A decision support tool for optimal configuration of critical infrastructures

Andrea Tortorelli*, Andrea Fiaschetti,
Roberto Germanà and Alessandro Giuseppi

Department of Computer, Control and
Management Engineering (DIAG),
'Antonio Ruberti' of the Sapienza University of Rome,
Via Ariosto 25, 00185 Rome, Italy
Email: tortorelli@diag.uniroma1.it
Email: fiaschetti@diag.uniroma1.it
Email: germana@diag.uniroma1.it
Email: giuseppi@diag.uniroma1.it
*Corresponding author

Vincenzo Suraci

Department of Computer, Control and
Management Engineering (DIAG),
'Antonio Ruberti' of the Sapienza University of Rome,
Via Ariosto 25, 00185 Rome, Italy
and
Università degli Studi eCampus,
Via Isimbardi 10, 22060, Novedrate (CO), Italy
Email: suraci@diag.uniroma1.it

Andrea Andreani and Francesco Delli Priscoli

Department of Computer, Control and
Management Engineering (DIAG),
'Antonio Ruberti' of the Sapienza University of Rome,
Via Ariosto 25, 00185 Rome, Italy
Email: andreani.1792911@studenti.uniroma1.it
Email: dellipriscoli@diag.uniroma1.it

Abstract: In this work, a decision support system aimed at suggesting to critical infrastructure (CI) operators the optimal configuration in terms of deployed security functions Ali ties is presented. Two specific problems have been addressed: the security evaluation problem and the security configuration computation problem. Concerning the former problem, the framework provided by the Open Source Security Testing Methodology Manual (OSSTMM) has been retained and extended to capture innovative security features providing CI operators with a holistic insight on the system security level. Concerning the latter problem, the DSS has been provided with an optimisation framework based on a genetic algorithm (GA) for exploring the solution space; in this respect, three different implementations of the adopted GA have been developed and evaluated in realistic operation scenarios. Finally, the outputs of the DSS have been validated from a security point of view.

Keywords: critical infrastructures; cyber-physical security; decision support systems; DSSs; genetic algorithms.

Reference to this paper should be made as follows: Tortorelli, A., Fiaschetti, A., Germanà, R., Giuseppi, A., Suraci, V., Andreani, A. and Delli Priscoli, F. (2022) 'A decision support tool for optimal configuration of critical infrastructures', *Int. J. Critical Infrastructures*, Vol. 18, No. 2, pp.105–127.

Biographical notes: Andrea Tortorelli received his MSc (Summa Cum Laude) in Control Engineering in 2017 from the Sapienza University of Rome. Currently, he is a PhD student in Automatica at the Department of Computer, Control and Management Engineering 'Antonio Ruberti' of the same institute. He participated in four EU-funded projects. Currently, he is an Adjunct Professor of 'Automatic Controls' and 'Instrumentation for Automation' at the eCampus University (CO). His current research areas include cyber-security, network control, smart mobility, operations management, predictive maintenance.

Andrea Fiaschetti received his PhD in Systems Engineering from the Sapienza University of Rome in 2012. He currently is a R&D System Engineer at the Thales Alenia, Rome. He has been from 2008 to 2012 the R&D Project Manager at the CRAT, Rome. His research interests are satellite telecommunication systems, management of satellite resources, and complex systems control. He his member of the Integrated Mission Group for Security in support of aerospace and defence industry association of Europe.

Roberto Germanà received his MSc (Summa Cum Laude) from the Sapienza University of Rome in 2017. He is currently a PhD student in Automatica at the Department of Computer, Control, and Management Engineering 'Antonio Ruberti' of the University 'La Sapienza'. He specialises in optimisation solutions and model predictive control applied to power systems.

Alessandro Giuseppi received from the Sapienza University of Rome, Italy, his BSc in Computer and Automation Engineering in 2014 and his MSc in Control Engineering in 2016, both summa cum laude. Currently, he is a PhD candidate in Automatica at the Department of Computer, Control, and Management Engineering 'Antonio Ruberti' of the same university. Since 2016, he has participated in four EU-funded projects. He authored about 13 journal and conference papers on the application of control theory to networked systems.

Vincenzo Suraci received his MSc (Summa Cum Laude) in Computer Engineering and PhD in Systems Engineering from the Department of Computer, Control, and Management Engineering 'Antonio Ruberti', Sapienza University of Rome, in 2004 and 2008, respectively. He is currently an Associate Professor with eCampus University, Novedrate (CO), Italy and a Project Manager with CRAT, Rome. His current research interests include the development of advanced control and operation research methodologies (e.g., reinforcement learning, column generation, hybrid automata, and discrete event systems) for the solution of emerging engineering problems, including, connection admission control, access technologies selection, quality of experience/quality of service cognitive control, resource management, and convergence of heterogeneous networks.

Andrea Andreani received his Bachelor's in Computer Engineering (Summa Cum Laude) in 2020 from the University of Rome 'La Sapienza'. Currently, he is a Master's student of Computer Engineering at the same institute. His research areas are mainly cyber-security and neural networks for image recognition.

Francesco Delli Priscoli received his MSc (Summa Cum Laude) in Electronics Engineering and PhD in Systems Engineering from the Sapienza University of Rome, in 1986 and 1991, respectively. From 1986 to 1991, he was with Telespazio, Rome. Since 1991, he has been with the University of Rome 'La Sapienza', where he is currently a Full Professor of 'Automatic Control', 'Control of Autonomous Multiagent Systems', and 'Control of Communication and Energy Networks'. He is an Associate Editor of *Control Engineering Practice* (Elsevier). He was the scientific responsible at Sapienza University of Rome for 35 projects funded by the European Union and by the European Space Agency. His current research interests include closed loop multiagent learning techniques in advanced communication and energy networks. He is a member of the IFAC Technical Committee on 'Networked Systems'.

1 Introduction

In the context of critical infrastructures, a proper selection of the security functionalities to deploy represents a crucial yet intrinsically complex problem due to several reasons.

First, CIs are complex multi-domain systems characterised by lots of internal interactions and processes (operations). The security functionality selection problem is indeed characterised by a huge space of solution and requires a clear insight on the impact of each functionality on the overall security level. In addition, the security evaluator should have a deep understanding of different domains and be constantly updated about newly discovered vulnerabilities. There exist automatic tools for vulnerability identification such as Greenbone Networks (2019), BeyondTrust (2019), Rapid7 (2019) and SWASCAN (2019). However, these solutions are ICT oriented and fail to provide a holistic characterisation of the security level in the multi-domain context CIs.

Second, the evaluation of given security configurations is not an easy task itself. In this respect, compliance with security standards and guidelines, such as the IEC 62443 and the ISO/IEC 27000 series of standards and the ISO/IEC 15408 standard (Common Criteria: New CC Portal, 2019), guarantees that best practices are followed. With this approach, the security level of a system can be characterised in a qualitative way or by discrete levels which renders difficult to compare similar security configurations or to track the evolution of the security level in presence of relatively small improvements. To address this issue, methodologies such as Herzog (2010), FIRST (2019) and Manadhata and Wing (2011) can be adopted as they provide mathematical frameworks allowing to characterise from a quantitative point of view security features concurring to the definition of the security level.

