Received: 31 January 2018 | Revised: 23 August 2018 | Accepted: 24 September 2018

DOI: 10.1002/rnc.4375

SPECIAL ISSUE ARTICLE

WILEY

Security measure allocation for industrial control systems: Exploiting systematic search techniques and submodularity

Jezdimir Milošević[1] | André Teixeira[2] | Takashi Tanaka[3] | Karl H. Johansson[1] | Henrik Sandberg[1]

[1]Department of Automatic Control, School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Stockholm, Sweden

[2]Signals and Systems, Department of Engineering Sciences, Uppsala University, Uppsala, Sweden

[3]Department of Aerospace Engineering and Engineering Mechanics, The University of Texas at Austin, Austin, Texas

Correspondence
Jezdimir Milošević, Malvinas väg 10, 100 44 Stockholm, Sweden.
Email: jezdimir@kth.se

Funding information
The Swedish Civil Contingencies Agency (CERCES project); the Swedish Research Council; The Swedish Foundation for Strategic Research; Knut och Alice Wallenbergs foundation

Summary

To protect industrial control systems from cyberattacks, multiple layers of security measures need to be allocated to prevent critical security vulnerabilities. However, both finding the critical vulnerabilities and then allocating security measures in a cost-efficient way become challenging when the number of vulnerabilities and measures is large. This paper proposes a framework that can be used once this is the case. In our framework, the attacker exploits security vulnerabilities to gain control over some of the sensors and actuators. The critical vulnerabilities are those that are not complex to exploit and can lead to a large impact on the physical world through the compromised sensors and actuators. To find these vulnerabilities efficiently, we propose an algorithm that uses the nondecreasing properties of the impact and complexity functions and properties of the security measure allocation problem to speed up the search. Once the critical vulnerabilities are located, the security measure allocation problem reduces to an integer linear program. Since integer linear programs are NP-hard in general, we reformulate this problem as a problem of minimizing a linear set function subject to a submodular constraint. A polynomial time greedy algorithm can then be applied to obtain a solution with guaranteed approximation bound. The applicability of our framework is demonstrated on a control system used for regulation of temperature within a building.

KEYWORDS

cybersecurity, industrial control systems, risk, security measures, submodularity

1 | INTRODUCTION

Reliable operation of industrial control systems (ICSs) is of crucial importance for our society. ICSs are used to operate power plants, manufacturing systems, water distribution networks, and oil production facilities, for instance. Despite their importance, cybersecurity of ICSs was rarely discussed in the past.[1] This is partly because ICSs used proprietary equipment and were not connected to other systems. Hence, it was the isolation that protected these systems from cyberattacks. In contrast, modern ICSs have become interconnected and have started resembling ordinary IT systems. Although these

Abbreviations: EEV, electronic expansion valve; ICS, industrial control system; IT, information technology; PLC, programmable logic controller.

changes have made ICSs more efficient and less costly to operate, they have also exposed them to cyberattacks. How dangerous this can be is illustrated by a few fatal incidents reported in recent years: the cyberattack against sewage control system lead to an environmental hazard,[2] Stuxnet malware managed to damage the equipment in a nuclear facility,[3] and the cyberattack against Ukranian power grid operators left thousands of consumers without electrical energy.[4]

To prevent cyberattacks, the recommended practice is to implement a defense-in-depth strategy, which consists of multiple layers of security measures.[5-7] Examples of these measures include installation and maintenance of antivirus software, segmentation and segregation of the control network, encryption of communication links, and deployment of better physical protection. However, deployment of these measures in an ICS is challenging and costly. For example, in contrast to ordinary IT systems that have a typical life span of two to five years, ICSs are designed to last for decades. Thus, support for some of the equipment found within ICSs may not exist anymore.[5] Additionally, due to real-time requirements of ICSs, stopping them in order to deploy or update security measures needs to be planned well in advance.[5] Moreover, control equipment is in many cases computationally constrained, which introduces additional difficulties. For instance, adding encryption tasks on a controller may cause delays in the system and result in instability.[8]

Overall, due to the difficulties in implementing security measures, potentially large number of vulnerabilities, and limited budget, we expect not to be able to deploy all of the security measures. Therefore, a risk assessment[9] should be conducted prior to their deployment. In this assessment, one should prioritize among vulnerabilities, that is, one should decide which vulnerabilities are critical. The security measures can then be allocated to prevent only these critical vulnerabilities, instead of preventing them all. Unfortunately, due to the curse of dimensionality, both finding critical vulnerabilities and selecting the least expensive security measures to prevent them become challenging when the number of vulnerabilities and measures is large. The problem addressed in this paper is therefore how to allocate security measures in a cost-efficient way in this case.

Literature review. Cybersecurity of ICSs has become a well-established research topic. Experimental[10-12] and theoretical works[13-15] showed that existing methods in control cannot handle carefully designed cyberattacks, which opened up for many new research challenges. For example, novel approaches for attack-resilient estimators,[16-18] detectors,[15,19-21] controllers,[22-24] and consensus protocols[25,26] have been studied throughout the literature. The physical model of the system also proved to be useful for modeling and analyzing different types of attacks, such as zero dynamics,[15,27] covert,[28] replay,[20] denial of service,[29] rerouting,[30] and optimal linear attacks.[31] Another problem that has attracted a lot of attention is security measure allocation, which we consider in this paper. We also remark that other types of allocation problems have been considered within the control community. For example, allocating actuators to maximize different controllability metrics,[32] allocating leader agents in multiagent systems,[33] or allocating sensors to detect and isolate faults.[34] What is common for all these works on allocation is that they use the so-called submodularity property of a set function to obtain a suboptimal solution with performance guarantees.

A significant amount of work on allocating security measures has been developed for ICSs monitoring power grids. The grid is often modeled as a *static* linear system, and a particular combination of an estimator and an anomaly detector is often used. The assumption was in most of the cases that the sensors are vulnerable and that can be protected by deploying some security measure. The security measure allocation problem was formulated as selecting some of the existing sensors to secure, and/or placing additional secured sensors, to make undetectable attacks introduced in the work of Liu et al[13] harder to achieve. To solve the problem once the number of vulnerabilities and measures is large, which corresponds to a large number of sensors, a number of approaches have been proposed.[35-42] For example, Bobba et al proved that it suffices to protect the set of so-called basic sensors to prevent undetectable attacks and used LU decomposition to find these sensors.[35] Kim and Poor approximated the attacker's effort with a solution of a linear program and used greedy algorithms to select sensors to maximize this effort.[36] Greedy algorithms that allocate measures based on the so-called security index were proposed in the work of Dán and Sandberg.[37] In the sequel of this work,[38] more detailed models of communication network and security measures were introduced.

In contrast, the problem of allocating security measures based on *dynamical* models of control systems has attracted less attention. Cárdenas et al introduced several methods to estimate the attack impact and then mentioned that these methods can be used to select sensors/actuators to protect.[43] In the work of Teixeira et al,[44] a flexible risk model based on which security measures can be allocated was proposed. In the work of Milošević et al,[45] a Kalman filtering problem in the presence of bias injection attacks was considered, and a method for selecting sensors to secure was proposed to mitigate the impact of these attacks.

Open challenges. We now identify directions in which the existing literature can be extended. Firstly, the framework for allocating security measures based on dynamical models of control systems is lacking. The methods developed for power grid monitoring systems heavily rely on the static model of the grid and the attack model introduced in the work of

Liu et al.[13] Thus, these methods are not straightforward to extend to dynamical models of ICSs and attacks developed for these models. Additionally, the works[43-45] that consider dynamical models do not explain how to allocate security measures when the number of measures and vulnerabilities is large. Secondly, the modeling frameworks proposed in previous works can be improved. For instance, the connection between the cyber and physical part of an ICS is missing in most of the publications. Thus, it is unclear in what way the attacker gains control over the sensors/actuators and how the defender protects the sensors/actuators. This issue is partially addressed in the work of Vukovíc et al,[38] but the authors were mostly concerned with the models of vulnerabilities and measures in the communication infrastructure. Moreover, most of the previous publications do not allocate security measures based on a risk model, which is the recommended practice.[5-7] The measures are usually allocated relying on the attack impact but not that much attention is given to the attack complexity. Therefore, unnecessary amount of security budget can be spent by preventing unlikely attacks. A flexible risk model was proposed in the work of Teixeira et al,[44] but how to use this model once the number of vulnerabilities and security measures is large was not explained. Finally, optimality of the algorithms that are proposed for solving security measure allocation problem is rarely discussed. This problem is in general NP-hard, so the solution obtained in polynomial time can be arbitrarily far from the optimal one, except if some special structure of the problem is identified.

Contributions. As the first contribution, we propose a flexible modeling framework for allocating security measures in ICSs. Our framework is suitable for dynamical models of ICSs, models the cyber-physical interaction in more details, and uses a risk model to allocate security measures. Most importantly, it can be used once the number of security vulnerabilities and measures is large. In our framework, the attacker can gain control over sensors and actuators by exploiting a combination of vulnerabilities and, in that way, endanger the physical world. The defender wants to deploy security measures to prevent these vulnerabilities but, due to limited budget, cannot prevent all the vulnerabilities at once. Thus, he/she uses a risk model to prioritize among vulnerabilities. In the risk model we adopt, each subset of exploited vulnerabilities is a possible attack *scenario*, and its risk is determined based on the *impact* and *complexity* set functions, which are assumed to be nondecreasing with the number of vulnerabilities. Except imposing that the impact and complexity functions are nondecreasing, the proposed framework is not restricted to any particular instance of these functions, which makes it flexible. The scenarios with low complexity and potentially large impact are defined as critical scenarios. The security measure allocation problem is then how to prevent all the critical scenarios with minimal cost. The main two challenges concerning this problem are explained next.

The first challenge is to construct an instance of the security measure allocation problem since to do this, all the critical scenarios need to be found. Given that the number of possible scenarios is equal to the number of subsets of the vulnerability set, it is not feasible to simply search all the subsets to find those that are critical. To reduce the search space, we show how to use the nondecreasing property of the complexity function and prove that it suffices to find the smaller set of critical scenarios (*the sufficient representation of minimal cardinality*) instead of the whole set of critical scenarios. Additionally, given that the impact set function can be expensive to evaluate, we use the nondecreasing property of this function to reduce the number of its evaluations. Finally, all the combinations of security vulnerabilities are modeled with the search tree introduced in the work of Rymon[46] for the purpose of systematic and efficient search. To explore the tree, a breadth-first search algorithm that uses the aforementioned properties is proposed (Algorithm 1), and it is proven that this algorithm returns the sufficient representation of minimal cardinality (Theorem 1), which we outline as the second major contribution of this paper.

The second challenge is how to solve the security measure allocation problem once the critical attack scenarios are found. In our framework, the security measure allocation problem reduces to an integer linear program, which is NP-hard to solve in general. However, we show that the security measure allocation problem can be reformulated as a minimization of a linear function subject to a submodular constraint (Theorem 2). It is then known that a polynomial time greedy heuristic can be used to provide a suboptimal solution with known performance bound.[47] We also prove that the set of critical scenarios returned by Algorithm 1 provides the best performance guarantees for the greedy algorithm (Theorem 3). We outline Theorem 2 and Theorem 3 as the third major contribution. Finally, the applicability of the framework is demonstrated on an ICS that is used for regulating temperature within a building.[48]

We remark that a preliminary version of this paper appeared in the work of Milošević et al.[49] The results of the aforementioned work[49] are extended in the following aspects: (i) the modeling framework is more general; (ii) a section on finding the critical attack scenarios is added; (iii) the security measure allocation problem is formulated in different way; (iv) the performance of the greedy algorithm was not analyzed in the work of Milošević et al[49]; (v) a more detailed simulation section is included.

Paper organization. In the remainder of this section, we introduce notation used throughout the paper. The modeling framework and the problem formulation are introduced in Section 2. An algorithm that systematically searches for critical scenarios is proposed in Section 3. The submodular nature of security measure allocation problem is proven in Section 4. In Section 5, we illustrate on a simulation study applicability of our approach. We conclude this paper in Section 6.

Notation. Let $\mathcal{V} = \{v_1, \dots, v_n\}$ be a finite set. The set of all subsets of $\mathcal{V}$ is referred to as the *power set* and it is denoted with $2^{\mathcal{V}}$. We denote with $|\mathcal{V}|$ the cardinality of the set (the number of elements in $\mathcal{V}$). A *set function* represents a mapping $F : 2^{\mathcal{V}} \to \mathbb{R}$, that is, it maps every subset of the set $\mathcal{V}$ to a real number. A set function F is *nondecreasing* if for all $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}' \subseteq \mathcal{V}$, we have $F(\bar{\mathcal{V}}) \le F(\bar{\mathcal{V}}')$. Sets of real, real positive, and complex numbers are denoted with $\mathbb{R}$, $\mathbb{R}^+$, and $\mathbb{C}$, respectively.

2 | MODEL SETUP AND PROBLEM FORMULATION

In this section, we introduce the modeling framework and formulate the security measure allocation problem. We first model security vulnerabilities within an ICS, their connections with sensors and actuators, and their connection with security measures. To prioritize among vulnerabilities, we introduce a model of risk. Based on the risk model, the security measure allocation problem is formulated, and two main challenges regarding this problem are outlined.

2.1 | Security vulnerabilities and security measures

An ICS can be divided into the field layer and supervisory layer,[5] as illustrated in Figure 1. The field layer consists of field stations that interact with the physical process. The components of this layer are well known in the control community and include sensors, actuators, and control devices. The supervisory layer consists of one or more control centers, which are responsible for monitoring and supervisory control of the physical process. An operator from a control center can remotely gain control over actuators in cases of emergency, set reference signals for controllers, or change control algorithms implemented on control devices. The control center is also responsible for collecting and storing process information, centralized alarming, and communicating with other IT systems (eg, manufacturing execution systems and enterprise resource planning systems).

Various security vulnerabilities can be identified within an ICS. We model these vulnerabilities with the set

$$\mathcal{V} = \{v_1, \dots, v_{n_v}\}.$$

A security vulnerability $v \in \mathcal{V}$ can model a communication link without protection, lack of antivirus software on some of the computers in the control center, lack of physical protection of control equipment, etc. Throughout this paper, we will be interested in attacks that exploit multiple vulnerabilities. Therefore, we make the following definition.

Definition 1. A subset $\bar{\mathcal{V}} \subseteq \mathcal{V}$ of vulnerabilities actively used in a possible attack is called a *scenario*.

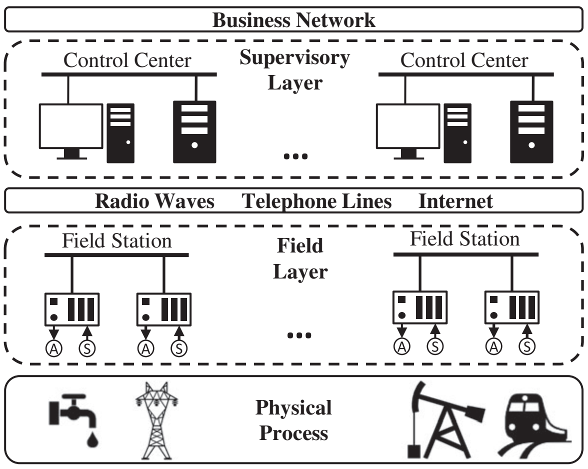

FIGURE 1 Common architecture of an industrial control system. The physical process is controlled through the field layer, which consists of control devices, sensors, and actuators. The supervisory layer is used for monitoring and control of the field layer, consists of ordinary IT equipment, and is often connected to other networks

Remark 1. The model of vulnerabilities in this paper assumes that vulnerabilities are known. In other words, undiscovered (zero-day) vulnerabilities are not captured with the model. However, since every security strategy needs to be updated over time, newly discovered vulnerabilities can be taken into consideration once a new strategy is deployed.

