Digital Object Identifier TBD

Towards an ideal Access Control Strategy for Industry 4.0 Manufacturing Systems

BJÖRN LEANDER[1,2], (Member, IEEE), AIDA ČAUŠEVIĆ[2], HANS HANSSON[2], (Member, IEEE), AND TOMAS LINDSTRÖM[1]
[1]ABB AB Process Automation, Ängsgärdsg. 6, 721 80 Västerås, Sweden (e-mail: {bjorn.leander, tomas.lindstrom}@se.abb.com)
[2]Mälardalen University, IDT, Box 883, 721 23 Västerås, Sweden (e-mail:{bjorn.leander, aida.causevic, hans.hansson}@mdh.se)
Corresponding author: Björn Leander (e-mail: bjorn.leander@mdh.se).

This work was supported in part by the EU Commission and Vinnova under grant agreement No 876038.

ABSTRACT

Industrial control systems control and supervise our most important and critical infrastructures, such as power utilities, clean water plants and nuclear plants, as well as the manufacturing industries at the base of our economy. These systems are currently undergoing a transformation driven by the Industry 4.0 evolution, characterized by increased connectivity and flexibility.
Consequently, the cybersecurity threat landscape for industrial control systems is evolving as well. Current strategies used for access control within industrial control systems are relatively rudimentary. It is evident that some of the emerging cybersecurity threats related to Industry 4.0 could be better mitigated using more fine-grained access control policies.
In this article we discuss and describe a number of access control strategies that could be used within manufacturing systems. We evaluate the strategies in a simulation experiment, using a number of attack-scenarios. Moreover, a method is outlined for automatic policy-generation based on engineering-data, which is aligned with one of the best performing strategies.

INDEX TERMS Access Control, Cybersecurity, Industry 4.0, Modular Automation

I. INTRODUCTION

MANUFACTURING systems developed within the Industry 4.0 (I4.0) paradigm are evolving both the business and technological side of the manufacturing domain. The business side is focused on *servitization* [1, 2] and *mass customization* [3], while the technical side aims at self-organized systems of autonomous devices, inclusions of Internet and Internet of Things (IoT) technology, use of edge and cloud services, etc.

The automation systems related to the I4.0 domain therefore exhibits a number of new characteristics related to modularization, inclusion of novel technologies, increased number of stakeholder and growing system complexity [4–7]. These characteristics have implications on the security of the system, e.g., making correct behavior less predictable, and rendering some of the traditional security controls based on anomaly detection and white-listing less useful.

A compromised device may cause significant damage within a manufacturing environment, leading to economical as well as safety-related harm. In an Industrial Automation and Control System (IACS), the class of highly motivated and resourceful attacks executed by *Advanced Persistent Threat* (APT) [8] are among the most difficult to counter. These attacks are often orchestrated directly or indirectly through persons or devices residing inside the attacked network.

Strict and fine-grained access control is a security mechanism that can be used to minimize the maneuverability and increase visibility of insider as well as external attackers, thus mitigating several of the issues related to emerging cybersecurity challenges in I4.0 systems. However, access control is relatively undeveloped in the context of IACS, especially for control of Machine-to-Machine (M2M) interactions. Formulating policies and tracking identities is difficult and requires a considerable management efforts for complex and dynamic systems.

Modular Automation (MA) [9] is one of the I4.0 manufacturing system types [10]. It is a subset of the Smart Manufacturing paradigm, focusing on continuous production, in contrast to discrete manufacturing. One of the leading ideas behind MA is to enable adaptable manufacturing within process industries, in order to shorten the lead time between innovation and production, adapt to changing market requirements and customize products. On the production level an MA system is comprised of a number of modules that are built to perform specific tasks autonomously, and orchestrating units that coordinate the high level production scheme. The high level of independence combined with well defined interfaces of the modules allows for arbitrary combinations of modules to fulfill shifting manufacturing requirements.

In this article, five access control strategies for manufacturing systems are presented and graded on a progressive scale towards an ideal strategy. The first three steps in the scale account for currently used strategies, while the last two steps are aligned with requirements of I4.0 manufacturing systems. Based on the fourth step, an algorithm is presented for access control policy formulation intended for an MA system. All the strategies are evaluated using four attack scenarios in a simulation experiment implemented utilizing the JADE framework [11]. The simulations indicate that, in dynamic manufacturing systems, such as MA systems, access control strategies being equally dynamic outperforms the traditional static strategies, from a security perspective.

The remainder of this article is organized as follows. Section II introduces related work. Section III presents a number of strategies for access control. Section IV describes an attack scenario used to evaluate introduced strategies, while Section V contains a comparison of these strategies. In Section VI an algorithm for automated access control policy generation for MA systems is outlined and aligned with one of the discussed strategies. Results from a simulation of the different access control strategies are presented in Section VII. The results of the evaluation and the implications of the suggested approach are discussed in Section VIII. Section IX concludes the paper.

II. RELATED WORK

In the area of access control for modern manufacturing systems, a number of publications are discussing issues closely related to the work presented in this article. Watson *et al.* [12] discuss the use of a number of different access control models in conjunction with the OPC Unified Architecture (OPC UA) [13]. The authors advocate the use of Attribute Based Access Control (ABAC) [14, 15] or a combination of ABAC and Role Based Access Control (RBAC) [16] as a good match for protection against privilege escalation for both inside and outside attackers within IACS. Ruland *et al.* [17] describe an eXtended Access Control Markup-Language (XACML)-based access control system [18] for smart energy grids, including attributes related to system state, allowing to some extent dynamicity with regards to privilege deduction. The main use of the system state is for policy decisions on safety-related functionality. Both these works touch upon our suggested solutions for access control policy formulation. However, none of them support the use cases related to dynamic system composition, being a key characteristic we target in our research.

In the work of Martinelli *et al.* [19], a framework for implementing the Usage Control (UCON) [20] access control model in conjunction with OPC UA is suggested to allow fine-grained access control in industrial control systems. The suggested framework is based on expressing policies in XACML, and describes an enforcement architecture. The work considers access control in systems similar to Modular Automation, but provides mainly solutions related to the enforcement layer, while we provide solutions regarding policy formulation. The suggestion to express policies using UCON could be an interesting replacement of Next Generation Access Control (NGAC) [21] in our approach.

In the area of access control for dynamic and modular systems, there are some proposed solutions. Task Based Access Control (TBAC) [22] is an access control model aiming at limiting privileges to a just-in-time and need-to-do basis, similar to what we try to achieve with respect to authorization within modular manufacturing systems. The idea is to have a set of trustees validating each privilege request. Granting privileges is also limited by expected usage, e.g., number of allowed resource accesses. However, as far as we understand, TBAC never materialized in any expression language or reference implementation, making it an infeasible choice for an industrial system. In our work, we try to reach the same objectives by using a standardized expression language for the policies.

Uddin *et al.* [23] describe a dynamic access control model utilizing XACML and a combination of TBAC and RBAC. The work has similarities with ours, but uses static workflows, and applies to a banking system only containing human initiators, as opposed to our approach with dynamically changing workflows and industrial systems containing autonomous machine-to-machine interactions. In their work they discuss the principle of *segregation of duties*, which also is of importance for certain aspects of IACS.

The work by Bhatt *et al.* [24] builds upon the principles described by Sandhu *et al.* [25] related to the next-generation RBAC, but suggests some alterations and additions in the light of the evolving Smart Community (SC). Argumentation is provided for the need of a Convergent access control model, as required by the emerging SC utilizing IoT technologies, smart healthcare, artificial intelligence, etc. These principles are partly coinciding with the requirements for dynamic manufacturing systems, e.g., dynamic authorization, high level of flexibility and scalability. The article does not provide any particular solutions, but rather presents the principles and requirements, discusses the challenges and suggests future research directions.

Chiquito *et al.* [26] discuss the need for flexible solutions for authorizing access to times-series databases in industrial systems, suggesting the use of NGAC for the policy model description. The use case of providing external access to time-series data is an important aspect of I4.0 systems not directly covered in this article, which instead is focusing on the protection and integrity of the workflows and devices directly involved in process control and supervision.

Process-Aware Information Systems (PAIS) are a common type of systems for managing business processes with use cases such as document handling, procurement, financial, etc. A review of research on security related to PAIS is provided by Leitner *et al.* [27]. One aspect being a common issue in recent PAIS research is on the workflow-satisfiability problem, e.g., discussed by Gutin *et al.* [28], concerning the problem of knowing if a specific workflow can be executed in a system. This brings in the access control requirements related to dynamic segregation-of-duties, which is typically important in PAIS. We have not considered those requirement in this work, but there are scenarios, typically related to safety-critical functionality, where workflows contain actions that must be performed by different principals.