The aim of the present work is to present a decision support system (DSS) allowing to address the above-mentioned issues. To tackle the computational costs of the security functionality selection problem the DSS adopts a genetic algorithm (GA) aimed at providing (sub) optimal solutions in relatively short time. To tackle the security evaluation problem, the DSS adopts extended versions of the security indicators developed by the Open Source Security Testing Methodology Manual (OSSTMM) (Herzog, 2010) (referred to as extended OSSTMM indicators) allowing to capture security features not covered by the standard OSSTMM, such as components lifecycle and to exploit the knowledge stored in accessible databases such as the common vulnerability and exposures (CVE) database (MITRE, 2019).

2 Related works

For suggesting security configurations to a CI operator, the DSS must address two specific issues. First, a methodology for evaluating the security level of a configuration must be embedded in the DSS in order to

1 compare the candidate configurations

2 provide the CI operator with a characterisation of the system security level.

Second, an optimisation framework for selecting the optimal security configurations must be set up.

With respect to the security evaluation problem, it must be noted that it is an inherently complex problem since security itself is a subjective property. Indeed, the literature lacks a unique definition of security as well as those elements concurring to its evaluation. Many security standards and guidelines, such as Common Criteria: New CC Portal (2019), Scarfone et al. (2008) and IEC (2018), focus on the identification of good practices to follow or security requirements to verify for guaranteeing that given security levels are met. However, these security levels are often qualitative or discrete which does not allow to evaluate the impact of single security functionality nor to automatise, at a certain extent, the security evaluation procedure. Other approaches address the security evaluation problem by focusing on specific attack scenarios which translates in studying the system resilience in response to given attacks. Several solutions, based on attack graphs (such as Lye and Wing, 2005; Johnson et al., 2016), attack trees (such as Gadyatsaya et al., 2016; Kumar et al., 2018) or control theory (such as Teixeira et al., 2012; Pasqualetti et al., 2013), can be found in the literature. The results obtained with these approaches, however, are either limited to the considered attack scenarios, and thus do not allow to characterise intrinsic security, or require a model of the system, which in the context of CIs can be difficult. To address these issues, some approaches, such as Herzog (2010), FIRST (2019) and Manadhata and Wing (2011) defined mathematical approaches providing quantitative measures of security. As detailed in Section 4, the mathematical framework defined by the OSSTMM (Herzog, 2010) constitutes the backbone of the DSS security evaluation process and elements of the CVSS methodology (FIRST, 2019) have been embedded in such procedure as well.

Concerning the identification of features allowing to characterise security, in the literature a certain consensus can be observed around specific aspects such as the need of considering the attackers resources, easiness in exploiting vulnerabilities and the components lifecycle (Common Criteria: New CC Portal, 2019; Manadhata and Wing, 2011; Scarfone et al., 2008; IEC, 2018).

It must be noted that the complexity of the problem to be solved grows exponentially with the number of available security functionalities (Floudas, 1995). In addition, safety and security related applications must envisage the presence of a human in the loop. In this respect, the solutions provided by a security management system should always be validated by the CI operator (a security expert). However, DSSs for security management are useful for providing a sound and complete security evaluation. First, due to the definition of a formal evaluation procedure, security evaluation and system configuration selection biases are reduced. Second, when the dimension of the system grows, the availability of a tool for exploring the solution space allows a significant reduction of errors guaranteeing that all the internal interactions and mutual influences are properly addressed.

With respect to the security evaluation problem, the automatic tools mentioned in Section 1 are able to suggest security configurations but only from a software point of view (e.g., in terms of update detection, misconfigurations or wrong settings). In Almeida and Respício (2018), the authors addressed the security configuration problem from an economic perspective integrating guidelines of the ISO/IEC 27001 in an optimisation framework for the selection of information security controls aimed at minimising the investment costs in security. However, the resulting DSS did not allow to take in consideration topological nor security constraints. Security management has been mostly addressed from a risk management perspective (e.g., ISO, 2018; Zhang et al., 2010) which implies knowing not only the likelihood of an adverse event to occur, but also to determine its impact on the CI. Although these solutions may work for specific systems, the impact's fortuity and likelihood definition impairs the consistency and repeatability of the security evaluation process thus preventing the establishment of a common ground which can be used by CI operators to compare results.

With respect to the above-mentioned issues, the objective of the present work consists in developing a DSS with the following features. First, the CI operator should be able to specify constraints from the security, economic, topological and functional points of view (see Sections 3 and 5). Second, the security configuration computation (SCC) problem should be driven by quantitative measures in order to set up a formal security evaluation framework (see Section 4). Third, the CI operator should be provided with a holistic characterisation of the security level also able to capture state of the art security information (see Subsections 4.2 and 4.3).

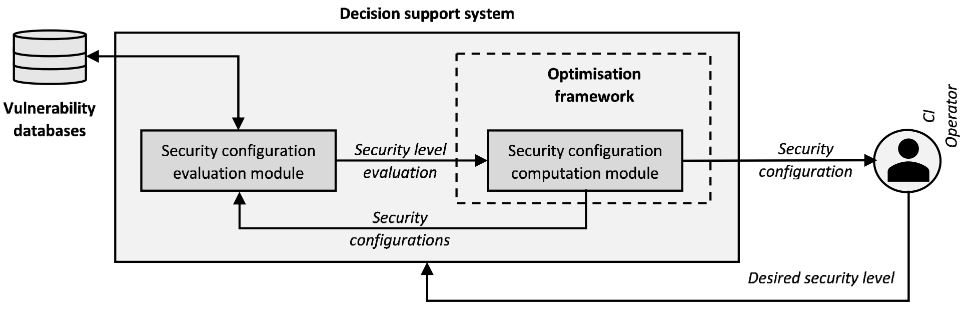
3 Framework overview

As anticipated, to provide CI operators with optimal security configurations, the DSS must address two specific issues. First, a methodology to evaluate the security level of system must be defined. Second, an optimisation framework for properly selecting the security functionalities must be setup. Following on these considerations, the internal architecture of the proposed DSS has been structured into two main blocks as depicted in Figure 1:

- the *security configuration evaluation* (SCE) module, in charge of characterising the security level of a given configuration
- the *SCC* module, in charge of determining feasible optimal configurations matching given security levels.

The CI operator specifies a desired security level in terms of proper security indicators (see Section 4) and interrogates the DSS for a security configuration matching such security level. In addition, the CI operator can also specify topological or functional constraints. At this point, the SCC module explores the solution space for finding candidate configurations which are fed to the SCE module for being evaluated. Based on the configuration evaluation, the SCC may continue in the search for feasible solutions or, if the target security level and the constraints are satisfied, return the computed configuration to the CI operator (see Section 5) who will then decide if accepting such configuration or not. Note that the DSS has been designed following the man-in-the-loop paradigm. This is a natural choice for safety and security related decision problems.

Figure 1 DSS internal architecture



The role of the SCE module is to provide the DSS with a mean to evaluate the security level of a given system. Such evaluation should be able to

1 capture and quantify relevant security features

2 combine these measures to obtain numerical values reflecting the security level of the system.