By exploiting some of the vulnerabilities, an attacker can gain control over the sensors and actuators and use these components to conduct the attack against the physical process. Let the sets of sensors and actuators be denoted by

$$\mathcal{S} = \{s_1, \ldots, s_{n_y}\} \qquad \mathcal{A} = \{a_1, \ldots, a_{n_u}\},$$

respectively. With each vulnerability $v \in \mathcal{V}$, we associate the subsets of compromised sensors $\bar{\mathcal{S}}_v \subseteq \mathcal{S}$ and actuators $\bar{\mathcal{A}}_v \subseteq \mathcal{A}$. These sets model the components the attacker gains control over once it exploits vulnerability v. In the scenario $\bar{\mathcal{V}} \subseteq \mathcal{V}$, where the attacker exploits multiple security vulnerabilities, the sensors and actuators under its control are

$$\bar{\mathcal{S}}(\bar{\mathcal{V}}) = \bigcup_{v \in \bar{\mathcal{V}}} \bar{\mathcal{S}}_v \qquad \bar{\mathcal{A}}(\bar{\mathcal{V}}) = \bigcup_{v \in \bar{\mathcal{V}}} \bar{\mathcal{A}}_v.$$

The attacker can then freely change the measurements of the sensors $\bar{\mathcal{S}}(\bar{\mathcal{V}})$ and the control actions sent to the actuators $\bar{\mathcal{A}}(\bar{\mathcal{V}})$.

The security vulnerabilities can be prevented by investing in security measures. We denote the set of security measures by

$$\mathcal{M} = \{m_1, \ldots, m_{n_m}\}.$$

Each security measure $m \in \mathcal{M}$ can model a communication link encryption, installation and maintenance of antivirus software, or deployment of better physical protection. It can also capture the scenarios where multiple measures are deployed at the same time. With each security measure $m \in \mathcal{M}$, we associate a number $c_m \in \mathbb{R}^+$ that models the cost of deploying m and a set of vulnerabilities $\bar{\mathcal{V}}_m$ prevented by m. In the case when we deploy the subset of security measures $\bar{\mathcal{M}} \subseteq \mathcal{M}$, the set of prevented vulnerabilities $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ and the total cost $c(\bar{\mathcal{M}})$ are defined by

$$\bar{\mathcal{V}}(\bar{\mathcal{M}}) = \bigcup_{m \in \bar{\mathcal{M}}} \bar{\mathcal{V}}_m \qquad c(\bar{\mathcal{M}}) = \sum_{m \in \bar{\mathcal{M}}} c_m. \tag{1}$$

The assumption is that the security measures provide perfect prevention of the vulnerabilities $\bar{\mathcal{V}}(\bar{\mathcal{M}})$.

Assumption 1. Let $\bar{\mathcal{V}}(\bar{\mathcal{M}}) \subseteq \mathcal{V}$ be the subset of vulnerabilities prevented by deploying corresponding security measures $\bar{\mathcal{M}}$. The vulnerabilities from the set $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ cannot be taken advantage of by an attacker.

Based on Assumption 1, we say that scenario $\bar{\mathcal{V}}$ is *prevented* if it requires at least one of the prevented vulnerabilities from the set $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ to be conducted, that is, $\bar{\mathcal{V}} \cap \bar{\mathcal{V}}(\bar{\mathcal{M}}) \neq \emptyset$. Similarly, we say that a set of attack scenarios $\tilde{\mathcal{V}} = \{\bar{\mathcal{V}}_1, \ldots, \bar{\mathcal{V}}_l\} \subseteq 2^{\mathcal{V}}$ is prevented, if, for every $\bar{\mathcal{V}} \in \tilde{\mathcal{V}}$, it holds $\bar{\mathcal{V}} \cap \bar{\mathcal{V}}(\bar{\mathcal{M}}) \neq \emptyset$. To clarify this further, we introduce an example.

Example 1. Say that we have a scenario $\bar{\mathcal{V}} = \{v_1, v_2, v_3\}$. In case that we prevent vulnerability v_1, this scenario is prevented. However, note that the scenario $\bar{\mathcal{V}}' = \{v_2, v_3\}$ is still possible. That is, the attacker can still use the remaining two vulnerabilities to conduct an attack. Assume now that we have a set of attack scenarios $\tilde{\mathcal{V}} = \{\{v_1, v_2, v_3\}, \{v_1, v_2\}, \{v_3\}\}$ that we want to prevent. In this case, it is not sufficient to prevent only v_1 because the scenario $\{v_3\}$ is not prevented. The possible sets of vulnerabilities that have to be prevented to prevent $\tilde{\mathcal{V}}$ are $\{v_1, v_3\}$, $\{v_2, v_3\}$, or $\{v_1, v_2, v_3\}$.

We also assume that each security vulnerability can be prevented by deploying a suitable security measure from the set $\mathcal{M}$. This assumption is needed in order to be able to guarantee the feasibility of security measure allocation problem that we define later.

Assumption 2. Let $v \in \mathcal{V}$ be an arbitrary selected vulnerability. There exists at least one security measure $m \in \mathcal{M}$ that prevents this vulnerability, that is, $v \cap \bar{\mathcal{V}}_m \neq \emptyset$.

We now introduce an example to clarify the notation.

Example 2. We consider an ICS consisting of the control center and two programmable logic controllers (PLCs), as shown in Figure 2. From the Figure, we see that the sets of sensors and actuators are given by $\mathcal{S} = \{s_1, s_2, s_3, s_4\}$ and $\mathcal{A} = \{a_1, a_2\}$, respectively. The following security vulnerabilities are identified within the system. PLC 1 is lacking physical protection (vulnerability v_1), connection between PLC 1, and the control center is unprotected (vulnerability v_2) as well as the connection between PLC 2 and the control center (vulnerability v_3). The set of vulnerabilities

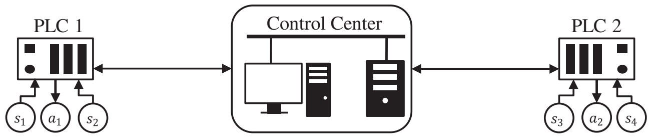

FIGURE 2 Example of an industrial control system consisting of the control center, two programmable logic controllers (PLC 1 and PLC 2), two actuators (a_1, a_2), and four sensors (s_1, s_2, s_3, s_4)

is then $\mathcal{V} = \{v_1, v_2, v_3\}$. In case that the attacker exploits vulnerability v_1, it gains control over $\bar{\mathcal{S}}_{v_1} = \{s_1, s_2\}$ and $\bar{\mathcal{A}}_{v_1} = \{a_1\}$. If the vulnerability v_2 is exploited, then $\bar{\mathcal{S}}_{v_2} = \{s_1, s_2\}$ and $\bar{\mathcal{A}}_{v_2} = \{a_1\}$. If v_3 is taken advantage of, then $\bar{\mathcal{S}}_{v_3} = \{s_3, s_4\}$ and $\bar{\mathcal{A}}_{v_3} = \{a_2\}$. An example of attack scenario could be $\bar{\mathcal{V}} = \{v_1, v_3\}$. In that case, the components the attacker controls are $\bar{\mathcal{S}}(\bar{\mathcal{V}}) = \{s_1, s_2, s_3, s_4\}$ and $\bar{\mathcal{A}}(\bar{\mathcal{V}}) = \{a_1, a_2\}$.

The security measure that prevents vulnerability v_1 could be locking the PLC in the cabinet (measure m_1). The vulnerability v_2 could be prevented by implementing encryption and authentication schemes on the corresponding communication link (measure m_2). Same holds for v_3 (measure m_3). There is also an option to protect both of the links for more favorable price (measure m_4). The set of security measures is then $\mathcal{M} = \{m_1, m_2, m_3, m_4\}$ and sets of vulnerabilities prevented by each of the measures are $\bar{\mathcal{V}}_{m_1} = \{v_1\}$, $\bar{\mathcal{V}}_{m_2} = \{v_2\}$, $\bar{\mathcal{V}}_{m_3} = \{v_3\}$, and $\bar{\mathcal{V}}_{m_4} = \{v_2, v_3\}$. If m_1 and m_2 are deployed, then $\bar{\mathcal{M}} = \{m_1, m_2\}$ and $\bar{\mathcal{V}}(\bar{\mathcal{M}}) = \{v_1, v_2\}$.

2.2 | Model of risk

To protect an ICS from cyberattacks, we want to select a subset of security measures to deploy. As discussed earlier, in most cases, the total budget would not be large enough to deploy all the security measures. Thus, we should focus our budget in preventing the most critical attack scenarios.

In this section, we introduce a model of risk based on which we prioritize among attack scenarios. We use the risk model introduced in the work of Kaplan and Garrick,[50] where it was proposed to model risk as a triplet (scenario, impact, complexity). This model can be applied in our context by defining the triplets as

$$\left(\bar{\mathcal{V}}, I\left(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}})\right), \pi(\bar{\mathcal{V}})\right),$$

where scenarios are represented by the subset of vulnerabilities $\bar{\mathcal{V}} \subseteq \mathcal{V}$ that the attacker exploits to conduct a certain attack, $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}}))$ represents the impact that occurs in that scenario and $\pi(\bar{\mathcal{V}})$ represents the complexity of the scenario.

Remark 2. In general, what we mean by complexity of the attack scenario is often referred to as attack likelihood throughout the literature. We avoid using the term likelihood of attack scenarios to avoid confusing this term with likelihood in statistics, which is different in nature.

The impact function $I(\cdot)$ should reflect how dangerous it is if the vulnerabilities $\bar{\mathcal{V}}$ are exploited, and it can be estimated based on a physical model of the system.[15,27,51,52] We assume that the attacks conducted with more components are more severe than those conducted with less.

Assumption 3. Let $I : 2^{\mathcal{A}} \times 2^{\mathcal{S}} \to \mathbb{R}^+$ be the impact set function. If $\bar{\mathcal{A}} \subseteq \bar{\mathcal{A}}' \subseteq \mathcal{A}$ and $\bar{\mathcal{S}} \subseteq \bar{\mathcal{S}}' \subseteq \mathcal{S}$, then $I(\bar{\mathcal{A}}, \bar{\mathcal{S}}) \leq I(\bar{\mathcal{A}}', \bar{\mathcal{S}}')$.

The complexity function $\pi(\cdot)$ models how hard it is for the attacker to exploit all the vulnerabilities from $\bar{\mathcal{V}}$ simultaneously. The function can be estimated based on a security expert knowledge.[9,53,54] We assume that the more vulnerabilities the attacker exploits, the more complex the attack scenario becomes.

Assumption 4. The complexity function $\pi : 2^{\mathcal{V}} \to \mathbb{R}^+$ is a nondecreasing set function, that is, $\pi(\bar{\mathcal{V}}) \leq \pi(\bar{\mathcal{V}}')$ for any $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}'$.

We remark that the framework we propose in this paper is quite flexible and can be used for any impact (complexity) functions that satisfy Assumption 3 (Assumption 4). In the next section, these assumptions are used to construct algorithm that systematically searches for the most critical attack scenarios. Naturally, the most critical attack scenarios are those that require relatively low complexity to be conducted and can lead to large impact. We formally define these scenarios as follows.

Definition 2. A scenario $\bar{\mathcal{V}} \subseteq \mathcal{V}$ is said to be *critical* if $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}})) \geq I_{\min}$ and $\pi(\bar{\mathcal{V}}) \leq \pi_{\max}$, where $I_{\min} \in \mathbb{R}^+$ and $\pi_{\max} \in \mathbb{R}^+$ are some predefined thresholds.

Remark 3. In general, the choice of $I_{\min}$ and $\pi_{\max}$ depends on the complexity and impact functions used and the particular problem instance. Hence, the thresholds should be seen as tuning parameters in the allocation. One way to choose the thresholds would be to set $I_{\min}$ relatively high and $\pi_{\max}$ relatively low. In this way, we restrict our attention to those scenarios that have very large impact and are simple to conduct. In case that these scenarios are inexpensive to prevent, one can then decrease $I_{\min}$ and increase $\pi_{\max}$ and re-solve the problem iteratively to prevent less dangerous scenarios.

We now introduce concrete examples of impact and complexity functions.

2.2.1 | Attack impact

To define the impact set function, we use model of zero-dynamics attacks.[15,27] These attacks are serious because from the sensor data, the attacks are indistinguishable from the normal system operation. In that way, the attacker is able to potentially make some of the system states arbitrarily large while staying undetected by the system operator at the same time.

To determine if it is possible for the attacker to conduct a zero-dynamic attack, we introduce the physical model of the system

$$\begin{aligned} x(k+1) &= Ax(k) + B\tilde{u}(k) \\ y(k) &= Cx(k), \end{aligned} \tag{2}$$

where $x(k) \in \mathbb{R}^{n_x}$ is the state of the system, $\tilde{u}(k) \in \mathbb{R}^{n_u}$ is the control signal applied to the process, and $y(k) \in \mathbb{R}^{n_y}$ is the vector of sensor measurements collected from the process. Due to attacks, the signal $\tilde{u}(k)$ is different from the control signals calculated by the controllers, which we denote with $u(k)$. Similarly, due to attacks against sensors, the operators receive false measurements $\tilde{y}(k)$ instead of the original ones $y(k)$. The signals $\tilde{y}(k)$ and $\tilde{u}(k)$ can then be modeled as

$$\tilde{y}(k) = y(k) + D_y(\bar{\mathcal{V}})a_y(k) \qquad \tilde{u}(k) = u(k) + D_u(\bar{\mathcal{V}})a_u(k),$$

where $a_y(k) \in \mathbb{R}^{|\bar{\mathcal{S}}(\bar{\mathcal{V}})|}$ is the attack signal added to the measurements from attacked sensors $\bar{\mathcal{S}}(\bar{\mathcal{V}})$, and $a_u(k) \in \mathbb{R}^{|\bar{\mathcal{A}}(\bar{\mathcal{V}})|}$ is the attack signal sent to the attacked actuators $\bar{\mathcal{A}}(\bar{\mathcal{V}})$. It is important to understand that the matrices $D_u(\bar{\mathcal{V}}) \in \mathbb{R}^{n_u \times |\bar{\mathcal{A}}(\bar{\mathcal{V}})|}$ and $D_y(\bar{\mathcal{V}}) \in \mathbb{R}^{n_y \times |\bar{\mathcal{S}}(\bar{\mathcal{V}})|}$ are dependent on the attack scenario $\bar{\mathcal{V}}$. Let $\bar{\mathcal{S}}(\bar{\mathcal{V}}) = \{s_{j_1}, \dots, s_{j_p}\}$ be the sensors controlled by the attacker once vulnerabilities $\bar{\mathcal{V}}$ are exploited. Then, the elements $(j_1, 1), \dots, (j_p, p)$ of $D_y(\bar{\mathcal{V}})$ are equal to one and the remaining elements are equal to zero. The matrix $D_u(\bar{\mathcal{V}})$ is defined in an analogous way but based on $\bar{\mathcal{A}}(\bar{\mathcal{V}})$. We also assume that the matrix $BD_u(\bar{\mathcal{V}})$ has a full column rank, which is an assumption adopted to exclude the attack signals that cancel each other and do not lead to any impact. Undetectable attacks can then be defined as follows.

Definition 3. The nonzero attack (a_u, a_y) is *undetectable*, if there exists an initial state $x(0)$ such that $\tilde{y}(k) = 0$ for $k \geq 0$.