III. THE IDEAL ACCESS CONTROL POLICY, AND DIFFERENT STRATEGIES AIMING TOWARDS IT

Access control is one of the basic cybersecurity mechanisms in a software system. It restricts access to resources only to legitimate and authorized subjects. One of the guiding principles within access control is the *principle of least privilege* [29], stipulating that no entity in the system should hold privileges outside the scope of what is required for the entity to perform its intended tasks within the system. The point of the least-privilege principle is to minimize the harm a system user could do, on purpose, or unintentionally, e.g., through malware. The *principle of containment* [25] is an extension of the least-privilege principle, which also includes separation of duties, rate- and usage-limits, etc.

Access control is divided into three distinct mechanisms [8], as follows:

- Identification: deals with the issue of tying a unique identity to a specific entity.
- Authentication: methods for proving that a specific entity is in possession of an identity.
- Authorization: mechanisms for granting (or denying) an entity privileges on a resource in the system, based on a set of rules.

The rules governing access control are often referred to as an access control policy [30].

Assuming we would be able to formulate any kind of access control policy, the ideal policy should be a set of rules that would only allow an entity to perform actions for a set of precisely and well-defined resources at exactly the time mandated by the currently executing workflow. Furthermore, access to a resource shall never be denied to an entity in need of access to it, unless there are justified reasons for that, e.g., the resource is technically unavailable. According to this ideal policy, no actor would be able to perform actions or read data outside the scope of its current tasks, but would not be denied to perform actions in line with expected tasks.

Implementing the ideal strategy may be very difficult, but a number of incremental policy strategies can be defined aiming towards this goal, in the following way:

A Anyone within the network is trusted.
B Anyone with trusted credentials is trusted.
C Access is allowed to entities within a certain group.
D Entities assigned to a certain workflow are allowed to perform operations contained by the workflow.
E Entities assigned to a certain workflow are allowed to perform operations, following the sequence of the workflow.

The concept of a well-defined workflow that every acting entity in the system is following is of course also an ideal. In many systems such workflows may not be articulated or feasible. There are however situations where workflows are well-defined and formalized. Within manufacturing systems, which are the focus for this article, the production process is typically strictly defined. In the process industries, recipes are used to define the sequence of operations needed to produce a specific substance, and in discrete manufacturing, process scheduling is a common concept.

Different versions of strategies **A**-**C** are widely used in current IACSs. Strategies **D** and **E** are additional steps towards the described ideal policy. To the best of our knowledge, using a workflow as determinant for access control in an industrial control system is not used in any commonly available product. There are however other types of systems where workflow-based access control is used, e.g., in the domain of workflow management systems [31], including for example document life-cycle handling systems.

Further and even more restrictive steps towards the perceived ideal policy could be taken by including rate and usage limits on privileges, following the principle of containment. Such limits could be applied to strategies **C**-**E**. For Strategy **C**, rate limits could e.g., be applied to sensitive operations, and for strategies **D**-**E** usage limits could be applied as described by the workflow. In the evaluations and implementations of policies presented in this work, we have however left these mechanisms out. The reason for that is the added complexity both with regards to formulating policies and for analyzing attack impact. The event ordering between legitimate and illegitimate privilege requests will have influence over the outcome, as well as the selected numbers or time-frames for the limiters. A malicious use of these limiting mechanisms could potentially work as means for an availability attack.

IV. ATTACK SCENARIOS

Let us assume that there is a malicious actor that has somehow gained control over a unit somewhere within a manufacturing system. The goal of the attacker is to perform actions or read data from other devices in the system, without raising alarms related to failed access control requests. This quite closely describes the desired operations of an Advanced Persistent Threat (APT) [8]. A high-level network architecture with a target device and four different attack scenarios is depicted in Fig. 1. The device, denoted as **A.D1**, is the intended attack target in all scenarios. The network is segmented into two logical groups, **A** and **B**. The scenarios are based on different points from which the attacks are launched, and are denoted Scenario 1 - Scenario 4 (**S1-S4**). At the time of the attack there is a running workflow, described by a scheme including devices **A.D1**, **A.D2** and **A.D3**.

Scenario **S1** is launched from device **A.D2**, Scenario **S2** is launched from device **A.D6**, Scenario **S3** is launched from device **B.D2**, and Scenario **S4** is launched from a *Rogue* device. The *Rogue* device is an entity placed in the network as an entry point by a potential attacker, but without holding legitimate credentials. Finally, all attack points are being fully controlled by the attacker, including the device credentials. The attacker is modeled according to the definition declared in IEC 62443-3-3 for Security Level 4: *"... using sophisticated means with extended resources, IACS specific skills and high motivation"* [32], i.e., holding detailed knowledge about the communication protocol, network set-up, etc. This is based on the assumption of an internal attacker, or an attacker informed by internal sources.

V. COMPARISON OF STRATEGIES

This section contains a comparison of the strategies shown in Section III, in the context of a workflow-based system, and different attack scenarios mentioned in Section IV. A formal description of such a workflow-based system, strategies, and how they relate to the least privilege principle is described in Appendix A.

A. STRATEGY A: ANYONE WITHIN THE NETWORK IS TRUSTED

This strategy is what is currently common, at least in the control network of a traditional IACS, as several of the currently used field communication protocols, e.g., Manufacturing Message Specification (MMS) [33], Modbus [34], have no direct support for access control. Any device present in the network can generate traffic that will be parsed and trusted by other entities on the network, as long as it conforms to the expected protocol. The strategy is clearly very far away from the previously formulated ideal policy, but it is a sensible choice for a physically isolated network containing trusted components with hard-wired interconnections.

This strategy acts as access control only in the sense that access is granted exclusively to physically present entities. Therefore, it holds in itself no resilience against the described attack scenarios. If an entity on the network is compromised, it will be allowed to read data and perform actions, as long as the protocol used for communication is followed. This means that the system is susceptible to attacks originating at any of the defined attack points (scenarios **S1 - S4**), including the one from the *Rogue* device.

In this case, the system is not protected by the access control policy, but rather by the physical and logical perimeter protection of the network, and possibly other additional security measures, such as intrusion detection mechanisms.

B. STRATEGY B: ANYONE WITH TRUSTED CREDENTIALS IS TRUSTED

This strategy implies that an entity recognized as a valid member of the system will also be trusted. Depending on the security of the credentials, the strategy can be rather powerful against an attacker. If the credentials cannot be forged or stolen, the attacker must be able to corrupt the device to perform the malicious action while still adding e.g., a correct cryptographic signature.

One example from the OT domain is using Open Platform Communication Unified Architecture (OPC UA)[1] [12, 13] with application instance certificates for inter-device authentication, without specifying privileges for the entities.

It is worth noting that this strategy provides an access control without authorization, i.e., it only includes mechanisms for identification and authentication.

Strategy **B** will be resilient against attacks launched from *Rogue* devices, where the attacker has not been able to steal credentials of a trusted user. Attacks originating from any of the legitimate devices that have been compromised, may be both successful and stealthy, based on the definition of the attackers being in full control of the compromised devices.

Secure device provisioning, secure boot, attestation, malware protection and hardware-protection for certificate keys may be used as additional mitigating methods, making attacks from legitimate devices more difficult to perform.

C. STRATEGY C: ACCESS ALLOWED TO ENTITIES WITHIN A CERTAIN GROUP

This strategy is supported by RBAC [16], and is often used to describe policies for human users in both Information Technology (IT) and OT networks. Typically there are roles for operators, engineers, service personnel, etc., granting appropriate privileges to each group of users. It is not commonly used for device interactions, but it could very well be, e.g., by defining roles for devices allowing only interactions within a group of physically adjacent entities. The strategy relies on Strategy **B** being already in place, i.e., the included entities must have unique identities, and there must be a method for providing proof of identity.

For a static automation system that always performs the same set of operation, this strategy could be implemented to closely match the ideal scenario, except that the order of operations cannot be enforced.

For the sake of this evaluation, Strategy **C** is defined as: Entities within a group are trusted only within that group. This makes sense for horizontal M2M interactions, if the group is defined as a localized section of the network. For the described attack-scenarios, this set-up is meaningful, as all the entities reside at the same horizontal level. Trusting entities only within the group may make less sense for vertical communication, i.e., from the HMI layer down to the

[1]IEC 62541, Part 2, 4.8: OPC UA provides a mechanism to exchange user credentials but does not specify how the applications use these credentials.

