Journal of Information Security and Applications 52 (2020) 102471

Contents lists available at ScienceDirect

Journal of Information Security and Applications

journal homepage: www.elsevier.com/locate/jisa

Measuring cyber-physical security in industrial control systems via minimum-effort attack strategies

Martín Barrère[a,*], Chris Hankin[a], Nicolas Nicolaou[b], Demetrios G. Eliades[b], Thomas Parisini[c]

[a] Institute for Security Science and Technology, Imperial College London, UK
[b] KIOS Research and Innovation Centre of Excellence, University of Cyprus, Cyprus
[c] Department of Electrical and Electronic Engineering, Imperial College London, UK

ARTICLE INFO

Keywords:
Security metric
Industrial control systems
Cyber-physical systems
AND/OR graphs
Hypergraphs
MAX-SAT resolution

ABSTRACT

In recent years, Industrial Control Systems (ICS) have become increasingly exposed to a wide range of cyber-physical attacks, having massive destructive consequences. Security metrics are therefore essential to assess and improve their security posture. In this paper, we present a novel ICS security metric based on AND/OR graphs and hypergraphs which is able to efficiently identify the set of critical ICS components and security measures that should be compromised, with minimum cost (effort) for an attacker, in order to disrupt the operation of vital ICS assets. Our tool, META4ICS (pronounced as *metaphorics*), leverages state-of-the-art methods from the field of logical satisfiability optimisation and MAX-SAT techniques in order to achieve efficient computation times. In addition, we present a case study where we have used our system to analyse the security posture of a realistic Water Transport Network (WTN).

1. Introduction

From water and energy plants, to oil, gas, power, manufacturing, and automotive facilities, Industrial Control Systems (ICS) have become an appealing target for attackers over the last years [1–3]. Reasons for that include mostly their increased connectivity to the outside world, their lack of preparedness for cyber attacks, and the huge impact these attacks may have on many aspects of modern society. As a vital part of critical national infrastructure, protecting ICS from cyber threats has become a high priority since their compromise can result in a myriad of different problems, from service disruptions and economical loss, to jeopardising natural ecosystems and putting human lives at risk. Stuxnet, BlackEnergy 3, Industroyer, WannaCry, and later NotPetya, exemplify the devastating consequences this type of attack may have on critical ICS infrastructures [2–6]. In particular, cyber attacks on these systems can lead, for example, to flooding, blackouts, or even nuclear disasters [1].

Although guidance and standard best practices are available to increase ICS security [7], the amount of cyber incidents just keeps increasing [8,9]. This landscape comes as no surprise since ICS environments, originally designed to work in isolation, suddenly became immersed into a hyper-connected world, just a few commands away from malicious actors. We argue that the integration of these complex environments, involving tangled ensembles of dependencies between cyber-physical components, has produced convoluted ecosystems that are hard to control and protect.

As an example, Fig. 1 shows an open benchmark Water Distribution Network (WTN) that resembles a real city *C-Town* [10], and illustrates the scale and structural complexity of these networks (discussed later in the paper). In that context, security metrics play a fundamental role since they allow us to understand the exposure and vulnerability of ICS environments, and improve their security posture [7]. In particular, the ability to identify critical cyber-physical components that should be prioritised and addressed from a security standpoint becomes essential.

AND/OR graphs have proven very useful in this domain as they are able to semantically grasp intricate logical interdependencies among ICS components. However, identifying critical nodes in AND/OR graphs is an NP-complete problem [11–14]. In addition, ICS settings normally involve various cyber and physical security measures that simultaneously protect multiple ICS components in overlapping manners, which makes this problem even harder. In this paper, we are interested in the identification of security-critical cyber-physical components, which are defined as a balance

* Corresponding author.
E-mail addresses: m.barrere@imperial.ac.uk (M. Barrère), c.hankin@imperial.ac.uk (C. Hankin), nicolasn@ucy.ac.cy (N. Nicolaou), eldemet@ucy.ac.cy (D.G. Eliades), t.parisini@imperial.ac.uk (T. Parisini).

https://doi.org/10.1016/j.jisa.2020.102471

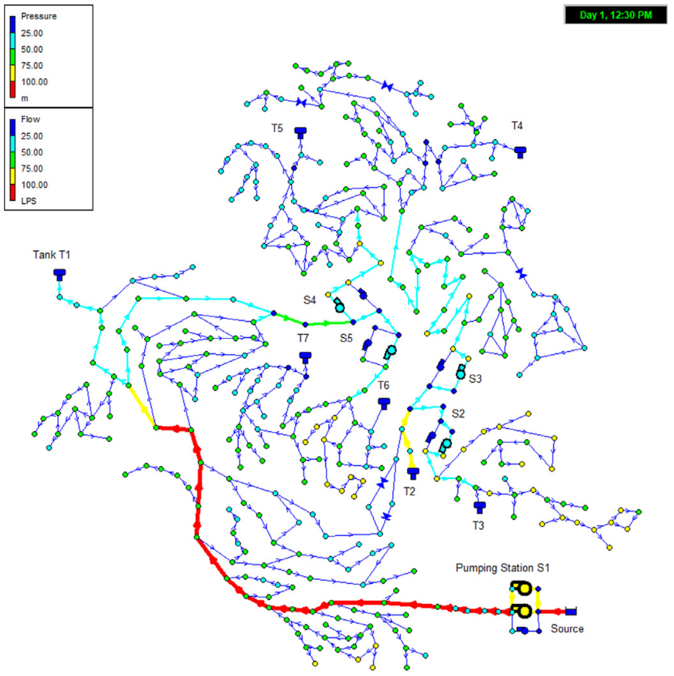


Fig. 1. Large-scale Water Distribution Network (C-Town Benchmark) [10].

between the integral role they have in the operation of the system and the relative security strength used to protect them. More specifically, we define the minimum-effort attack strategy for a given ICS environment as the set of critical nodes and security measures that should be compromised, with minimum cost (effort) for an attacker, in order to disrupt the operation of vital ICS assets. We use this concept as a baseline to measure the security level of ICS systems.

1.1. Scope of the paper

The identification of critical nodes not only allows analysts to define a metric to measure the security level of the system but also provides actionable information that can be used to decide how and where to improve security as well as adding redundant and fallback components. In that context, this paper provides a unified extended framework that builds upon our previous contributions [15,16]. Our approach relies on AND/OR graphs and hypergraphs to model ICS environments with multiple overlapping security measures as well as MAX-SAT techniques to optimally compute critical network nodes. We have incorporated our technique in META4ICS, a Java-based security metric analyser for ICS that we also describe in this paper. In addition, we provide a thorough performance evaluation that shows the feasibility of our method. In particular, our experimental results indicate that the proposed security metric can efficiently scale to networks with thousands of nodes and be computed in seconds. We also illustrate our methodology through a case study in which we analyse the security posture of a realistic Water Transport Network (WTN).

1.2. Contributions

Our main contributions are:

- a flexible and robust mathematical model able to represent complex dependencies in ICS environments protected by multiple overlapping security measures (Section 4),
- a novel security metric and efficient algorithms to identify critical cyber-physical components (Sections 5 and 6),

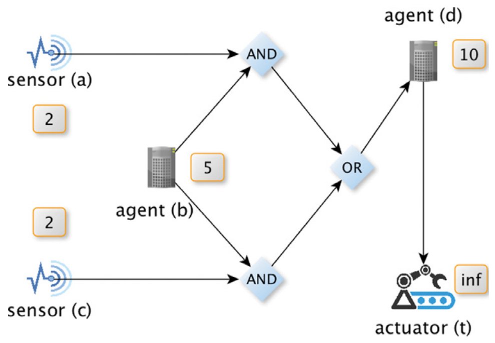


Fig. 2. Base case (weighted AND/OR graph).

- an open source tool called META4ICS to analyse real ICS models (Section 7),
- an extensive experimental evaluation on performance and scalability aspects (Section 8),
- a thorough case study conducted on a realistic water transport network that shows the applicability of our security metric (Section 9).

In the next section, we illustrate the problem and our approach with a simple example.

2. An overview of our approach

2.1. Motivating example

Fig. 2 shows a simple AND/OR graph involving five CPS components in the form of atomic nodes, namely, 2 sensors (a and c), 2 software agents (b and d), and 1 actuator (t). In addition, the graph includes 2 AND nodes and 1 OR node that express the logical dependencies among the CPS components. The graph reads as follows: the actuator t depends on the output of software agent d, e.g. a programmable logic controller (PLC). Agent d in turn has two alternatives to work properly; it can use either the readings of sensor a and the output from agent b together, or the output from agent b and the readings of sensor c together.

Now, let us assume that each CPS component also has an associated value (or weight) that represents its compromise cost (attacker's effort) where *inf* means infinite. Considering these compromise costs, the question we are trying to answer is: which nodes should be compromised in order to disrupt the operation of actuator t, with minimal effort (cost) for the attacker? In other words, what is the least-effort attack strategy to disable actuator t?

This base example involves many attack alternatives, however, only one is minimal. For example, the attacker could compromise node d and thus, the target node t would be successfully disconnected from the graph. However, this strategy has cost 10. Another option would be to compromise node b, with a lower cost of 5. Because node b feeds both AND nodes, these will be disrupted and consecutively the OR node, which in turn will affect node d and finally node t. In terms of costs, however, the optimal strategy for the attacker in this case is to compromise nodes a and c with a total cost of 4. From a defence perspective, we understand this minimal cost as a metric that represents the security level of the system we are trying to protect.

While quite useful in some cases, assigning individual costs on each node can only capture cyber-physical security measures that are applied independently to each ICS component. For example, it can capture that sensors a and c are protected by fenced areas, each one with cost 2, but it cannot model that both sensors are protected by one single fenced area with cost 2. In the latter case, the attacker's effort (cost) required to compromise the secu-

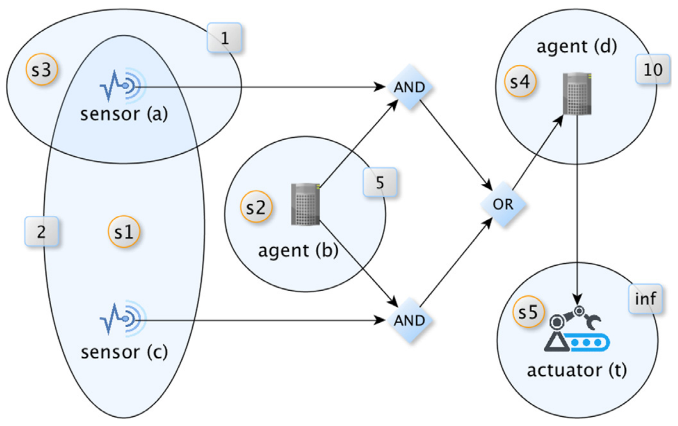


Fig. 3. General case (overlapping weighted measures).

rity mechanism should be considered only once. This leads to a more general case, as illustrated in Fig. 3, where the problem is two-fold: on the one hand, we need to identify the critical components that can disconnect the target from its dependencies in the AND/OR graph, and on the other hand, minimise the attack effort imposed by the security measures that are jointly applied to protect multiple ICS components simultaneously.

In this second scenario, sensors a and c are protected by the same security measure instance $s1$ (e.g. fenced area). Therefore, the cost of bypassing $s1$ to compromise sensor a, sensor c, or both, is 2. However, sensor a is also protected by the security measure $s3$ (e.g. locked container). As a consequence, compromising sensor a would imply to bypass both protective measures $s1$ and $s3$. Therefore, the best strategy in this case is to compromise the security measures $s1$ and $s3$, involving the critical nodes a and c, with a total cost of $2+1=3$. These simple examples are intended to illustrate the problem. As shown later in this paper, however, identifying critical components in larger scenarios with multiple overlapping security controls becomes significantly harder to the naked eye. In that sense, our approach aims at providing useful support for security decision-making, prioritising mitigation plans, and increasing the resilience of ICS environments.

2.2. Overall technical approach

Roughly stated, our approach takes as input an AND/OR graph, in the form of a digraph $G=(V,E)$ that represents the operational dependencies of the ICS environment, and a target node t. Then, we transform the graph into an equivalent logical formula that fulfils node t. Security measures are integrated into the model via AND/OR hypergraphs so we expand this formula by incorporating a logical representation of the security measures applied on each node. The negation of this formula corresponds to the objective of the attacker, i.e. to disable t, which is later converted to an equisatisfiable formula in conjunctive normal form (CNF) by using the Tseitin transformation [17]. Finally, the CNF formula is used as a basis to build a Weighted Partial MAX-SAT problem. The solution to this problem is a minimal weighted vertex cut in G with regards to node t and all of the security measures that protect the ICS components. This cut represents the set of critical components with minimal cost for the attacker such that, if compromised, it will render the cyber-physical system into a non-operational state.

3. Related work

Since the early 2000s, many research efforts have been produced to understand and improve the security of industrial control systems and critical national infrastructure [11,12]. These works have inspired the need for taking into account the cyber-physical dependencies between ICS components and being able to combine them in order to provide quantifiable measurements [1,4,18]. In this paper, we consider the insightful methodology presented in [19], whose objective is to compute the cyber-physical security level of a system based on the number of cyber-physical elements that needs to be compromised in order to disrupt the normal operation of the target system. However, the latter does not consider concrete algorithms to identify critical CPS components in ICS networks nor a model able to capture logical combinations among cyber-physical components with AND/OR connectives. This paper builds upon these ideas and previous contributions [15,16], and provides a complete AND/OR graph-based framework capable of grasping complex interdependencies among CPS components and the security measures used to protect them.

From a graph-theoretical perspective, our approach looks for a minimal weighted vertex cut in AND/OR graphs. This is an NP-complete problem as shown in [11–14]. While well-known algorithms such as Max-flow Min-cut [20,21] and variants of it could be used to estimate such metric over OR graphs in polynomial time, their use for general AND/OR graphs is not evident nor trivial as they may fail to capture the underlying logical semantics of the graph. In that context, we take advantage of state-of-the-art techniques which excel in the domain of logical satisfiability and boolean optimisation problems [22]. Various previous research efforts have addressed the problem of identifying critical nodes in complex networks [23–28]. However, these works are focused either on undirected graphs or directed graphs without AND/OR semantics. In addition, our approach is novel in that we use MAX-SAT-based techniques to identify critical nodes protected by multiple security measures. Other attempts to identify critical components have been made in the domain of network centrality measurements [29]. While useful in many types of scenarios [30], most of them are focused on OR-only graph-based models for IT networks and do not cover AND/OR semantics.