To check if the attacker can conduct undetectable attack using $\bar{\mathcal{S}}(\bar{\mathcal{V}})$ and $\bar{\mathcal{A}}(\bar{\mathcal{V}})$, the Rosenbrock matrix[55] of the system

$$P(z) = \begin{bmatrix} A - zI & BD_u(\bar{\mathcal{V}}) & 0_{n_x \times |\bar{\mathcal{S}}(\bar{\mathcal{V}})|} \\ C & 0_{n_y \times |\bar{\mathcal{A}}(\bar{\mathcal{V}})|} & D_y(\bar{\mathcal{V}}) \end{bmatrix}$$

can be used.[15,27,56] In particular, the undetectable attack can be conducted if and only if there exist $z_0 \in \mathbb{C}$, $x_0 \in \mathbb{C}^{n_x}$, $a_u \in \mathbb{C}^{|\bar{\mathcal{A}}(\bar{\mathcal{V}})|}$, $a_y \in \mathbb{C}^{|\bar{\mathcal{S}}(\bar{\mathcal{V}})|}$, and $[\, a_u^T \;\; a_y^T \,] \neq 0$, such that $P(z_0)[\, x_0^T \;\; a_u^T \;\; a_y^T \,]^T = 0$. Two cases of undetectable attacks are particularly dangerous. If

$$P(z_0)\begin{bmatrix} x_0^T & a_u^T & a_y^T \end{bmatrix}^T = 0 \qquad \text{for some } |z_0| > 1 \tag{3}$$

is satisfied, the attacker can make some of the system states arbitrarily large while remaining undetected. An even more dangerous scenario occurs when

$$\text{normalrank } P = \max_z \text{rank}(P(z)) < n_x + |\bar{\mathcal{S}}(\bar{\mathcal{V}})| + |\bar{\mathcal{A}}(\bar{\mathcal{V}})|. \tag{4}$$

TABLE 1 Impact set function $I(\cdot)$ based on zero-dynamic attacks

$I(\bar{\mathcal{A}}, \bar{\mathcal{S}})$	Description
0	An undetectable attack cannot be conducted with components $\bar{\mathcal{A}}, \bar{\mathcal{S}}$
1	An undetectable attack can be conducted with components $\bar{\mathcal{A}}, \bar{\mathcal{S}}$ but only for some $\lvert z_0 \rvert < 1$
2	An undetectable attack can be conducted with components $\bar{\mathcal{A}}, \bar{\mathcal{S}}$ for some $\lvert z_0 \rvert \geq 1$
3	An undetectable attack can be conducted with components $\bar{\mathcal{A}}, \bar{\mathcal{S}}$ for any z_0

In that case, the attacker can conduct undetectable attack for any complex frequency z_0. Due to the importance of the Rosenbrock matrix, both the condition (3) and (4) can be checked efficiently for a given scenario $\bar{\mathcal{V}}$ using well-established algorithms. We also remark that similar analysis can be conducted for continuous-time systems.

Based on the previously introduced attack strategy, one way to define the attack impact function would be as shown in Table 1. Note that this impact metric satisfies Assumption 3. Mainly, if the attacker is able to conduct the zero-dynamic attack using the components $\bar{\mathcal{S}}$ and $\bar{\mathcal{A}}$, then it can always conduct this same attack with the larger set of components $\bar{\mathcal{S}}' \supseteq \bar{\mathcal{S}}$ and $\bar{\mathcal{A}}' \supseteq \bar{\mathcal{A}}$. It just needs to send the same signals to sensors $\bar{\mathcal{S}}$ and actuators $\bar{\mathcal{A}}$, while keeping the attack signals that corresponds to sensors $\bar{\mathcal{S}}' \setminus \bar{\mathcal{S}}$ and actuators $\bar{\mathcal{A}}' \setminus \bar{\mathcal{A}}$ equal to zero.

We remark that other methods developed for estimating attack impact can also be used in our framework. For example, in the works of Umsonst et al[51] and Milošević et al,[52] the impact is measured through the infinity norm of so-called critical states, and it is obtained by solving a set of convex problems. To form these convex problems for every scenario $\bar{\mathcal{V}}$, we again use the physical model of the system under attack. This physical model is adjusted for every scenario $\bar{\mathcal{V}}$ by changing the matrices $D_u(\bar{\mathcal{V}})$ and $D_y(\bar{\mathcal{V}})$, same as in the case with zero-dynamics attacks presented here. Other methods compatible with our framework include those developed for estimating impact of attacks in monitoring systems.[57-59] In these works, the impact of attacks is measured through the ellipsoidal approximation of a reachable region.

2.2.2 | Attack complexity

Estimation of complexity (likelihood) of adversarial attack scenarios and nonadversarial fault scenarios is different. In case of nonadversarial scenarios, the likelihood of a scenario is usually probabilistic in nature, and it is estimated based on historical evidence or empirical data.[9] In contrast, the complexity of a malicious scenario is typically a score representing the belief of such a scenario occurring relative to other scenarios.[9] This score is formed based on security expert knowledge[9,53,54] or using the tools developed for this purpose.[60,61] The factors that can be used to estimate the complexity include site architecture, security measures that are already installed, cost of attack, and technical difficulty.[53]

Within the control community, attack complexity is sometimes approximated based on the number of compromised sensors and actuators, which is reasonable assumption in certain cases.[44] In this paper, we use a more detailed method presented in the work of Byres et al[53] to illustrate a possible way to form a complexity function. There, the authors used the so-called attack trees to estimate the attack complexity. The idea was to represent a large attack scenario as a tree and break it into smaller subtrees. The complexities of subtrees were first estimated and then combined to calculate the complexity of the whole tree.

This method can be applied in our framework as follows. The first step is to assign complexity $\pi_v > 0$ to each of the vulnerabilities $v \in \mathcal{V}$. As we mentioned, this is done based on a security expert opinion[53] or using some of the tools for vulnerability ranking.[61] One possible assignment for these complexities is shown in Table 2. In the second step, the complexities π_v are combined to estimate the complexity $\pi(\bar{\mathcal{V}})$ of a more complex scenario $\bar{\mathcal{V}} \subseteq \mathcal{V}$ consisting of multiple vulnerabilities. For example, $\pi(\bar{\mathcal{V}})$ can be defined as

$$\pi(\bar{\mathcal{V}}) = \sum_{v \in \bar{\mathcal{V}}} \pi_v. \tag{5}$$

Although simple, this function captures the essence of the problem. Scenarios containing vulnerabilities with higher values of π_v have a larger total complexity than those with equal number of vulnerabilities but with lower values of π_v. This complexity function is also nondecreasing because it represents a nonnegative sum. Thus, it satisfies Assumption 4.

TABLE 2 Assigning complexity π_v of exploiting individual vulnerabilities $v \in \mathcal{V}$

Complexity	Very Low	Low	Medium	Difficult	Very Difficult
π_v	1	2	3	4	5

Other functions proposed in the literature[9] such as weighted sum $\pi(\bar{\mathcal{V}}) = \sum_{v\in\bar{\mathcal{V}}} w_v\pi_v$ or max function $\pi(\bar{\mathcal{V}}) = \max_{v\in\bar{\mathcal{V}}}\pi_v$ can be used instead of (5).

2.3 | Problem formulation

Let the set of all critical scenarios be $\tilde{\mathcal{V}}_C$. Our goal is then to find the least expensive set of security measures $\bar{\mathcal{M}} \subseteq \mathcal{M}$ that prevents all the critical scenarios from $\tilde{\mathcal{V}}_C$. This problem can be formulated as an integer linear program, as explained next.

The set of deployed security measures $\bar{\mathcal{M}}$ can be represented as an integer-valued decision vector $x_m \in \{0,1\}^{n_m}$. In case that we chose to deploy security measure m_i, then $x_m(i)$ is set to 1. Otherwise, $x_m(i)$ is equal to 0. The objective function we want to minimize can then be modeled as $c^T x_m$, where $c = [c_{m_1}, \dots, c_{m_{n_m}}]^T$ is a vector containing individual costs of security measures.

The constraints for the problem are that the deployed set of security measures x_m should prevent all of the critical scenarios. Thus, we introduce the matrix $F \in \mathbb{R}^{n_v\times n_m}$, which is the incidence matrix modeling which vulnerabilities are prevented by the deployed security measures x_m. The vector of prevented vulnerabilities is denoted with x_v and equality $x_v = Fx_m$ must hold. In case that $x_v(i) > 0$, we know that vulnerability v_i is prevented. We then join a vector $f_{\bar{\mathcal{V}}} \in \mathbb{R}^{n_v}$ with each of the critical scenarios $\bar{\mathcal{V}} \in \tilde{\mathcal{V}}_C$ defined as

$$f_{\bar{\mathcal{V}}}(i) = \begin{cases} 1\,, & \text{if } v_i \in \bar{\mathcal{V}} \\ 0\,, & \text{otherwise.} \end{cases}$$

The condition $f_{\bar{\mathcal{V}}}^T x_v \geq 1$ is then equivalent to perfect prevention of the scenario $\bar{\mathcal{V}}$. We are now ready to introduce the security measure allocation problem.

Problem 1. Security measure allocation

$$\begin{aligned} \underset{x_m}{\text{minimize}} \quad & c^T x_m \\ \text{subject to} \quad & x_v = Fx_m \\ & f_{\bar{\mathcal{V}}}^T x_v \geq 1 \qquad \forall \bar{\mathcal{V}} \in \tilde{\mathcal{V}}_C. \end{aligned}$$

The two main difficulties with solving this problem are the following. Firstly, constructing Problem 1 represents an issue. To form constraints of this problem, we need to find the set of critical scenarios $\tilde{\mathcal{V}}_C$. The number of possible attack scenarios is equal to the number of subsets of the set $\mathcal{V}$. Thus, simply going through all the subsets of $\mathcal{V}$ and deciding whether they are critical or not are not feasible when the cardinality of the set $\mathcal{V}$ is large. Secondly, even if the set $\tilde{\mathcal{V}}_C$ is found, Problem 1 is NP-hard in general. Therefore, polynomial time algorithms that return the optimal or suboptimal solution are in general unknown, unless some special structure of the problem is identified.

In the following two sections, we tackle these issues. In Section 3, we present an algorithm that systematically generates the so-called sufficient representation of minimal cardinality $\hat{\mathcal{V}}_C^*$. The set $\hat{\mathcal{V}}_C^*$ is the subset of $\tilde{\mathcal{V}}_C$, with the property that, if $\hat{\mathcal{V}}_C^*$ is prevented, then we know that $\tilde{\mathcal{V}}_C$ is prevented as well. We show that, by using the properties of $\hat{\mathcal{V}}_C^*$ together with the properties of the complexity and the impact functions, we can potentially significantly reduce the execution time of the search. To resolve the second issue, we use the fact that Problem 1 can be reformulated as the problem of minimizing a linear function subject to a submodular constraint. In that case, a polynomial time greedy algorithm can be used to find a suboptimal solution of the problem with guaranteed performance, as shown in Section 4.

3 | SYSTEMATICALLY SEARCHING FOR CRITICAL SCENARIOS

In this section, we address the issue of constructing a set of critical scenarios $\tilde{\mathcal{V}}_C$. As we mentioned, the number of possible attack scenarios in our model is equal to the number of subsets of $\mathcal{V}$, that is, $2^{|\mathcal{V}|}$. Thus, simply going through all the scenarios and deciding whether they are critical or not is not feasible for large cardinalities of $\mathcal{V}$. In this section, we propose an algorithm that systematically searches for critical scenarios. Before we move to the design of the algorithm, we first explain the rules that the algorithm uses to reduce the execution time of the search. The first way of reducing the execution time is by reducing the number of scenarios we explore. For this purpose, we use the property of complexity function $\pi(\cdot)$

introduced in Assumption 4, and we show that it suffices to find a suitable subset $\hat{\mathcal{V}}_C^*$ of the set of critical scenarios, instead of the whole set $\tilde{\mathcal{V}}_C$. To further improve the total execution time, the algorithm avoids evaluating possibly computationally intensive impact function $I(\cdot)$ for every scenario. In particular, the property of $I(\cdot)$ introduced in Assumption 3 can be used to reuse the information from the previously explored scenarios. We then formulate the algorithm that systematically constructs $\hat{\mathcal{V}}_C^*$.

3.1 | Reducing number of explored scenarios

The first way to reduce the number of scenarios to explore is by using the properties of the complexity function. Since the complexity function is nondecreasing, we conclude that, if we find a scenario $\bar{\mathcal{V}}$ that has complexity $\pi(\bar{\mathcal{V}})$ higher than the threshold $\pi_{\max}$, we do not need to investigate any other scenario $\bar{\mathcal{V}}'$ that contains this scenario. The reason is that these scenarios have complexity that is higher than or equal to $\pi(\bar{\mathcal{V}})$, hence they do not belong to critical scenarios.

Lemma 1. *Assume that a scenario $\bar{\mathcal{V}}$ satisfies $\pi(\bar{\mathcal{V}}) > \pi_{\max}$. Then, any scenario $\bar{\mathcal{V}}'$ that satisfies $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}'$ is not a critical scenario.*

Proof. From Assumption 4, $\pi(\bar{\mathcal{V}}') \geq \pi(\bar{\mathcal{V}}) > \pi_{\max}$ for $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}'$. Thus, scenario $\bar{\mathcal{V}}'$ does not satisfy Definition 2. □

The second way to reduce the number of explored scenarios is by observing that we do not need to find the whole set $\tilde{\mathcal{V}}_C$. Instead, it is sufficient to find a suitable subset of this set. We use an example to explain the idea.

Example 3. Assume that we have a set of vulnerabilities $\mathcal{V} = \{v_1, v_2, v_3\}$ and let the set of critical scenarios be given by $\tilde{\mathcal{V}}_C = \{\{v_1\}, \{v_1, v_3\}, \{v_1, v_2\}, \{v_2, v_3\}\}$. Consider now the subset $\hat{\mathcal{V}}_C = \{\{v_1\}, \{v_2, v_3\}\}$. This subset is prevented if the set of prevented vulnerabilities $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ is one of the following: $\{v_1, v_2\}$, $\{v_1, v_3\}$, or $\{v_1, v_2, v_3\}$. What is important to realize is that every choice of $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ that prevents $\hat{\mathcal{V}}_C$ prevents $\tilde{\mathcal{V}}_C$ as well. Thus, instead of constructing the complete set of critical scenarios, it suffices to find a subset of smaller cardinality $\hat{\mathcal{V}}_C$. The reason is that, every time we prevent $\hat{\mathcal{V}}_C$, we know that $\tilde{\mathcal{V}}_C$ is prevented.

Motivated by the previous example, we introduce a notion of sufficient representation of the set of critical scenarios. In general, a sufficient representation $\hat{\mathcal{V}}_C$ is a subset of $\tilde{\mathcal{V}}_C$ with the property that, once we prevent all the scenarios in $\hat{\mathcal{V}}_C$, all the critical scenarios $\tilde{\mathcal{V}}_C$ are prevented, as stated in the following definition.

Definition 4. A set $\hat{\mathcal{V}}_C \subseteq 2^{\mathcal{V}}$ is a *sufficient representation* of a set $\tilde{\mathcal{V}}_C$ if the following two conditions are satisfied: (i) $\hat{\mathcal{V}}_C \subseteq \tilde{\mathcal{V}}_C$; (ii) for every $\bar{\mathcal{V}}(\bar{\mathcal{M}}) \subseteq \mathcal{V}$, it holds that $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ prevents $\tilde{\mathcal{V}}_C$ if and only if $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ prevents $\hat{\mathcal{V}}_C$.

Remark 4. A sufficient representation is not unique in general. Furthermore, from the definition, it follows that if the set of security measures prevents one sufficient representation of the set $\hat{\mathcal{V}}_C$, then it also prevents any other sufficient representation $\hat{\mathcal{V}}_C'$. We use this property in some of the proofs in this section.