To solve this problem, the SCE module exploits the mathematical framework setup by the OSSTMM (Herzog, 2010) which provides a quantitative mean to characterise the security level of a given system (see Subsection 4.1). This is achieved by means of several security indicators computed by combining three elements: the system exposure to threats (porosity), the deployed protection mechanisms (controls) and the deficiencies of protection mechanisms as well as the problems in maintaining a separation between assets and threats (limitations). For evaluating and comparing security configurations, the SCE module uses innovative security indicators in addition to those defined by the OSSTMM. These additional indicators are obtained by extending the OSTTMM indicators (standard OSSTMM indicators) with the aims of capturing the impact of components' lifecycle on the system security level (see Subsection 4.2) and exploiting the knowledge of available vulnerability databases (see Subsection 4.3). The former aspect, as further detailed in Section 4, has been identified as crucial by several security standards and methodologies (e.g., IEC 62443 and ISO/IEC/IEEE 15288). On the other hand, the latter aspect allows to embed in the security evaluation procedure the possibility of exploiting the knowledge stored in vulnerability databases thus empowering, at a certain extent, the automatisation of the security evaluation procedure always guaranteeing updated information.

The configuration computation problem solved by the SCC module can be modelled as an integer optimisation problem. The decision variables can be selected as Booleans modelling the deployment, or not, of the available security functionalities. The objective function can be defined in terms of economic factors [as done in Almeida and Respício (2018)] or security aspects (e.g., in terms of the standard OSSTMM security indicators). In the present work, the latter option has been chosen and, in particular, the above-mentioned security indicators will be used aiming at associating real numbers to given security configurations. Due to the definition of these security indicators, the optimisation problem becomes nonlinear. For what concerns the constraints, the CI operator should be able to specify them in terms of security, economic, topological, and functional aspects. All these aspects put together render the problem NP-hard. Due to this complexity, the SCC module uses a GA for solving the nonlinear integer optimisation problem (see Section 5). The solutions adopted by the SCC and SCE modules to address the above-mentioned issues are detailed in Sections 4 and 5, respectively.

4 Security configuration evaluation

The security evaluation procedure adopted in the SCE module is based on the OSSTMM which will be briefly described in Subsection 4.1 [for a complete description the reader can refer to Herzog (2010)].

In Subsection 4.2, the OSSTMM methodology will be extended with two purposes. The first objective consists in including in the security evaluation process lifecycle aspects. The consideration of such aspect has been identified as crucial in several security standards and methodologies (e.g., IEC 62443 and ISO/IEC/IEEE 15288) and represents an important feature of the proposed DSS. Indeed, from the CI operator point of view, it is important to have an insight on the actual effectiveness of the security functionalities which may differ from the nominal one. Such mismatch may originate from unsecure value chain: the security functionalities implemented by a component whose lifecycle has not been verified and properly controlled may be not fully effective and may also introduce additional entry points that attackers can use to access the system. The second objective consists in exploiting the available knowledge coming from vulnerabilities databases such as the CVE Database (MITRE, 2019), the National Vulnerability Database (NVD) (National Institute of Standards and Technologies, 2019) and the CERT/CC Vulnerability Notes Database (CM University, 2019). By doing so, it is possible to properly address vulnerabilities, to avoid overspending on countermeasures and to have a continuously updated matching between vulnerabilities and countermeasures. In this work, the CVE database is retained as source of information and the vulnerabilities are accounted for by means of the Common Vulnerability Scoring System (CVSS) methodology whose scores are embedded in the OSSTMM methodology as detailed in Subsection 4.3.

4.1 OSSTMM methodology

The OSSTMM methodology is based on the concept that security is a measure of the separation degree between assets (to be protected) and threats. The output of the evaluation process consists of a set of security indicators which, as a whole, describe the security level of a given system. Such indicators are obtained by combining three elements: the system attack surface (referred to as porosity, see Subsection 4.1.1), the countermeasures (referred to as controls, see Subsection 4.1.2) and the vulnerabilities (referred to as limitations, see Subsection 4.1.3) reducing the effectiveness of countermeasures or increasing the system exposure.

4.1.1 Porosity

The porosity (or operational security, *OpSec*) as defined by the OSSTMM, provides a measure of the lack of separation between assets and threats which is needed for operational reasons and is defined as

$$OpSec_{\Sigma} = P_V + P_A + P_T \tag{1}$$

where P_A, referred to as access, is the number of points of the system from which an interaction with the external world may occur, P_T, referred to as trust, are the internal interactions needed for the system to be operative and P_V, referred to as visibility, are the known number of assets which are in danger of being targeted by attackers.

4.1.2 Controls

In the OSSTMM, all the possible controls have been casted into ten control types organised in two classes:

- The interactive controls impact on the system attack surface and include authentication, indemnification, resilience, subjugation, and continuity.
- The process controls are aimed at protecting an asset once that threats are present and include non-repudiation, confidentiality, privacy, integrity, and alarm.

For each of the ten control types, the number of deployed countermeasures is referred to as loss controls (LC_i). The total number of implemented countermeasures (LC_{sum}) is thus defined as the sum of all the loss controls:

$$LC_{sum} = \sum_{i=1}^{10} LC_i \tag{2}$$

For each of the ten control types, it is possible to compute the number of countermeasures to be implemented to address the lack of separation between assets and threats. By doing so, it is possible to define the missing controls (MC_i) as:

$$MC_i = \begin{cases} 0, & \text{if } OpSec_{\Sigma} - LC_i \leq 0 \\ OpSec_{\Sigma} - LC_i, & \text{else} \end{cases} \tag{3}$$

The total number of missing controls MC_{sum} is obtained by summing the missing controls for all control types.

The number of unprotected operations (i.e., the missing controls) can be put in relation with the operational security. By doing so, it is possible to define the missing coverage (*MCvg*) as

$$MCvg = \begin{cases} 0, & \text{if } OpSec_{\Sigma} \leq 0 \\ \dfrac{MC_{sum} \times 0.1}{OpSem_{sum}}, & \text{else} \end{cases} \tag{4}$$

The opposite of missing controls is true controls (TC_i) which account for the protected operations and, for each control type, are defined as

$$TC_i = OpSec_{\Sigma} - MC_i \tag{5}$$

The total number of missing controls TC_{sum} is obtained by summing the true controls for all control types. In analogy with what done with missing controls, the total number of true controls can be referred to the operational security so to obtain a percentage. By doing so, it is possible to define true coverage ($TCvg$) as

$$TCvg = \begin{cases} 0, & \text{if } OpSec_{\Sigma} \leq 0 \\ 1 - \dfrac{TC_{sum}}{10 \times OpSec_{sum}}, & \text{else} \end{cases} \tag{6}$$

True controls account for the protected operations for each control type. Full controls (FC), on the other hand, provide a measure of the countermeasures put in place regardless from their type and are defined as

$$FC_{base} = \log^2 \left(1 + 10 \times LC_{sum}\right) \tag{7}$$

4.1.3 Limitations

To capture the deficiencies of the protection mechanisms and the problems in maintaining a separation between assets and threats, the OSSTMM methodology defines five types of limitations: vulnerabilities (L_V), weaknesses (L_W), concerns (L_C), exposures (L_E) and anomalies (L_A). Each of them is individually weighted as shown in Herzog (2010). Table 1 shows how such weights are computed.