Attack trees and fault tree analysis constitute a major domain in this area [31–35,103]. A fundamental difference between these works and our approach is that strict logical trees are a particular case of AND/OR graphs. Our approach is able to cover tree-based cases as well as general dependency graphs (i.e. internal nodes can have more than one parent). This is an important aspect since diamond-shaped structures, for example, cannot be represented with trees. In addition, the use of tree forests to separately model different parts of the system requires a delicate analysis on components that are shared across multiple trees [35]. The survey presented in [35] provides a comprehensive analysis on current techniques, tools and approaches used on fault tree analysis, including boolean manipulation and binary decision diagrams (BDDs) [36] to find minimal cut sets. Directed acyclic graphs (DAGs), on the other hand, enable richer representations though they cannot have cycles by definition [37]. However, real cyber-physical models might also be cyclic, thus presenting the interdiction problem [38]. As shown in Appendix A, our approach allows AND/OR graphs with cycles between components and are solved using a similar approach to that considered in [39].

Another close research area to our problem includes the domain of attack graphs [40–45]. Attack graphs are mainly focused on depicting the many ways in which an attacker may compromise assets in a computer network. Well-known approaches include logical attack graphs [39,46–50], state-based attack graphs [51,52], hierarchical attack graphs [53,54], conservative attack graphs [55], multiple-prerequisite graphs [56], exploit dependency graphs [57], among others. Mathematically, AND/OR graphs are similar to the structures considered in logical attack graphs [58]. The difference in this work is that we use AND/OR graphs from an operational dependency perspective. In particular, we aim at identifying the set of critical components (vertex cut) whose ensemble of protective measures involves the minimum compromise cost (among all vertex cuts) for an attacker. If such a vertex cut is compromised, it would disrupt the operation of the entire system. Attack graphs, on the other hand, are focused on modelling how multi-stage attacks can be carried out through a network towards the attacker's objective. Therefore, our approach is essentially different as we consider that network nodes can be equally compromised. In addition, attack graphs usually take into account only cyber lateral movements, without considering operational cyber-physical dependencies among components [1]. Bayesian attack graphs also constitute an important research area [59,60]. As opposed to finding potential critical nodes, these works are focused on understanding the likelihood of an attacker to compromise a given asset. SAT-based techniques have also been used in the area of IT network hardening and security management [61–63]. Our work is complementary to attack graphs. However, we plan to incorporate attack graphs into our approach as explained in Section 10, as a means to better estimate risk due to cyber attacks.

Attack graph generation is also a challenging task [44]. Among others, this activity requires a thorough automated analysis of existing vulnerabilities [64], managing unknown weaknesses [65,66], and the involvement of useful tools to systematise the mapping and discovery of IT assets and security aspects [67,68]. Although the automated generation of AND/OR graph-based models for industrial control systems is beyond the scope of this paper, we understand that many lessons can be learnt from attack graph generation in IT networks. Nonetheless, cyber-physical networks pose further generation challenges such as the difficulties to automatically map mechanical (non cyber) devices, or tasks that may be too intrusive (e.g. active probing, scanning) and thus raising concerns about operational disruptions. The latter is an issue that many security platforms already take into account, e.g., in the form of passive monitoring [2,67]. As stated in [2], understanding the full cyber-physical ecosystem is vital to maintain healthy Operational Technology (OT) networks [69]. Other important approaches targeting cyber-physical systems include formal method-based techniques [70–72], security metrics and graph-based methods [73–76], system design analysis [77], structural controllability aspects [78–81], deep learning techniques [82], among others. This paper presents a complementary approach to these related works and aims at measuring the security level of industrial control systems via the identification of least-effort attack strategies that may disrupt the operation of the entire system. In the next section, we introduce our graph-based modelling approach for ICS cyber-physical environments.

4. Cyber-physical network model

4.1. AND/OR dependency graph

We model an industrial network W as a directed AND/OR graph $G = (V, E)$ that represents the operational dependencies in W. The graph involves three types of basic vertices, called atomic nodes (V_{AT}), that model the different classes of components in the network:

S represents the set of sensor nodes, C represents the set of actuator nodes, and A represents the set of software agents.

We define $V_{AT} = S \cup C \cup A$. In addition, the graph also involves two artificial node types that model logical dependencies between network components: Δ represents the set of logical AND nodes, and Θ represents the set of logical OR nodes.

Based on the previous types, we have that $V(G) = V_{AT} \cup \Delta \cup \Theta$. $E(G)$ corresponds to the set of edges among nodes and their semantics depend on the type of nodes they connect. We consider three types of basic edges as follows:

- $E_{A,C} = \{(a, c) : a \in A \wedge c \in C\}$ represents that agent a controls the operation of actuator c,
- $E_{S,A} = \{(s, a) : s \in S \wedge a \in A\}$ means that agent a requires measurements from sensor s to fulfil its purpose,
- $E_{A,A} = \{(a_i, a_j) : a_i, a_j \in A, a_i \neq a_j\}$ represents that agent a_j requires input from agent a_i to operate normally.

In addition, we consider another two types of edges involving artificial AND and OR nodes. These nodes act as special connectors and shall be interpreted from a logical perspective. A node v reached by an OR node means that the operational purpose of v can be satisfied, i.e. v operates normally, if *at least one* of the incoming nodes to the OR node is also satisfied. Alike, a node w reached by an AND node will be satisfied if *all* of the incoming nodes to the AND node are also satisfied. The following two edge types are also allowed in $E(G)$:

- $E_{i\Delta\Theta} = \{(v, x) : v \in V - C, x \in \Delta \cup \Theta\}$ represents incoming connections to AND/OR nodes from any type of graph node except actuators (C),
- $E_{o\Delta\Theta} = \{(x, v) : x \in \Delta \cup \Theta, v \in V - S\}$ represents outgoing connections from AND/OR nodes to any type of graph node except sensors (S).

Graph properties. We use $\mathcal{G}$ to denote the domain of AND/OR graphs. Let $d_{in} : V \to \mathbb{N}$ and $d_{out} : V \to \mathbb{N}$ be two functions that compute the in-degree and out-degree of a node respectively. The following properties hold in every instance of $G = (V, E)$, with $G \in \mathcal{G}$:

- $d_{out}(c) = 0, \forall c \in C$ (no outgoing edges from actuators)
- $d_{in}(v) = 1, \forall v \in V_{AT}$ (only one incoming edge on atomic nodes)
- $d_{out}(v) \geq 1, \forall v \in \Theta \cup \Delta$ (logical nodes are linked to some destination node)
- $d_{in}(v) \geq 2, \forall v \in \Theta \cup \Delta$ (logical nodes combine two or more nodes)

4.2. Adversarial model and assumptions

Our adversarial model considers that an attacker can compromise any network node $n \in V_{AT}$ at a certain cost $\varphi(n)$, with $\varphi : V_{AT} \to \mathbb{R}_{\geq 0}$. A *compromised node* in this context shall be understood as a CPS component unable to operate properly, that is, a node incapable of fulfilling the purpose it was designed for. The cost function $\varphi(n)$ is intended to provide a means to quantitatively express the efforts required by an attacker to individually compromise a given node n. We realise that the instantiation of this cost function may be difficult in some cases. However, there have been many remarkable advances in this direction over the last years. For example, the *Common Vulnerability Scoring System (CVSS)* is a standard security effort that provides means to quantify software vulnerabilities in the form of a numerical score [83]. This score reflects the severity of a vulnerability and considers aspects such as complexity, exploitability, impact, among others.

Since CPS components (graph nodes) logically depend on others to work properly, our adversarial model considers that the compromise of a node will also affect the operation of the nodes that depend on it. Therefore, such impact is passed on to other nodes following a logic-style propagation. Algorithm 1 describes this process from a node removal standpoint in the form of a function $\sigma(G, X)$.

The function $\sigma(G, X)$ takes as input an AND/OR graph G and a set of nodes to remove X, and returns G after deleting the nodes in X as well as the nodes that logically depend on them, and every edge related to the removed nodes. Essentially, for each node n that must be removed, Algorithm 1 analyses the nodes that depend on n (set M, line 3). A node x that depends on n will be affected

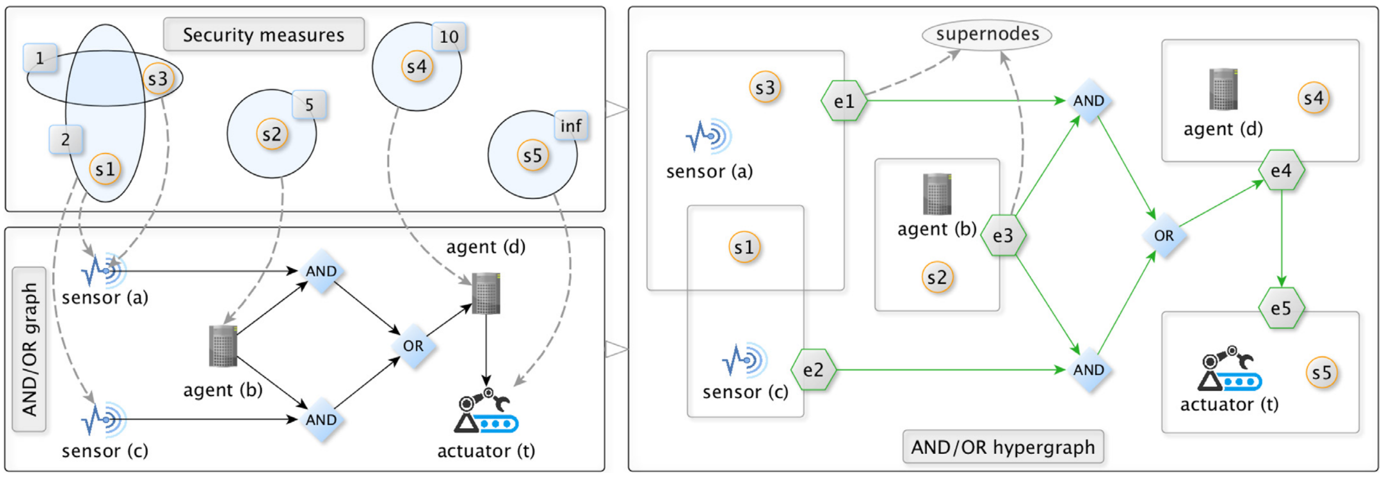


Fig. 4. AND/OR hypergraph (general case example).

Algorithm 1: Logical node removal $\sigma(G, X)$

```
Name : $\sigma(G, X)$
Input: Graph $G = (V, E)$, Nodes to remove $X$
Output: Updated graph $G = (V, E)$
1  while $X$ is not empty do
2      Node $n \leftarrow X.pop()$                         // get first node
3      $M \leftarrow \{x \in V : (n, x) \in E\}$          // nodes reached by $n$
4      for $x \in M$ do
5          if $(x \in V_{AT})$ or $(x \in \Delta)$ or $(x \in \Theta \wedge d_{in}(x) = 1)$ then
6              $X.append(x)$                              // $x$ must be removed
7          end
8      end
9      $V = V - \{n\}$                                    // remove $n$ from $G$
10     $E = E - \{(v, w) \in E : v = n \vee w = n\}$      // remove edges
11 end
12 return $G = (V, E)$
```

only if x is an atomic node, an AND node, or an OR node with only one input left (line 5). In any of these cases, node x is queued for removal (line 6). In other words, only an OR node with more than one input will remain in the graph when one of its inputs is removed. Finally, each node marked for removal is deleted from G along with its respective edges (lines 9-10). We use $\sigma(G, X)$ in the definition of our security metric (Section 5) to express the impact of compromised nodes as well as the resulting graph when such nodes are removed from the network.

4.3. Defence model and security measures

From a defence perspective, we model security measures as an additional layer (or overlay) that is placed on top of the AND/OR dependency graph, as illustrated before in Fig. 3 (and also in Fig. 4, described later in Section 6.2). In particular, each ICS component is protected by one or more security measure instances s_j of type M_i. Table 1 exemplifies some typical security measures.

Given a specific scenario, we define $S = \{s1, s2, \ldots\}$ as the set of involved security measure instances. In addition, each measure type M_i involves a cost for the attacker that quantifies the effort that he or she has to make in order to bypass the measure. We model this aspect for measure instances as a function $\psi : S \rightarrow \mathbb{R}_{\geq 0}$. We call *protection range* to the set of ICS components protected by a single measure instance s_j. Considering the initial example illustrated in Fig. 3, the security measures are as shown in Table 2.

Table 1
Examples of protection measures.

Measure	Cost (attacker)	Description
$M1$	1	Sound alarm
$M2$	2	Fenced area
$M3$	5	Locked container
$M4$	10	Tamper-resistant container
$M5$	inf	Alarmed locked building

Table 2
Security measures for general example in Fig. 3.

Measure instance	s1	s2	s3	s4	s5
Measure type	$M2$	$M3$	$M1$	$M4$	$M5$
Attacker's cost $\psi(s_j)$	2	5	1	10	*inf*
Protection range	$\{a, c\}$	$\{b\}$	$\{a\}$	$\{d\}$	$\{t\}$

We also define the function m: $V_{AT} \rightarrow 2^S$ which, given a node n, returns the set of measure instances that protect n. From a graph-theoretical standpoint, we use an AND/OR hypergraph-based approach to link each node in the AND/OR dependency graph with the set of security measures instances that are used to protect it. This concept is later formalised in Section 6.2.