Naturally, we are interested in finding a sufficient representation that has minimal cardinality. Besides helping us to reduce the number of scenarios to explore, in the next section, we show that this sufficient representation of minimal cardinality is also beneficial for the problem of preventing the critical scenarios using the minimal budget. In the following lemma, we introduce the condition that the sufficient representation of minimal cardinality needs to satisfy. In particular, we show that none of the scenarios in this representation should contain any other scenario from the representation. We also prove that the sufficient representation of minimal cardinality is unique.

Lemma 2. *Let a set $\hat{\mathcal{V}}_C^*$ be a sufficient representation of the set of critical scenarios $\tilde{\mathcal{V}}_C$. The set $\hat{\mathcal{V}}_C^*$ is the unique sufficient representations of minimal cardinality if and only if for any two nonempty sets $\bar{\mathcal{V}} \in \hat{\mathcal{V}}_C$ and $\bar{\mathcal{V}}' \in \hat{\mathcal{V}}_C$, $\bar{\mathcal{V}} \neq \bar{\mathcal{V}}'$, it holds $\bar{\mathcal{V}} \not\subseteq \bar{\mathcal{V}}'$.*

Proof. (⇒) We prove that necessity holds by using a contradiction argument. Let $\hat{\mathcal{V}}_C^*$ be the sufficient representation of minimal cardinality, and assume that there exists $\bar{\mathcal{V}} \in \hat{\mathcal{V}}_C^*$ and $\bar{\mathcal{V}}' \in \hat{\mathcal{V}}_C^*$, $\bar{\mathcal{V}}' \neq \bar{\mathcal{V}}$, such that $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}'$. In what follows, we prove that the set $\hat{\mathcal{V}}_C' = \hat{\mathcal{V}}_C^* \setminus \bar{\mathcal{V}}'$ is a sufficient representation of $\tilde{\mathcal{V}}_C$, with cardinality smaller than $\hat{\mathcal{V}}_C^*$. Let $\bar{\mathcal{V}}_P$ be the set of prevented vulnerabilities, and assume that $\hat{\mathcal{V}}_C'$ is prevented by this set. Given that $\hat{\mathcal{V}}_C' = \hat{\mathcal{V}}_C^* \setminus \bar{\mathcal{V}}'$, this implies that $\bar{\mathcal{V}}_P$ prevents any scenario from $\hat{\mathcal{V}}_C^*$, except perhaps $\bar{\mathcal{V}}'$. However, since $\bar{\mathcal{V}} \in \hat{\mathcal{V}}_C'$, we know that $\bar{\mathcal{V}} \cap \bar{\mathcal{V}}_P \neq \emptyset$, which implies

$\bar{\mathcal{V}}' \cap \bar{\mathcal{V}}_P \neq \emptyset$. Hence, we proved that any set $\bar{\mathcal{V}}_P$ that prevents $\hat{\mathcal{V}}'_C$ automatically prevents $\hat{\mathcal{V}}^*_C$. Given that $\hat{\mathcal{V}}'_C \subset \hat{\mathcal{V}}^*_C$, any set $\bar{\mathcal{V}}_P$ that prevents $\hat{\mathcal{V}}^*_C$ prevents $\hat{\mathcal{V}}'_C$ as well. Thus, $\bar{\mathcal{V}}_P$ prevents $\hat{\mathcal{V}}^*_C$ if and only if $\bar{\mathcal{V}}_P$ prevents $\hat{\mathcal{V}}'_C$. We then conclude that $\hat{\mathcal{V}}'_C$ represents a sufficient representation of $\tilde{\mathcal{V}}_C$. Since $|\hat{\mathcal{V}}'_C| = |\hat{\mathcal{V}}^*_C| - 1$, it follows that $\hat{\mathcal{V}}^*_C$ is not the sufficient representation of minimal cardinality, which contradicts the initial assumption.

($\Leftarrow$) Same as in the case of necessity, we use the contradiction argument to prove sufficiency. Let $\hat{\mathcal{V}}^*_C = \{\bar{\mathcal{V}}_1, \ldots, \bar{\mathcal{V}}_l\}$ be a sufficient representation of $\tilde{\mathcal{V}}_C$, which satisfies $\bar{\mathcal{V}} \not\subseteq \bar{\mathcal{V}}'$ for any $\bar{\mathcal{V}} \neq \bar{\mathcal{V}}'$ from $\hat{\mathcal{V}}^*_C$. Assume now that there exists a sufficient representation $\hat{\mathcal{V}}'_C = \{\bar{\mathcal{V}}'_1, \ldots, \bar{\mathcal{V}}'_m\}$, $\hat{\mathcal{V}}'_C \neq \hat{\mathcal{V}}^*_C$, with $|\hat{\mathcal{V}}'_C| \leq |\hat{\mathcal{V}}^*_C|$. In that case, there has to be at least one scenario $\bar{\mathcal{V}}_n$ that satisfies $\bar{\mathcal{V}}_n \in \hat{\mathcal{V}}^*_C$ and $\bar{\mathcal{V}}_n \notin \hat{\mathcal{V}}'_C$. In what follows, we prove that $\bar{\mathcal{V}}_n$ does not exists. Assume first that for any $\bar{\mathcal{V}}'_i \in \hat{\mathcal{V}}'_C$, we have $\bar{\mathcal{V}}'_i \setminus \bar{\mathcal{V}}_n \neq \emptyset$. If we choose prevented vulnerabilities to be $\bar{\mathcal{V}}_P = \{v_{p_1}, \ldots, v_{p_m}\}$, where $v_{p_i} \in \bar{\mathcal{V}}'_i \setminus \bar{\mathcal{V}}_n$, we see that $\bar{\mathcal{V}}_P$ prevents $\hat{\mathcal{V}}'_C$. However, this set does not prevent $\hat{\mathcal{V}}^*_C$ since it does not prevent $\bar{\mathcal{V}}_n$. This is inconsistent with the fact that both $\hat{\mathcal{V}}^*_C$ and $\hat{\mathcal{V}}'_C$ are sufficient representations of $\tilde{\mathcal{V}}_C$ since any $\bar{\mathcal{V}}_P$ that prevents one sufficient representation needs to prevent any other as well. Therefore, there has to be at least one set $\bar{\mathcal{V}}'_k \in \hat{\mathcal{V}}'_C$ such that $\bar{\mathcal{V}}'_k \subset \bar{\mathcal{V}}_n$. Let the set of prevented vulnerabilities be now $\bar{\mathcal{V}}_P = \{v_{p_1}, \ldots, v_{p_l}\}$, where $v_{p_i} \in \bar{\mathcal{V}}_i \setminus \bar{\mathcal{V}}_n$ for $\bar{\mathcal{V}}_i \in \hat{\mathcal{V}}^*_C, \bar{\mathcal{V}}_i \neq \bar{\mathcal{V}}_n$ and $v_{p_n} \in \bar{\mathcal{V}}_n \setminus \bar{\mathcal{V}}'_k$. Note that $\bar{\mathcal{V}}_i \setminus \bar{\mathcal{V}}_n$ is always nonempty since $\bar{\mathcal{V}}_i \not\subseteq \bar{\mathcal{V}}_n$ by the assumption. This set prevents $\hat{\mathcal{V}}^*_C$. However, this set does not prevent $\hat{\mathcal{V}}'_C$ since it does not prevent $\bar{\mathcal{V}}'_k$, which is again in contradiction with the fact that both $\hat{\mathcal{V}}^*_C$ and $\hat{\mathcal{V}}'_C$ are sufficient representations. Thus, we conclude that the set $\bar{\mathcal{V}}_n$ does not exist, which contradicts the existence of $\hat{\mathcal{V}}'_C$ with $|\hat{\mathcal{V}}'_C| \leq |\hat{\mathcal{V}}^*_C|$. □

3.2 | Reducing number of executions of impact set function

As we stated, the impact set function is estimated based on a physical model of the system, and it can become expensive to calculate for models of large dimension. Therefore, it is desirable to reduce the number of executions of this function as much as possible. One way to do this would be to store the combinations of sensors and actuators for which we have already evaluated the impact function. These combinations can then be further divided into two lists, which we denote with $\mathcal{C}_+$ and $\mathcal{C}_-$. These lists can be updated online but also in a preprocessing step based on some a priori knowledge.

The list $\mathcal{C}_+$ contains combinations of sensors and actuators that lead to the impact greater than or equal to $I_{\min}$. For instance, say that the list contains combination $(\bar{\mathcal{A}}', \bar{\mathcal{S}}')$, and we need to check if the impact for some scenario $\bar{\mathcal{V}}$ is greater than or equal to $I_{\min}$. If $\bar{\mathcal{A}}' \subseteq \bar{\mathcal{A}}(\bar{\mathcal{V}})$ and $\bar{\mathcal{S}}' \subseteq \bar{\mathcal{S}}(\bar{\mathcal{V}})$, then we know from Assumption 3 that $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}})) \geq I_{\min}$. In other words, if the combination $(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}}))$ for which we need to calculate impact contains any combination from $\mathcal{C}_+$, its impact is greater than or equal to $I_{\min}$. The list $\mathcal{C}_-$ contains those combinations that lead to impact less than $I_{\min}$. Let the combination $(\bar{\mathcal{A}}', \bar{\mathcal{S}}')$ be the element of $\mathcal{C}_-$. If $\bar{\mathcal{A}}(\bar{\mathcal{V}}) \subseteq \bar{\mathcal{A}}'$ and $\bar{\mathcal{S}}(\bar{\mathcal{V}}) \subseteq \bar{\mathcal{S}}'$, then it follows from Assumption 3 that $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}})) < I_{\min}$. Thus, if the combination $(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}}))$ for which we need to calculate impact is the subset of a combination from $\mathcal{C}_-$, its impact is less than $I_{\min}$.

Remark 5. Since the number of combinations of sensors and actuators investigated rapidly grows, we can store in $\mathcal{C}_+$ ($\mathcal{C}_-$) only combinations that contain relatively small (relatively large) number of sensors and actuators.

3.3 | Algorithm for constructing the sufficient representation of minimal cardinality

We now introduce a systematic way to find the sufficient representation of minimal cardinality $\hat{\mathcal{V}}^*_C$. We begin with introducing a set enumeration tree of the power set $2^{\mathcal{V}}$ shown in Figure 3. This tree representation was introduced in the work of Rymon[46] for the purposes of systematically searching through the power set.

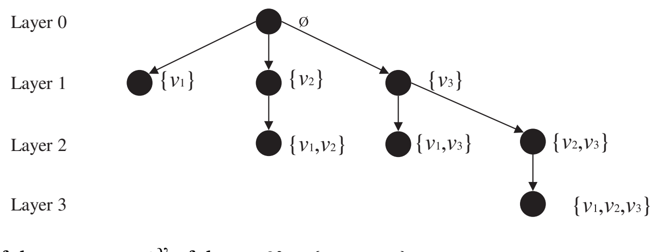

FIGURE 3 Tree representation of the power set $2^{\mathcal{V}}$ of the set $\mathcal{V} = \{v_1, v_2, v_3\}$

In our case, each node of the tree represents one attack scenario. The tree has $|\mathcal{V}| + 1$ layers enumerated with $p = 0, 1, \ldots, |\mathcal{V}|$, where layer p only contains the subsets of $\mathcal{V}$ with the cardinality equal to p. Another important property of the tree is that the connections are based on the following two principles. Firstly, the node $\bar{\mathcal{V}}$ in layer p can be connected only to nodes $\bar{\mathcal{V}} \cup v_j$, $v_j \notin \bar{\mathcal{V}}$ in layer $p + 1$. Secondly, the node $\bar{\mathcal{V}}$ is connected to node $\bar{\mathcal{V}} \cup v_j$, only if for all $v_i \in \bar{\mathcal{V}}$, we have $j < i$. For instance, in the case of the graph shown Figure 3, the node $\{v_2\}$ is connected to $\{v_1, v_2\}$ but not to $\{v_2, v_3\}$.

To explore the tree for the set $\hat{\mathcal{V}}_C^*$, we adopt a *Breadth-first search* algorithm.[62] This algorithm explores the tree by layers, that is, it does not move to the next layer before all the scenarios from the current layer are explored. For each scenario in the current layer, the algorithm first performs *classification* of that scenario. Once all the scenarios are classified, the algorithm *generates new scenarios* that will be searched in the next layer. If there are no more scenarios to be explored, the algorithm terminates. In the following, we explain the classification and generation steps.

3.3.1 | Classification step

The algorithm keeps the list of scenarios to be explored in the current layer, which is denoted by $\tilde{\mathcal{V}}_L$. For every scenario $\bar{\mathcal{V}}$ from $\tilde{\mathcal{V}}_L$, the algorithm first calculates $\pi(\bar{\mathcal{V}})$. If $\pi(\bar{\mathcal{V}}) > \pi_{\max}$, the algorithm moves to the next scenario. Otherwise, it checks if $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}})) \geq I_{\min}$. The algorithm first tries to determine this based on the lists C_+ and C_- introduced in Section 3.2. If that is not possible, the impact function is evaluated. It is also indicated that impact was calculated to update C_+ and C_- later on.

Based on the impact, the scenario is classified in one of the following two categories. The first category is critical scenarios that we want to find, and these scenarios are stored in the set $\hat{\mathcal{V}}_C^*$. If the scenario is not critical, it is stored in the list that we denote with $\tilde{\mathcal{V}}_O$. The reason for this is that although the scenario is not critical by itself, combining this scenario with additional vulnerabilities may result in critical scenario. Once all the scenarios are classified, the algorithm empties $\tilde{\mathcal{V}}_L$.

The important observation is that the classification of scenarios within a layer can be performed independently for each scenario. Thus, if we have N cores available, the classification step can be executed in parallel and, in that way, reduce the execution time of this step N times. Since the classification step involves evaluating impact and complexity function large number of times, which is expected to be the most time-consuming action in the algorithm, significant time savings can be achieved with parallelization.

Once the classification step is finished, the lists C_+ and C_- are updated with the combinations of sensors and actuators for which the impact was evaluated. The reason why we do not update the lists within the classification step is in order to be able to execute this step in parallel. The algorithm then moves to the generation step

3.3.2 | Generation step

To reduce the number of scenarios to generate in the next layer, the algorithm relies on Lemmas 1 and 2. Mainly, if a scenario has a complexity larger than $\pi_{\max}$ the algorithm does not generate any other new scenarios based on it since these are not critical (Lemma 1). Additionally, if the algorithm classifies a scenario as critical, it does not generate other scenarios that contain this one since $\hat{\mathcal{V}}_C^*$ does not contain these (Lemma 2). In other words, every time we find a critical scenario or a scenario with complexity larger than $\pi_{\max}$, we eliminate the branch of the tree starting from it. Thus, only scenarios from $\tilde{\mathcal{V}}_O$ are used to generate new scenarios. In this way, the number of scenarios we search through in the next layer is potentially significantly reduced.

The generation of scenarios is performed according to the following rules. Firstly, for each scenario $\bar{\mathcal{V}}$ from the list $\tilde{\mathcal{V}}_O$, the algorithm considers a scenario $v_i \cup \bar{\mathcal{V}}$ as a candidate to be added to the list $\tilde{\mathcal{V}}_L$, only if for all $v_j \in \bar{\mathcal{V}}$ it holds $i < j$. This rule follows from the specific tree structure. However, from the tree structure, it also follows that a scenario in the layer $p + 1$ can contain a critical scenario from the previous layer, although the branch starting from that critical scenario was eliminated. Thus, to obtain a sufficient representation of minimal cardinality, the second rule is to add the candidate scenario $v_i \cup \bar{\mathcal{V}}$ to the list $\tilde{\mathcal{V}}_L$ only if there does not exist a critical scenario in the list $\hat{\mathcal{V}}_C^*$ that is contained in $v_i \cup \bar{\mathcal{V}}$. Once the new scenarios are generated for all the scenarios from $\tilde{\mathcal{V}}_O$, $\tilde{\mathcal{V}}_O$ is set to $\emptyset$.