Table 1 Computation of limitations weights

	Weights
Vulnerabilities	$W_V = \dfrac{\left(OpSec_{\Sigma} + MC_{sum}\right)}{OpSec_{\Sigma}}$
Weaknesses	$W_W = \dfrac{\left(OpSec_{\Sigma} + MC_A\right)}{OpSec_{sum}}$
Concerns	$W_C = \dfrac{\left(OpSec_{\Sigma} + MC_B\right)}{OpSec_{\Sigma}}$
Exposures	$W_E = \dfrac{\left(\left(P_V + P_A\right) \times MCvg + L_V + L_W + L_C\right)}{OpSec_{\Sigma}}$
Anomalies	$W_A = \dfrac{\left(P_T \times MCvg + L_V + L_W + L_C\right)}{OpSec_{\Sigma}}$

Security limitations (*SecLim_{sum}*) accounts for all the Limitations types and is defined as

$$SecLim_{sum} = L_V \times W_V + L_W \times W_W + L_C \times W_C + L_E \times W_E + L_A \times W_A \tag{8}$$

4.1.4 Additional security indicators

Based on the three OSSTMM categories (porosity, controls and limitations) and related parameters (described in Subsections 4.1.1–4.1.3), the OSSTMM defines several security indicators for characterising the security level of a given system.

The first indicator which can be defined is the actual security delta ($ActSec\Delta$), which provides a measure of the countermeasures that should be put in place in order to balance the system exposure, is defined as

$$ActSec\Delta = FC_{base} - OpSec_{base} - SecLim_{base} \tag{9}$$

where the subscript base specifies that the values have been reported on a logarithmic scale.

A second relevant indicator that can be defined is the true protection (*TruPro*) which allows to characterise the actual coverage of the protection mechanisms with respect to all the control types and is defined as

$$TruPro = 100 + TC_{base} - OpSec_{base} - SecLim_{base} \tag{10}$$

The third indicator that can be defined, referred to as actual security (*ActSec*) provides a measure of the system security level taking in consideration the system exposure, the applied countermeasures and the discovered limitations (i.e., this indicator considers all the three OSSTMM categories) and is defined as

$$\begin{aligned} ActSec = 100 + ActSec\Delta - \frac{1}{100}\big(& OpSec_{base} \times FC_{base} - OpSec_{base} \\ & \times SecLim_{base} + FC_{base} \times SecLim_{base}\big) \end{aligned} \tag{11}$$

A value of 100 corresponds to a perfect balance between the OSSTMM categories, while values lower than 100 indicate that there are some not addressed security aspects. It should be noted that values higher than 100 are also possible and correspond to a situation in which there are more controls than necessary.

4.2 Inclusion of lifecycle aspects

As anticipated, one of the aims of the SCE module is to enhance the security evaluation capabilities of the proposed DSS by providing innovative quantitative measures of security features. To do so, the standard OSSTMM indicators can be extended for taking in consideration lifecycle aspects and, in particular, to provide the CI operators with a prediction on the security level evolution. The resulting security indicators will be referred to as extended OSSTMM indicators.

The consideration of the components' lifecycle is a critical aspect especially in the context of CIs which are characterised by the presence of many legacy components. Indeed, the security of a component, and thus the reliability of the protection mechanisms implemented by it, must be evaluated taking in consideration all the value chain, i.e., the design, production, delivery, and maintenance processes. These aspects characterise an intrinsic property of components which will be referred to as background lifecycle. In addition to these considerations it should be noted that unsecure value chains may add additional entry and exit points in the system and its effects are hidden in nominal operations and postponed in time. Indeed, the security level of a component decreases with time if no control actions are implemented since it is reasonable to assume that new vulnerabilities will be discovered through the years to come. This second aspect will be referred to as the forecast lifecycle.

In Subsections 4.2.1 and 4.2.2, these qualitative lifecycle descriptions will be translated into quantitative ones and it will be shown how can be embedded in the OSSTMM framework.

4.2.1 Background lifecycle

Background lifecycle refers to an intrinsic property of components and is defined based on the quality of its value chain. This concept will be used in order to discriminate between security functionalities. By doing so, CI operators will be provided with a characterisation of the CI security level which takes in consideration the fact that not all the deployed protection mechanisms have the same reliability degree. In view of embedding this concept in the OSSTMM framework, it is clear that the element that should be modified is represented by the loss controls (defined in Subsection 4.1.2) which accounts for the protection mechanisms put in place.

To derive a quantitative measure accounting for the background lifecycle, the qualitative measures provided by the common criteria are used as reference.

Common criteria define seven discrete levels, referred to as evaluation assurance levels (EALs), for evaluating the security of a component. To achieve a given EAL, a component must verify several security requirements, ordered in hierarchical way, which are organised in assurance classes each focusing on a different security level. The common criteria assurance class addressing lifecycle aspects is called lifecycle support class (referred to as ALC). The security requirements of the ALC class specify a set of actions which, if progressively verified, guarantee increasing security levels. In few words, the more requirements are satisfied, the higher is the guarantee that the component is secure. Within the ALC class, security requirements are organised in seven families each focusing on a different aspect (e.g., documentation of the value chain, techniques used during the design process, method of delivery). If all the requirements of all the families are verified, the component has the highest security level. It may happen that a component satisfies all the requirements of a family, but none of the requirements of other families. To characterise the security level of a component in such situation, let r_i be the number of verified requirements of the ith ALC family and R_i be the maximum number of requirements that are specified by the ith ALC family. Then, it is possible to derive a quantitative characterisation of the component lifecycle (γ_{LC}) as

$$\gamma_{LC} = \sum_{i=1}^{N} l_i \left(\frac{r_i}{R_i} \right) \tag{12}$$

where N is the number of considered families (in the current modelling $N = 7$) and l_i is the weight associated to each family. In the current modelling each family is considered equally important and thus $l_i = 1/N$.

The lifecycle parameter γ_{LC} can thus be used to characterise the reliability of the deployed protection mechanisms. To embed this aspect in the OSSTMM framework, it is possible to define the actual controls (*AC*) which are equal to the loss controls (as defined in Subsection 4.1.2) when considering the background lifecycle of components. ACs are computed for all the ten types of controls; by considering, for example, authentication controls, one has:

$$AC_{Au} = \sum_{j=1}^{LC_{Au}} \sum_{i=1}^{N} \frac{r_i}{R_i} \cdot l_i \tag{13}$$

with $j = 1, \ldots, LC_{Au}$ being the number of authentication controls. By substituting ACs in place of loss controls in the equations presented in Subsections 4.1.2–4.1.4 it is possible to define additional security indicators. As an example, by substituting ACs in equation (10), it is possible to define a modified version of true protection, referred to as actual true protection (*ATruPro*) allowing to provide a measure of the coverage of the deployed protection mechanisms with respect to the ten types of controls.

4.2.2 Forecast lifecycle

Forecast lifecycle refers to the decay of the security level induced by the presence of components with unsecure value chains. The underlying idea of the forecast lifecycle is that, since new vulnerabilities are constantly discovered, if a component is not updated or replaced, the effectiveness of its security functionalities will decay and, in addition, it may also introduce new vulnerabilities in the system.