5. Security metric

5.1. Problem definition

Let W be an industrial network, $G = (V, E)$ a directed AND/OR graph representing the operational dependencies in W, and t a target network node. The objective of our security metric, $\mu(G, t)$, is to identify the set of nodes $X = \{x_1, \ldots, x_h\}$ protected by measures $S(X) = \{s_1, \ldots s_k\} \subseteq S$ that must be compromised in order to disrupt the normal operation of target node t, with minimal cost for the attacker. More formally, we define $S(X) = \bigcup_{x_i \in X} m(x_i)$, and $\mu : \mathcal{G} \times V \rightarrow 2^V$ as:

$$\mu(G, t) = \underset{X \subseteq V_{AT}}{\operatorname{argmin}} \left(\sum_{x_i \in X} \varphi(x_i) + \sum_{s_j \in S(X)} \psi(s_j) \right) \quad \text{s.t.} \quad wcc(\sigma(G, X)) \geq 2 \vee X = \{t\} \tag{1}$$

where the solution with minimal cost must be either node t or a set of nodes X such that, if removed (with function σ), t gets disconnected from the graph. As explained before, function $\sigma(G, X)$

removes from G each node $x \in X$ and the nodes that depend on them following a logic-style propagation. The result is then analysed with function $wcc(G)$, which computes the number of weakly connected components in G. If G contains two or more components, it means that the target node t gets disconnected from a non-empty set of nodes on which t depends (directly or indirectly) to function properly. Note that $S(X)$ returns the *set* of security measure instances used to protect the nodes in X. Therefore, measure instances that protect more than one node in X appear only once, and thus their costs are considered only once in the second summation of Eq. (1). Note also that $\varphi(n)$ can be neutral (e.g. $\varphi(n) = 0, \forall n \in V_{AT}$) to only consider the costs of the security measures, or it can be instantiated with cyber costs, e.g. CVSS scores [83].

5.2. Overall resolution process

We address our problem from a logical perspective, and more precisely, from a satisfiability point of view. The resolution process of the metric involves three main steps, which are fully detailed in the next section. In particular, the first step (Section 6.1) involves the transformation of the input AND/OR graph into a logical formulation that represents the logical dependencies on which the target t relies on. The second step (Section 6.2) expands this formulation to capture the security measure instances that protect each node in the CPS system. To do so, we use an AND/OR hypergraph-based model that allows us to grasp how security measures and CPS components are interrelated. Finally, the third step (Section 6.3) involves the assignment of compromise costs (weights) to the variables within the expanded logical formulation. The resulting weighted formula is then used to solve a Weighted Partial MAX-SAT problem, whose solution will indicate the set of critical nodes and security measures that should be compromised, with minimal cost for an attacker, in order to disrupt the operation of the CPS system. As shown in Section 8, current SAT solvers are able to handle this family of problems at a very decent large scale (dozens of thousands of variables), and they normally involve state-of-the-art techniques to tackle satisfiability problems, pseudo-boolean problems and optimisation procedures [84].

6. Computation strategy

6.1. Logical AND/OR dependency graph transformation

Given a target node t, the input graph G can be used as a map to decode the dependencies that node t relies on. Since these dependencies are presented as a logical combination of components connected with AND and OR operators, we say that node t is fulfilled (or can operate normally) if the logical combination is satisfied. In turn, these dependencies may also have previous dependencies, and therefore, they must be also satisfied. In that sense, G can be traversed backwards in order to produce a propositional formula that represents the different ways in which node t can be fulfilled. We call this transformation $f_G(t)$, whose algorithm is fully described in Appendix A. To illustrate this idea, let us consider our base example (Fig. 2). In this case, $f_G(t)$ returns the following formula:

$$f_G(t) = t \wedge (d \wedge ((a \wedge b) \vee (b \wedge c)))$$

The goal of the attacker, however, is precisely the opposite, i.e., to disrupt node t somewhere along the graph. Therefore, we are actually interested in satisfying $\neg f_G(t)$, which describes the means to disable t, as follows:

$$\neg f_G(t) = \neg(t \wedge (d \wedge ((a \wedge b) \vee (b \wedge c))))$$

Under that perspective, a logical assignment such that $\neg f_G(t) = true$ will indicate which nodes must be compromised (i.e. logically falsified) in order to disrupt the operation of the system. Finding such an assignment constitutes a Satisfiability (SAT) problem [85]. A SAT problem essentially looks for an assignment of truth values to the variables of a logical formula such that the formula evaluates to *true*. Our general problem, however, involves security measures that are used to protect these nodes. Therefore, compromising a node also implies to compromise the security measures protecting it, which in turn can be shared with other nodes, as exemplified in Fig. 3. To address this aspect, we consider a generalised model that cover both the base and general problem using AND/OR hypergraphs, as explained in the next section.

Table 3
Hypergraph H (general case example).

e_1	e_2	e_3	e_4	e_5
$\{a, s1, s3\}$	$\{c, s1\}$	$\{b, s2\}$	$\{d, s4\}$	$\{t, s5\}$

6.2. Integration of overlapping security measures through AND/OR hypergraphs

Hypergraphs are a generalisation of standard graphs where graph edges, called hyperedges, can connect any number of vertices [86,87]. More formally, let X be a set of vertices $X = \{x_1, x_2, \ldots, x_n\}$. A hypergraph on X, denoted $H = (X, E)$, is a family of subsets of X, with $E = \{e_1, e_2, \ldots, e_m\}$, such that: (1) there are no empty edges in H, i.e. $e_i \neq \emptyset$, $\forall e_i \in E$; and (2) X is covered by E, i.e. $\bigcup_{i=1}^{m} e_i = X$.

In this work, we propose the use of a hybrid type of hypergraph, called AND/OR hypergraph, which essentially combines properties of hypergraphs and the logical structure of AND/OR graphs. Roughly stated, the nodes of an AND/OR hypergraph are the hyperedges of a standard hypergraph, and these are linked using logical AND/OR nodes as done in classical AND/OR graphs.

We use standard hypergraphs to model groups of security measures that are applied to each ICS component in the network. For example, let us consider our initial scenario illustrated in Fig. 3. In this case, the hypergraph is defined as $H = (X, E)$ where $X = V_{AT} \cup S$ is the set of nodes of the hypergraph, and $E = \{e_1, e_2, e_3, e_4, e_5\}$ is the set of hyperedges. Table 3 details the members of each hyperedge $e_i \in E$.

Hyperedges combine each network node with the instances of the security measures that are used to protect them. The advantage of using hypergraphs is that we can capture multiple overlapping security measures in the hyperedges of the hypergraph. In addition, we can easily model protection ranges, that is, how a specific measure instance, e.g. a fenced area, protects multiple ICS components simultaneously, e.g. $s1 \mapsto \{a, c\}$. At a semantic level, the interpretation of a hyperedge e_i is that the original node n is accompanied by the security measures that protect it, and therefore, node n can only be disrupted if every security measure in e_i is compromised too. Now hyperedges can be understood as super nodes that represent each original node and their protective measures. Therefore, we can follow the same logical structure as in the original graph and combine these super nodes via AND/OR connectives as illustrated in Fig. 4.

From a logical perspective, we use a function $f_H(t)$ to map the dependency model of the AND/OR hypergraph in the same way we did before with $f_G(t)$ over AND/OR graphs:

$$f_H(e5) = e5 \wedge e4 \wedge ((e1 \wedge e3) \vee (e3 \wedge e2))$$

As explained before, the objective of the attacker is to falsify the previous formula (or satisfy $\neg f_H(e5)$) in order to make the target $e5$ non-functional. Since each hyperedge e_i involves many security measures plus the original node n, the only way to falsify e_i is

Table 4
Hard clauses.

$\neg t \vee \neg d \vee \neg a \vee \neg b$
$\neg t \vee \neg d \vee \neg b \vee \neg c$

to falsify every member in it. Therefore, we logically capture this aspect by replacing each hyperedge e_i by a disjunctive construct $(n \vee s_i \vee \ldots \vee s_j)$, where $s_i \vee \ldots \vee s_j$ is the disjunction of measure instances that protect node n. We call this transformation $h_G(t)$, which in our example is as follows:

$$h_G(t) = (t \vee s5) \wedge (d \vee s4) \\ \wedge (((a \vee s1 \vee s3) \wedge (b \vee s2)) \vee ((b \vee s2) \wedge (c \vee s1)))$$

Therefore, given a node n, such a disjunctive construct actually forces a SAT solver to make *false* every security measure protecting n, which essentially equals to the fact that the attacker must compromise all of the measures to take control of the ICS component n.

6.3. Weighted partial MAX-SAT resolution approach

6.3.1. CNF conversion

Normally, SAT formulations consider the input formula in conjunctive normal form (CNF). Converting an arbitrary boolean formula to CNF can be naively tackled by using De Morgan and distributive laws. For example, the CNF formulation of $\neg f_G(t)$ in our base example (Fig. 2) is as follows:

$$CNF(\neg f_G(t)) = (\neg t \vee \neg d \vee \neg a \vee \neg b) \wedge (\neg t \vee \neg d \vee \neg b \vee \neg c)$$

However, such an approach might lead to exponential computation times over large graphs, thus only being able to scale up to a few hundred nodes. To avoid this issue, we use the Tseitin transformation [17], which essentially produces a new formula in CNF that is not strictly equivalent to the original formula (because it introduces new variables) but is equisatisfiable. This means that, given an assignment of truth values, the new formula is satisfied if and only if the original formula is also satisfied. An example of how the Tseitin transformation works can be found in Appendix B. Since the Tseitin transformation adds new variables during the process, the new formula is larger in size than the original one (we omit the transformed formula for our base example since it has 15 variables and 27 clauses). However, the Tseitin transformation can be done in polynomial time, as opposed to the naive CNF conversion approach that can ramp up to exponential computation times in the worst case.

6.3.2. Satisfiability formulation

When a CNF formula also involves weights on each clause, the problem is called MAX-SAT [22]. A MAX-SAT problem consists in finding a truth assignment that maximises the weight of the satisfied clauses. Equivalently, MAX-SAT minimises the weight of the clauses it falsifies [22]. When a set of clauses must be forcibly satisfied (called *hard clauses*), the problem is denominated Partial MAX-SAT and it works on a subset of clauses (denominated *soft clauses*) that can be falsified if necessary. If the *soft clauses* have non-unit weights, the problem is called Weighted Partial MAX-SAT and it will try to minimise the penalty induced by falsified weighted variables. We use the latter to address our problem.

Let us now exemplify this process using our base example involving weighted AND/OR graphs (Fig. 2) and the general case with AND/OR hypergraphs (Fig. 3 and Table 3).

◇ *Base case (weighted AND/OR graph).* In this example, the hard clauses are those involved in the CNF formula as shown in Table 4.

Soft clauses correspond to each atomic node in the graph with their corresponding penalties (costs) as shown in Table 5.

Table 5
Individual compromise costs.

a	b	c	d	t
$\varphi(a) = 2$	$\varphi(b) = 5$	$\varphi(c) = 2$	$\varphi(d) = 10$	$\varphi(t) = inf$

Table 6
Falsification penalty scores for security measures.

Measure instance	$s1$	$s2$	$s3$	$s4$	$s5$
Attacker's cost $\psi(s_i)$	2	5	1	10	*inf*

Therefore, a MAX-SAT solver will try to minimise the number of falsified variables as well as their weights (costs), which in our problem equals to minimise the compromise cost for the attacker. As mentioned before, the solution in this case is to compromise nodes a and c with a total cost of 4.

◇ *General case (AND/OR hypergraphs).* Our general model with AND/OR hypergraphs not only involves individual costs on atomic nodes (function φ) but also compromise costs on each security measure instance (function ψ). Table 6 shows the attacker's costs for each measure instance within our general example that are used as the falsification penalty scores.

If we consider, for example, a neutral cost on each atomic node n, e.g. $\varphi(n) = 1, \forall n \in V_{AT}$, the solution of the Weighted Partial MAX-SAT problem to disrupt the system $h_G(t)$:

$$h_G(t) = (t \vee s5) \wedge (d \vee s4) \\ \wedge (((a \vee s1 \vee s3) \wedge (b \vee s2)) \vee ((b \vee s2) \wedge (c \vee s1)))$$

is composed of nodes a and c, and security controls $s1$ and $s3$, with a total cost of 5. Informally speaking, we are trying to find a portion of $h_G(t)$ that can be falsified (so $\neg h_G(t)$ is *true*) with minimal cost. We can observe that if the last big clause of $h_G(t)$ is falsified, then $h_G(t)$ is falsified. We can choose to falsify the whole disjunction by making, for example, the sub-sentence $(b \vee s2)$ *false*. However, the penalty here is $1 + 5 = 6$. If $(a \vee s1 \vee s3)$ and $(c \vee s1)$ are falsified instead, the cost corresponds to the penalty paid for the set $\{a, s1, s3, c\}$ with a total cost of $1 + 2 + 1 + 1 = 5$. The other two options, $(t \vee s5)$ and $(d \vee s4)$, have costs infinite and 11 respectively, so the final solution involves the critical node set $\{a, c\}$ with a total cost of 5. As a final remark, note that if we want to only consider the costs of security measures, we can define $\varphi(n) = 0, \forall n \in V_{AT}$, in which case our example would yield 3 as the total attack cost.

6.3.3. Method summary

Given an input AND/OR graph $G = (V, E)$ and a target node t, we model our security metric $\mu(G, t)$ as a Weighted Partial MAX-SAT problem where the weights are provided by the cost functions $\varphi(v)$ for each node $v \in V_{AT}$ and $\psi(s)$ for each security measure $s \in S$. Our method includes the following steps:

1. We first transform the dependency graph G into an equivalent logical representation, $f_G(t)$, as described in Section 6.1.
2. $f_G(t)$ is transformed into a new formula $h_G(t)$ as described in Section 6.2, where each atomic node $n \in V_{AT}$ in $f_G(t)$ is replaced with $(n \vee s_i \vee \ldots \vee s_j)$, where $s_i \vee \ldots \vee s_j$ is the disjunction of security controls that protect node n.
3. The objective of the attacker, $\neg h_G(t)$, is then converted to CNF using the Tseitin transformation, i.e., $\widehat{g}(t) = CNF(\neg h_G(t))$, where $\widehat{g}(t) = (v_{1i} \vee \ldots \vee v_{1j}) \wedge \ldots \wedge (v_{hi} \vee \ldots \vee v_{hj})$.
4. Finally, we define a soft clause for each atomic node $v \in V_{AT}$ and each security measure $s \in S$, and assign their corresponding weights as $(v, \varphi(v))$ and $(s, \psi(s))$ respectively.