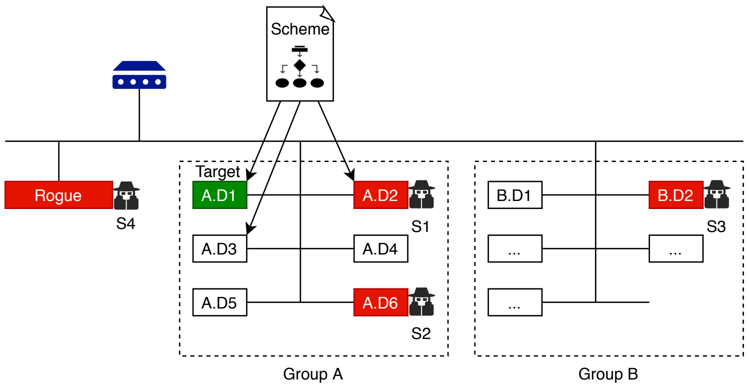

FIGURE 1. A network architecture with annotated attack scenarios.

control network, where typically the upstream entities will hold higher privileges than those downstream.

Accordingly, the strategy will be resilient against attacks launched from *Rogue* devices. Attacks originating from compromised legitimate entities will be limited to targets within the trusted entity's resource group, as decided by the policy. However, within that group, an attacker can freely read data and execute actions. This means that the strategy would give resilience against attacks in scenarios **S3** and **S4**, but not against attacks in the other scenarios.

One clear drawback with this strategy in relation to modern manufacturing systems, is that it limits the possibility to freely restructure the manufacturing system, e.g., combining devices from both groups **A** and **B** in a manufacturing scheme is not possible without altering the groups.

D. STRATEGY D: WORKFLOW-LIMITED ACCESS CONTROL

This strategy limits the permissive operations within the system, to what is described by currently executing workflows. There is however no control on the order of the operations. None of the available access control models have direct support for policy formulation adhering to a workflow scheme; instead, available primitives must be used to formulate rules that follow the strategy.

To the best of our knowledge, there is currently no support for this kind of access control strategy in any commercially available IACS. It is however a necessary step towards the ideal access control policy for systems with a dynamic workflow behavior, i.e., where the entities interactions to fulfill required operations within the system is shifting over time. Such behavior is becoming increasingly common in modern manufacturing systems. Both Smart Manufacturing [35, 36] and Modular Automation [10] are realized through Service Oriented Architecture (SOA) solutions, where included autonomous entities are capable of dynamic re-organization to fulfill current production requirements.

Strategy **D** states that only privileges required by the active workflow will be granted. This means that the system is resilient against attacks from outside the scope of the current workflow, and within the workflow only actions and data flows mandated by the scheme will be allowed. This is indeed a much more restrictive approach than what is allowed by the previously described strategies. The attack would only be successful if the compromised device is scheduled to perform operations on, or if data from the intended target is read within the current workflow. Therefore, only attacks from the attack point **S1** have a chance of being successful, but only if the desired action is permitted by a specific workflow, and the attack occurs while that workflow is executing.

E. STRATEGY E: WORKFLOW-LIMITED ACCESS CONTROL INCLUDING OPERATION SEQUENCES

This strategy is very close to the formulated ideal policy. As far as we know, no available access control model allows this kind of fine-grained temporal behavior, even though there are primitives that can support event-based behaviors in some of the available models, e.g., obligations in NGAC and XACML. For this strategy to be stringently enforceable, the access control enforcement mechanism must be able to execute a copy of the state-machine for the workflow, e.g., as a digital twin.

Worth noticing is that even with this nearly ideal strategy, there are classes of attacks that the system is susceptible to. As an example, a compromised or faulty device could request execution actions according to the defined workflow, but using parameters leading to degraded or faulty products, or possibly safety-related incidents.

Strategy **E** is vulnerable only to attacks in line with the execution of the workflow, implying that only attacks from **S1** have a chance of being successful, but the time-frame to perform the malicious action is limited to precisely when the workflow is executed within a step allowing that action.

F. COMPARISON SUMMARY

Table 1 lists the impact of the attack scenarios on the different strategies. A High (H) impact indicates that the attack will be able to perform any action, read any data, without risk of detection by the access control mechanism. A Medium (M) impact indicates that the set of available actions and data will be limited for an attacker from that point. Low (L) impact indicates that the available time-span to execute actions is additionally highly limited. Finally, character × indicates that the attack (as defined above) is not possible. Of course there are other types of attacks capable to impact the target, such as Denial-of-Service (DoS) attacks, not discussed in this article.

A formalized comparison of the strategies, related to the least privilege principle is provided in Appendix A-I.

Strategy	Attack Scenario			
	S1	**S2**	**S3**	**S4**
A	H	H	H	H
B	H	H	H	×
C	H	H	×	×
D	M	×	×	×
E	L	×	×	×

TABLE 1. Impact of attack scenarios on target **A.D1**, related to the access control strategies (H: high, M: medium, L: low, ×: attack not possible).

As can be seen from Table 1, strategies **A**-**C** are all highly vulnerable for two or more of the described attack scenarios. Strategy **E** has no obvious support among available access control models. This prompts us to look at the possibility of realizing Strategy **D**. Section VI-A outlines an approach implementing this strategy in the context of a Modular Automation system.

VI. PROPOSED ALGORITHM

In a previous study [37], requirements on access control models for use in Smart Manufacturing systems are presented and discussed. One of the main issues raised is the management effort required to uphold a policy framework following the principle of least privilege in dynamic manufacturing systems.

The following section outlines an approach to formulate fine grained access control rules with minimal management effort, within one type of dynamic manufacturing system, with the purpose to alleviate some of the expected challenges related to management effort. The method is described more in detail in a previous technical report [38], and is outlined here as a part-solution aligned with Strategy **D**, and it is applicable for device interactions in a Modular Automation system.

In the following necessary background needed to understand the technological components used in the algorithm is introduced.

At a technical level, the MA system components interact using a SOA following the orchestration concept [39], where the orchestrating unit executes commands on the modules according to the current manufacturing workflow. The workflow schemes are formally described as recipes, and are typically expressed as Sequential Functional Charts (SFC) [9, 40]. SFC is a graph-based programming language typically used in Programmable Logical Controls (PLC). A SFC describes a workflow as a sequence of steps, containing operations and conditioned transitions between the steps. An example of a recipe described by an SFC is provided in Fig. 2. In this work, a simplified model of the SFC is used, assuming that an SFC has a starting step; each step contains a set of zero or more operations, and a step has zero or more immediate successor steps. Each operation of an SFC step is directed at a target module. In relation to the defined attack scenarios, an SFC could be the workflow scheme depicted in Fig. 1.

To minimize the required management effort for policy formulation, an algorithm could automate part of that work, namely formulation of the policies needed for device-to-device interactions within an MA system, which are described by the set of currently active workflows. As the set of active workflows changes over time, so will the access control policies change over time.

To be able to formalize such detailed policies, a flexible enough access control model is required. The ABAC [14] model family is a potentially promising candidate for such detailed policies, as in ABAC, the policy inference is done by evaluating logical expressions using attributes on the subject, object and environment, allowing for a very high flexibility in policy expression.

NGAC is an access control model based on ABAC, and standardized by the American National Institute of Standards and Technologies (NIST) [21, 41]. In NGAC, the policies are defined using a graph, where attributes can form hierarchies, similar to the concept of membership. The attribute hierarchies are described using the term *containment*, annotated so that $attr1 \rightarrow attr2$ indicates that $attr1$ is contained by $attr2$. Allowed operations are modeled as associations between object and subject attributes so that $sa - \{op\} - oa$ indicates that a subject being contained (directly or indirectly) by subject attribute sa is allowed to perform the operation op on any object contained by the object attribute oa.

A. DESCRIPTION

The proposed algorithm takes as input a recipe expressed as an SFC, and generates a sub-graph in NGAC, containing required attributes and privilege associations. Upon recipe activation, i.e., when production using the recipe is started, attributes are assigned to the appointed modules and orchestrator, while on recipe deactivation, attributes are deassigned, effectively limiting the allowed actions within the system to what is prescribed in the currently active recipes. This means that only privileges related to active recipes will be granted.

The algorithm works by traversing the SFC of a recipe, and for each step creates attributes, attribute assignments and associations in the NGAC-graph, as required for the module operations related to that step. After algorithm completion, a NGAC sub-graph is present uniquely representing policies related to the recipe.

In the following, a simplified pseudo-code for the proposed algorithm is presented.