The quantitative characterisation of this aspect has been performed on an empirical base by analysing statistics relative to the vulnerability discovery rate based on public and proprietary vulnerability databases [such as MITRE (2019) and National Institute of Standards and Technologies (2019), both sponsored by the US Department of Homeland Security[1], and VulnDB[2]]. From such analysis, it is possible to state that, on average, each year the number of newly discovered vulnerabilities sees a 15% increase. In other words, the effectiveness of a security configuration is almost halved after three years of operations without upgrades or system maintenance. Following on these considerations, it is possible to define an empirical coefficient for the forecast lifecycle as

$$FL(t) = FL(t-1) \cdot 0.85 \tag{14}$$

The forecast lifecycle coefficient $FL(t)$ is used by the DSS to provide CI operators with a prediction on the system security level evolution. In other words, the DSS provides a security configuration matching the target security level specified by the CI operator and an estimate of the moment in which such security level will be not guaranteed anymore. This is achieved by inducing the evolution described in equation (14) in the component lifecycle parameter as defined in equation (12).

4.3 Inclusion of vulnerabilities databases

As already mentioned, one of the features of the proposed DSS consists in extending the framework provided by the OSSTMM methodology to exploit the knowledge stored in vulnerability databases and, in particular, in the CVE database. To achieve this result, the OSSTMM Limitations can be weighted accordingly to the severity scores provided by the

National Institute of Security and Technologies (NIST) in compliance with the CVSS methodology. By doing so, the parameters L_V, L_W, L_C, L_E and L_A defined in Subsection 4.1.3 (accounting for the number of the five types of limitations) can be modified as follows

$$L_j' = 0.1 * \sum_{i=1}^{L_j} L_{jbase_i} \tag{15}$$

where L_{jbase_i} is the base CVSS severity score (MITRE, 2019) of the ith limitation of type L_j. By considering these new parameters it is possible to modify the security limitations indicator, as defined in equation (8). By doing so, it is possible to define adjusted security limitation (*AdjSecLim*) which is computed as

$$AdjSecLim_{sum} = L_V' \times W_V + L_W' \times W_W + L_C' \times W_C + L_E' \times W_E + L_A' \times W_A \tag{16}$$

By substituting *SecLim* with *AdjSecLim* in equations (9), (10) and (11), it is possible to derive modified security indicators which take in consideration the fact that not all the Limitations have the same impact on the system security level.

5 SCC and solutions ranking

In Section 4 the (quantitative) framework that the SCE module implements for evaluating security configurations has been introduced. In this section, the optimisation framework adopted by the DSS and implemented in the SCC module is presented.

The configuration computation problem requires to understand the optimal placement of security functionalities taking in consideration

1 logical and topological relations

2 economic, functional and security requirements.

In general terms, the underlying optimisation problem can be formulated as a binary optimisation problem in which the decisions variables x_j are equal to 1, if the jth security functionality is deployed and 0 otherwise. As anticipated, the proposed DSS is focused on security and thus the objective function is defined as to maximise the security indicators defined in Section 4 (more on the selected indicators in Section 0). It can be observed that such indicators are nonlinear in the decision variables and thus the optimisation problem is nonlinear as well.

The optimisation problem, as formulated, is in general NP-hard (Floudas, 1995) and thus the solution algorithm must be able to address this complexity. Following on these considerations, the SCC adopts a GA as it allows to determine high quality solutions in a relatively short amount of time. GAs are inspired to biological evolution processes and have been successfully applied to several search problems (Davis, 1991). The huge dimension of the solution space of the security configuration problem renders the adoption of such algorithms particularly suited.

In Subsection 5.1 the main characteristics of the adopted solution algorithm are introduced while in Subsection 5.2 the mathematical formulation of the problem is presented.

5.1 Genetic algorithms

GAs are heuristic search algorithms which allow to obtain high quality solutions.

To model natural selection processes, the possible solutions of a problem are seen as individuals of a population. GAs combine two or more solutions (referred to as chromosomes or parents) to generate a new offspring which will inherit some properties (genes) of the parents. To foster diversity in the new offspring, GAs are also able to simulate the mutation process which is a key aspect of evolution processes. That is, GAs can be characterised in terms of the selection operator (specifying the rules for parent selection), the crossover operator (specifying how to combine parents' genes) and the mutation operator (specifying in which measure the new offspring is different from the parents). The function used by the selection operator to evaluate individuals is referred to as fitness function and the goal of the induced population evolution (i.e., the application of the three above mentioned operators) is to identify the individuals with highest fitness value.

In the present work the fitness function is represented by the security indicators defined in Section 4. For what concerns the selection, crossover and mutation operators, several choices are possible (e.g., tournament and truncation selection, single/multi-point crossover, uniform/non-uniform crossover). The solutions adopted are detailed in Subsection 5.2 and in more in detail in Section 0.

5.2 Problem formulation

As anticipated, the configuration computation problem solved by the DSS can be formulated as an integer nonlinear optimisation problem:

$$\max_{x \in X} F(x) \quad \text{s.t.} \quad \begin{array}{l} F(x) < l_b \\ g(x) < 0 \end{array} \tag{17}$$

where $X = \{0, 1\}^N$ is the vector of decision variables defined as $X = \{x_j{:}x_j = 1$ if the j^{th} security functionality is deployed, $x_j = 0$ otherwise$\}$, $F(x)$: $X \to \mathbb{R}$ is the objective function (i.e., the fitness function) associating a real number to a given security configuration according to the selected security metric (see Section 4), $l_b \in \mathbb{R}$ is the target security level expressed according to the selected security metric and $g(x)$ is a function modelling security, topological, functional and economic constraints.

That is, a vector $x \in X$ represents a generic security configuration or, by adopting the terminology introduced in the previous section, an individual of the population. The GA solution algorithm, by simulating an evolutionary process, iteratively computes populations. Hereinafter, with j and m the j^{th} component of vector $x \in X$ (i.e., the j^{th} security functionality) and the m^{th} population generated by the solution algorithm will be denoted.

Given the formulation (17), in order to apply a GA, it is necessary to manipulate the objective function since GAs does not allow to manage constraints. Among the several solutions present in the literature (e.g., Ponsich et al., 2008), the one adopted in the present work is based on what done in Deb (2000) and consists in augmenting the objective function $F(x)$ in the following way:

$$F_m(x) = \begin{cases} F(x), & \text{if } x \text{ is feasible} \\ f_m^w + \sum_{i=1}^{L} D_i(x), & \text{otherwise} \end{cases} \tag{18}$$

where m denotes the current generation, f_m^w is the worst value among the fitness function values associated to the individuals of the current generation, L is the number of constraints and $D_i(x)$ is an increasing penalty function associated to the ith constraint. The augmented objective function $F_m(x)$ as defined in equation (18) guarantees that

- any feasible solution is preferred with respect to unfeasible solutions
- among two feasible solutions, the one having better objective function value is preferred
- among two unfeasible solutions, the one having smaller constraint violation is preferred.

In order to foster a good exploration of the solution space, it is possible to act on the mutation operator. However, this reduces the correlation between parents and their offspring thus increasing the risk of losing good aspects (parents' genes) in terms of solution quality and constraints violation. As further described in Section 0, the proposed DSS resolves this trade off by adopting a dynamic mutation allowing to combine good exploration capabilities while reducing the risk of slowing convergence. The details of the adopted GA are detailed in Section 0.