The last step tells the MAX-SAT solver that each soft clause (each node v of the graph and each security measure s) can be

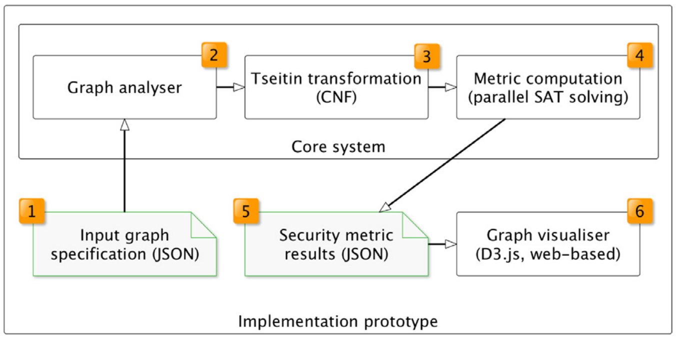

Graph analyser
2
Tseitin transformation (CNF)
3
Metric computation (parallel SAT solving)
4
Core system
1
Input graph specification (JSON)
5
Security metric results (JSON)
Graph visualiser (D3.js, web-based)
6
Implementation prototype

Fig. 5. META4ICS overall architecture.

falsified with a certain penalty $\varphi(v)$ or $\psi(s)$, which is the cost required for the attacker to compromise v or a security control s. Since the solver tries to minimise the total weight of falsified variables, a solution to this problem yields a minimum vertex cut in the graph that models our CPS system. Therefore, the overall process provides a *set of critical components and security measures with minimal cost for the attacker such that, if compromised, it would render the cyber-physical system into a non-operational state*. In practice, steps 1 and 2 are implemented simultaneously; here we present them separately for the sake of clarity.

7. META4ICS – implementation prototype

In this section, we describe our implementation prototype called META4ICS (Metric Analyser for Industrial Control Systems), pronounced as *metaphorics*. The tool is open source and publicly available at [88]. The overall architecture of META4ICS is illustrated in Fig. 5, and it involves two main modules:

(i) a core system in charge of analysing the input graph and computing the security metric, and (ii)a Web-based visualisation system that displays the graph as well as the critical nodes indicated by the security metric.

The core system has been purely developed in Java and it can be executed from the command line as a single runnable JAR. This is fully documented online [88]. Initially (step 1), the tool consumes an input graph represented in JSON (JavaScript Object Notation). Listing 1 depicts the JSON specification file for our general scenario (Figs. 3 and 4), and a neutral cost assignment on network nodes ($\varphi(n) = 1, \forall n \in V_{AT}$).

The graph encoding essentially involves a list of nodes, the edges among them, the security measures used to protect the ICS components, and the target node under analysis. If the input graph has more than one node with no incoming edges, the graph analyser creates an artificial source node, denoted as *s*, which is linked to each one of these nodes in order to produce a single-source graph (step 2). This is done to simplify graph-processing algorithms. Within our example, nodes *a*, *b*, and *c* have no incoming edges, and therefore, the following edges are also added: (*s*, *a*), (*s*, *b*), and (*s*, *c*). This can be observed in Fig. 6 (described later).

Afterwards, our prototype translates the AND/OR graph and the involved security measures into a logical representation that is then converted to an equisatisfiable CNF formula using the Tseitin transformation (step 3) [17]. The Weighted Partial MAX-SAT problem is then built and solved (step 4) using the appropriate clauses and costs, as described in Section 6.3.2. For the MAX-SAT resolution process, we have experimentally observed that different SAT solvers may behave quite differently with the same AND/OR graphs due to the use of distinct techniques. To address this issue, META4ICS executes multiple pre-configured solvers in parallel and picks up the solution of the solver that finishes first. This method provides a more stable behaviour in terms of performance

```
{"graph": {
  "target":"t",
  "nodes": [
    { "id":"t", "type":"actuator", "value":"1",
        "measures": [{"id":"s5", "type":"M5"}] },
    { "id":"d", "type":"agent", "value":"1",
        "measures": [{"id":"s4", "type":"M4"}] },
    { "id":"or-d", "type":"or", "value":"none" },
    { "id":"c", "type":"sensor", "value":"1",
        "measures": [{"id":"s1", "type":"M2"}] },
    { "id":"b", "type":"agent", "value":"1",
        "measures": [{"id":"s2", "type":"M3"}] },
    { "id":"a", "type":"sensor", "value":"1",
        "measures": [{"id":"s1", "type":"M2"},
           {"id":"s3", "type":"M1"}] },
    { "id":"a-b", "type":"and", "value":"none" },
    { "id":"b-c", "type":"and", "value":"none" } ],
  "edges": [
    { "source":"d", "target":"t" },
    { "source":"or-d", "target":"d" },
    { "source":"a-b", "target":"or-d" },
    { "source":"b-c", "target":"or-d" },
    { "source":"a", "target":"a-b" },
    { "source":"b", "target":"a-b" },
    { "source":"b", "target":"b-c" },
    { "source":"c", "target":"b-c" } ],
  "measures": [
    { "id":"M1", "cost":"1", "desc":"Sound alarm" },
    { "id":"M2", "cost":"2", "desc":"Fenced area" },
    { "id":"M3", "cost":"5", "desc":"Locked container" },
    { "id":"M4", "cost":"10", "desc":"Tamper-resistant container" },
    { "id":"M5", "cost":"inf", "desc":"Alarmed locked building"} ]
}}
```

Listing. 1. AND/OR graph JSON specification.

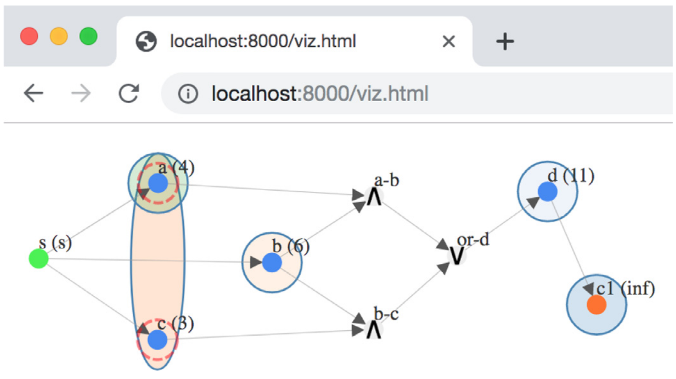


Fig. 6. META4ICS viewer – metric resolution.

and scalability as discussed later in Section 8. Currently, the tool uses two different MAX-SAT solvers, namely SAT4J [84,89] and a Python-based linear programming approach using Gurobi [90]. In the case of ties (even cost for two or more solutions), the tool selects the solution with the minimum number of nodes. When executed, META4ICS displays the least-effort attack strategy in the command line, which in our example is composed of critical nodes *a* and *c*, each with cost 1, and the security measure instances *s*1 and *s*3 with costs 2 and 1 respectively, totalling a minimal cost of 5. The tool also outputs a JSON file that includes the original graph and the minimum weighted vertex cut identified as the solution (step 5). This output is then used to feed the visualisation component (step 6). The latter is an interactive graph visualiser, built on top of the D3.js technology [91], whose objective is to provide visual means to understand dependencies among nodes and manipulate critical nodes. Fig. 6 shows the metric resolution for our example scenario where each node includes its total aggregated cost. The tool displays critical nodes surrounded by dashed red circles and allows the user to validate the solution by interactively removing them until the target is disabled.

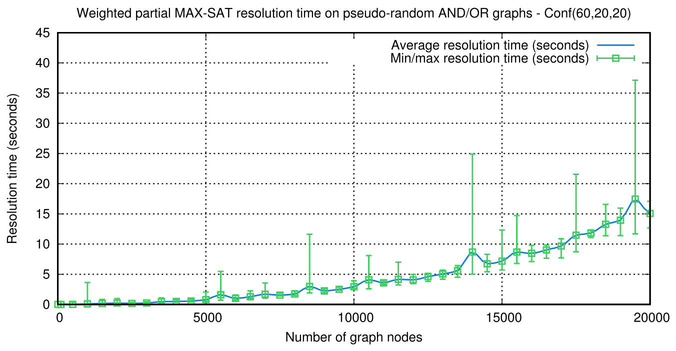


Fig. 7. Scalability evaluation up to 20000 nodes.

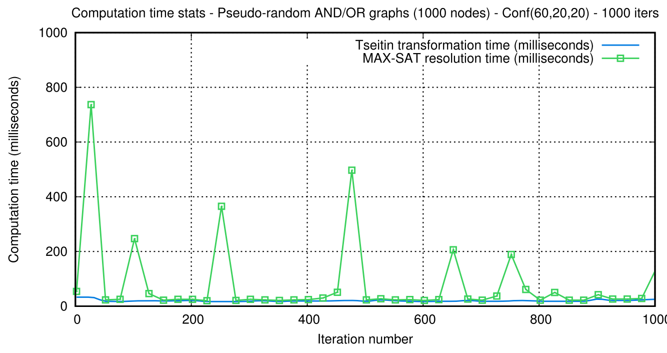


Fig. 8. Performance evaluation (1000 nodes x 1000 iters.).

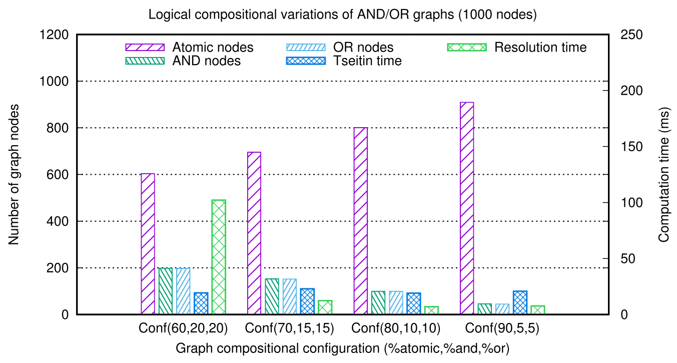


Fig. 9. Analysis of different logical compositions (1000n).

8. Performance evaluation

In order to evaluate the feasibility of our approach, we have conducted an extensive set of experiments based on synthetic pseudo-random AND/OR graphs of different size and composition. These experiments have been performed using a MacBook Pro (15-inch, 2018), 2.9 GHz Intel Core i9, 32 GB 2400 MHz DDR4. The construction procedure for an AND/OR graph of size n is as follows. We first create the target node. Afterwards, we create a predecessor which has one of the three types (atomic, AND, OR) according to a probability given by a compositional configuration predefined for the experiment. For example, a configuration of (60,20,20) means 60% of atomic nodes, 20% of AND nodes and 20% of OR nodes. We repeat this process creating children on the respective nodes until we approximate the desired size of the graph, n.

In this section, we first explore the behaviour of our method over simple weighted AND/OR graphs. Afterwards, we describe the obtained results using independent security measures over AND/OR hypergraphs. Finally, we study the use of various security measures applied to multiple nodes simultaneously and the impact this overlapping poses in terms of computation time.

8.1. Experimental analysis over weighted AND/OR graphs

Fig. 7 shows the behaviour of our methodology over weighted AND/OR graphs (graphs with one single cost per node, e.g. Fig. 2), when the size of the input graph increases.

In this experiment, we produce pseudo-random AND/OR graphs of size n and a compositional configuration of (60,20,20). The size n varies as $n \in [0, 500, 1000, 1500, \ldots, 20000]$, and we iterate the evaluation process 10 times for each value of n. The solid line shows the average values obtained for graphs of size n while the vertical bars indicate shortest and largest computation times for each value of n. As we explore later in this section, the structure of the logical formulation varies according to the structure of the input graph (e.g. more ANDs than ORs), and therefore, the time required by the CNF converter and SAT solver might vary as well, which explains the vertical bars. In general, however, we have observed very good results in terms of performance and scalability. For example, for graphs with 10000 nodes, the average resolution time is about 3 seconds, while for graphs with 20,000 nodes, the average time is around 15 seconds. Note that scalability here is understood from a computational standpoint rather than a control systems perspective.

In order to analyse the variability observed in computation times due to the structure of the formulas, we have taken a closer look at the two main processes that govern the overall behaviour of the strategy: the Tseitin transformation and the MAX-SAT resolution. Fig. 8 shows the results of a 1000-iteration experiment using weighted AND/OR graphs with 1000 nodes and a (60,20,20) configuration.

In general, we have observed that while the Tseitin transformation time is stable across all iterations, the MAX-SAT resolution process requires more time to solve the problem in some graphs than others. This happens because some graphs induce formulas with longer sequences of AND or OR operators connecting with different combinations of graph nodes, which incurs in variable computation times.

We have also observed that the number of clauses and variables within the transformed Tseitin formulas involve similar patterns for this and other configurations. For example, for graphs with 1000 nodes and the (60,20,20) configuration, the Tseitin formulas involve in average 1500 clauses and 3000 variables. This means that each clause generally involves the disjunction of two or three variables. In order to better understand how the complexity of the formulas may impact the overall strategy, we have also experimented with different composition configurations for graphs with 1000 nodes. Fig. 9 shows the obtained results on four different configurations including the previous (60,20,20) distribution.

While the Tseitin transformation shows almost a constant behaviour, we can observe a dramatic reduction in the average MAX-SAT resolution time as the number of AND/OR nodes decreases. This phenomenon occurs because the graphs now involve more dependent nodes in sequence with less AND/OR nodes among them. In addition, OR nodes have a higher impact in the resolution time since any fulfilled input may enable this connector, while AND nodes only require one disconnected input to be disabled. In order to confirm these observations, we have conducted the same scalability experiment up to 20000 nodes, but now with a different configuration using a (80,10,10) distribution. The results are shown in Fig. 10. As expected, we can observe that the variability of the experiments (vertical green bars) is now much lower since the structure of the formulas pose less restrictions to find the optimal solutions.

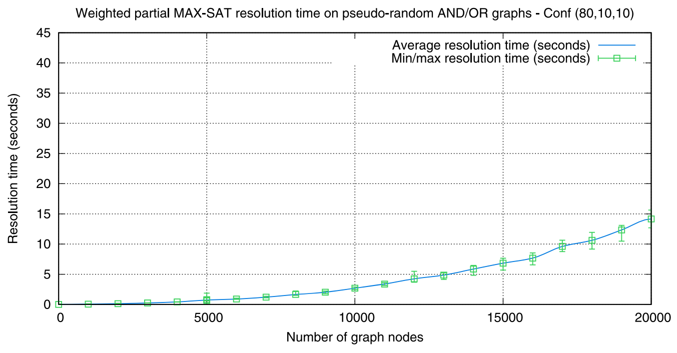


Fig. 10. Scalability evaluation – Conf(80,10,10).