3.3.3 | Algorithm — formulation and analysis

Based on the previous discussion, we formulate Algorithm 1 for constructing the sufficient representation of minimal cardinality. It is important to emphasize that the running time of the algorithm depends on the choice of $I(\cdot)$ and $\pi(\cdot)$, thresholds $I_{\min}$ and $\pi_{\max}$, and the size of the set $\mathcal{V}$ in a nontrivial way. In the worst case, the algorithm may end up searching every subset of the set $\mathcal{V}$. Thus, based on the available time, the search can be restricted to only the first p layers.

In that case, the algorithm searches in the worst case $O(n_v^p)$ scenarios. However, this number is expected to be reduced by searching systematically.

We conclude this section by formally proving that the algorithm returns the sufficient representation of minimal cardinality. In case that the algorithm is restricted to search the first p layers, the algorithm returns the sufficient representation of minimal cardinality for the set $\tilde{\mathcal{V}}_C^{(p)}$, which contains all the critical scenarios with cardinality less than or equal to p.

Algorithm 1 Finding the sufficient representation of minimal cardinality

```
1:  **Input:** $\mathcal{V}$, $\pi(\cdot)$, $I(\cdot)$, $\pi_{\max}$, $I_{\min}$
2:  **Output:** $\hat{\mathcal{V}}_C^*$
3:  $\hat{\mathcal{V}}_C^* \leftarrow \emptyset$;        % The sufficient representation of minimal cardinality
4:  $\tilde{\mathcal{V}}_L \leftarrow \emptyset$;        % The list of scenarios to be searched in the current layer
5:  $\tilde{\mathcal{V}}_O \leftarrow \emptyset$;        % The list of scenarios that are used for generating new scenarios
6:  $\mathcal{C}_+, \mathcal{C}_- \leftarrow \emptyset$;   % The lists in which we store explored combinations of sensors and actuators
7:  % Initialize the list $\tilde{\mathcal{V}}_L$ with scenarios containing one vulnerability
8:  **for** $i = 1:1:n_v$ **do**
9:      add $\{v_i\}$ to the bottom of $\tilde{\mathcal{V}}_L$
10: **end for**
11: **while** $\tilde{\mathcal{V}}_L \neq \emptyset$ **do**
12:     % Classification of scenarios that is executed in parallel on $N$ cores
13:     **for** every scenario $\bar{\mathcal{V}}$ in $\tilde{\mathcal{V}}_L$ **do**
14:         **if** $\pi(\bar{\mathcal{V}}) \leq \pi_{\max}$ **then**
15:             determine if $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}}))$ is larger than $I_{\min}$, either using $\mathcal{C}_+$ and $\mathcal{C}_-$ or by evaluating impact function
16:             indicate if impact function was evaluated
17:             **if** $I(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}})) \geq I_{\min}$ **then**
18:                 add $\bar{\mathcal{V}}$ to $\hat{\mathcal{V}}_C^*$
19:             **else**
20:                 add $\bar{\mathcal{V}}$ to $\tilde{\mathcal{V}}_O$
21:             **end if**
22:         **end if**
23:     **end for**
24:     $\tilde{\mathcal{V}}_L \leftarrow \emptyset$
25:     update $\mathcal{C}_+$ and $\mathcal{C}_-$ with combinations $(\bar{\mathcal{A}}(\bar{\mathcal{V}}), \bar{\mathcal{S}}(\bar{\mathcal{V}}))$ for which the impact was evaluated
26:     % Generation of new scenarios
27:     **for** every scenario $\bar{\mathcal{V}}$ in $\tilde{\mathcal{V}}_O$ **do**
28:         find the vulnerability with minimal index $j$ in $\bar{\mathcal{V}}$
29:         **for** $i = 1:1:j-1$ **do**
30:             **if** there is no scenario from $\hat{\mathcal{V}}_C^*$ contained in $\{\bar{\mathcal{V}} \cup v_i\}$ **then**
31:                 add $\{\bar{\mathcal{V}} \cup v_i\}$ to the bottom of $\tilde{\mathcal{V}}_L$
32:             **end if**
33:         **end for**
34:     **end for**
35:     $\tilde{\mathcal{V}}_O \leftarrow \emptyset$
36: **end while**
```

Theorem 1. *Algorithm 1 returns the sufficient representation of minimal cardinality $\hat{\mathcal{V}}_C^*$.*

Proof. We first show that the set $\hat{\mathcal{V}}_C^*$ formed by Algorithm 1 contains the sufficient representation of minimal cardinality. We use an induction argument for this purpose. Let $p = 1$. Since the set $\tilde{\mathcal{V}}_L$ is initialized with all the scenarios of the first layer, all of the critical scenarios among these are for sure found, and added to the set $\hat{\mathcal{V}}_C^*$, so the claim holds for the first layer. Suppose now that the algorithm reaches the layer p with a given set $\hat{\mathcal{V}}_C^*$. Assume that the list contains all the scenarios with the cardinality less than or equal to p that are contained in the sufficient representation of minimal cardinality. In the layer $p + 1$, all the critical scenarios among generated scenarios are added to $\hat{\mathcal{V}}_C^*$. The scenarios from the layer $p + 1$ that are not generated fall in one of the two categories. The first category are the scenarios that contain the scenarios with the complexity larger than $\pi_{\max}$. From Lemma 1, we know that these scenarios cannot be critical, so these scenarios do not belong to the sufficient representation of minimal cardinality. The second category are those scenarios that contain some of the critical scenarios that are already added to $\hat{\mathcal{V}}_C^*$. These scenarios do not belong to the sufficient representation of minimal cardinality based on Lemma 2. Thus, the claim holds for the layer $p + 1$ as well, so $\hat{\mathcal{V}}_C^*$ contains all the scenarios from the sufficient representation of minimal cardinality.

Note that $\hat{\mathcal{V}}_C^*$ is a sufficient representation of $\tilde{\mathcal{V}}_C$. The reason is that every time we prevent all the scenarios within $\hat{\mathcal{V}}_C^*$, we know that the sufficient representation of minimal cardinality is prevented since it is contained in $\hat{\mathcal{V}}_C^*$. This implies that $\tilde{\mathcal{V}}_C$ is prevented. On the other hand, $\hat{\mathcal{V}}_C^*$ is subset of $\tilde{\mathcal{V}}_C$, so every time we prevent $\tilde{\mathcal{V}}_C$, we know that $\hat{\mathcal{V}}_C^*$ is prevented.

It remains to be proven that $\hat{\mathcal{V}}_C^*$ does not contain any two scenarios $\bar{\mathcal{V}}$ and $\bar{\mathcal{V}}'$, such that $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}'$. Firstly, for the two scenarios $\bar{\mathcal{V}} \neq \bar{\mathcal{V}}'$ from the same layer, the condition $\bar{\mathcal{V}} \subseteq \bar{\mathcal{V}}'$ cannot be satisfied. On the other hand, once we generate scenarios to be explored in the layer p, we always check if these contain any other scenarios from the lower layers previously added to $\hat{\mathcal{V}}_C^*$ (Line 30). Thus, the overlaps are not possible either for scenarios from different layers, so we conclude that $\hat{\mathcal{V}}_C^*$ represents the sufficient representation of minimal cardinality. □

Corollary 1. *If we restrict Algorithm 1 to search the first p layers of the tree, the sufficient representation of minimal cardinality of the set* $\tilde{\mathcal{V}}_C^{(p)}$ *is returned.*

Proof. To prove the claim, we introduce the complexity function that is defined as

$$\pi'(\bar{\mathcal{V}}) = \begin{cases} \pi(\bar{\mathcal{V}}), & \text{if } |\bar{\mathcal{V}}| \leq p \\ \max\left\{\pi(\bar{\mathcal{V}}), \pi'_{\max}\right\}, & \text{if } |\bar{\mathcal{V}}| > p, \end{cases}$$

where $\pi'_{\max} > \pi_{\max}$. This function satisfies Assumption 4 and represents a possible candidate for the complexity function. For any $\bar{\mathcal{V}} \subseteq \mathcal{V}$ and $|\bar{\mathcal{V}}| \leq p$, $\pi'(\bar{\mathcal{V}}) = \pi(\bar{\mathcal{V}})$. Thus, the sets of critical scenarios with the cardinalities less than or equal to p are the same for both $\pi'(\cdot)$ and $\pi(\cdot)$. However, the scenarios with the cardinality greater than p are with complexity larger than $\pi_{\max}$ if the function $\pi'(\cdot)$ is used. In that case, scenarios with cardinality greater than p cannot be critical based on Lemma 1, which implies that the set of critical scenarios is equal to $\tilde{\mathcal{V}}_C^{(p)}$. Therefore, if we apply Algorithm 1 with $\pi'(\cdot)$ as a complexity function, then it follows from Theorem 1 that the sufficient representation of minimal cardinality of $\tilde{\mathcal{V}}_C^{(p)}$ is returned. □

4 | SUBMODULAR NATURE OF SECURITY MEASURE ALLOCATION PROBLEM

In this section, we address solving the security measure allocation problem. This problem is an integer linear program and, thus, NP-hard to solve in general. The first important result of this section is to show that this problem can be casted as a minimization of linear function subject to a submodular constraint. In that case, a polynomial time greedy algorithm can be used to find a suboptimal solution with a guaranteed performance bounds. The second important result of this section is to show that the greedy algorithm gives the best performance guarantees on the solution once the sufficient representation of minimal cardinality $\hat{\mathcal{V}}_C^*$ is used to represent the set of critical scenarios $\tilde{\mathcal{V}}_C$. In case that Algorithm 1 was stopped due to time constraints after p layers, the same results hold but for $\tilde{\mathcal{V}}_C^{(p)}$. Before we start proving these claims, we introduce a necessary theoretical background.

4.1 | Submodularity

Submodularity is referred to as a diminishing returns property of a set function. If a set function is submodular, adding an element to a set $\bar{\mathcal{M}}$ results in a larger increase of the function than adding that element to a larger set containing $\bar{\mathcal{M}}$.[63]

Definition 5. Let $\mathcal{M} = \{m_1, \dots, m_n\}$ be a finite nonempty set. A set function $G : 2^{\mathcal{M}} \to \mathbb{R}$ is said to be *submodular* if for all $\bar{\mathcal{M}} \subseteq \bar{\mathcal{M}}' \subseteq \mathcal{M}$ and $m \notin \bar{\mathcal{M}}'$, we have $G(\bar{\mathcal{M}} \cup m) - G(\bar{\mathcal{M}}) \geq G(\bar{\mathcal{M}}' \cup m) - G(\bar{\mathcal{M}}')$.

The submodularity property plays an important role in combinatorial optimization. Mainly, certain classes of combinatorial optimization problems that have a submodular structure can be approximately solved with performance guarantees in polynomial time. One of these problems is the minimization of a linear function subject to a submodular constraint, which is defined as follows.

Problem 2. Minimization of linear function under a submodular constraint

$$\begin{aligned} \underset{\bar{\mathcal{M}}\subseteq\mathcal{M}}{\text{minimize}} \quad & b(\bar{\mathcal{M}}) = \sum_{m\in\bar{\mathcal{M}}} b_m \\ \text{subject to} \quad & G(\bar{\mathcal{M}}) \geq G_{\min}, \end{aligned}$$

where $b_m \in \mathbb{R}^+$ is a cost assigned to each $m \in \mathcal{M}$, $G_{\min} > 0$ is a bound, and $G(\cdot)$ is submodular, nondecreasing, and integer-valued set function. It is shown in the work of Wolsey[47] that this problem can be approximately solved by Algorithm 2. The algorithm first forms the greedy set $\bar{\mathcal{M}}_G = \emptyset$. In each iteration, the cost benefit ratio

$$\frac{b_m}{G\left(\bar{\mathcal{M}}_G \cup m\right) - G\left(\bar{\mathcal{M}}_G\right)}$$

for each $m \in \mathcal{M}\backslash\bar{\mathcal{M}}_G$ is calculated. At the end of each iteration, the element m^* that achieves the lowest value of the cost benefit ratio is added to $\bar{\mathcal{M}}_G$. This process is repeated at most $|\mathcal{M}|$ times, thus, it can be executed fast even for large sets of security measures. The performance guarantees of the algorithm are provided in Lemma 3.

Algorithm 2 A greedy heuristics for Problem 2[47]

```
1: Input: $\mathcal{M}$, $G(\cdot)$, $b_1, \dots, b_n$, $G_{\min}$
2: Output: $\bar{\mathcal{M}}_G$
3: $\bar{\mathcal{M}}_G \leftarrow \emptyset$
4: while $G(\bar{\mathcal{M}}_G) < G_{\min}$ do
5:     $m^* \leftarrow \text{argmin}\{\frac{b_m}{G(\bar{\mathcal{M}}_G\cup\{m\})-G(\bar{\mathcal{M}}_G)} : m \in \mathcal{M} \setminus \bar{\mathcal{M}}_G\}$
6:     $\bar{\mathcal{M}}_G \leftarrow \bar{\mathcal{M}}_G \cup m^*$
7: end while
```

Lemma 3. (See theorem 1 in the work of Wolsey[47])
Let $G(\cdot)$ be a nondecreasing, submodular, and integer-valued set function with $G(\emptyset) = 0$. Consider Problem 2 and denote by $b(\bar{\mathcal{M}}_O)$ the optimal value and, by $b(\bar{\mathcal{M}}_G)$, the value found by Algorithm 2. Then,

$$\frac{b(\bar{\mathcal{M}}_G)}{b(\bar{\mathcal{M}}_O)} \leq H\left(\max_{m\in\mathcal{M}} G(m)\right) \qquad H(d) = \sum_{i=1}^{d}\frac{1}{i}. \tag{6}$$

Remark 6. Note that bound (6) is dependent on the function $G(\cdot)$. In particular, the bound grows logarithmically in the value of $\max_{m\in\mathcal{M}} G(m)$, so the performance guarantees remain relatively good even for large values of $G(m)$. Furthermore, this bound represents the worst case performance guarantees. Algorithm 2 can perform much better in practice.

4.2 | Submodularity and security measure allocation problem

In what follows, we prove that the security measure allocation problem is an instance of Problem 2. In that case, we can use polynomial time Algorithm 2 for finding an approximate solution with guaranteed performance.

Recall that $\bar{\mathcal{M}}$ represents the set of security measures we want to choose and that $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ is the subset of vulnerabilities prevented by the security measures $\bar{\mathcal{M}}$. From Assumption 1, it follows that all the attack scenarios that exploit vulnerabilities from the set $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ become prevented. To model this relation, with each scenario $\bar{\mathcal{V}}_i \in \hat{\mathcal{V}}_C^*$, we define a gain function

$$g_i(\bar{\mathcal{M}}) = \min\left\{|\bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i|, 1\right\}.$$

If $\bar{\mathcal{V}}_i$ requires any of the vulnerabilities from $\bar{\mathcal{V}}(\bar{\mathcal{M}})$ to be conducted, it is prevented, and $g_i(\bar{\mathcal{M}}) = 1$. Otherwise, we do not prevent $\bar{\mathcal{V}}_i$, so $g_i(\bar{\mathcal{M}}) = 0$. The global gain function is then

$$G(\bar{\mathcal{M}}) = \sum_{\bar{\mathcal{V}}_i \in \hat{\mathcal{V}}_C^*} g_i(\bar{\mathcal{M}}) = \sum_{\bar{\mathcal{V}}_i \in \hat{\mathcal{V}}_C^*} \min\left\{|\bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i|, 1\right\}.$$

The security measure allocation problem can now be reformulated as follows.