6 Simulations

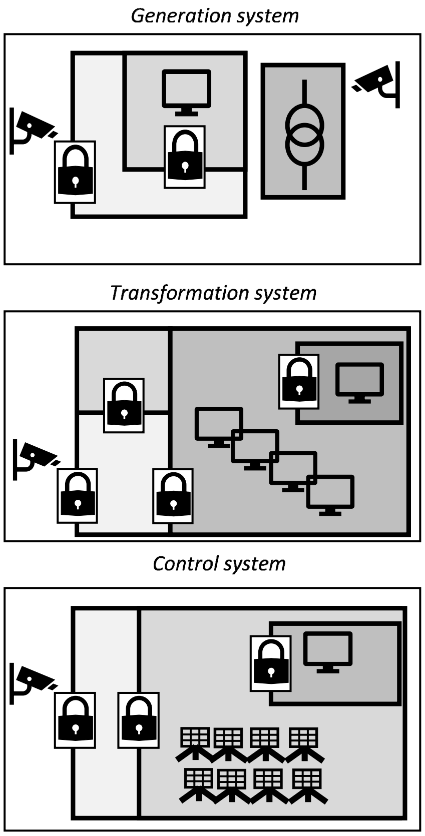
In this section the developed DSS is tested in a realistic operational scenario: a photovoltaic generation system. The objective of the simulations is to show that the security configuration suggested by the DSS is compliant with commonly adopted best practices. In addition, the performance of three different implementations of the solution algorithm are discussed.

The goal of the DSS is to compute a security configuration (i.e., deciding which security functionalities have to be deployed) in order to match a given security level. In other words, the DSS will suggest if given assets providing protection (such as cameras and firewalls) should be deployed and/or if given security functionalities (such as authentication mechanisms and data encryption) should be activated. That is, the DSS allows to match the target security level both by acting on the system topology and by deciding which functionality should be activated.

The application scenario, depicted in Figure 2, consists in a photovoltaic electric generation system and is characterised by three subsystems: the field where photovoltaic panels are deployed, the control room and the transformation system.

The generation system hosts photovoltaic panels and an RTU used to regulate their movement. The communication with the control room occurs via fibre and/or via a wireless communication system. The choice of the communication system to implement in the system configuration is demanded to the DSS for then being validated by the CI operator. In other words, it has been assumed that the CI operator specifies a topological constraint requiring that at least one between the fibre and wireless communication systems must be operative. In this respect, best practices specify that infrastructure redundancy should be implemented in order to increase the resiliency of the system. This means that the DSS is expected to select both communication systems in order to be able to deal with failures of one communication channel. In addition, if only one communication system is selected, the optimal choice should be the fibre since is less prone to attacks. Beyond the choice of the communication system, the DSS should suggest about the deployment of IP cameras to monitor the access to the system. It should be noted that introducing an asset may increase the system attack surface since it may introduce additional entry and exit points to the system. This translates in the fact that deploying all the cameras is not an optimal choice. In this respect, in the simulations it has been assumed that the CI operator has specified a minimum and maximum number of cameras. The DSS is also in charge of deciding, for each deployed asset, which security functionalities should be activated. For example, the doors may be equipped with fingerprint and/or badge readers, with an alarm mechanism for preventing their tampering and so on.

Figure 2 Schematisation of application scenario



The control room hosts the SCADA servers and several HMI positions for the operators. The communication with the transformation system occurs through optical fibre. Again, the DSS should suggest about the deployment of security camera and several other security functionalities and assets providing protection. Among them there is a firewall to restrict the access to the network of the control system and a monitoring system enabling incident response. It should be noted that from a security point of view, the segregation of networks (i.e., the deployment of the firewall) and the infrastructure monitoring are highly suggested to reduce the system attack surface. Following on these considerations, the DSS should suggest deploying both.

The transformation system is responsible of the power conversion process and comprises high-voltage switches, transformers and an RTU to control them. Among the assets that can be deployed to provide protection there are security cameras and network monitoring systems.

6.1 Algorithm

As anticipated, three different implementations of the solution algorithm will be tested. More in detail, the performance of a GA with different crossover operators will be discussed (see Subsection 5.1). Since GAs are heuristics, different simulation on the same scenario may provide different sets of solution. The performance evaluation will be thus performed considering average behaviours for the different implementations.

The optimisation problem underlying the scenario described in the previous section involves almost one hundred of decision variables. The security indicator that will be maximised is the actual security [as defined in equation (11) with the extensions presented in Subsections 4.2 and 4.3].

As anticipated, the three implementations of the solution algorithm (whose pseudo code is reported in Table 2) differ for the crossover operator.

That is, each new population consists of three groups of individuals. The first group are the best individuals among the parents (elitism-based selection); the second group consists of individuals obtained by combining parents (crossover children); the third group consists of mutations of the crossover children (fostering the exploration of the solution space). In the simulations the portion of individuals of the first group is kept constant. For the remaining groups, instead, the number of individuals varies accordingly to three different crossover fractions functions:

- in the first case (linear function) the crossover fraction value starts from a 0 (meaning that all the initial offspring is generated with mutations) and increases linearly until the value of 1 (meaning that that all the last offspring consists of crossover children)
- in the second case (quadratic concave function) the crossover fraction value starts from 0, increases until 1 and then decreases until a value of 0.7
- in the third case (quadratic convex function) the crossover fraction value starts from a value of 1, then decreases until 0 and finally increases until the value of 0.3.

The behaviour of the GA implementing the three above-mentioned crossover fraction functions are in Table 2.

In the first case (linear behaviour of the crossover fraction function), the exploration of the solution space is fostered in the first generations by the low value of the crossover fraction; when the value of the crossover fraction increases, the correlation between children and parents increases as well in order to preserve the good aspects coming from the past generations.

Table 2 Pseudo code of the solution algorithm

Genetic algorithm pseudocode

```
        Input: instance $G$
        Size $P$ of population,
        Rate $E$ of elitism,
        Rate $\gamma$ of mutation,
        Crossover fraction $C(I)$;
        Number $I$ of iterations
        Output: solution
// Initialisation
1       Generate $P$ solutions randomly;
2       save them in the population $Pop$;
// Loop until the terminal condition
3       for $i = 1$ to $I$ do
// Elitism-based selection
4       number of elitism $ne = P * E$;
// Crossover
5       number of crossovers $nc = (P - ne) * C(i)$
6       for $j = 1$ to $nc$ do
7       randomly select two solutions $X_A$ and $X_B$ from $Pop$;
8       generate $X_C$ and $X_D$ by crossover operator to $X_A$ and $X_B$;
9       save $X_C$ and $X_D$ to $Pop_2$;
10      endfor
11      number of mutations $nm = P - ne - nc$
// Mutation
12      For $j = 1$ to $nm$
13      select a solution $X_j$ from $Pop_2$;
14      mutate each bit of $X_j$ under the rate $\gamma$ and generate a new solution $X_j'$;
// Updating
15      update $Pop = Pop_1 + Pop_2$;
16      endfor
// Returning the best solution
17      return the best solution $X$ in $Pop$;
```