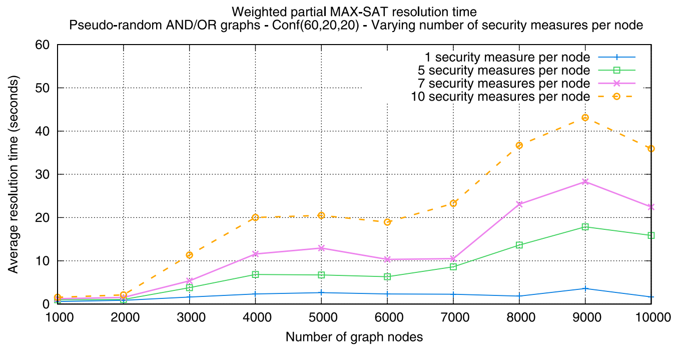


Fig. 11. Scalability while increasing graph size.

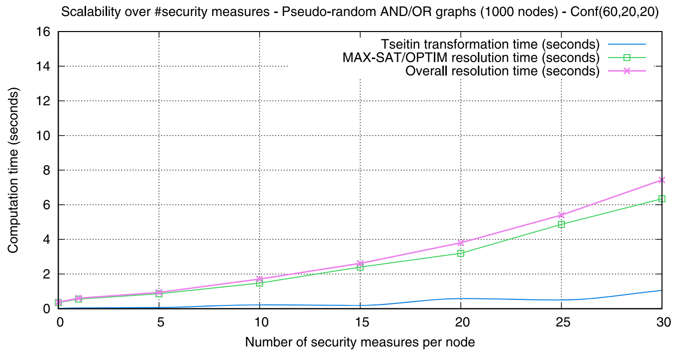


Fig. 12. Performance while increasing measures.

8.2. AND/OR hypergraphs with disjoint security measures

This section studies the scalability and performance of our general approach based on AND/OR hypergraphs when we increase the number of security measures applied independently on each network node. Fig. 11 shows the results of this evaluation over input AND/OR graphs with up to 10,000 nodes.

We have measured the MAX-SAT resolution time for graphs of different sizes in four sub-experiments that use a different number of independent security controls (1, 5, 7 and 10) on each graph node. Each sub-experiment has been repeated 10 times and we have taken the average results. As expected, we can observe that the more security measures we use to protect each node independently, the more time is required to compute the underlying security metric. As explained before, even when there are small time variations on each sub-experiment due to the compositional characteristics of some random AND/OR graphs (and therefore smaller graphs sometimes require more time than larger graphs), the overall behaviour remains relatively stable.

Fig. 12 shows a closer look at the logical transformation and MAX-SAT resolution times for graphs with 1000 nodes while increasing the use of disjoint security measures. We can observe that

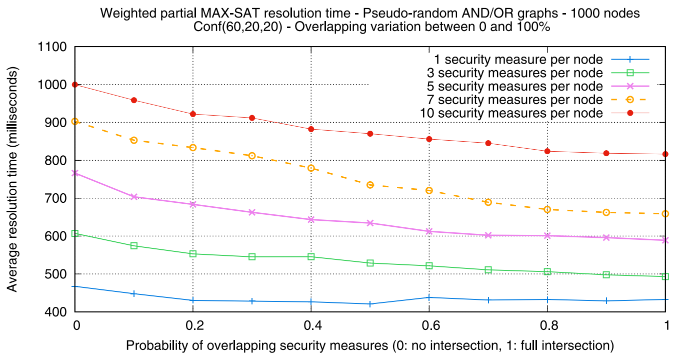


Fig. 13. Variation analysis of overlapping measures.

the MAX-SAT resolution time grows polynomially. This is essentially explained by the fact that each node variable n is replaced by a larger disjunction with x logical variables (for x security measures) plus the node variable itself. Because each node is protected by a different set of security measures (no overlapping), such replacement just increments the size of the formula by a factor of x. Therefore, the overall process is still solved in polynomial time. In hypergraph terms, the smaller the hyperedges, the lower the computation time.

This section is focused on security measures that are applied individually on each ICS component. As mentioned before, however, many security measures may be used to protect two or more components altogether in practice, e.g. fenced areas. In the next section, we evaluate our approach considering multiple overlapping security measures.

8.3. AND/OR hypergraphs with overlapping security measures

In order to analyse scenarios where two or more nodes may be protected by the same security control, we use a simple probabilistic method to generate a protection assignment as follows. Let x be the number of security measures to be applied on each graph node $n \in V_{AT}$. We then traverse the set V_{AT}, and for each node, we stochastically choose whether to assign the same security measure used with the last node, or conversely, to use a new one. In mathematical terms, we apply the same security control with probability p (positive overlapping), or we apply a new one with probability $1 - p$ (no overlapping). We repeat the above procedure x times. Fig. 13 shows the behaviour of the MAX-SAT resolution time over input graphs with 1000 nodes that have been protected following the previous assignment.

We can observe that, as the probability of overlapping increases from 0 to 1, the MAX-SAT resolution time decreases. In other words, the greater the level of overlapping, the easier is for the MAX-SAT solver engine to find the solution. In addition, this behaviour is observed independently of the number of security measures applied in the experiment. In logical terms, this happens because a security measure that protects many nodes will appear on the logical expansion of all of them (see Section 6.2), and therefore, the MAX-SAT solver leverages such interdependency to speed up the overall resolution process. We have performed a similar analysis on larger AND/OR graphs and the results indicate the same behavioural pattern, as shown in Fig. 14. The experiments involve AND/OR graphs with 1000 to 10000 nodes, using 5 security measures on each node. As expected, the results suggest that the more nodes are protected by the same security measures (i.e. higher probability), the faster is the resolution process.

Overall, the obtained results indicate that our approach can efficiently scale to large AND/OR graphs involving thousands of nodes protected by multiple security measures, and compute the proposed security metric in a matter of seconds. As a final note,

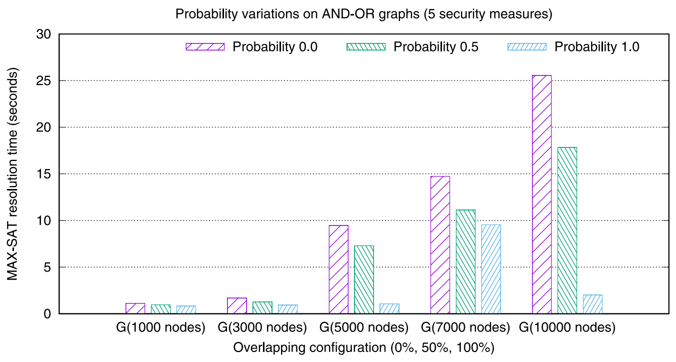


Fig. 14. Analysis on graphs of different sizes.

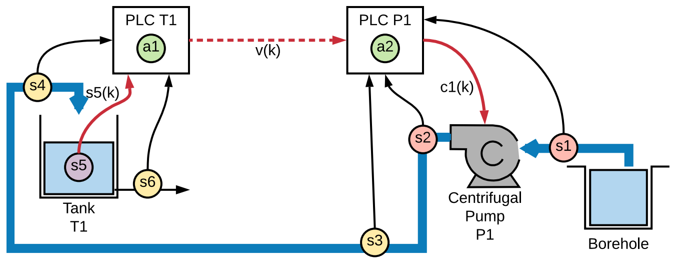


Fig. 15. Basic WTN component [19].

we have submitted a significant part of our evaluation benchmark [92] to the MaxSAT Evaluation 2019 [93] where our datasets have been part of the body of NP-hard optimization problems used to evaluate the participant MaxSAT solvers. Interestingly, none of the nine solvers evaluated with our dataset performed better than the others on every instance. The reason is that distinct MaxSAT solvers generally use very different resolution techniques. As a consequence, the result obtained from diversity (using parallel solving) is in fact quite good. Our datasets are available for download in DIMACS format [94] at [88]. In the next section, we validate our approach through a comprehensive case study.

9. Case study on water transport networks

Our case study is focused on water transport networks (WTNs) where we examine the applicability of our approach over real WTNs typically deployed in European countries.

9.1. Case study description

Typical WTNs are composed of the following main physical elements: (i) tanks, (ii) pumping stations, (iii) water sources (e.g., boreholes), and (iv) pipes. To monitor the status of each element, utilities deploy electronic sensing devices and collect measurements regarding the flow, pressure, level, and quality of the water that flows in the system. A typical configuration found in several water utilities (see [95]) is similar to the one shown in Fig. 15.

The same structure appears repeatedly in larger infrastructures thus forming patterns as illustrated in Fig. 16. For the sake of brevity and simplicity, we focus here and thereafter on the subsystem shown in Fig. 15.

In this setup, drinking water is extracted from a water source (e.g., a borehole or another tank) using a pump. The pump increases the water pressure which pushes the water into a tank, which may be located a few kilometres away at a higher elevation. The water tank is then used to provide water to consumers, as well as to transfer water to other subsystems, for instance, through another pump-tank subsystem. Additional details of this scenario can be found in [19].

The subsystem shown in Fig. 15 involves the following sensing elements: a pressure sensor before the pump ($s1$), a pressure sensor after the pump ($s2$), and a water flow sensor ($s3$) measuring the pump outflow. At the water tank, flow sensors ($s4$, $s6$) may also be installed for monitoring the inflow and outflow respectively. For its operation, the control system is comprised of two Programmable Logic Controllers (PLCs); one situated at the pump and the other at the water tank. These PLCs are connected to the system's sensors and actuators, and execute programs to achieve the control objectives. More specifically, the sensing node $s5$ provides the water level state measurement $s5(k)$ to the agent $a1$ in PLC-T1, where k is the discrete time step. Then, the control logic is executed, and the result $v(k)$ is transmitted to PLC-P1, where another control logic $a2$ is executed. Agent $a2$ instructs the contactor (i.e., an electrically operated relay) through a signal $c1(k)$ to turn on/off the pump, should the pump flow $s3$ be below a threshold.

9.2. Data collection and preparation

Various security measures are applied by water utilities in order to protect the components of their systems against malicious actors. We have acquired data from a number of water utilities and public information sources in order to: (i) determine typical measures used to protect their infrastructures, and (ii) identify components that are protected by multiple overlapping measures.

Table 7 presents a sample list of the measures acquired. We evaluate three different factors in order to calculate the cost of the attacker to compromise a security measure: (i) skills/knowledge required to design and execute the attack ($f1$), (ii) tools needed for the attack ($f2$), and (iii) time needed to execute the attack ($f3$). We use a three-point scale to rate the three factors for each measure, as shown in Table 8.

Then, for each collected measure m, we calculate the *attacker cost* $\psi(m)$ as the product of each individual rating: $\psi(m) = f1 \times f2 \times f3$. The cost of each component determines the level of difficulty an attacker will have to compromise it. The security measures along with their individual ratings and attack costs are depicted in Table 7.

Based on this information, we have used our methodology to determine the security level of such infrastructures.

9.3. Base WTN subsystem (no redundancy)

According to the collected data, the base WTN subsystem shown in Fig. 15 involves multiple security measures that simultaneously protect various components as shown in Table 9. For example, agent $a1$ is protected by a wired fence (F1-2), located inside a building with a security lock (B1-1), and an alarm system (A3-1). Sensor $s5$ is also protected by the same measure instances but also by a protection box (P2-2). In order to make the scenario even more interesting, we assume the special case where $c1$ has been heavily protected and cannot be compromised (infinite cost).

The total cost for an attacker to compromise a component n is computed as $\sum_{m \in S_n} \psi(m)$, where S_n is the set of security measures protecting n. Given the AND/OR specification of the base subsystem with no redundancy, we have run META4ICS in order to identify the set of critical ICS components and security measures, as shown in Fig. 17.

Fig. 17a shows the AND/OR graph of the network where, given the applied measures, META4ICS has identified agent $a1$ at PLC-T1 as the weakest point that can disable actuator $c1$. Its compromise implies to bypass three security measures (F1-2, B1-1, A3-1) with a total cost of 6. Fig. 17b shows the AND/OR hypergraph of the system involving its multiple overlapping measures. Agent $a1$ is responsible for measuring the water level of the tank and deciding whether to send a signal to turn on/off the pump. Note that sensor

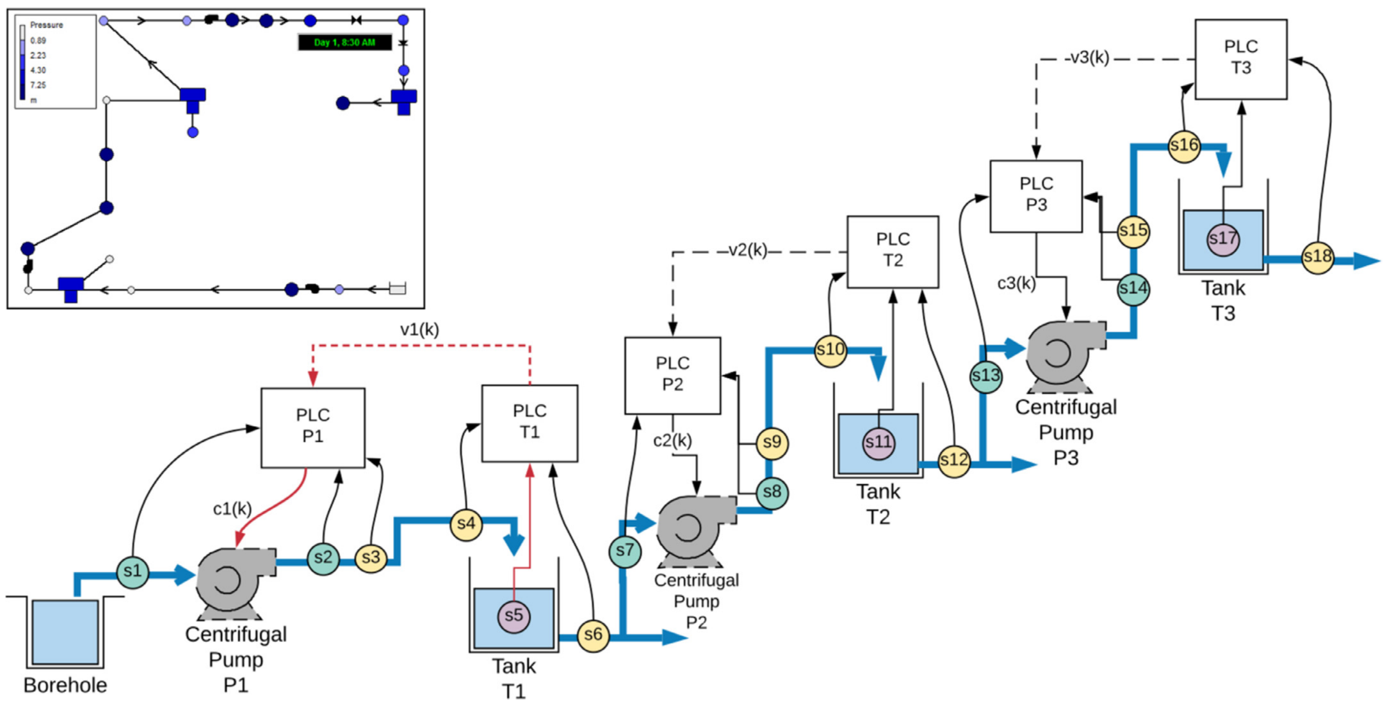


Fig. 16. Three-tank WTN architecture.