Problem 3. Security measure allocation

$$\begin{aligned} \underset{\bar{\mathcal{M}} \subseteq \mathcal{M}}{\text{minimize}} \quad & c(\bar{\mathcal{M}}) = \sum_{m \in \bar{\mathcal{M}}} c_m \\ \text{subject to} \quad & G(\bar{\mathcal{M}}) = \left|\hat{\mathcal{V}}_C^*\right|. \end{aligned}$$

The objective function $c(\bar{\mathcal{M}})$ we are trying to minimize represents the total cost of deployed security measures $\bar{\mathcal{M}}$. The constraint $G(\bar{\mathcal{M}}) = |\hat{\mathcal{V}}_C^*|$ comes from the fact that $G(\cdot)$ is equal to $|\hat{\mathcal{V}}_C^*|$ once all of the scenarios from $\hat{\mathcal{V}}_C^*$ are prevented. This automatically implies that all the critical scenarios $\tilde{\mathcal{V}}_C$ are prevented since $\hat{\mathcal{V}}_C^*$ represents the sufficient representation of $\tilde{\mathcal{V}}_C$.

We now introduce the first important result of this section, which is that Problem 3 has the same submodular structure as Problem 2. That implies that we can use Algorithm 2 to find a suboptimal solution with the guarantees given in Lemma 3.

Theorem 2. *Problem 1 is an instance of Problem 2.*

Proof. To show that the claim holds, we prove that $G(\cdot)$ satisfies the conditions stated in Lemma 3, that is, it is submodular, nondecreasing, and integer valued. We proved submodularity and nondecreasing property for similar set function in theorem 1.[49] We include the proof here because of completeness.

Submodularity. It suffices to show that $g_i(\cdot)$ is submodular since submodularity is preserved under a nonnegative sum.[63] We prove that $g_i(\cdot)$ is submodular by using the definition of submodularity and contradiction argument. Let

$$\Delta_m(\bar{\mathcal{M}}) = g_i(\bar{\mathcal{M}} \cup m) - g_i(\bar{\mathcal{M}})$$

be the gain achieved by adding the security measure $m \in \mathcal{M}$ to the set of already deployed security measures $\bar{\mathcal{M}}$. We show that this gain could be either 0 or 1. The gain is equal to zero in two situations. The first situation is when the scenario $\bar{\mathcal{V}}_i$ is already prevented by the deployed security measures $\bar{\mathcal{M}}$. We then have $g_i(\bar{\mathcal{M}}) = g_i(\bar{\mathcal{M}} \cup m) = 1$. Thus, $\Delta_m(\bar{\mathcal{M}}) = 0$. The second situation occurs when $\bar{\mathcal{M}} \cup m$ does not prevent $\bar{\mathcal{V}}_i$. We then have $g_i(\bar{\mathcal{M}}) = g_i(\bar{\mathcal{M}} \cup m) = 0$, hence $\Delta_m(\bar{\mathcal{M}}) = 0$. The value $\Delta_m(\bar{\mathcal{M}}) = 1$ is achieved when scenario $\bar{\mathcal{V}}_i$ is prevented by m but not $\bar{\mathcal{M}}$. In that case, $g_i(\bar{\mathcal{M}}) = 0$ and $g_i(\bar{\mathcal{M}} \cup m) = 1$, so $\Delta_m(\bar{\mathcal{M}}) = 1$. Based on the previous discussion, it follows:

$$\Delta_m(\bar{\mathcal{M}}) = \begin{cases} 1\,, & \bar{\mathcal{V}}_m \cap \bar{\mathcal{V}}_i \neq \emptyset \text{ and } \bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i = \emptyset \\ 0\,, & \text{otherwise.} \end{cases} \tag{7}$$

Assume that $g_i(\cdot)$ is not submodular. Then, there exist $\bar{\mathcal{M}} \subseteq \bar{\mathcal{M}}'$ and $m \notin \bar{\mathcal{M}}'$ such that $\Delta_m(\bar{\mathcal{M}}) < \Delta_m(\bar{\mathcal{M}}')$. That is only possible if $\Delta_m(\bar{\mathcal{M}}) = 0$ and $\Delta_m(\bar{\mathcal{M}}') = 1$. From (7), it follows $\bar{\mathcal{V}}(\bar{\mathcal{M}}') \cap \bar{\mathcal{V}}_i = \emptyset$ and $\bar{\mathcal{V}}_m \cap \bar{\mathcal{V}}_i \neq \emptyset$. However, since $\bar{\mathcal{V}}(\bar{\mathcal{M}}) \subseteq \bar{\mathcal{V}}(\bar{\mathcal{M}}')$, we have $\bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i = \emptyset$. It then follows from (7) that $\Delta_m(\bar{\mathcal{M}}) = 1$, which contradicts the assumption. Thus, $g_i(\cdot)$ is submodular.

Nondecreasing property. Let $\bar{\mathcal{M}} \subseteq \bar{\mathcal{M}}' \subseteq \mathcal{M}$. From (1), we have $|\bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i| \leq |\bar{\mathcal{V}}(\bar{\mathcal{M}}') \cap \bar{\mathcal{V}}_i|$ for any $\bar{\mathcal{V}}_i$, which implies that $g_i(\cdot)$ is nondecreasing. Thus, $G(\cdot)$ is nondecreasing as well, as a nonnegative sum of nondecreasing set functions.

Integer valued. Since $g_i(\cdot)$ can take only values 0 or 1, $G(\cdot)$ is integer valued set function. □

Remark 7. In the case when each security measure prevents exactly one vulnerability, Problem 1 becomes a weighted hitting set problem. If the sufficient representation of minimal cardinality contains all the vulnerabilities from the first layer, Problem 1 reduces to a weighted set cover problem. Both of the problems are known to be NP complete.[64]

Remark 8. Note that the submodularity property of Problem 1 follows from the submodularity of function $g_i(\cdot)$, which is disconnected from the choice of the impact and complexity set functions.

We now introduce the second important result of this section, which is to show the benefit of using the sufficient representation of minimal cardinality $\hat{\mathcal{V}}_C^*$ compared with other sufficient representations. Note that $G(\cdot)$ is dependent on the set $\hat{\mathcal{V}}_C^*$ and recall from Remark 6 that guarantees on performance stated in Lemma 3 are dependent on $G(\cdot)$. We now prove that $\hat{\mathcal{V}}_C^*$ provides the tightest guarantees on performance among the sufficient representations.

Theorem 3. *Let $\hat{\mathcal{V}}_C^*$ be the sufficient representation of minimal cardinality of the set $\tilde{\mathcal{V}}_C$, and let $\hat{\mathcal{V}}_C$ be any other sufficient representation. Then, $\hat{\mathcal{V}}_C^*$ guarantees the best performance in terms of the bound (6), that is,*

$$H\left(\max_{m\in\mathcal{M}} G(m)\right) \leq H\left(\max_{m\in\mathcal{M}} G'(m)\right),$$

where $G(\bar{\mathcal{M}}) = \sum_{\bar{\mathcal{V}}_i\in\hat{\mathcal{V}}_C^*} \min\{|\bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i|, 1\}$ and $G'(\bar{\mathcal{M}}) = \sum_{\bar{\mathcal{V}}_i\in\hat{\mathcal{V}}_C} \min\{|\bar{\mathcal{V}}(\bar{\mathcal{M}}) \cap \bar{\mathcal{V}}_i|, 1\}$.

Proof. To prove the claim of the theorem, we first prove the sufficient representation of minimal cardinality $\hat{\mathcal{V}}_C^*$ is contained in any other sufficient representation $\hat{\mathcal{V}}_C$. Assume that $\hat{\mathcal{V}}_C^* = \{\bar{\mathcal{V}}_1, \dots, \bar{\mathcal{V}}_m\}$, $\hat{\mathcal{V}}_C = \{\bar{\mathcal{V}}_1', \dots, \bar{\mathcal{V}}_l'\}$, and let $\bar{\mathcal{V}}_j$ be an arbitrary selected scenario from $\hat{\mathcal{V}}_C^*$. We prove this scenario belongs to $\hat{\mathcal{V}}_C$ as well. The first option is that, for any $\bar{\mathcal{V}}_i' \in \hat{\mathcal{V}}_C$, we have $\bar{\mathcal{V}}_i' \setminus \bar{\mathcal{V}}_j \neq \emptyset$. If we choose a set of prevented vulnerabilities to be $\bar{\mathcal{V}}_P = \{v_{p_1}, \dots, v_{p_l}\}$, where $v_{p_i} \in \bar{\mathcal{V}}_i' \setminus \bar{\mathcal{V}}_j$, we see that $\bar{\mathcal{V}}_P$ prevents $\hat{\mathcal{V}}_C$. However, $\hat{\mathcal{V}}_C^*$ is not prevented by $\bar{\mathcal{V}}_P$ since $\bar{\mathcal{V}}_j$ is not prevented. This is impossible since $\hat{\mathcal{V}}_C$ and $\hat{\mathcal{V}}_C^*$ are sufficient representations. Therefore, there has to be at least one scenario $\bar{\mathcal{V}}_t' \in \hat{\mathcal{V}}_C$ such that $\bar{\mathcal{V}}_t' \subseteq \bar{\mathcal{V}}_j$. Assume that $\bar{\mathcal{V}}_t' \subset \bar{\mathcal{V}}_j$ and define the set of prevented vulnerabilities as $\bar{\mathcal{V}}_P = \{v_{p_1}, \dots, v_{p_m}\}$, where $v_{p_i} \in \bar{\mathcal{V}}_i \setminus \bar{\mathcal{V}}_j$ for $i \neq j$ and $v_{p_j} \in \bar{\mathcal{V}}_j \setminus \bar{\mathcal{V}}_t'$. This set prevents $\hat{\mathcal{V}}_C^*$ but does not prevent $\hat{\mathcal{V}}_C$ since it does not prevent $\bar{\mathcal{V}}_t'$. This is again in contradiction with the fact that both $\hat{\mathcal{V}}_C$ and $\hat{\mathcal{V}}_C^*$ are sufficient representations. Thus, the only option that remains is that there exists $\bar{\mathcal{V}}_t' \in \hat{\mathcal{V}}_C$, such that $\bar{\mathcal{V}}_j = \bar{\mathcal{V}}_t'$. Since $\bar{\mathcal{V}}_j$ was arbitrarily selected, we conclude that any scenario contained in $\hat{\mathcal{V}}_C^*$ has to be contained in $\hat{\mathcal{V}}_C$ as well.

Based on the previous discussion, we can represent $\hat{\mathcal{V}}_C$ as $\hat{\mathcal{V}}_C = \hat{\mathcal{V}}_C^* \cup (\hat{\mathcal{V}}_C \setminus \hat{\mathcal{V}}_C^*)$, where $\hat{\mathcal{V}}_C \setminus \hat{\mathcal{V}}_C^*$ is nonempty since $\hat{\mathcal{V}}_C^*$ is unique. Let $m \in \mathcal{M}$ be an arbitrary security measure. We then have

$$\begin{aligned} G'(m) &= \sum_{\bar{\mathcal{V}}_i\in\hat{\mathcal{V}}_C} \min\left\{|\bar{\mathcal{V}}_m \cap \bar{\mathcal{V}}_i|, 1\right\} = \sum_{\bar{\mathcal{V}}_i\in\hat{\mathcal{V}}_C^*} \min\left\{|\bar{\mathcal{V}}_m \cap \bar{\mathcal{V}}_i|, 1\right\} + \sum_{\bar{\mathcal{V}}_i\in\hat{\mathcal{V}}_C\setminus\hat{\mathcal{V}}_C^*} \min\left\{|\bar{\mathcal{V}}_m \cap \bar{\mathcal{V}}_i|, 1\right\} \\ &= G(m) + \sum_{\bar{\mathcal{V}}_i\in\hat{\mathcal{V}}_C\setminus\hat{\mathcal{V}}_C^*} \min\left\{|\bar{\mathcal{V}}_m \cap \bar{\mathcal{V}}_i|, 1\right\} \geq G(m). \end{aligned}$$

Thus, for any m, we have $G(m) \leq G'(m)$. This implies $H(\max_{m\in\mathcal{M}} G(m)) \leq H(\max_{m\in\mathcal{M}} G'(m))$, which concludes the proof. □

5 | ILLUSTRATIVE EXAMPLE

In this section, we illustrate the applicability of our framework. We consider an ICS that regulates temperature within a building and assume multiple security vulnerabilities to be present in the system. The goal is to identify the most critical scenarios that have to be prevented and then to select the security measures to prevent these scenarios at the minimal cost.

5.1 | System model

The variable-refrigerant-flow system consists of a compressor, a condenser, N evaporators, and N electronic expansion valves (EEV), as shown in Figure 4. Each EEV and evaporator pair corresponds to an area of the building, and the objective of the system is to maintain desirable temperature within the areas. We now briefly introduce the model of the system, and we refer the interested reader to see the work of Jain et al[48] for more detailed treatment.

Each area of the building is modeled with three state-space variables $x_i = [\,T_{ai}\;\; T_{wi}\;\; P_i\,]^T$, where T_{ai} is the temperature of the corresponding area, T_{wi} is the temperature of the evaporator's lumped coil wall, and P_i is the refrigerant pressure after leaving the evaporator. These variables are controlled using the control signal $u_i = [\,\omega_{fi}\;\; a_{vi}\,]^T$, where ω_{fi} is the speed of evaporator's fan that is used to cool down the evaporator coil, and a_{vi} is the control signal that is used to change the fluid resistance of EEV. Other dynamical states of the system are $x_c = [\,P_C\;\; P_q\;\; T_{wc}\,]^T$, where P_C represents the refrigerant pressure after leaving the compressor, P_q is the junction pressure, and T_{wc} is the temperature of compressor's lumped coil

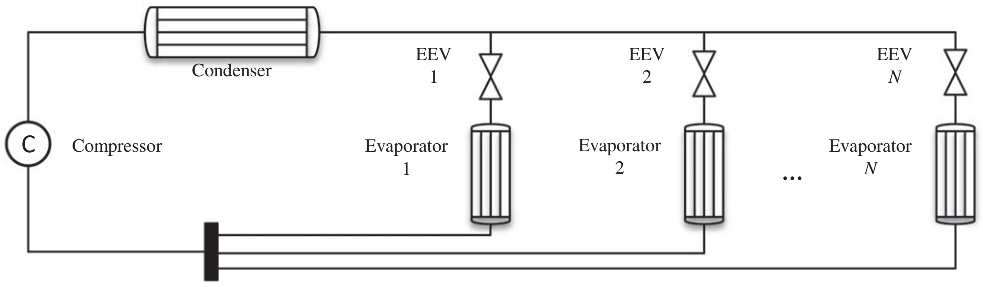

FIGURE 4 Physical infrastructure of the variable refrigerant flow control system. EEV, electronic expansion valve

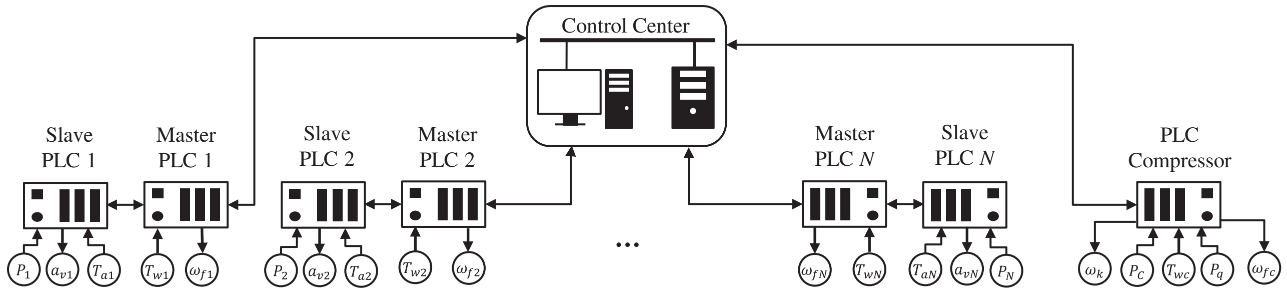

FIGURE 5 Cyber infrastructure of the variable refrigerant flow control system. PLC, programmable logic controller

wall. These variables are controlled using the control inputs $u_c = [\ \omega_k\ \ \omega_{fc}\ ]^T$, where ω_k represents the compressor speed, and ω_{fc} represents the speed of the compressor's fan. All of the states in the system are assumed to be measurable. Thus, the state, the control, and the measurement vectors are given by $x = [\ x_1^T\ ,\ \dots,\ x_N^T\ x_c^T\ ]^T$, $u = [\ u_1^T\ ,\ \dots,\ u_N^T\ u_c^T\ ]^T$, and $y = [\ x_1^T\ ,\ \dots,\ x_N^T\ x_c^T\ ]^T$, respectively.