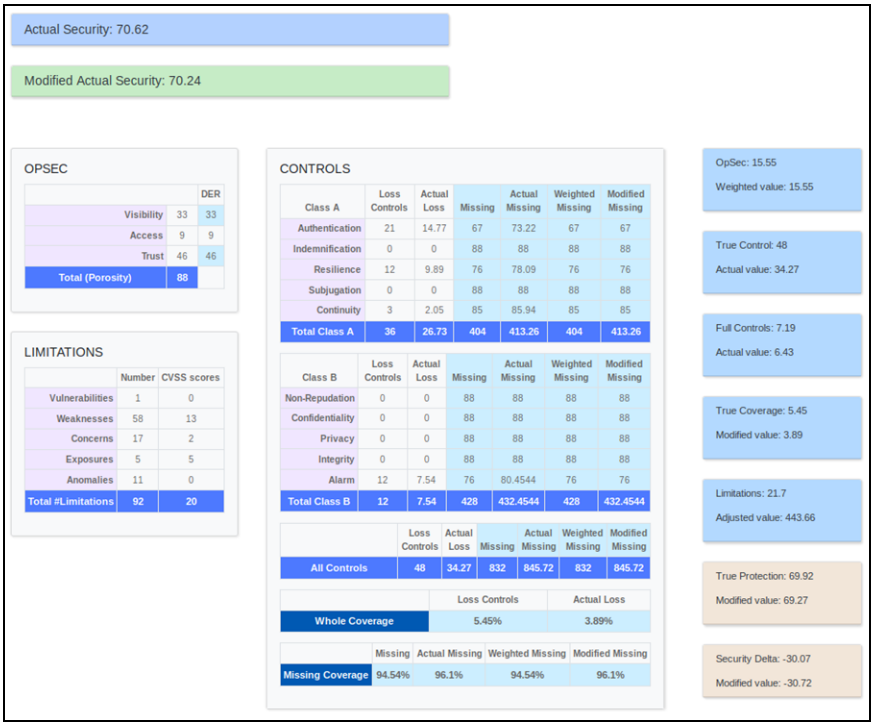
In the second case (quadratic concave behaviour of the crossover fraction function) the behaviour is similar to the linear case; the difference is that in the last populations the portion of mutations increases in order to allow an additional exploration phase starting from a new initial area in the configuration space potentially better than the initial one.

In the third case, (quadratic convex behaviour of the crossover fraction function) the behaviour is the opposite with respect to the quadratic concave case: correlation between children and parents is high in the first generations, then decreases to foster the exploration of the solution space and increases again to refine the last populations.

6.2 Results and comments

The simulations performed proved that in all its three implementations, the DSS was able to compute security configurations satisfying the target security level (expressed in terms of the extended actual security and set at 68). However, several differences can be highlighted.

Figure 3 Security Indicators as provided by the DSS to the CI operator (quadratic concave case) (see online version for colours)



Actual Security: 70.62

Modified Actual Security: 70.24

OPSEC

		DER
Visibility	33	33
Access	9	9
Trust	46	46
Total (Porosity)	88	

LIMITATIONS

	Number	CVSS scores
Vulnerabilities	1	0
Weaknesses	58	13
Concerns	17	2
Exposures	5	5
Anomalies	11	0
Total #Limitations	92	20

CONTROLS

Class A	Loss Controls	Actual Loss	Missing	Actual Missing	Weighted Missing	Modified Missing
Authentication	21	14.77	67	73.22	67	67
Indemnification	0	0	88	88	88	88
Resilience	12	9.89	76	78.09	76	76
Subjugation	0	0	88	88	88	88
Continuity	3	2.05	85	85.94	85	85
Total Class A	36	26.73	404	413.26	404	413.26

Class B	Loss Controls	Actual Loss	Missing	Actual Missing	Weighted Missing	Modified Missing
Non-Repudation	0	0	88	88	88	88
Confidentiality	0	0	88	88	88	88
Privacy	0	0	88	88	88	88
Integrity	0	0	88	88	88	88
Alarm	12	7.54	76	80.4544	76	76
Total Class B	12	7.54	428	432.4544	428	432.4544

	Loss Controls	Actual Loss	Missing	Actual Missing	Weighted Missing	Modified Missing
All Controls	48	34.27	832	845.72	832	845.72

	Loss Controls	Actual Loss
Whole Coverage	5.45%	3.89%

	Missing	Actual Missing	Weighted Missing	Modified Missing
Missing Coverage	94.54%	96.1%	94.54%	96.1%

OpSec: 15.55
Weighted value: 15.55

True Control: 48
Actual value: 34.27

Full Controls: 7.19
Actual value: 6.43

True Coverage: 5.45
Modified value: 3.89

Limitations: 21.7
Adjusted value: 443.66

True Protection: 69.92
Modified value: 69.27

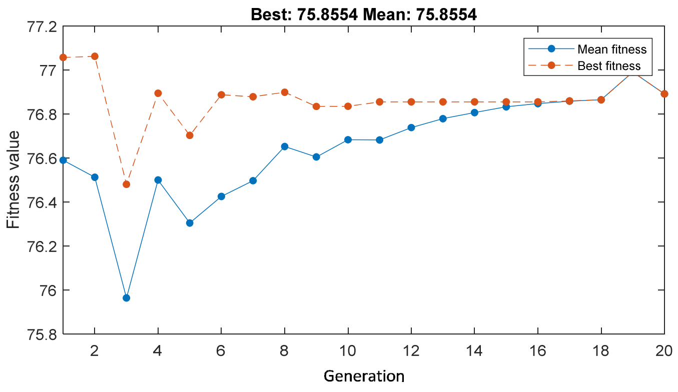
Security Delta: -30.07
Modified value: -30.72

On the average, the DSS implementing a linear crossover fraction function showed a lower security level (69.81) with respect to the quadratic concave (70.24) and quadratic convex (70.64) cases. Figure 3 reports the security indicators as provided by the DSS to the CI operator in the quadratic concave case. In all three cases, the DSS suggested to implement monitoring systems as expected (see Section 0). Furthermore, in the linear case both communication systems have been deployed, while in the other cases only the optical fibre. It should be noted that also deploying the wireless communication system increases the system resilience but also its attack surface (which is reflected by a lower security level). For what concerns the deployment of the monitoring systems, in all three cases they have been deployed. For what concerns constraints, in all three cases topological constraints are not violated. Simulations showed that in the linear and quadratic convex cases, constraints concerning the number of deployed security cameras are violated in an acceptable measure (they deploy, in general, one camera more or less than the requirements). In the quadratic concave case, instead, constraint violations also involve inconsistency of deployed assets (some of the deployed assets can be removed without affecting the system functionalities).

The most interesting differences between the different implementations can be observed between the quadratic concave and convex cases. In the concave case, the predominance of mutations in the first generations makes the selection of feasible solutions difficult; this approach fosters the search of the solution space but increases the risk of founding non-feasible solutions. In the convex case, the initial generations are characterised by low mutations thus fostering correlation between the first population and its offspring.

In general terms, the configurations provided by the DSS are all compliant with the best practices, highlighted in Section 0, in terms of network segregation, system monitoring and security of the communication system (fibre is always chosen against wireless).