Table 7
Typical security measures and attack costs.

Measure	Skills	Tools	Time	Attack cost	Description
F1	1	1	1	**1**	Fenced area (wire)
F2	1	2	1	**2**	Fenced area (locked underground facility)
B1	1	1	2	**2**	Building + regular lock
B2	2	2	2	**8**	Building + secure lock
A1	2	3	2	**12**	Door alarm
A2	3	2	3	**18**	Alarm on telemetry box
A3	1	1	3	**3**	Patrol unit
P1	1	2	1	**2**	Locked box
P2	2	2	2	**8**	Cable protection

Table 8
Attacker's cost – three-point rating scale.

Factor / Rate	1	2	3
Skills ($f1$)	no special skills/knowledge	advanced skills/knowledge	expert skills/knowledge
Tools ($f2$)	off-the-shelf tools	non-conventional tools required	specialized tools
Time ($f3$)	$\leq$ 10 min	10–30 min	$\geq$ 30 min

Table 9
Measures per component (base subsystem).

Components	Security measures	Total cost
$s3$	{F2-1, P1-2, A2-2}	22
$s5$	{F1-2, B1-1, A3-1, P2-2}	14
$a1$	{F1-2, B1-1, A3-1}	6
$a2$	{F1-1, B2-1, P1-1, A2-1}	29
$c1$	{F1-1, B2-1}	9 + inf (special case)

Table 10
Measures per component (redundant subsystem).

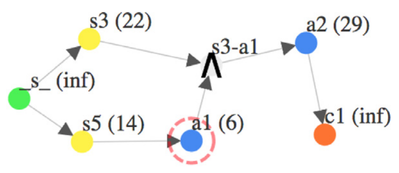
Components	Security measures	Total cost
$a2$, $a7$, $a8$, $a10$	{F1-1, B2-1, P1-1, A2-1}	29
$a1$, $a3$, $a9$	{F1-2, B1-1, A3-1}	6
$s1$, $s2$	{F1-1, B2-1}	9
$c1$	{F1-1, B2-1}	9 + inf (special case)
$s3$	{F2-1, P1-2, A2-2}	22
$s4$	{F1-2, B1-1, A3-1, P2-1}	14
$s5$	{F1-2, B1-1, A3-1, P2-2}	14
$s6$	{F2-2, P1-3, A2-3, A3-1}	25

$s5$, which also measures the level of the tank, was not identified as a critical node as it is guarded with stronger security measures and a total attack cost of 14 (see Table 9).

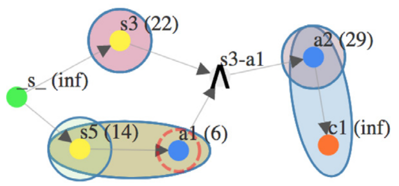
9.4. Extended WTN subsystem with redundancy

WTN systems are typically set up using the minimum configuration. However, additional sensors and agents can be used to introduce analytical redundancy in order to ensure the reliable operation of the system. In that context, we have analysed an extended scenario, detailed in [19], involving the components and security measures listed in Table 10.

Table 11 on the other hand shows the components protected by each measure instance and their costs.



(a) AND/OR graph



(b) AND/OR hypergraph

Fig. 17. Base scenario.

Table 11
Components per measure instance (redundant subsystem).

Measure instance	Measure type	Attacker cost	Protection range
F1-1	F1	1	{$a2$, $a7$, $a8$, $a10$, $c1$, $s1$, $s2$}
F1-2	F1	1	{$a1$, $a3$, $a9$, $s4$, $s5$}
F2-1	F2	2	{$s3$}
F2-2	F2	2	{$s6$}
B1-1	B1	2	{$a1$, $a3$, $a9$, $s4$, $s5$}
B2-1	B2	8	{$a2$, $a7$, $a8$, $a10$, $c1$, $s1$, $s2$}
A2-1	A2	18	{$a2$, $a7$, $a8$, $a10$}
A2-2	A2	18	{$s3$}
A2-3	A2	18	{$s6$}
A3-1	A3	3	{$a1$, $a3$, $a9$, $s4$, $s5$, $s6$}
P1-1	P1	2	{$a2$, $a7$, $a8$, $a10$}
P1-2	P1	2	{$s3$}
P1-3	P1	2	{$s6$}
P2-1	P2	8	{$s4$}
P2-2	P2	8	{$s5$}

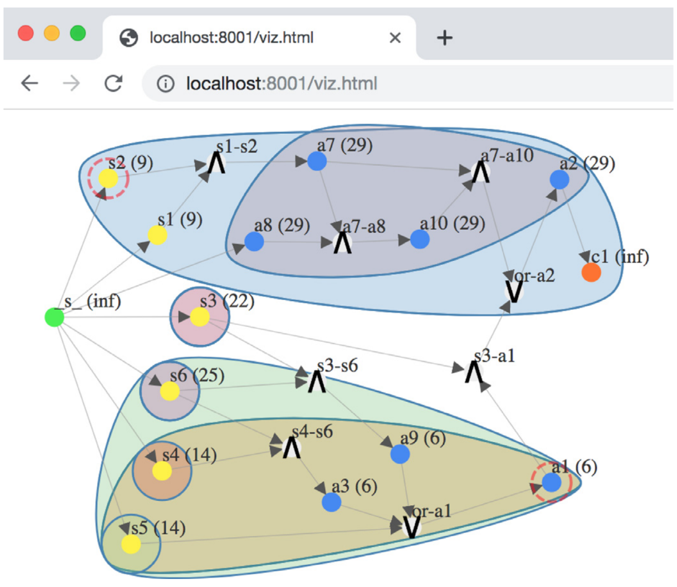


Fig. 18. AND/OR hypergraph with overlapping measures (redundant subsystem).

The structure of the network as well as the critical nodes identified by META4ICS are shown in Fig. 18. The optimal strategy indicated by the tool involves agent $a1$ and sensor $s2$ as the critical nodes and five different measure instances (F1-2, B1-1, A3-1, F1-1, B2-1) that should be violated so as to disable actuator $c1$, with a total attack cost of 15. Note that the security level of this configuration is much higher than the settings without redundancy.

10. Discussion, limitations and research challenges

10.1. ICS topological analysis

Scalability is an essential aspect when dealing with evolving and growing environments. However, it is also certain that ICS networks have been designed with some underlying structure in mind, i.e., they have not been created chaotically. In that sense, they usually present some organisational characteristics that we can leverage to address complexity even more. We argue that insightful structural information about the network (e.g. clusters, zones, regions, subnets), may be used to reduce large graphs into smaller problems and compose their solutions. However, understanding and generalising structural properties of real-world industrial settings is a challenging goal. For example, while oil and gas facilities may involve kilometres of pipes and sensors depicting elongated and thin graphs with clear articulation points, other industrial settings may be translated into more dense graphs with highly interconnected components. Additional research questions such as what numbers are representative regarding size and classes of components (hundreds, thousands, tens of thousands?) are also important and may help improve the analysis of the graphs used to represent ICS environments.

10.2. Extended cyber-physical integration

Due to the complexity of CPS environments and the diversity of attack vectors (combining social, cyber and physical methods), the construction of a unified cyber-physical security model is a hard challenge. Our model provides a robust framework to combine logical CPS dependencies with on-site security measures that are often disregarded in physical environments. At the cyber level, however, attackers may also remotely compromise software agents or affect the integrity of communication channels. We plan to integrate these aspects in the form of attack graphs, a subject widely explored in the domain of IT networks [39,43,49]. However, this integration is not trivial. A forest of individual attack graphs focused on software agents may provide good estimations on the complexity of their compromise. However, an attacker may need to compromise a subset of cyber resources only once to get to different software agents on a target facility. Therefore, identifying critical resources with aggregated costs at the cyber level also presents hard challenges. In addition, the sequential nature of multi-step attacks combining both cyber and physical perspectives [96], which in turn involve controllability aspects [97,98], must be also considered in order to produce an homogeneous extended model.

10.3. Further research challenges

Automating the generation of input AND/OR graphs for ICS is also a challenging activity, which we plan to further investigate over real-world settings. In particular, we aim at a hybrid approach involving three main aspects: using semantic inference techniques to produce analytical redundancies [99], IT-like network mapping and discovery mechanisms at the cyber level, and semi-automated methods to consolidate expert knowledge from operators. We also plan to extend our approach to consider multi-target attacks, socio-technical aspects, and defence budget constraints. Redundant components sometimes handle only a fraction of the functions provided by main components. We plan to refine our model to cover this aspect as well as standard fault-tolerant techniques such as triple modular redundancy (TMR) [100]. At the optimisation side, our computation strategy already considers a tie-break algorithm that selects the solution with minimum amount of nodes when two or more solutions with equal cost are found. However, in complex dense cases, deciding among minimal solutions with the same cost and the same amount of nodes requires further analysis. We also aim at studying the criticality of ICS components when nodes may be partially compromised or involve faulty signals, i.e., nodes that might partially operate under the presence of an attack (as opposed to be completely disconnected from the graph) [13]. Finally, we have shown that adding redundancy can increase the resiliency of an ICS environment. However, adding more components might also translate into an extended attack surface [101]. This aspect generates an interesting research problem whose solution might lead to a Pareto frontier regarding security levels and countermeasures.

11. Conclusion

Industrial control systems typically involve a large spectrum of overlapping cyber-physical security measures used to protect their operational components. As such, understanding which security measures and ICS components should be compromised so as to disturb the normal operation of the system with minimal cost for an attacker is a challenging task. In this paper, we solve this problem via an efficient mechanism based on AND/OR graphs and hypergraphs, which is able to capture complex interdependencies among ICS components and the measures used to protect them. Our strategy involves an efficient transformation of AND/OR graphs into weighted logical formulas that are then used to build and solve a Weighted Partial MAX-SAT problem. We have presented our tool META4ICS as well as an extensive experimental evaluation. The obtained results indicate that our computation strategy can properly scale to graphs with thousands of nodes in seconds. In addition, we have described a thorough case study conducted over a realistic water transport network that shows the applicability of our method. Finally, we have presented a comprehensive discussion on open problems and future research directions to further improve this work.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant no 739551 (KIOS CoE).

Appendix A. AND/OR graph transformation $f_G(t)$ and cycle handling

Given a directed AND/OR graph $G = (V, E)$ and a target node $t \in V_{AT}$, we first produce a propositional formula that represents the logical semantics of G with regards to t, i.e. the logical conditions that must be satisfied to fulfil t. We denote this transformation as $f_G(t)$, which is described in Algorithm 2. The formulation process starts at t and traverses G backwards, expanding logical conditions as needed, until nodes with no incoming edges are reached.

Algorithm 2 moves recursively through the graph and builds a valid logical sentence considering three main cases that depend on the type of node being analysed. Atomic nodes ($n \in V_{AT}$) constitute the first case, which is expanded recursively if node n has a predecessor. Atomic nodes only have one incoming edge by definition, with the exception of the source that has none. The other two cases correspond to AND/OR nodes respectively, and are treated in a similar way. In these cases, the algorithm calls a second function, described in Algorithm 3, which essentially builds sub-sentences for each predecessor of the AND/OR node (stored in NodeList X) and joins them using the appropriate operator $op \in \{\wedge, \vee\}$. There are two important aspects about the logical transformation that are worth to mention.

Algorithm 2: Main logical sentence builder (recursive)

```
Name : getSentence
Global: Graph $G = (V, E)$
Input: Node $n$, Visited nodes $M$
Output: Logical sentence $p$

1  $M' \leftarrow M \cup \{n\}$                         // mark $n$ as visited
2  if $n \in V_{AT}$ then                              // node $n$ is atomic
3      $x \leftarrow incomingNode(G, n)$                 // predecessor
4      if $not(x)\ ||\ x \in M$ then                   // $x$ null|visited
5          $p \leftarrow n$                             // atomic sentence
6      else
7          $s \leftarrow getSentence(x, M')$             //recursive call
8          $p \leftarrow (\cdot\ n \cdot \wedge \cdot s\ \cdot)$          // concat with $\cdot$
9      end
10 end
11 $X \leftarrow incomingNodes(G, n)$                  // nodes reaching $n$
12 $X \leftarrow filter(X, visited)$                   // unseen nodes only
13 if $n \in \Delta$ then $p \leftarrow getMultiSentence(X, \wedge, M')$
14 if $n \in \Theta$ then $p \leftarrow getMultiSentence(X, \vee, M')$
15 return p
```

Algorithm 3: Logical multi-sentence builder ($\wedge$, $\vee$)

```
Name : getMultiSentence
Input: NodeList $X$, Operator $op$, Visited nodes $M$
Output: Logical sentence $p$

1  if $X = \{\}$ then                                  // empty set of nodes
2      return $true$
3  end
4  $p \leftarrow ($                                     // open sentence
5  for $i = 0;\ i < |X| - 1;\ i = i + 1$ do
6      $x \leftarrow X.get(i)$                          // get node from list
7      $s \leftarrow getSentence(x, M)$                 //build sub-sentence
8      $p \leftarrow p \cdot s \cdot op$                      // concat with $\cdot$
9  end
10 $x \leftarrow X.get(|X| - 1)$                        // get last node
11 $s \leftarrow getSentence(x, M)$                     // build sub-sentence
12 $p \leftarrow p \cdot s \cdot )$                          // close sentence
13 return $p$
```

1. **The $\wedge$operator on atomic recursive calls**. In line 8, Algorithm 2 builds a sentence with the current node (n) and the sentence obtained from its predecessor (x), by using the AND operator ($\wedge$). The reason for using the AND operator relies on the semantics of the graph G, which represents dependencies between components. Because n depends on its predecessor x, node n can only be fulfilled if its predecessor x is fulfilled (x can be an AND/OR node as well), and therefore, we state so using the $\wedge$ operator.