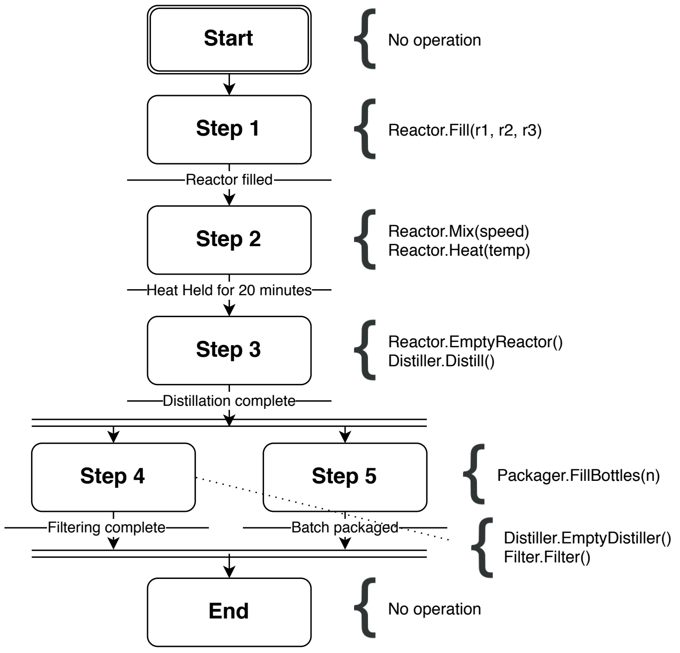

FIGURE 2. An example of a recipe expressed as an SFC

Algorithm 1 `PolicyGeneration`(*Recipe*)

```
1:  function GENERATESTEPPOLICIES(step)
2:      if step has operations then
3:          $step_{id}$ := unique attribute for step,
4:          assign attribute orch to $step_{id}$: $orch \rightarrow step_{id}$
5:          for all operations $op \in step$ do
6:              target := attribute for target of operation
7:              associate $step_{id}$ with target: $step_{id} - \{op\} - target$
8:          end for
9:      end if
10: end function
11:
12: function VISITSTEP(step)
13:     if $\neg$visited(step) then
14:         visit(step)
15:         GenerateStepPolicies(step)
16:         for all consecutive steps $step' \in step$ do
17:             VisitStep(step')
18:         end for
19:     end if
20: end function
21:
22: begin algorithm
23:     orch :=unique attribute for orchestrator of Recipe
24:     VisitStep (start step of Recipe SFC)
25: end algorithm
```

As can be seen, the orchestrating unit and the recipe modules are not directly assigned in the algorithm, instead synthetic attributes are created, *orch* for the orchestrator and unique *target*-attributes for each module. Upon recipe activation, the designated orchestrator is assigned to the *orch* attribute, and each target module is assigned to the respective *target* attribute. As the SFC may contain loops, the marker `visited` is used in the pseudo-code to indicate that a step of the SFC is already processed by the algorithm. Fig. 3 depicts the NGAC sub-graph resulting in the execution of the algorithm on the recipe described by the SFC in Fig. 2, directly after recipe activation. The sub-graph also contains the policy-class *Module Control Policies*, as required by NGAC, but not in detail described in this article.

B. PROOF OVERVIEW

A proof overview indicating that the algorithm creates policies as needed for recipe activation is provided here, details can be found in [38].

The proof uses the deduced relationship `Priv`(*orchestrator*, *recipe*), required of an NGAC-graph for an orchestrator to be allowed to execute exactly the operations defined in a formal description of a recipe. The relationship is based on the definition of granted privileges in NGAC, as described by NIST [21]. For a subject s, a target object o, an operation op and a policy class pc, the privilege of s executing op on o is granted only if there exist an association between a subject attribute sa and an object attribute oa containing operation op, where s is contained by attribute sa, and o is contained by attribute oa, and both s, o, and oa are contained by the same policy class pc.

The proof makes use of the transitive property of the containment operator, i.e.,

$$(a \rightarrow b) \wedge (b \rightarrow c) \implies a \rightarrow c. \quad (1)$$

The theorem states that:

Theorem 1. *Algorithm 1 will create policies fulfilling definition* `Priv`*(subj, R) for a recipe* $R = (id, F)$*, with SFC* F*, an orchestrator* $subj$*, and a set of target modules* T_m*,*

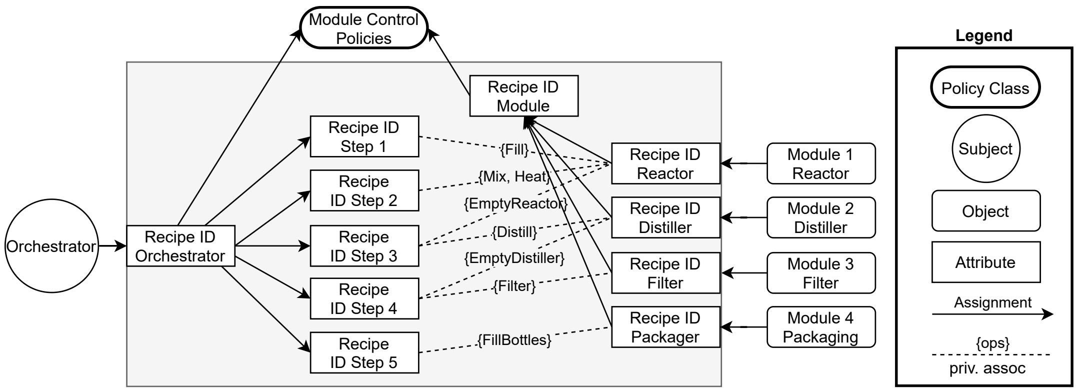

FIGURE 3. Example of NGAC policy and attribute setup for the recipe described in Fig. 2. The sub-graph in the grey rectangle is generated by the algorithm, the other assignments are related to recipe activation. The policy-class is expected to be pre-existing.

using an NGAC graph containing the policy class pc, under assumptions related to attribute assignments to orchestrator and target modules upon recipe activation.

The proof of the theorem is using three lemmas:

Lemma 1 (Policy generation for a single SFC step)**.** *Function `GenerateStepPolicies` generates access control policies fulfilling `Priv`(subj, step) for any step in an SFC used as parameter.*

Lemma 2 (Policy generation for Visited steps)**.** *A step p visited by procedure `VisitStep`$(p, ...)$, will imply that `Priv`(subj, p) is fulfilled.*

Lemma 3 (Policy generation for an SFC)**.** *For a recipe $R = (id, F)$ where SFC $F = (S, s_0)$ with steps S and starting step s_0, a call to function `VisitStep`$(s_0, ...)$ will generate policies such that `Priv`$(subj, R)$ is fulfilled.*

Finally, the theorem is proven to follow by the algorithm invoking the lemmas and the fulfillment of the initial assumptions.

C. ALIGNING THE ALGORITHM WITH STRATEGIES

The use of the suggested algorithm for access control policy generation in an MA system will limit permissive actions for the device interactions to what is described in the active workflow descriptions. However, it does but not consider the ordering of the events, thus aligning the approach with the previously described Strategy **D**. The generative approach using engineering data as input for policy formulations makes this method very light-weight from the management perspective. This algorithm is implemented as part of the strategy simulation described in Section VII, where it is used for Strategy **D**.

With a slight modification, the algorithm is also used to implement Strategy **E**. This is achieved by postponing the assignment between the orchestrator attribute and the recipe step attributes. For the example depicted in Fig. 3 none of the assignment-arrows between 'Recipe ID Orchestrator' and the 'Recipe ID Step X' attribute would be present after the algorithm according to Strategy **E** is executed. These attribute assignments are instead driven by the orchestrating unit, which will assign the orchestrator-attribute to the attribute(s) representing the active step(s). Intuitively, this would make the solution less secure, as the method is designed to limit the privileges of the orchestrator. The approach could however be useful, if the attribute assignment operation is monitored and audited in a way so that suspicious behavior would be visible. It is also one way of supporting the principle of automation, as described by Sandhu *et al.* [25].

VII. SIMULATION

To provide evidence on the analytical results discussed in Section V-F, a simulation experiment is designed. Strategies **A** - **E** are implemented and evaluated in a simulated MA environment, based on the JADE framework [11]. The enforcement architecture used in the simulation environment is very rudimentary, with each agent holding a complete set of the policy data, a policy decision point and a policy enforcement point. Strategies **A** - **C** are implemented as static access control matrices, Strategy **D** is implemented using the algorithm outlined in Section VI, utilizing the Policy Machine[2] for the NGAC representation. Strategy **E** is realized as an extension of Strategy **D**, with assignments to recipe step attributes delayed until the orchestrator is ready to start executing the corresponding step.

The simulation environment contains most aspects of an MA system that are relevant in the context of this work, including recipe activation, execution and deactivation. All the described attack scenarios are also included in the simulated environment, modeled as a change of behavior of the attacking component at arbitrary points in time during simulation.