For illustration purposes, we assume the cyber part of the system to be as shown in Figure 5. In this configuration, the equipment in each of the N areas is controlled using the two PLCs, the master PLC and the slave PLC. The master PLC is used to control the evaporator. It collects the temperature of the evaporator's lumped coil wall T_{wi} and controls the speed of the evaporator's fan ω_{fi}. It is also assumed that master PLCs receive and execute commands from the control center. The tasks of slave PLCs include collecting the measurements P_i and T_{ai} and controlling the actuator a_{vi}. It is assumed that slave PLCs communicate with the control center through their corresponding master PLCs. The compressor is controlled using a single PLC. This device collects the measurements P_C, P_q, and T_{wc} and controls the actuators ω_k and ω_{fc}. It also exchanges the measurements and executes the commands from the control center.

5.2 | Security vulnerabilities and security measures

To model the sets of security vulnerabilities $\mathcal{V}$ and security measures $\mathcal{M}$, we used the list of common security vulnerabilities provided in the work of Stouffer et al.[5] In particular, we assumed that the vulnerabilities listed in Table 3 are present in the system. Table 3 also contains the list of sensors and actuators that the attacker gains control over if the vulnerability v is exploited and the complexity π_v of exploiting v. The security measures are listed in Table 4 together with the vulnerabilities $\bar{\mathcal{V}}_m$ prevented by the security measure m and price c_m of implementing m. The way of selecting π_v and c_m is explained in further sections. We now briefly explain the vulnerabilities and security measures considered.

Firstly, computers within the control center are connected to other IT networks without adequate protection, and physical ports on the computers are not secured. These vulnerabilities open the space for the attacker to spread malware within the system, either through the other networks or by USB sticks. In both of the cases, the attacker gains absolute control over all the sensors and actuators within the system. The first vulnerability can be prevented by deploying and properly adjusting firewalls and the second one by locking or removing the ports.

Secondly, it was identified that the communication links between master and slave PLCs, PLCs and the control center, and sensors/actuators and all the PLCs are unsecured. This opens a space for the attacker to intercept the communication and conduct man-in-the-middle attacks. The sensors and actuators that the attacker gains control over are dependent on a communication link attacked. If a link between a sensor/actuator and PLC is attacked, the attacker gains control over

TABLE 3 List of vulnerabilities $\mathcal{V}$ identified within the system, sensors $\bar{S}_v$, and actuators $\bar{A}_v$ that the attacker gains control over if vulnerability v is exploited and complexity π_v of exploiting v

Vulnerability v	**Sensors $\bar{S}_v$ and Actuators $\bar{A}_v$ Controlled by Attacker**	**Complexity π_v**
The control center computers connected to other networks without protection	All sensors and actuators	1
Unsecured physical ports in the control center	All sensors and actuators	1
Insecure connection between sensor/actuator *i* and a PLC	Sensor/Actuator *i*	1-5
Insecure connection between slave PLC *i* and master PLC *i*	Sensors and actuators attached to slave PLC *i*	1-5
Insecure connection between master PLC *i* and the control center	Sensors and actuators attached to master PLC *i* and slave PLC *i*	1-5
Insecure connection between the compressor PLC and the control center	Sensors and actuators attached to the compressor PLC	1
Lack of physical protection of slave PLC *i*	Sensors and actuators attached to slave PLC *i*	1-5
Lack of physical protection of master PLC *i*	Sensors and actuators attached to master PLC *i* and slave PLC *i*	1-5
Lack of physical protection of the compressor PLC	Sensors and actuators attached to the compressor PLC	1
Lack of physical protection of sensor/actuator *i*	Sensor/Actuator *i*	1-5

Abbreviations: PLC, programmable logic controller.

TABLE 4 List of security measures $\mathcal{M}$ and vulnerabilities $\bar{\mathcal{V}}_m$ prevented by the security measure m and cost c_m of deploying m

Security Measure m	**Vulnerabilities $\mathcal{V}_m$ Prevented by m**	**Cost c_m**
Installing and properly adjusting firewalls between the control center and other IT networks	The attacker cannot gain access to the computers in the control center from other networks	3-5
Locking and removing the physical ports	The attacker cannot inject malware through the physical ports within the control center	1-2
Protecting an unsecured communication link	The attacker cannot intercept and modify messages going through the corresponding link	1-4
Physical protection of an individual component (a sensor, an actuator, or a PLC controller)	The attacker cannot gain physical access to the corresponding component	1-2
Physical protection of group of components (PLC controller and sensors and actuators attached to it)	The attacker cannot gain unauthorized access to the group of components and cannot exploit unprotected communication between PLC and sensors/actuators	5-10

Abbreviations: IT, information technology; PLC, programmable logic controller.

that sensor/actuator. If the attacker intercepts the communication between master and slave PLCs, it gains control over all the sensors and actuators attached to the slave PLC. If the attacker compromises a link between a PLC and the control center, it gains control over the sensors and actuators attached to that PLC and the corresponding slave PLC attached to it. The vulnerabilities of this type can be prevented by implementing encryption and authentication schemes.

Finally, lack of physical protection is identified within the system. If the attacker has physical access to a sensor/actuator, we assume that the attacker can gain control over that sensor/actuator. If the attacker gains unauthorized access to a slave PLC, it gains control over all the sensors and actuators attached to that PLC. In case of unauthorized access to a master PLC, the attacker gains control over all the sensors and actuators attached to both the master and the slave PLC. The unauthorized access can be prevented by introducing additional physical protection. The first alternative is to protect the group of components at the same time. The assumption is that physical protection prevents the attacker from gaining unauthorized access to the PLC and all the sensors and the actuators attached to that PLC. In addition, it also prevents the attacker from exploiting unsecured connections between the corresponding PLC and sensors/actuators because these components are usually connected with wires and lie in proximity of each other. The second alternative is to protect components individually.

5.3 | Critical scenarios

Checking if attack scenarios are critical (checking Definition 2) was done as follows. We used the impact function given in Table 1, and we set the threshold $I_{\min} = 2$. To check if the attack impact is above or below the threshold, we used the function *tzero* implemented in MATLAB, which returns both the normal rank of the system and the list of transmission zeros.

To check if a scenario is of complexity smaller than $\pi_{\max}$, we used the complexity function (5). We first assigned individual complexity π_v to each of the vulnerabilities. We assumed that the security vulnerabilities related to the equipment in areas:

- 1 to $\frac{N}{5}$ are of very low complexity;
- $\frac{N}{5} + 1$ to $\frac{2N}{5}$ are of low complexity;
- $\frac{2N}{5} + 1$ to $\frac{3N}{5}$ are of medium complexity;
- $\frac{3N}{5} + 1$ to $\frac{4N}{5}$ are difficult to exploit; and
- $\frac{4N}{5} + 1$ to N are very difficult to exploit.

The vulnerabilities related to the control center and the equipment controlling compressor were assumed to be of very low complexity. In this way, we achieved approximately the equal number of vulnerabilities belonging to each of the five groups from Table 2. To calculate the complexity of attack scenario consisting of more than one vulnerability, we used the complexity function (5). The threshold was set to be $\pi_{\max} = 5$. Note that this choice of the threshold $\pi_{\max}$ implies that all the scenarios with six or more vulnerabilities are with the complexity larger than $\pi_{\max}$. Thus, we need to explore only the first five layers of the power set $2^{\mathcal{V}}$ to find the critical scenarios.

5.4 | Searching for critical scenarios

To find the sufficient representation of minimal cardinality, we used a computer cluster consisting of 4 Intel® Core™ i7-4470S computer processors with 16 cores in total. To increase or decrease the number of vulnerabilities, we varied the number of areas in the range $N = 5$ to $N = 25$. By counting, it can be verified that the total number of vulnerabilities is equal to $14N + 14$, whereas the number of security measures is equal to $16N + 15$.

For the aforementioned specifications, we measured the execution time of Algorithm 1. The results are shown in Table 5. The execution time was the highest for the case of 364 vulnerabilities, where it reached 21.54 minutes. For the sake of comparison, the brute force search through the first five layers of the power set when vulnerability set consisted of 84 measures already took more than 8 minutes, whereas the estimated time of the brute force search for the vulnerability set consisting of 364 elements is more than 20 days. This demonstrates that systematic search can allow us to explore large sets of vulnerabilities in reasonable time.

The number of scenarios in the sufficient representation of minimal cardinality is reported in Table 6, together with the number of scenarios in each of the layers. As it can be seen, the attacker can conduct undetectable attack by exploiting only a single vulnerability. Naturally, if the attacker exploits vulnerabilities within the control center, it is able to conduct an undetectable attack because it controls all of the sensors and actuators. It also turned out that attacking any of the master PLCs leads to undetectable attacks as well. For instance, if the attacker exploits lack of physical protection of master PLC i, it can increase or decrease the temperatures T_{ai} and T_{wi} by an arbitrary value. This is possible since changes in T_{ai} and T_{wi} influence neither pressures nor temperatures in other areas. The number of scenarios in the second layer was equal to four for all the values of N. The reason is that these scenarios involved only the vulnerabilities of equipment related to the

TABLE 5 Execution times of Algorithm 1 with respect to number of areas

Number of Areas N	Number of Vulnerabilities n_v	Execution Time Algorithm 1, sec
5	84	4.230
10	154	13.954
15	224	80.285
20	294	366.030
25	364	1292.560

TABLE 6 The number of critical scenarios within $\hat{\mathcal{V}}_C^*$ in total and in each of the layers

Number of Areas N	Scenarios Total $\lvert\hat{\mathcal{V}}_C^*\rvert$	Scenarios Layer 1	Scenarios Layer 2	Scenarios Layer 3	Scenarios Layer 4	Scenarios Layer 5
5	226	14	4	16	0	192
10	508	24	4	32	64	384
15	854	34	4	48	192	576
20	1264	44	4	64	384	768
25	1738	54	4	80	640	960

compressor, which does not change by increasing or decreasing the number of areas. We also observe that scenarios from the layer 5 involved, exploiting vulnerabilities of the equipment related to the compressor. For example, since there exists a coupling between the pressures $P_1, \ldots, P_N, P_c$, and P_q, attacking P_i requires manipulating the compressor equipment to cover the attack.

5.5 | Allocating security measures

Once the sufficient representation of minimal cardinality was found, we moved to solving the security measure allocation problem. We used both the greedy algorithm introduced in the previous section but also the specialized integer linear program solver included in the Gurobi package. For each case of N, we performed simulations 500 times for different values of costs c_m of security measures. The values of c_m were randomly selected from the intervals given in Table 4.

We first compare the algorithms in terms of the execution time. The plot of the worst-case execution times of the algorithms is shown in Figure 6. The maximal execution time was reached for 415 security measures, and it was approximately 8.4 seconds for the Gurobi solver and 0.09 seconds for Algorithm 2. The explanation for short execution times lies in the fact that sufficient representation of minimal cardinality contained relatively small number of scenarios, and all the scenarios were with cardinality not larger than five. However, it can also be observed from Figure 6 that the worst-case execution time increased much faster for the Gurobi solver than for Algorithm 2. This may indicate that the restriction of using the Gurobi solver once the number of scenarios to prevent is very large. In that case, we can rely on Algorithm 2 to solve the problem with known performance guarantees. However, in this particular case, we conclude that the execution time did not represent an issue for any of the approaches.

Next, we compare the algorithms in terms of the solution obtained. We first remark that the Gurobi solver managed to find the optimal objective value of the problem $b(\bar{\mathcal{M}}_O)$ in all the five cases, for all the realizations of costs. Again, we find the reason for this to be relatively small number of scenarios contained in the sufficient representation of minimal cardinality and sparsity of the scenarios. Thus, although integer linear programs are NP-hard in general, in this case, we managed to find the optimal solution for the problem in a matter of seconds. To compare the solution $b(\bar{\mathcal{M}}_G)$ returned

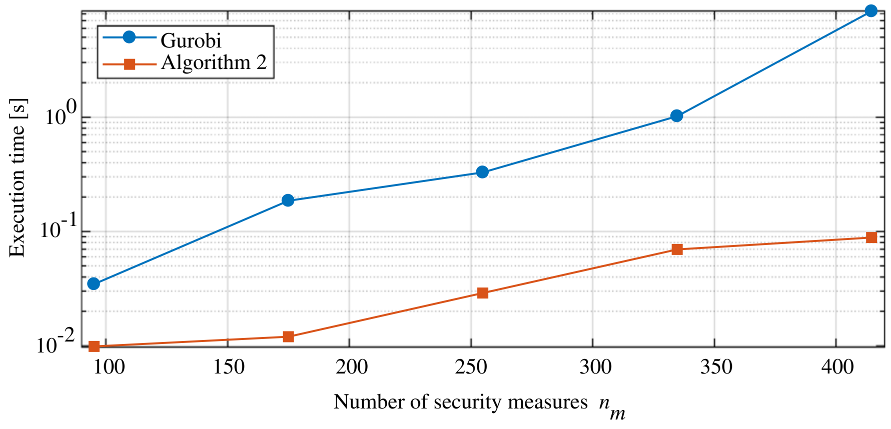

FIGURE 6 Comparison of Algorithm 2 and the Gurobi solver in terms of execution time [Colour figure can be viewed at wileyonlinelibrary.com]

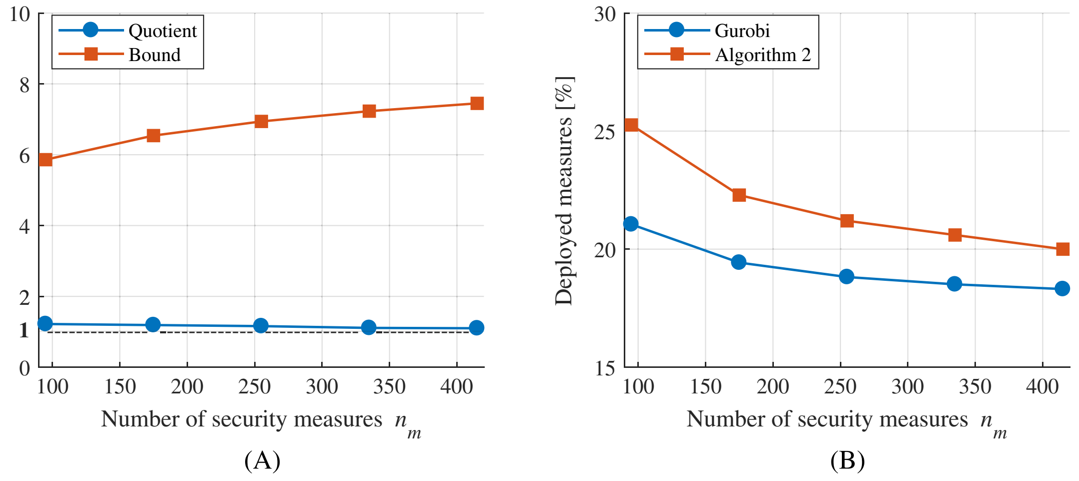

FIGURE 7 In Figure 7A, we plotted the quotient $b(\bar{\mathcal{M}}_G)/b(\bar{\mathcal{M}}_O)$ of the solutions $b(\bar{\mathcal{M}}_G)$ obtained by Algorithm 2 and $b(\bar{\mathcal{M}}_0)$ obtained by the Gurobi solver. The values of the bound introduced in Lemma 3 are also provided. In Figure 7B, we plotted the the largest percentages of deployed security measures obtained in simulations for both the algorithms [Colour figure can be viewed at wileyonlinelibrary.com]

by Algorithm 2 with the solution $b(\bar{\mathcal{M}}_O)$ returned by the Gurobi solver, we recorded the worst-case values of the quotient $b(\bar{\mathcal{M}}_G)/b(\bar{\mathcal{M}}_O)$ in Figure 7A. We also calculated bound from Lemma 3. From Figure 7A, we see that the solution $b(\bar{\mathcal{M}}_G)$ returned by Algorithm 2 was close to the optimal. In particular, it was at most 1.22 times larger than the one returned by the Gurobi solver, which demonstrates that the bound stated in Lemma 3 may be quite conservative.