2. **Cycles**. Normally, AND/OR graphs are acyclic [14]. However, the meaning of cycles in AND/OR graphs representing dependencies might be debatable. In this work, we aim at tackling the general case where the input graph $G = (V, E)$ may also contain cycles.

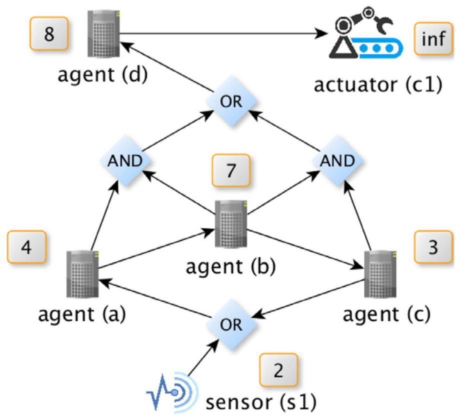


Fig. A.19. Cycle example (nodes *a*, *b*, *c*).

Our approach to deal with cycles is to keep a record of the nodes that have been analysed so far. In that sense, Algorithm 2 controls cycles by using a set of visited nodes. Nodes that have already been visited are not expanded again in deeper explorations coming from them. This is possible because the truth value of an already visited node is present in an earlier part of the formula and connected to their predecessors via the $\wedge$ operator. For the atomic case, this is directly implemented in line 5, while for complex AND/OR cases, visited predecessors are previously filtered in line 12.

As an example, let us consider the cyclic graph illustrated in Fig. A.19. As explained before, the logical transformation $f_G(c1)$ starts at a target node $c1$ and traverses the graph backwards until all the components in the graph have been covered. It is easy to see that at some point, the partial formula in this example will have the following aspect (where nodes *a*, *b* and *c* have not yet been expanded):

$$f_G(c1) = c1 \wedge d \wedge ((a \wedge b) \vee (b \wedge c))$$

When node a is expanded with the nodes it depends on, we can see that a depends on $(s1 \vee c)$. While $s1$ is a terminal node and does not depend on any node, node c still depends on b, and b eventually depends on a. Because node a was already visited, the transformation process stops the exploration at this point (does not go further from a again), with the following partial formula:

$$f_G(c1) = c1 \wedge d \wedge (((a \wedge (s1 \vee (c \wedge b \wedge a))) \wedge b) \vee (b \wedge c))$$

Clearly, the last a can be removed since it already appears early in the formula as a predecessor connected with the AND operator. Hypothetically, if we follow the loop indefinitely, we would see the same pattern again and again. From a satisfiability perspective, it would yield the same result since the same variables are conjunctively joined in the sentence. A similar situation occurs when nodes b and c are expanded. At the end, the final transformation looks as follows:

$$\begin{aligned} f_G(c1) = c1 \wedge d \wedge (((a \wedge (s1 \vee (c \wedge b))) \wedge (b \wedge a \wedge (s1 \vee c))) \\ \vee((b \wedge a \wedge (s1 \vee c)) \wedge (c \wedge b \wedge a \wedge s1))) \end{aligned}$$

Fig. A.20 shows the metric resolution for this graph.

It is important to note that a typical approach to deal with cycles, widely used in many graph-related works, is to analyse each cycle as a whole. That is, treat loopy formations as clusters that can be collapsed and analysed as one super node where its cost is equal to the minimum cost among its member nodes. From a graph-theoretical perspective, these clusters are strongly connected components (SCC) and can be efficiently identified in linear time using, for example, Tarjan's algorithm [102]. However, such an approach does not properly work with AND/OR graphs. The previous scenario is a counterexample. The minimum node cost within the

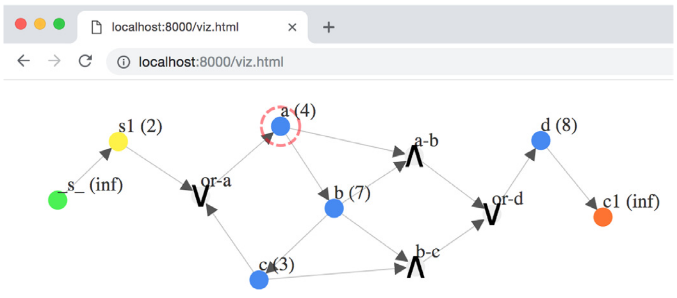


Fig. A.20. Cycle example (between *a*, *b*, and *c*) - META4ICS viewer.

cycle is 3 (node c), however, the solution to that problem is node a with cost 4. This is because dependencies outside the loop may affect the overall optimal solution.

Appendix B. Tseitin transformation example

The Tseitin transformation [17] is an important technical part of our approach to produce equisatisfiable CNF formulas. From a practical perspective, we have experimentally observed that the naive CNF conversion method barely scales to a few hundred nodes before running out of memory. Instead, the Tseitin transformation enables our approach to scale to graphs with thousand of nodes in a matter of seconds. In this section, we exemplify how the Tseitin transformation works and the kind of formulas we obtain from it. Let us consider the following logical formula: $\phi = (p \vee q) \wedge r$.

The subformulas involved in ϕ (non-atomic) are: i. $p\vee q$ and ii. $(p\vee q)\wedge r$. We now introduce a new variable for each subformula as follows: i. $x_1 \leftrightarrow p\vee q$ and ii. $x_2 \leftrightarrow x_1 \wedge r$ (note that we are using x_1 instead of $p\vee q$). Putting all substitutions together (including x_2 as the substitution of ϕ), we obtain the following transformed formula: $\tau(\phi) = x_2 \wedge (x_2 \leftrightarrow x_1 \wedge r) \wedge (x_1 \leftrightarrow p \vee q)$. Now each conjunct in $\tau(\phi)$ can be individually converted to its conjunctive normal form (CNF). For $x_1 \leftrightarrow p\vee q$, we have that:

$$\begin{aligned} x_1 \leftrightarrow p \vee q &\equiv (x_1 \rightarrow (p \vee q)) \wedge ((p \vee q) \rightarrow x_1) \\ &\equiv (\neg x_1 \vee p \vee q) \wedge (\neg(p \vee q) \vee x_1) \\ &\equiv (\neg x_1 \vee p \vee q) \wedge ((\neg p \wedge \neg q) \vee x_1) \\ &\equiv (\neg x_1 \vee p \vee q) \wedge (\neg p \vee x_1) \wedge (\neg q \vee x_1) \end{aligned}$$

It can be observed that after applying a few logical equivalence rules, the obtained formula is in CNF. For $x_2 \leftrightarrow x_1 \wedge r$, we have that:

$$\begin{aligned} x_2 \leftrightarrow x_1 \wedge r &\equiv (x_2 \rightarrow (x_1 \wedge r)) \wedge ((x_1 \wedge r) \rightarrow x_2) \\ &\equiv (\neg x_2 \vee (x_1 \wedge r)) \wedge (\neg(x_1 \wedge r) \vee x_2) \\ &\equiv (\neg x_2 \vee x_1) \wedge (\neg x_2 \vee r) \wedge (\neg(x_1 \wedge r) \vee x_2) \\ &\equiv (\neg x_2 \vee x_1) \wedge (\neg x_2 \vee r) \wedge (\neg x_1 \vee \neg r \vee x_2) \end{aligned}$$

Finally, substituting each clause in $\tau(\phi)$ by its corresponding CNF conversion as shown before, we obtain a new CNF formula with additional variables that is not equivalent to the original one, but is equisatisfiable. This means that for any truth assignment, ϕ and $\tau(\phi)$ will always be either both *true* or both *false*. The expanded new CNF formula is as follows:

$$\begin{aligned} \tau(\phi) = x_2 \wedge (\neg x_1 \vee p \vee q) \wedge (\neg p \vee x_1) \wedge (\neg q \vee x_1) \\ \wedge(\neg x_2 \vee x_1) \wedge (\neg x_2 \vee r) \wedge (\neg x_1 \vee \neg r \vee x_2) \end{aligned}$$

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.jisa.2020.102471.