A. SIMULATION RESULTS

The simulation execution is identical between each strategy and attack scenario, and is run according to the following schedule: 1) an operator activates a recipe, and orders an

[2]https://github.com/PM-Master/policy-machine-core

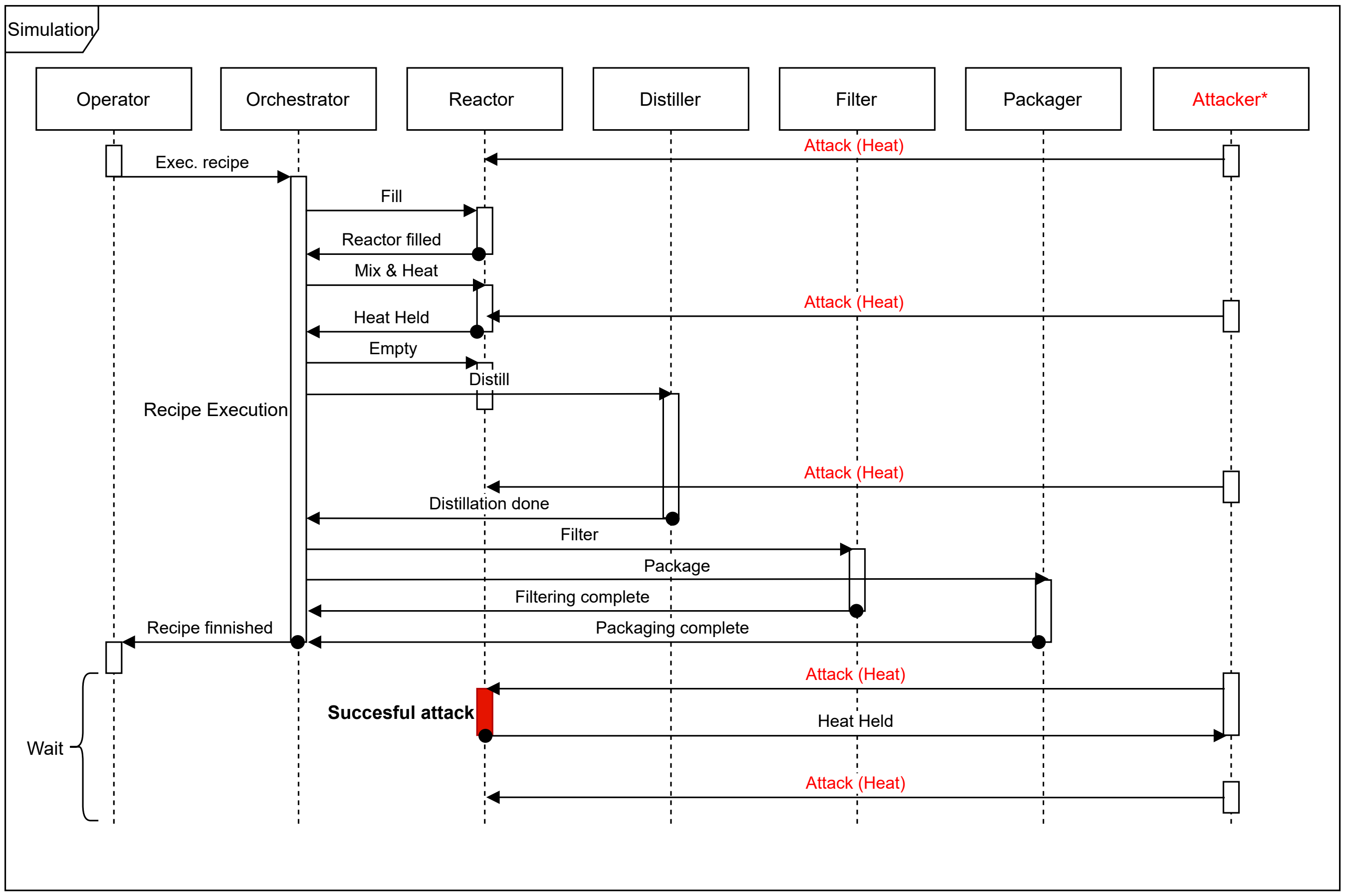

FIGURE 4. A sequence diagram of a simulation. The occurrence of attempted and successful attacks in the diagram are merely illustrative.

orchestrator to execute it, 2) the orchestrator executes the recipe by iterating all the steps in the correlating SFC (duration about 2:10 min), 3) the operator de-activates the recipe, and there is a waiting time (duration about 1:30 min), after which 4) the schedule is repeated. At every 0.5s during the simulation there is a 1/25 probability that an attack will be launched. This interval is chosen substantially shorter than the timing of the applications, and since the JADE agents have randomly distributed start times, this ensure that attacks can occur at potentially any time during the simulation execution. The simulation is ended after a fixed amount of attack attempts, which occurs after approx. 20 mins. Fig. 4 shows the sequence diagram for a recipe-execution cycle of a simulation that executes the SFC in Fig. 2. In the simulation, the Heat-method of the reactor module is used as the attack target. Each module will have one or more signals used to indicate that an action is finalized. The orchestrator reads these signals and uses them for triggering step transitions. Therefore, the arrows pointing from the modules towards the orchestrator are a simplified indication of a changing value for a signal being continuously read by the orchestrator. Similarly, the amount of attack attempts visualized in the sequence diagram are reduced for readability reasons.

The measurement collected from the simulation is the percentage of successful attacks. The simulation results (see Table 2) in principle corroborate on the analytical results discussed in Section V. As expected, the ratio of successful attacks for **S1**, Strategy **D**, correlates with the time the recipe is active. Likewise for Strategy **E**, the amount of time that the step containing the attack target action is active correlates with the amount of successful attacks from **S1**. The attacker is modeled to have full knowledge of the system, and the system is implemented with access control as the only security measure. The ideal modeling of the attacker and the system will result in a worst case scenario, where the only determinant for a successful attack is the selected access control strategy. This explains the binary results of either 100% or 0% successful attacks for the static access control strategies (Strategy **A**-**C**). In a real-world example the amount of successful attacks would most likely be lower for all attack scenarios, due to mistakes done by the attacker, additional security measures and other technical particularities in the system.

B. DISCUSSIONS ON VALIDITY

In this experiment, the number of successful attacks is used as a measurement for evaluation. This measurement is meaningful from the perspective of providing a comparable number for ranking the strategies, it is however only one

Strategy	Attack Scenario			
	S1	**S2**	**S3**	**S4**
A	100%	100%	100%	100%
B	100%	100%	100%	0%
C	100%	100%	0%	0%
D	58% $\sigma = 2.4\%$	0%	0%	0%
E	19% $\sigma = 2.7\%$	0%	0%	0%

TABLE 2. Average percentage of successful attacks per strategy and scenario, standard deviation (σ)=0 unless otherwise stated. Results are based on 7-10 executions of each combination of strategy and attack scenario.

of several possible aspects to consider when evaluating an access control strategy. Several quantitative and qualitative aspects exist, such as administrative transparency, speed of inference, and perceived management effort of the strategy. While these are important factors, the simulation model is not designed to provide any measures of them.

The part of the simulation environment representing the physical behavior of the system is very simple, and is lacking some important characteristics, such as material flow between modules. The simulation is therefore not credible from the perspective of how the attack scenarios impact the physical environment. This is not the goal of this experiment, but could be an interesting future expansion.

The provided numerical results are average values from a limited number of runs of the simulation (7-10 executions for each combination of strategy and attack scenario). However, as the numerical results are only valid for the specific recipe and simulation schedule used in the recipe - these numbers are to be seen mainly as indicative examples. For strategies **D** and **E** the exact selection of attack point for **S1** has an impact on the result. Selecting a device that does not trigger the attack operation on the target as part of normal recipe execution would result in 0% successful attack attempts for both these strategies.

Even though the simulation is limited to a simple case of orchestrating a recipe in an MA system, the results could be used as indications of the effectiveness of evaluated strategies also in a wider context.

Please note that the simulation environment is currently under development, but can be made available on request.

VIII. DISCUSSION

The currently used strategies for access control (Strategy **A**-**C**) provide some basic security for inter-device interactions in traditional IACS. They are however highly sensitive to the attacks described in this article, considering the attacker as a knowledgeable insider. In the manufacturing systems prescribed by I4.0, the probability of an internal device on the network being compromised is increasing, which follows from the following factors:

- Increased attack surface, due to interconnections between devices, systems and the outside world [4].
- Increased flexibility and dynamicity leading to a difficulty in detecting anomalous system behavior [5].
- New groups of stakeholders and users of data, and functionality within the system [7, 42], consequently increasing the potential for social engineering attacks.