In Figure 7B, we recorded the maximal percentages of security measures deployed (number of deployed security measures divided by the total number of security measures). The number of deployed security measures varied from 18.31% to 21.05% for the Gurobi solver. For Algorithm 2, the percentage of deployed security measures varied from 20.00% to 25.26%. We also see that the algorithms performed quite similar in this aspect as well. The security measures that were selected by the algorithm were mostly those protecting the major components in the system, for instance, implementing protection within the control center, communication links between master PLCs and the control center, and physical protection of master PLCs. As expected, security measures that were not implemented were mostly those preventing vulnerabilities of slave PLCs and individual sensors/actuators with the high values of π_v.

6 | CONCLUSION

In this paper, we proposed a modeling framework that can be used for allocating security measures in ICSs. The framework is suitable for dynamical models of ICSs, captures the cyber-physical interaction within the system, and allocates security measures based on a risk model. Moreover, the framework targets the case when the number of security vulnerabilities and measures is large. In our framework, the security measure allocation problem was divided into two parts. The first part consisted of conducting risk assessment. For this purpose, we proposed the algorithm that systematically searches for critical attack scenarios (Algorithm 1). In addition, we proved that this algorithm returns the sufficient representation of minimal cardinality (Theorem 1). The second part is to prevent the critical scenarios by deploying the least expensive combination of security measures, which can be done by solving an integer linear program. Given that integer linear programs are NP-hard in general, submodular structure of the problem was outlined (Theorem 2). In that case, a polynomial time greedy algorithm can be used to obtain a solution with guaranteed approximation bound. Additionally, we showed that the sufficient representation of minimal cardinality returned by Algorithm 1 provides the tightest guarantees on performance for the greedy algorithm (Theorem 3).

The applicability of the framework was illustrated through simulations. It was verified that Algorithm 1 allows us to search through a large number of attack scenarios efficiently. We also demonstrated that solving the security measure allocation problem is not necessarily complex. In the experiment, the exact solution was obtained using a standard integer linear program solver in a matter of seconds. Furthermore, it was shown that the greedy algorithm can return a solution close to the optimal and considerably better than the worst-case theoretical bound. The greedy algorithm also proved to be faster than an integer linear program solver, which indicates that we can rely on this method in the case that the use of the solver becomes time consuming.

ACKNOWLEDGEMENTS

This work was supported by The Swedish Civil Contingencies Agency (CERCES project), the Swedish Research Council, The Swedish Foundation for Strategic Research, and Knut och Alice Wallenbergs foundation.

ORCID

Jezdimir Milošević http://orcid.org/0000-0002-2045-5665

REFERENCES

1. Samad T, McLaughlin P, Lu J. System architecture for process automation: review and trends. *J Process Control*. 2007;17(3):191-201.
2. Slay J, Miller M. Lessons learned from the Maroochy water breach. In: *Critical Infrastructure Protection*. Boston, MA: Springer Boston; 2008:73-82.
3. Kushner D. The real story of Stuxnet. *IEEE Spectr*. 2013;50(3):48-53.
4. Analysis of the cyber attack on the Ukrainian power grid. Defense Use Case. Washington, DC: Electricity Information Sharing and Analysis Center; 2016.
5. Stouffer K, Falco J, Scarfone K. Guide to industrial control systems (ICS) security. Gaithersburg, MD: National Institute of Standards and Technology; 2011.
6. Recommended practice: improving industrial control systems cybersecurity with defense-in-depth strategies. Washington, DC: US Department of Homeland Security; 2016.
7. Guide to increased security in industrial information and control systems. Karlstad, Sweden: Swedish Civil Contingencies Agency; 2014.
8. Zheng B, Deng P, Anguluri R, Zhu Q, Pasqualetti F. Cross-layer codesign for secure cyber-physical systems. *IEEE Trans Comput Aided Des Integr Circuits Syst*. 2016;35(5):699-711.
9. Blank RM. Guide for conducting risk assessments. Gaithersburg, MD: National Institute of Standards and Technology; 2011.
10. Amin S, Litrico X, Sastry S, Bayen AM. Cyber security of water SCADA systems – part I: analysis and experimentation of stealthy deception attacks. *IEEE Trans Control Syst Technol*. 2013;21(5):1963-1970.
11. Teixeira A, Dán G, Sandberg H, Johansson KH. A cyber security study of a SCADA energy management system: stealthy deception attacks on the state estimator. *IFAC Proc Vol*. 2011;44(1):11271-11277.
12. Ahmed CM, Ochoa M, Zhou J, et al. NoisePrint: attack detection using sensor and process noise fingerprint in cyber physical systems. In: Proceedings of the 2018 Asian Conference on Computer and Communications Security (ASIACCS); 2018; Incheon, South Korea.
13. Liu Y, Ning P, Reiter MK. False data injection attacks against state estimation in electric power grids. *ACM Trans Inf Syst Secur*. 2011;14(1):13:1-13:33.
14. Bai CZ, Gupta V, Pasqualetti F. On Kalman filtering with compromised sensors: attack stealthiness and performance bounds. *IEEE Trans Autom Control*. 2017;62(12):6641-6648.
15. Pasqualetti F, Dorfler F, Bullo F. Attack detection and identification in cyber-physical systems. *IEEE Trans Autom Control*. 2013;58(11):2715-2729.
16. Ding K, Li Y, Quevedo DE, Dey S, Shi L. A multi-channel transmission schedule for remote state estimation under DoS attacks. *Automatica*. 2017;78:194-201.
17. Pajic M, Lee I, Pappas GJ. Attack-resilient state estimation for noisy dynamical systems. *IEEE Trans Control Netw Syst*. 2017;4(1):82-92.
18. Fawzi H, Tabuada P, Diggavi S. Secure estimation and control for cyber-physical systems under adversarial attacks. *IEEE Trans Autom Control*. 2014;59(6):1454-1467.
19. Boem F, Gallo AJ, Ferrari-Trecate G, Parisini T. A distributed attack detection method for multi-agent systems governed by consensus-based control. In: Proceedings of the 56th IEEE Conference on Decision and Control (CDC); 2017; Melbourne, Australia.
20. Mo Y, Weerakkody S, Sinopoli B. Physical authentication of control systems: designing watermarked control inputs to detect counterfeit sensor outputs. *IEEE Control Syst*. 2015;35(1):93-109.
21. Miao F, Zhu Q. A moving-horizon hybrid stochastic game for secure control of cyber-physical systems. In: Proceedings of the 53rd IEEE Conference on Decision and Control (CDC); 2014; Los Angeles, CA.
22. Feng S, Tesi P. Resilient control under denial-of-service: robust design. *Automatica*. 2017;79:42-51.
23. Yan Y, Antsaklis P, Gupta V. A resilient design for cyber physical systems under attack. In: Proceedings of the American Control Conference (ACC); 2017; Seattle, WA.
24. Persis CD, Tesi P. Input-to-state stabilizing control under denial-of-service. *IEEE Trans Autom Control*. 2015;60(11):2930-2944.
25. Zhao C, He J, Chen J. Resilient consensus with mobile detectors against malicious attacks. *IEEE Trans Signal Inf Process Netw*. 2018;4(1):60-69.
26. Sundaram S, Hadjicostis CN. Distributed function calculation via linear iterative strategies in the presence of malicious agents. *IEEE Trans Autom Control*. 2011;56(7):1495-1508.
27. Teixeira A, Shames I, Sandberg H, Johansson KH. A secure control framework for resource-limited adversaries. *Automatica*. 2015;51:135-148.

28. Smith RS. Covert misappropriation of networked control systems: presenting a feedback structure. *IEEE Control Syst*. 2015;35(1):82-92.
29. Cárdenas AA, Amin S, Sastry S. Research challenges for the security of control systems. In: Proceedings of the 3rd USENIX Workshop on Hot Topics in Security (HotSec); 2008; San Jose, CA.
30. Teixeira A, Paridari K, Sandberg H, Johansson KH. Voltage control for interconnected microgrids under adversarial actions. In: Proceedings of the 2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA); 2015; Luxembourg City, Luxembourg.
31. Guo Z, Shi D, Johansson KH, Shi L. Optimal linear cyber-attack on remote state estimation. *IEEE Trans Control Netw Syst*. 2017;4(1):4-13.
32. Summers T. Actuator placement in networks using optimal control performance metrics. In: Proceedings of the 2016 IEEE 55th Conference on Decision and Control (CDC); 2016; Las Vegas, NV.
33. Clark A, Bushnell L, Poovendran R. A supermodular optimization framework for leader selection under link noise in linear multi-agent systems. *IEEE Trans Autom Control*. 2014;59(2):283-296.
34. Sela Perelman L, Abbas W, Koutsoukos X, Amin S. Sensor placement for fault location identification in water networks. *Automatica*. 2016;172(C):166-176.
35. Bobba R, Rogers K, Wang Q, Khurana H, Nahrstedt K, Overbye T. Detecting false data injection attacks on dc state estimation. In: Proceedings of the Preprints of the First Workshop on Secure Control Systems (CPSWEEK); 2010; Stockholm, Sweden.
36. Kim T, Poor H. Strategic protection against data injection attacks on power grids. *IEEE Trans Smart Grid*. 2011;2(2):326-333.
37. Dán G, Sandberg H. Stealth attacks and protection schemes for state estimators in power systems. In: Proceedings of the 2010 First IEEE International Conference on Smart Grid Communications (SmartGridComm); 2010; Gaithersburg, MD.
38. Vuković O, Sou K, Dán G, Sandberg H. Network-aware mitigation of data integrity attacks on power system state estimation. *IEEE J Sel Areas Commun*. 2012;30(6):1108-1118.
39. Deka D, Baldick R, Vishwanath S. Data attack on strategic buses in the power grid: design and protection. In: Proceedings of the 2014 IEEE PES General Meeting and Conference Exposition; 2014; National Harbor, MD.
40. Bi S, Zhang Y. Graphical methods for defense against false-data injection attacks on power system state estimation. *IEEE Trans Smart Grid*. 2014;5(3):1216-1227.
41. Liu X, Li Z, Li Z. Optimal protection strategy against false data injection attacks in power systems. *IEEE Trans Smart Grid*. 2017;8(4):1802-1810.
42. Deng R, Xiao G, Lu R. Defending against false data injection attacks on power system state estimation. *IEEE Trans Ind Inform*. 2017;13(1):198-207.
43. Cárdenas A, Amin S, Lin Z, Huang Y, Huang C, Sastry S. Attacks against process control systems: risk assessment, detection, and response. In: Proceedings of the 6th ACM Symposium on Information Computer and Communications Security; 2011; Hong Kong, China.
44. Teixeira A, Sou K, Sandberg H, Johansson KH. Secure control systems: a quantitative risk management approach. *IEEE Control Syst*. 2015;35(1):24-45.
45. Milošević J, Tanaka T, Sandberg H, Johansson KH. Analysis and mitigation of bias injection attacks against a Kalman filter. *IFAC Pap*. 2017;50(1):8393-8398.
46. Rymon R. Search through systematic set enumeration. In: Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning (KR); 1992; Cambridge, MA.
47. Wolsey L. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*. 1982;2(4):385-393.
48. Jain N, Koeln J, Sundaram S, Alleyne A. Partially decentralized control of large-scale variable-refrigerant-flow systems in buildings. *J Process Control*. 2014;24(6):798-819.
49. Milošević J, Tanaka T, Sandberg H, Johansson KH. Exploiting submodularity in security measure allocation for industrial control systems. In: Proceedings of the 1st ACM Workshop on the Internet of Safe Things; 2017; Delft, The Netherlands.
50. Kaplan S, Garrick B. On the quantitative definition of risk. *Risk Anal*. 1981;1(1):11-27.
51. Umsonst D, Sandberg H, Cárdenas AA. Security analysis of control system anomaly detectors. In: Proceedings of the 2017 American Control Conference; 2017; Seattle, WA.
52. Milošević J, Umsonst D, Sandberg H, Johansson KH. Quantifying the impact of cyber-attack strategies for control systems equipped with an anomaly detector. In: Proceedings of the European Control Conference; 2018; Limassol, Cyprus.
53. Byres E, Franz M, Miller D. The use of attack trees in assessing vulnerabilities in SCADA systems. In: Proceedings of the International Infrastructure Survivability Workshop (IISW); 2004; Lisbon, Portugal.
54. Permann M, Rohde K. Cyber assessment methods for SCADA security. In: Proceedings of the 15th Annual Joint ISA POWID/EPRI Controls and Instrumentation Conference; 2005; Nashville, TN.
55. Zhou K, Doyle J, Glover K. *Robust and Optimal Control*. Upper Saddle River, NJ: Prentice Hall; 1996.
56. Sandberg H, Teixeira A. From control system security indices to attack identifiability. In: Proceedings of the 2016 Science of Security for Cyber-Physical Systems Workshop (SOSCYPS); 2016; Vienna, Austria.
57. Mo Y, Sinopoli B. On the performance degradation of cyber-physical systems under stealthy integrity attacks. *IEEE Trans Autom Control*. 2016;61(9):2618-2624.
58. Jovanov I, Pajic M. Sporadic data integrity for secure state estimation. In: Proceedings of the 2017 IEEE 56th Conference on Decision and Control (CDC); 2017; Melbourne, Australia.
59. Murguia C, van de Wouw N, Ruths J. Reachable sets of hidden CPS sensor attacks: analysis and synthesis tools. *IFAC Pap*. 2017;50(1):2088-2094. 20th IFAC World Congress.

60. Sommestad T, Ekstedt M, Holm H. The cyber security modeling language: a tool for assessing the vulnerability of enterprise system architectures. *IEEE Syst J*. 2013;7(3):363-373.

61. Bozorgi M, Saul LK, Savage S, Voelker GM. Beyond heuristics: learning to classify vulnerabilities and predict exploits. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD); 2010; Washington, DC.

62. Cormen T, Leiserson C, Rivest R, Stein C. *Introduction to Algorithms*. Cambridge, MA: MIT press; 2009.

63. Bach F. Learning with submodular functions: a convex optimization perspective. *Found Trends Mach Learn*. 2013;6(2-3):145-373.

64. Karp R. Reducibility among combinatorial problems. In: *Complexity of Computer Computations*. Boston, MA: Springer Boston; 1972:85-103.

How to cite this article: Milošević J, Teixeira A, Tanaka T, Johansson KH, Sandberg H. Security measure allocation for industrial control systems: Exploiting systematic search techniques and submodularity. *Int J Robust Nonlinear Control*. 2018;1–25. https://doi.org/10.1002/rnc.4375