References

[1] Humayed A, Lin J, Li F, Luo B. Cyber-physical systems security – a survey. IEEE Internet Things 2017;4(6):1802–31.
[2] CyberXReport2019. 2019 Global ICS & IIoT Risk Report. https://cyberx-labs.com/resources/risk-report-2019/ Cited December 2019.
[3] CyberXReport2020. 2020 Global ICS & IIoT Risk Report. https://cyberx-labs.com/resources/risk-report-2020/ Cited December 2019.
[4] Hankin C. Game theory and industrial control systems. In: Essays dedicated to Hanne Riis Nielson and Flemming Nielson on the occasion of their 60th birthdays on semantics, logics, and calculi – Volume 9560. Springer-Verlag New York, Inc.; 2016. p. 178–90. doi:10.1007/978-3-319-27810-0_9.
[5] Lee RM, Assante MJ, Conway T. Analysis of the cyber attack on the Ukrainian power grid. Defense Use Case. Tech. Rep. SANS, E-ISAC; 2016.
[6] Falliere N, Murchu LO, Chien E. W32.Stuxnet Dossier. Tech. Rep. Symantec; 2011.
[7] Stouffer KA, Falco JA, Scarfone KA. Guide to Industrial Control Systems (ICS) security, SP 800-82 r2. Tech. Rep. NIST; 2015.
[8] Positive Technologies. Cybersecurity threatscape – Q2 2018. Tech. Rep. Positive Technologies; 2018.
[9] Andreeva O, Gordeychik S, Gritsai G, Kochetova O, Potseluevskaya E, Sidorov SI., et al. Industrial Control Systems vulnerabilities statistics – Kaspersky Lab. https://media.kasperskycontenthub.com/wp-content/uploads/sites/43/2016/07/07190426/KL_REPORT_ICS_Statistic_vulnerabilities.pdf Cited December 2019.
[10] Ostfeld A, et al. Battle of the water calibration networks. J Water Resour Plan Manag 2012;138(5):523–32.
[11] Desmedt Y, Wang Y. Maximum flows and critical vertices in AND/OR graphs. In: Ibarra OH, Zhang L, editors. Computing and combinatorics. Berlin Heidelberg: Springer; 2002. p. 238–48.
[12] Desmedt Y, Wang Y. Analyzing vulnerabilities of critical infrastructures using flows and critical vertices in And/Or graphs. Int J Found Comput Sci 2004;15(1):107–25. doi:10.1142/S0129054104002339.
[13] Jakimoski G, Burmester M. Using faulty flows in AND/OR graphs to model survivability and reliability in distributed systems. https://www.cs.fsu.edu/files/reports/TR-060318.pdf 2004.
[14] dos Santos Souza U, Protti F, da Silva MD. Revisiting the complexity of and/or graph solution. J Comput Syst Sci 2013;79(7):1156–63.
[15] Barrère M, Hankin C, Nicolaou N, Eliades D, Parisini T. Identifying security-critical cyber-physical components in industrial control systems. arxiv:1905.04796, 2019a.
[16] Barrère M, Hankin C, Eliades D, Nicolaou N, Parisini T. Assessing cyber-physical security in industrial control systems. In: Proceedings of the 6th international symposium for ICS & SCADA cyber security research 2019; 2019. p. 49–58.
[17] Tseitin GS. On the complexity of derivation in propositional calculus. In: Slisenko A, editor. Studies in constructive maths and mathematical logic, Part II, seminars in mathematics. Steklov Mathematical Institute; 1970. p. 234–59.
[18] Nicol DM, Sanders WH, Trivedi KS. Model-based evaluation: from dependability to security. IEEE Trans Dep Sec Comput 2004;1(1):48–65.
[19] Nicolaou N, Eliades DG, Panayiotou C, Polycarpou MM. Reducing vulnerability to cyber-physical attacks in water distribution networks. In: 2018 international workshop on cyber-physical systems for smart water networks (CySWater); 2018. p. 16–19.
[20] Ford LR, Fulkerson DR. Flows in networks. University Press; 1962.
[21] Dantzig GB, Fulkerson DR. On the max flow min cut theorem of networks. Tech. Rep. Santa Monica, CA: RAND Corporation; 1955.
[22] Davies J, Bacchus F. Solving MAXSAT by solving a sequence of simpler SAT instances. In: Lee J, editor. Principles and practice of constraint programming – CP 2011. Springer; 2011. p. 225–39.
[23] Arulselvan A, Commander CW, Elefteriadou L, Pardalos PM. Detecting critical nodes in sparse graphs. Comput Oper Res 2009;36(7):2193–200.
[24] Shen S, Smith JC. Polynomial-time algorithms for solving a class of critical node problems on trees and series-parallel graphs. Networks 2012;60(2):103–19.
[25] Addis B, Summa MD, Grosso A. Identifying critical nodes in undirected graphs: complexity results and polynomial algorithms for the case of bounded treewidth. Discret Appl Math 2013;161(16):2349–60. doi:10.1016/j.dam.2013.03.021.
[26] Chen X. Critical nodes identification in complex systems. Complex Intell Syst 2015;1(1):37–56.
[27] Aringhieri R, Grosso A, Hosteins P, Scatamacchia R. A general evolutionary framework for different classes of critical node problems. Eng Appl ArtifIntell 2016;55:128–45.
[28] Lalou M, Tahraoui MA, Kheddouci H. The Critical Node Detection Problem in networks: asurvey. Comput Sci Rev 2018;28:92–117.
[29] Deng Y, Song L, Zhou Z, Liu P. Complexity and vulnerability analysis of critical infrastructures: a methodological approach. Mathematical Problems in Engineering 2017;2017:12. Article ID 8673143.
[30] Steiner R, Barrère M, Lupu E. WSNs under attack! How bad is it? Evaluating connectivity impact using centrality measures. In: Proceedings of the IET living in the Internet of Things conference, cybersecurity of the IoT; 2018.
[31] Schneier B. Attack trees – modeling security threats. http://www.schneier.com/paper-attacktrees-ddj-ft.html; 1999.
[32] Xie F, Lu T, Guo X, Liu J, Peng Y, Gao Y. Security analysis on cyber-physical system using attack tree. In: 9th international conference on intelligent information hiding and multimedia signal processing; 2013. p. 429–32.
[33] Davis KR, Davis CM, Zonouz SA, Bobba RB, Berthier R, Garcia L, et al. A cyber–physical modeling and assessment framework for power grid infrastructures. IEEE Trans Smart Grid 2015;6(5):2464–75.
[34] Depamelaere W, Lemaire L, Vossaert J, Naessens V. CPS security assessment using automatically generated attack trees. In: Proceedings of the 5th international symposium for ICS & SCADA cyber security research 2018; 2018. p. 1–10.
[35] Ruijters E, Stoelinga M. Fault tree analysis: a survey of the state-of-the-art in modeling, analysis and tools. Comput Sci Rev 2015;15–16:29–62.
[36] AkersS. B. Binary decision diagrams. IEEE Trans Comput 1978;C-27(6):509–16. doi:10.1109/TC.1978.1675141.
[37] Kordy B, Pitre-Cambacds L, Schweitzer P. DAG-based attack and defense modeling: don't miss the forest for the attack trees. Comput Sci Rev 2014;13–14:1–38.
[38] Altner DS, Ergun O, Uhan NA. The maximum flow network interdiction problem: valid inequalities, integrality gaps, and approximability. Oper Res Lett 2010;38(1):33–8.
[39] Wang L, Jajodia S, Singhal A. Network Security Metrics. 1st. Springer Publishing Company, Incorporated; 2017. 3319665049, 9783319665047.
[40] Lippmann R, Ingols K. An annotated review of past papers on attack graphs. Project report IA; 2005.
[41] Singhal A, Ou X. Security risk analysis of enterprise networks using probabilistic attack graphs. NIST Interagency Rep. 7788; 2011.
[42] Bopche GS, Mehtre BM. Attack graph generation, visualization and analysis: issues and challenges. In: Mauri JL, Thampi SM, Rawat DB, Jin D, editors. Security in computing and communications. SSCC 2014. Berlin Heidelberg: Springer; 2014. p. 379–90. ISBN 978-3-662-44966-0.
[43] Shandilya V, Simmons CB, Shiva S. Use of attack graphs in security systems. J Comput Netw Commun 2014;1(1).
[44] Kaynar K. A taxonomy for attack graph generation and usage in network security. J Inf Secur Appl 2016.
[45] Hong JB, Kim DS, Chung C-J, Huang D. A survey on the usability and practical applications of Graphical Security Models. Comput Sci Rev 2017;26:1–16.
[46] Ammann P, Wijesekera D, Kaushik S. Scalable, graph-based network vulnerability analysis. In: Proceedings of the 9th ACM conference on computer and communications security (CCS'02); 2002. p. 217.
[47] Ou X, Boyer WF, McQueen MA. A scalable approach to attack graph generation. In: Proceedings of the 13th ACM conference on computer and communications security (CCS'06). ACM Press; 2006. p. 336–45.
[48] Wang L, Albanese M, Jajodia S. Network hardening – an automated approach to improving network security. Springer; 2014.
[49] Barrère M, Steiner RV, Mohsen R, Lupu EC. Tracking the bad guys: an efficient forensic methodology to trace multi-step attacks using core attack graphs. In: 13th IEEE int. conference on network and service management; 2017. p. 1–7.
[50] Barrère M, Lupu EC. Naggen: a network attack graph generation tool. In: Proc. of the IEEE conf. on communications and network security; 2017. p. 378–9.
[51] Sheyner O, Haines J, Jha S, Lippmann R, Wing JM. Automated generation and analysis of attack graphs. In: Proceedings of the 2002 IEEE symposium on security and privacy. IEEE Computer Society; 2002. p. 273–84. ISBN 0-7695-1543-6.
[52] Jha S, Sheyner O, Wing J. Two formal analyses of attack graphs. In: Proc. of the 15th IEEE computer security foundations workshop; 2002. p. 49–63.
[53] Hong J, Kim DS. HARMs: hierarchical attack representation models for network security analysis. In: Proc. of the 10th Australian information security management conference. Australia: SRI Security Research Institute; 2012. p. 74–81.
[54] Noel S, Jajodia S. Managing attack graph complexity through visual hierarchical aggregation. In: Proceedings of the 2004 ACM workshop on visualization and data mining for computer security. New York, NY, USA; 2004. p. 109–18. 1-58113-974-8.
[55] DeLoach SA, Ou X, Zhuang R, Zhang S. Model-driven, moving-target defense for enterprise network security. In: Models@ run. time. Springer; 2014. p. 137–61.
[56] Ingols K, Lippmann R, Piwowarski K. Practical attack graph generation for network defense. In: In Proc. of the 22nd annual computer security applications conference, ACSAC '06.; 2006. p. 121–30.
[57] Noel S, Jajodia S, O'Berry B, Jacobs M. Efficient minimum-cost network hardening via exploit dependency graphs. In: Proceedings of the 19th annual computer security applications conference; 2003. p. 86–95.
[58] Alhomidi MA, Reed MJ. Attack graphs representations. In: 4th computer science and electronic engineering conference (CEEC); 2012. p. 83–8.
[59] Poolsappasit N, Dewri R, Ray I. Dynamic security risk management using Bayesian attack graphs. IEEE Trans Depend Secure Comput 2012;9(1):61–74.
[60] Muoz-Gonzlez L, Sgandurra D, Barrère M, Lupu EC. Exact inference techniques for the analysis of Bayesian attack graphs. IEEE Trans Depend Secure Comput 2019;16(2):231–44.
[61] Homer J, Ou X. Sat-solving approaches to context-aware enterprise network security management. IEEE J Sel Areas Commun 2009;27(3):315–22.
[62] Huang H, Zhang S, Ou X, Prakash A, Sakallah K. Distilling critical attack graph surface iteratively through minimum-cost SAT solving. In: 27th Annual computer security applications conference. ACM; 2011. p. 31–40. ISBN 978-1-4503-0672-0.

[63] Barrère M, Badonnel R, Festor O. A SAT-based autonomous strategy for security vulnerability management. In: 2014 IEEE network operations and management symposium (NOMS); 2014. p. 1–9.
[64] Barrère M, Badonnel R, Festor O. Vulnerability assessment in autonomic networks and services: a survey. IEEE Commun Surv Tutor 2014;16(2):988–1004.
[65] Wang L, Jajodia S, Singhal A, Cheng P, Noel S. k-Zero day safety: a network security metric for measuring the risk of unknown vulnerabilities. IEEE Trans Depend Secure Comput 2014;11(1):30–44.
[66] Li T, Hankin C. Effective defence against zero-day exploits using Bayesian networks. In: Havarneanu G, Setola R, Nassopoulos H, Wolthusen S, editors. Critical information infrastructures security. Cham: Springer International Publishing; 2017. p. 123–36.
[67] Nessus. Nessus vulnerability scanner. https://www.tenable.com/products/nessus Cited December 2019.
[68] CyberXAssessmentTool2018. Automated risk & vulnerability assessment for ICS networks. https://cyberx-labs.com/en/automated-vulnerability-assessments-for-ics-networks/ Cited December 2019.
[69] Ginter A. Secure operations technology. Lulu.com; 2019. ISBN 9780995298439.
[70] Elderhalli Y, Hasan O, Ahmad W, Tahar S. Formal dynamic fault trees analysis using an integration of theorem proving and model checking. In: Dutle A, Muñoz C, Narkawicz A, editors. NASA formal methods. Cham: Springer International Publishing; 2018. p. 139–56. ISBN 978-3-319-77935-5.
[71] Ibrahim A, Kacianka S, Pretschner A, Hartsell C, Karsai G. Practical causal models for cyber-physical systems. In: Badger JM, Rozier KY, editors. NASA formal methods. Cham: Springer International Publishing; 2019. p. 211–27. ISBN 978-3-030-20652-9.
[72] Askarpour M, Ghezzi C, Mandrioli D, Rossi M, Tsigkanos C. Formal methods in designing critical cyber-physical systems. Springer; 2019. p. 110–30. ISBN 978-3-030-30985-5.
[73] Vellaithurai C, Srivastava A, Zonouz S, Berthier R. Cpindex: cyber-physical vulnerability assessment for power-grid infrastructures. IEEE Trans Smart Grid 2015;6(2):566–75.
[74] Tippenhauer NO, Temple WG, Vu AH, Chen B, Nicol DM, Kalbarczyk Z, et al. Automatic generation of security argument graphs. In: IEEE 20th Pacific rim int. symposium on dependable computing; 2014. p. 33–42.
[75] Rahman MA, Shaer EA, Kavasseri RG. Security threat analytics and countermeasure synthesis for power system state estimation. In: 44th annual IEEE/IFIP int. conf. on dependable systems and networks; 2014. p. 156–67.
[76] Friedberg I, McLaughlin K, Smith P. A cyber-physical resilience metric for smart grids. In: 2017 IEEE power energy society innovative smart grid technologies conference (ISGT); 2017. p. 1–5.
[77] Chung E, Hanks JS. Fault tree analyses as a tool for flight control system architecture design. In: Annual reliability and maintainability symp; 2016. p. 1–6.
[78] Rahimian MA, Aghdam AG. Structural controllability of multi-agent networks: Robustness against simultaneous failures. Automatica 2013;49(11):3149–57.
[79] Alcaraz C, Wolthusen S. Recovery of structural controllability for control systems. In: Butts J, Shenoi S, editors. Critical infrastructure protection VIII. Berlin, Heidelberg: Springer; 2014. p. 47–63. ISBN 978-3-662-45355-1.
[80] Liu Y-Y, Barabási A-L. Control principles of complex systems. Rev Mod Phys 2016;88:035006.
[81] Pequito S, Khorrami F, Krishnamurthy P, Pappas GJ. Analysis and design of actuation sensing communication interconnection structures toward secured/resilient lti closed-loop systems. IEEE Trans Control Netw Syst 2019;6(2):667–78.
[82] Ferrag MA, Maglaras L, Moschoyiannis S, Janicke H. Deep learning for cyber security intrusion detection: approaches, datasets, and comparative study. J Inf Secur Appl 2020;50.
[83] CVSS. CVSS, common vulnerability scoring system. http://www.first.org/cvss/ Cited December 2019.
[84] Berre DL, Parrain A. The sat4j library, release 2.2.. JSAT 2010;7(2-3):56–9.
[85] Cook SA. The complexity of theorem-proving procedures. In: Proceedings of the third annual ACM symposium on theory of computing. New York, NY, USA: ACM; 1971. p. 151–8.
[86] Berge C. Graphs and hypergraphs. Oxford, UK, UK: Elsevier Science Ltd.; 1985. ISBN 0720404797.
[87] Berge C. Hypergraphs: combinatorics of finite sets. North-Holland; 1989.
[88] Barrère M. META4ICS – metric analyser for industrial control systems. https://github.com/mbarrere/meta4ics; 2019.
[89] SAT4J. SAT4J. http://www.sat4j.org/ Cited December 2019.
[90] Gurobi. Gurobi. https://www.gurobi.com/ Cited December 2019.
[91] d3.js. D3.js –data driven documents. https://d3js.org/ Cited December 2019.
[92] Barrère M, Hankin C, Nicolaou N, Eliades D, Parisini T. MaxSAT evaluation 2019 – benchmark: identifying security-critical cyber-physical components in weighted AND/OR graphs. In: Proceedings of the MaxSAT evaluation 2019 (MSE19); 2019. https://arxiv.org/abs/1911.00516.
[93] MaxSAT Evaluation 2019. https://maxsat-evaluations.github.io/2019/ Cited December 2019.
[94] DIMACS. Satisfiability Suggested Format. http://dimacs.rutgers.edu/ Cited December 2019.
[95] Trifunovic N. Introduction to Urban Water Distribution. 1st. London, UK: Taylor & Francis Group; 2006. 10 0415395178, 10 0415395186
[96] Hawrylak PJ, Haney M, Papa M, Hale J. Using hybrid attack graphs to model cyber-physical attacks in the smart grid. In: 5th international symposium on resilient control systems; 2012. p. 161–4.
[97] Pequito S, Khorrami F, Krishnamurthy P, Pappas G. Analysis and design of actuation-sensing-communication interconnection structures towards secured/resilient LTI closed-loop systems. IEEE Trans Control Netw Syst 2018.
[98] Liu Y, Barabási A. Control principles of complex networks. arxiv:1508.05384; 2016.
[99] Milis GM, Panayiotou CG, Polycarpou MM. SEMIoTICS: semantically enhanced IoT-enabled intelligent control systems. IEEE Internet Things J 2019;6(1):1257–66.
[100] Kastensmidt FL, Sterpone L, Carro L, Reorda MS. On the optimal design of triple modular redundancy logic for SRAM-based FPGAS. In: Proceedings of the conference on design, automation and test in Europe – Volume 2. Washington, DC, USA: IEEE Computer Society; 2005. p. 1290–5.
[101] Li T, Feng C, Hankin C. Improving ICS cyber resilience through optimal diversification of network resources. arxiv:1811.00142; 2018.
[102] Tarjan R. Depth-first search and linear graph algorithms. SIAM J Comput 1972.
[103] Haasl D F, Roberts N H, Vesely W E, Goldberg F F. Fault Tree Handbook". U.S. Nuclear Regulatory Commission 1981.