Therefore, security measures providing protection against insider threats are more relevant now than ever. One obvious way to decrease maneuverability and increase visibility of an insider attacker is by fine-grained access control. We argue that aiming towards Strategy **E** is a desirable target for increased security, as illustrated by the evaluation of the strategies against the attack scenarios in Section IV. Our proposed approach of reaching Strategy **D** is a first attempt towards that target.

The algorithm described in Section VI is a specific example useful for orchestration of an MA system, but there is a potential in generalizing this approach for use in other types of formal workflow descriptions. In that way, other dynamically re-configurable systems could benefit from a more fine-grained access control strategy.

There are aspects of SFC recipes that cannot be captured by the suggested method for policy generation, e.g., related to the difficulty to express transitions between steps. Another aspect is the actual logic within one step of the SFC. There may be IF conditions or loops that surrounds the module operations with additional logic, something not captured by the Access Control logic. The third aspect are parameters used for operations. A parameter set using a malicious value in an otherwise valid function call could have a harmful effect on the system.

As mentioned, there are other types of attacks and different attack objectives than those described in this article. For example, a DoS-attack could be successfully launched from any of the suggested attack points. Network data could be intercepted (and potentially altered) by a compromised network device. A compromised engineering station could create faulty and potentially dangerous workflows. For all of these types of attacks, fine grained Access Control will not provide direct mitigation. We should rather see Access Control as part of a *Defense in Depth* strategy [32], working together with a number of mutually reinforcing mechanisms, e.g., malware detection, perimeter protection, physical security, integrity and confidentiality of data.

Another potential threat that none of the described strategies will offer protection against is a faulty or maliciously formulated workflow. Other protective mechanisms are required to counter such a threat. These mechanisms are of great importance, but outside the scope of this article.

The described strategies (Strategy **B**-**E**), and the approach formulated in this article, are presented under the assumption that a well implemented policy enforcement architecture is in place, as well as that the mechanisms for identification and authentication are implemented following best practices.

IX. CONCLUSION

Outlined in this article are a number of strategies for access control in industrial control systems. The different strategies have been evaluated using a number of attack scenarios in the context of recipe orchestration in a Modular Automation system. We have also presented an algorithm for access control policy generation in the context of Modular Automation systems.

In modern manufacturing, especially in the I4.0 paradigm, cybersecurity is one of the key aspects for providing trustworthy systems. The strategies for access control described in this article are the currently used strategies in IACS, as well as potential future developments. We argue that the current strategies (Strategy **A**-**C**) are far from the ideal solution for dynamic manufacturing systems. Our suggested approach, in line with Strategy **D**, is a solution that will provide policies closer to the *least privilege principle*, without increasing the management effort related to formulation of access control policies.

As the next step, we are currently implementing the presented algorithm in a real system, using OPC UA as the communication protocol. This will provide answers on how to create a working access control enforcement architecture in a modern manufacturing system, as well as provide answers on scalability and usability of the suggested approach. We additionally intend to further develop the algorithm towards alignment with Strategy **E**.

Future directions of the work include investigating access control policy generation for other dynamic systems, e.g., in smart manufacturing using Petri-nets as a formalized workflow description, and introducing access control according to Strategy **D** and **E** in additional domains.

ACKNOWLEDGMENT

This work is supported by ABB AB; the industrial postgraduate school Automation Region Research Academy (ARRAY), funded by The Knowledge Foundation; and the Horizon 2020 project InSecTT. InSecTT (www.insectt.eu) has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876038. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Sweden, Spain, Italy, France, Portugal, Ireland, Finland, Slovenia, Poland, Netherlands, Turkey.

The document reflects only the author's view and the Commission is not responsible for any use that may be made of the information it contains.

.

APPENDIX A FORMALIZATION AND STRATEGY COMPARISON

In this appendix a formal definition is given for a time-dependent system using workflows, together with a formal definition of the least privilege for this type of system. Furthermore, strategies **A** - **E** are described and compared to the least privilege. The formalism used in this section is inspired by the work by Bell and LaPadula [43].

A. PRELIMINARY DEFINITIONS

The unions of vectors of sets of the same type is treated as element-wise unions, if the vectors are of the same size, i.e.,

$$P \cup Q = \{\{p_1 \cup q_1\}, ...\{p_n \cup q_n\}\}, \tag{2}$$

and undefined for vectors of different size.

B. SYSTEM DESCRIPTION

Assume that we have a workflow-based system:

$$X = (S, O, A, W, T), \tag{3}$$

where S are subjects, O objects, A actions, W valid workflows and T is an ordered set of discrete points in time, representing times when the state of the system changes[3].

All possible privileges in this system are defined as

$$P = S \times A \times O, \tag{4}$$

and a privilege $p \in P$ is denoted:

$$p = s \xrightarrow{a} o, \tag{5}$$

which is read: subject $s \in S$ is allowed to perform action $a \in A$ on object $o \in O$.

A workflow $w \in W$ is defined as an ordered set of steps E, where each step consists of a set of operations expected to be executed during this step, and a function L defining which steps are active at a specific point in time.

$$w = (E, L) \text{ where } (e \in E) \subset P \text{ and } L : T \to 2^E \tag{6}$$

Stating that the workflow contains a set of steps E, where each step $e \in E$ is a subset of P, i.e., each step consist of 0 or more allowed operations. Furthermore, we will for illustration purposes use a matrix (a binary activation matrix) notation to represent the function L, describing for each discrete point in time which of the steps of the workflow are active. The product between the binary matrix $\mathbf{L}$ and the set E results in a vector, where each item in the vector represents a set of valid operations for a given point in time.

Such time indexed vectors are sub-scripted with a capital T, e.g., X_T, and the notation $X(t_i)$ is used to refer to an element in X_T related to the discrete time $t_i \in T$.

The theoretical least privilege for such a system is clearly time-dependent, and can be defined as the union of all privileges being required by all currently active workflow-steps:

$$P_{l,T} = \bigcup_{\forall w_i \in W} \mathbf{L_i} E_i \text{ where } w_i = (E_i, \mathbf{L_i}), \tag{7}$$

yielding a time-indexed vector containing all required privileges for the system for each discrete point in time.[4]

[3]The discrete points in time in principle corresponds to meaningful states of the system, so T is not defined by monotonically fixed increments.

[4]The additions, being the result of the matrix multiplication, are between sets of the same type. This addition will be treated as the union ($\cup$) operator.

C. EXAMPLE

Assume a system described by the following sets:

$$S = \{s_1, s_2\} \quad (8)$$

$$O = \{o_1, o_2, o_3\} \quad (9)$$

$$A = \{a_1, a_2\} \quad (10)$$

$$W = \{w_1, w_2\} \quad (11)$$

$$T = \{t_1, t_1, ...t_6, ..., t_n\} \quad (12)$$

$$w_1 = (\left\{\{s_1 \xrightarrow{a_1} o_2\}, \{s_1 \xrightarrow{a_2} o_2, s_1 \xrightarrow{a_2} o_3\}\right\}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ 0 & 0 \\ \vdots & \vdots \end{bmatrix}) \quad (13)$$

$$w_2 = (\left\{\{s_2 \xrightarrow{a_1} o_1, s_1 \xrightarrow{a_1} o_1\}, \{s_2 \xrightarrow{a_2} o_1\}\right\}, \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \end{bmatrix}) \quad (14)$$

The matrices representing $\mathbf{L}$ should be read as each row representing a discrete instance in time $t_1, \ldots, t_n$, and each column representing a step in E. For example the first row in (13) indicates that at time t_1 the first step of the workflow is active.

Least privilege for our example is as follows:

$$P_l(t_1) = \{s_1 \xrightarrow{a_1} o_2\} \quad (15)$$

$$P_l(t_2) = \{s_1 \xrightarrow{a_2} o_2, s_1 \xrightarrow{a_2} o_3\} \quad (16)$$

$$P_l(t_3) = \{s_1 \xrightarrow{a_2} o_2, s_1 \xrightarrow{a_2} o_3, s_2 \xrightarrow{a_1} o_1, s_1 \xrightarrow{a_1} o_1\} \quad (17)$$

$$P_l(t_4) = \{s_2 \xrightarrow{a_1} o_1, s_1 \xrightarrow{a_1} o_1\} \quad (18)$$

$$P_l(t_5) = \{s_2 \xrightarrow{a_2} o_1\} \quad (19)$$

$$P_l(t \geq t_6) = \{\} \quad (20)$$

In the following we will discuss the privileges granted by different strategies **A**-**E**, and how they compare to each other and to $P_{l,T}$, using our simple example.

