp. 450–492, 2013.

[29] Skybox Security. (2012). Using risk modeling and attack simulation for proactive cybersecurity. [Online]. Available: http://www.skyboxsecurity.com

[30] 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.

[31] D. M. Nicol, W. H. Sanders, M. Seri, and S. Singh, "Experiences validating the access policy tool in industrial settings," in *Proc. 43rd IEEE Hawaii Int. Conf. System Sciences (HICSS)*, Koloa, HI, Jan. 8–10, 2010, pp. 1–8.

[32] H. Okhravi, R. H. Kagin, and D. M. Nicol, "Policy globe: A framework for integrating network and operating system security policies," in *Proc. 2nd ACM Workshop Assurable Usable Security Configuration (SafeConfig)*, Chicago, IL, Nov. 9–13, 2009, pp. 53–62.

[33] M. Cheminod, I. Cibrario Bertolotti, L. Durante, and A. Valenzano, "Automatic analysis of security policies in industrial networks," in *Proc. 8th IEEE Int. Workshop Factory Communication Systems (WFCS)*, Nancy, France, May 18–21, 2010, pp. 109–118.

[34] I. Cibrario Bertolotti, L. Durante, T. Hu, and A. Valenzano, "A model for the analysis of security policies in industrial networks," in *Proc. 1st Int. Symp. ICS SCADA Cybersecurity (ICS-CSR)*, Leicester, Sept. 16–17, 2013, pp. 1–11.

[35] M. Cheminod, L. Durante, and A. Valenzano, "System configuration check against security policies in industrial networks," in *Proc. 7th IEEE Int. Symp. Industrial Embedded Systems (SIES)*, Karlsruhe, Germany, June 20–22, 2012, pp. 247–265.

[36] I. Cibrario Bertolotti, L. Durante, L. Seno, and A. Valenzano, "A twofold model for the description of access policies in industrial systems," CNR-IEIIT Tech. Rep. CENG/2013-12/01, Torino, Italy, Dec. 2013.

[37] M. Cheminod, L. Durante, L. Seno, and A. Valenzano, "On the description of access control policies in networked industrial systems," in *Proc. 10th IEEE Int. Workshop Factory Communication Systems (WFCS)*, Toulouse, France, May 5–7, 2014, to appear.