D. STRATEGY A

Strategy **A** states that anyone who can talk in the system is trusted, i.e., the privileges permitted by Strategy **A**, P_A, contains all the permissions described in P, but would also permit operations for subjects outside S, "unknown" subjects, as long as they are connected to the system.

$$S_\delta = S \cup S_{unknown} \quad (21)$$

$$P_A = S_\delta \times A \times O \quad (22)$$

$$P \subset P_A \quad (23)$$

Furthermore, P_A is static over time, so transforming to a time-dependent vector results in a stuttering vector:

$$P_{A,T} = \{P_A, P_A, \ldots, P_A\} \quad (24)$$

E. STRATEGY B

Strategy **B** states that we trust known subjects, which in principle means that permissions granted by Strategy **B** is equal to P:

$$P_B = P \quad (25)$$

The time-dependent vector representation is analog with previous example:

$$P_{B,T} = \{P_B, P_B, \ldots, P_B\} \quad (26)$$

F. STRATEGY C

Strategy **C** is based on RBAC. Let us assume that we can define a role as a set of subjects and a set of object-actions. All roles R defined in the system could then be expressed as follows:

$$R = \{r_1, r_2, \ldots r_n\} \text{ where } r_i = (S_i, v_i)$$
$$\text{such that } S_i \subset S \text{ and } v_i \subset A \times O \quad (27)$$

In this case each element in v_i is representing "execution of action a on object o".

The granted privileges by the RBAC strategy can be expressed as:

$$P_{RBAC} = \bigcup_{\forall r_i \in R} S_i \times v_i \quad (28)$$

If we go back to our example, and express the roles as close to the least privilege as possible, the following could be the result:

$$R = \{r_1, r_2\} \quad (29)$$

$$r_1 = (\{s_1\}, \{\xrightarrow{a_1} o_2, \xrightarrow{a_2} o_3,\}) \quad (30)$$

$$r_2 = (\{s_1, s_2\}, \{\xrightarrow{a_1} o_1, \xrightarrow{a_2} o_1\}) \quad (31)$$

The permitted privileges using these role definitions would be:

$$P_C = \{s_1 \xrightarrow{a_1} o_2, s_1 \xrightarrow{a_2} o_3, s_1 \xrightarrow{a_1} o_1, s_1 \xrightarrow{a_2} o_1,$$
$$s_2 \xrightarrow{a_1} o_1, s_2 \xrightarrow{a_2} o_1\} \quad (32)$$

The time-dependent vector representation is analog with previous examples:

$$P_{C,T} = \{P_C, P_C, \ldots, P_C\} \quad (33)$$

In our example and in most scenarios (although there are simple scenarios in which they coincide) this set of privileges is strictly bigger than the least privilege for all points in time: $P_{l,T} \subset P_{C,T}$. Furthermore, $P_C \subset P$.

G. STRATEGY D

Strategy **D** permits all actions in active workflows. The set of active workflows $W_{A,T}$ is time-dependent and can be expressed as the product between the binary activation matrix corresponding to L and the set of workflows W, similarly as the step-activation mechanism for active steps in workflows, previously described:

$$W_{A,T} = \mathbf{Y}W, \tag{34}$$

where the elements of matrix $\mathbf{Y}$ are calculated using the cardinality of elements in the $(E\mathbf{L})_T$-vector of each workflow, as follows:

$$\mathbf{Y}_{i,j} = \begin{cases} 1 \text{ if } (|(\mathbf{L_i}E_i)(t_j)| > 0) \\ 0 \text{ otherwise} \end{cases}$$
$$\text{for } w_i = (E_i, \mathbf{L_i}) \text{ and } i \in (1, ...|W|), j \in (1, ...(|T|)) \tag{35}$$

That is, workflow w_i is part of the set of active workflows, if the number of elements in the $E_i\mathbf{L_i}$-vector related to the current point in time is greater than 0.

The permissions granted by Strategy **D** can be expressed as:

$$P_{D,T} = \mathbf{Y}H, \text{ where } H = (h_1, h_2, ...h_{|W|})$$
$$\text{and } h_i = \bigcup_{e \in E_i} e \text{ for } (w_i \in W) = (E_i, L_i) \tag{36}$$

In our example:

$$W_{A,T} = \Big\{\{w_1\}, \{w_1\}, \{w_1, w_2\}, \{w_2\}, \{w_2\}, \{\}, \dots\Big\}, \tag{37}$$

or expressed in matrix form:

$$W_{A,T} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} \tag{38}$$

which leads to the following privileges granted according to Strategy **D** ($P_{D,T}$)

$$P_D(t_1) = P_D(t_2) = \{s_1 \xrightarrow{a_1} o_2, s_1 \xrightarrow{a_2} o_2, s_1 \xrightarrow{a_2} o_3\} \tag{39}$$

$$P_D(t_3) = \{s_1 \xrightarrow{a_1} o_2, s_1 \xrightarrow{a_2} o_1, s_1 \xrightarrow{a_2} o_3, s_1 \xrightarrow{a_1} o_1, s_2 \xrightarrow{a_1} o_1, s_2 \xrightarrow{a_2} o_1\} \tag{40}$$

$$P_D(t_4) = P_D(t_5) = \{s_1 \xrightarrow{a_1} o_1, s_2 \xrightarrow{a_1} o_1, s_2 \xrightarrow{a_2} o_1\} \tag{41}$$

$$P_D(t \geq t_6) = \{\} \tag{42}$$

H. STRATEGY E

In Strategy **E**, only privileges related to the active steps are granted, implying that $P_{E,T} = P_{l,T}$.

I. STRATEGY COMPARISON

We define a set of workflows W as being "realistic" if there is at least one point in time where not all included workflows are active, i.e., $\exists t \in T$ for which $W_A(t) \subset W$, and at least one point in time where not all steps of all included workflows are active, i.e.,

$$\exists w \in W, \exists(i,j) \in (1, ..., |T|), w = (E, \mathbf{L})$$
$$\text{so that } (\mathbf{L}E)(t_i) \neq (\mathbf{L}E)(t_j) \tag{43}$$

For the time-dependent strategies, the following holds

$$P_{l,T} = P_{E,T} \tag{44}$$
$$P_{E,T} \subset P_{D,T} \subset P_{C,T}, \tag{45}$$

for "realistic" sets of workflows. Furthermore, for the static strategies:

$$P_C \subset P_B \subset P_A \tag{46}$$

Therefore we can say that, based on this system definition, Strategy **E** will result in the set of privileges closest matching the least privilege, followed by Strategy **D**. Using Strategy **C** results in the closest match to the least privilege for the static strategies, followed by Strategy **B**. Strategy **A** is furthest away from the least privilege, as it allows operations also for "unknown" subjects.

For any realistic systems, i.e., systems where the set of active workflows differ over time, the time varying access control strategies (Strategy **D** and **E**) outperforms the static strategies (Strategy **A**, **B**, **C**).

REFERENCES

[1] K.-d. Thoben, S. Wiesner, and T. Wuest, "Industrie 4.0 and smart manufacturing – a review of research issues and application examples," Intl. Journal of Automation Technology, January 2017.

[2] S. Vandermerwe and J. Rada, "Servitization of business: Adding value by adding services," European Management Journal, vol. 6, no. 4, pp. 314 – 324, 1988.

[3] Y. Lu, "Industry 4.0: A survey on technologies, applications and open research issues," Journal of Industrial Information Integration, vol. 6, pp. 1 – 10, 2017.

[4] M. Waidner and M. Kasper, "Security in industrie 4.0 - Challenges and solutions for the fourth industrial revolution," Design, Automation and Test in Europe Conference and Exhibition, DATE, pp. 1303–1308, 2016.

[5] N. Tuptuk and S. Hailes, "Security of smart manufacturing systems," Journal of Manufacturing Systems, vol. 47, pp. 93–106, April 2018.

[6] S. Mumtaz et al., "Massive Internet of Things for Industrial Applications: Addressing Wireless IIoT Connectivity Challenges and Ecosystem Fragmentation," IEEE Ind. Elec. Magazine, vol. 11, no. 1, 2017.

[7] A. Kusiak, "Service manufacturing: Basic concepts and technologies," Journal of Manufacturing Systems, vol. 52, pp. 198–204, 2019.
[8] E. D. Knapp and J. T. Langill, Industrial Network Security: Securing critical infrastructure networks for smart grid, SCADA, and other Industrial Control Systems. Syngress, 2014.
[9] J. Ladiges, A. Fay, T. Holm, U. Hempen, L. Urbas, M. Obst, and T. Albers, "Integration of modular process units into process control systems," IEEE Transactions on Industry Applications, vol. 54, pp. 1870–1880, March 2018.
[10] ZVEI—German Electrical and Electronic Manufacturers' Association, "Module-based production in the process industry—effects on automation in the "Industrie 4.0" environment," White Paper, March 2015.
[11] F. Bellifemine, A. Poggi, and G. Rimassa, "JADE: A FIPA2000 compliant agent development environment," in 5th Intl. Conference on Autonomous Agents, p. 216–217, Association for Computing Machinery, 2001.
[12] V. Watson, J. Sassmannshausen, and K. Waedt, "Secure granular interoperability with opc ua," in INFORMATIK 2019: 50 Jahre Gesellschaft für Informatik – Informatik für Gesellschaft (Workshop-Beiträge), (Bonn), pp. 309–320, Gesellschaft für Informatik e.V., 2019.
[13] "OPC unified architecture," standard, Internation Electrotechnical Commission, Geneva, CH, 2016.
[14] E. Yuan and J. Tong, "Attributed based access control (ABAC) for web services," in IEEE Intl. Conference on Web Services, vol. 2005, pp. 561–569, 2005.
[15] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, "Guide to Attribute Based Access Control (ABAC) Definition and Considerations," tech. rep., NIST, 2014.
[16] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Computer role-based access control models," Computer, vol. 29, no. 2, pp. 38–47, 1996.
[17] C. Ruland and J. Sassmannshausen, "Access Control in Safety Critical Environments," in 12th Intl. Conference on Reliability, Maintainability, and Safety, pp. 223–229, IEEE, 2018.
[18] "eXtensible Access Control Markup Language (XACML) version 3.0 plus errata 01," standard, OASIS, 2017.
[19] F. Martinelli, O. Osliak, P. Mori, and A. Saracino, "Improving security in industry 4.0 by extending OPC-UA with usage control," in 15th Intl. Conference on Availability, Reliability and Security, ACM, 2020.
[20] J. Park and R. Sandhu, "The UCON$_{ABC}$ usage control model," ACM Transactions on Information and System Security, vol. 7, no. 1, pp. 128–174, 2004.
[21] D. Ferraiolo, S. Gavrila, and W. Janse, "Policy Machine: Features, Architecture and Specification," white paper, NIST, October 2015.
[22] R. K. Thomas and R. S. Sandhu, "Task-based authorization controls (TBAC): A family of models for active and enterprise-oriented authorization management," in Database Security XI, pp. 166–181, Springer, 1998.
[23] M. Uddin, S. Islam, and A. Al-Nemrat, "A Dynamic Access Control Model Using Authorising Workflow and Task-Role-Based Access Control," IEEE Access, vol. 7, pp. 166676–166689, 2019.
[24] S. Bhatt and R. Sandhu, "Convergent Access Control to Enable Secure Smart Communities," pp. 148–156, 2020.
[25] R. Sandhu and V. Bhamidipati, "The ASCAA principles for next-generation role-based access control," ARES 2008 - 3rd International Conference on Availability, Security, and Reliability, Proceedings, 2008.
[26] A. Chiquito, U. Bodin, and O. Schelén, "Access Control Model for Time Series Databases using NGAC," IEEE International Conference on Emerging Technologies and Factory Automation, ETFA, vol. 2020-September, pp. 1001–1004, 2020.
[27] M. Leitner and S. Rinderle-Ma, "A systematic review on security in process-aware information systems – constitution, challenges, and future directions," Information and Software Technology, vol. 56, no. 3, pp. 273 – 293, 2014.
[28] G. Gutin and D. Karapetyan, "Constraint branching in workflow satisfiability problem," Proceedings of ACM Symposium on Access Control Models and Technologies, SACMAT, pp. 93–103, 2020.
[29] J. Saltzer and M. Schroeder, "The protection of information in computer systems," in IEEE, vol. 63, pp. 1278–1308, September 1975.
[30] R. S. Sandhu and P. Samarati, "Access control: Principles and Practice," IEEE Communications Magazine, vol. 32, pp. 40–48, September 1994.
[31] E. Bertino, E. Ferrari, and V. Atluri, "The specification and enforcement of authorization constraints in workflow management systems," ACM Transactions on Information and System Security, vol. 2, no. 1, pp. 65–104, 1999.
[32] "IEC 62443 security for industrial automation and control systems," standard, Internation Electrotechnical Commission, Geneva, CH, 2009-2018.
[33] K. Schwarz, "Introduction to the Manufacturing Message Specification (MMS, ISO/IEC 9506)." www.nettedautomation.com/standardization/ISO/TC184/SC5/WG2/mms_intro/, 2000. Online; Accessed: 2020-08-15.
[34] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the modbus protocols," Intl. Journal of Critical Infrastructure Protection, vol. 1, pp. 37 – 44, 2008.
[35] S. Mittal, M. A. Khan, and T. Wuest, "Smart manufacturing: Characteristics and technologies," in Product Lifecycle Management for Digital Transformation of Industries, pp. 539–548, Springer Intl. Publishing,

2016.
[36] J. Davis, T. Edgar, J. Porter, J. Bernaden, and M. Sarli, "Smart manufacturing, manufacturing intelligence and demand-dynamic performance," Computers and Chemical Engineering, vol. 47, pp. 145–156, 2012.
[37] B. Leander, A. Čaušević, H. Hansson, and T. Lindström, "Access control for smart manufacturing systems," in Software Architecture, pp. 463–476, Springer Intl. Publishing, 2020.
[38] B. Leander, A. Čaušević, and H. Hansson, "A recipe-based algorithm for access control in modular automation systems," Mälardalen Real-Time Research Centre, Mälardalen University, September 2020.
[39] C. Peltz, "Web services orchestration and choreography," Computer, vol. 36, pp. 46–52, October 2003.
[40] "IEC 61131-3:2013 Programmable Controllers - Part 3: Programming Languages," standard, IEC, 2013.
[41] D. Ferraiolo, R. Chandramouli, R. Kuhn, and V. Hu, "Extensible Access Control Markup Language (XACML) and Next Generation Access Control (NGAC)," ACM International Workshop on Attribute Based Access Control, pp. 13–24, March 2016.
[42] Y. Lu and F. Ju, "Smart manufacturing systems based on cyber-physical manufacturing services," IFAC-PapersOnLine, vol. 50, no. 1, pp. 15883–15889, 2017.
[43] D. Bell and L. LaPadula, "SECURE COMPUTER SYSTEMS: MATHEMATICAL FOUNDATIONS," Tech. Rep. MTR-2547, The MITRE Corporation, Bedford, Massachsetts, 1973.

BJÖRN LEANDER (M'20) Graduated as a M.Sc in Computer Science and Control Theory, from Luleå University of Technology in 2002. In 2019 he continued his academic career as an industrial PhD Student at Mälardalen University as part of the ARRAY postgraduate school. His research area is within the intersection of industrial automation systems, the industrial internet and cybersecurity, with an orientation toward access control. He is also a cybersecurity engineer at ABB Process Automation PCP, working mainly in R&D projects related to engineering and control.

AIDA ČAUŠEVIĆ is a control safety and cybersecurity engineer at CoE Propulsion Control at Alstom. She received Ph.D. from Mälardalen University, Sweden in 2014 and M.Sc from University of Sarajevo, B&H in 2007 in Computer Science. She has been promoted to Docent in November 2020 at Mälardalen University. Her academic contributions include domains of formal modeling and analysis, safety-critical engineering, and security informed safety-critical Cyber-Physical Systems.

HANS HANSSON Prof. Hans Hansson is full professor at MDH since 1997, as well as scientific leader at the research institute RISE in Västerås since 2014. In recent years he has been a driving force behind research efforts into functional safety and cybersecurity in industrial settings, as well online educational efforts focusing on a broad range of Engineering domains, primarily aimed at practitioners in the business sector. Hans' has additionally a track record of research contributions related to component-based design, real-time communication, real-time testing and debugging, execution-time analysis, automotive control software, and formal modeling.

TOMAS LINDSTRÖM Tomas Lindström has a M.Sc in Applied Physics and Electrical Engineering from Linköping Institute of Technology in 1991. He has over 25 years of experience working with development of industrial control systems, over the years focusing more and more on cybersecurity. Since 2010 he has held leading roles for cybersecurity within one of ABB's units, currently as Head of Cybersecurity in Process Control Platform within the Business Area Process Automation, working with cybersecurity for the products, projects and services from ABB, mainly related to ABB's Distributed Control Systems. He frequently cooperates with researchers from various universities in Sweden, e.g. in advisory boards.