Figure 4 Population evolution example (convex case) (see online version for colours)



7 Conclusions

A DSS for suggesting optimal security configurations to CI operators has been described. To achieve this result, two main issues have been addressed: the security evaluation problem (Section 4) and the security configuration problem (Section 5). The solutions of the proposed DSS proved to satisfy the target security level specified by the CI operator and to be compliant with the best practices in terms of security aspects. More in detail, the security configurations are computed by the DSS by exploiting

1 an extended version of the OSSTMM allowing to take in consideration relevant security features (such as the components' lifecycle) and exploiting the knowledge stored in constantly updated vulnerability databases (Subsections 4.2 and 4.3)

2 an optimisation framework based on GAs.

As a matter of fact, the proposed DSS can be adopted to prevent and mitigate risks for CIs. Indeed, the DSS can be triggered by a CI operator for

1 deciding about which protection mechanisms should be bought in order to match a given security level

2 understanding the current posture of the CI and compute a security configuration matching the desired security level based on the available protection mechanisms.

In addition, as discussed in Section 6, the DSS is able to suggest new configurations both in terms of system topology and deployed protection mechanisms.

The authors are currently working on a refinement of the security evaluation process with the aim of extending its description capabilities in line with the recommendations of recognised security standards and institutions. Future works will also investigate and compare the performances of different families and implementations of solution algorithms.

Acknowledgements

This work has been carried out in the framework of the ATENA project which has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 700581. This work reflects only the author's view. The EU Commission and the Research Executive Agency are not responsible for any use that may be made of the information it contains.

The authors would like to thank the partners involved in the H2020 ATENA project (Aubigny, 2019), and in particular the researchers working in the CRAT Team, for their precious help in the development of the present and the previous related works (e.g., Panfili et al., 2018; Giuseppi et al., 2019).

A special acknowledgment goes to Pete Herzog, the creator of the OSSTMM methodology, which provided useful comments that have been taken in consideration in the development of the present work and in the rest of the related activities of the H2020 ATENA project.

Finally, the authors would like to thank the reviewers for their precious comments which helped improving the quality of the present work.

References

Almeida, L. and Respício, A. (2018) 'Decision support for selecting information security controls', *Journal of Decision Systems*, Vol. 27, No. 1, pp.173–180, DOI: 10.1080/12460125.2018.1468177.

Aubigny, M. (2019) *ATENA – Keep your Eyes on your Infrastructure*, A European H2020 Project – ATENA [online] https://www.atena-h2020.eu/ (accessed 17 June 2019).

BeyondTrust (2019) *Enterprise Vulnerability Management – Find Network Security Threats and Software Vulnerabilities* [online] https://www.beyondtrust.com/vulnerability-management (accessed 13 June 2019).

CM University (2019) *Vulnerability Notes Database*, CERT Coordination Center [online] https://www.kb.cert.org/vuls/ (accessed 13 June 2019).

Common Criteria: New CC Portal (2019) [online] https://commoncriteriaportal.org/ (accessed 17 June 2019).

Davis, L. (1991) *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, DOI: 10.1016/S0004-3702(98)00016-2.

Deb, K. (2000) 'An efficient constraint handling method for genetic algorithms', *Computer Methods in Applied Mechanics and Engineering*, Vol. 186, Nos. 2–4, pp.311–338, DOI: 10.1016/S0045-7825(99)00389-8.

FIRST (2019) *Common Vulnerability Scoring System SIG*, FIRST – Forum of Incident Response and Security Teams [online] https://www.first.org/cvss/ (accessed 14 June 2019).

Floudas, C.A. (1995) *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Oxford University Press, DOI: 10.1023/A:1008256302713.

Gadyatsaya, O., Jhawar, R., Kordy, P., Lounis, K., Mauw, S. and Trujillo-Rausa, R. (2016) 'Attack trees for practical security assessment: ranking of attack scenarios with ADTool 2.0', in Agha, G. and Van Houdt, B. (Eds.): *Quantitative Evaluation Systems (QEST)*, *Lecture Notes in Computer Science*, Springer, Vol. 9862, pp.159–162, DOI: 10.1007/978-3-319-43425-4_10.

Giuseppi, A., Tortorelli, A., Germanà, R., Liberati, F. and Fiaschetti, A. (2019) 'Securing cyber-physical systems: an optimization framework based on OSSTMM and genetic algorithms', in *2019 27th Mediterranean Conference on Control and Automation (MED)*, Akko, Israel.

Greenbone Networks (2019) *OpenVAS – Open Vulnerability Assessment System* [online] Available: http://openvas.org/ (accessed 17 June 2019).

Herzog, P. (2010) *OSSTMM 3 – The Open Source Security Testing Methodology Manual*, Institute for Security and Open Methodologies (ISECOM).

IEC (2018) *IEC – 62443 Industrial Network and System Security*, International Electrotechnical Commission.

International Organization for Standardization (ISO) (2018) *ISO 31000 – Risk Management*, International Organization for Standardization (ISO).

Johnson, P., Vernotte, A., Ekstedt, M. and Lagerstrom, R. (2016) 'pwnPr3d: an attack-graph-driven probabilistic threat-modeling approach', *11th International Conference on Availability, Reliability and Security*, Vol. 283, p.278, DOI: 10.1109/ARES.2016.77.

Kumar, R., Schivo, S., Ruijters, E., Yildiz, B.M., Huistra, D., Brandt, J., Rensink, A. and Stoelinga, M. (2018) 'Effective analysis of attack trees: a model-driven approach', in Russo, A. and Schürr, A. (Eds.): *Fundamental Approaches to Software Engineering, FASE 2018, Lecture Notes in Computer Science*, Springer, Cham, pp.56–73, DOI: 10.1007/978-3-319-89363-1_4.

Lye, K-w. and Wing, J.M. (2005) 'Game strategies in network security', *International Journal of Information Security*, Vol. 4, pp.71–86, DOI: 10.1007/s10207-004-0060-x.

Manadhata, P.K. and Wing, J.M. (2011) 'An attack surface metric', *IEEE Transactions on Software Engineering*, Vol. 37, No. 3, pp.317–386, DOI: 10.1109/TSE.2010.60.

MITRE (2019) *CVE – Common Vulnerabilities and Exposures (CVE)* [online] https://cve.mitre.org/ (accessed 14 June 2019).

National Institute of Standards and Technologies (2019) *NVD – Home* [online] https://nvd.nist.gov/ (accessed 15 June 2019).

Panfili, M., Giuseppi, A., Fiaschetti, A., Al-Jibreen, H.B., Pietrabissa, A. and Delli Priscoli, F. (2018) 'A game-theoretical approach to cyber-security of critical infrastructures based on multi-agent reinforcement learning', in *2018 26th Mediterranean Conference on Control and Automation (MED)*, Zadar, Croatia.

Pasqualetti, F., Dorfler, F. and Bullo, F. (2013) 'Attack detection and identification in cyber-physical systems', *IEEE Transactions on Automatic Control*, November, Vol. 58, No. 11, pp.2715–2729, DOI: 10.1109/TAC.2013.2266831.

Ponsich, A., Azzaro-Pantel, C., Domenech, S. and Pibouleau, L. (2008) 'Constraint handling strategies in genetic algorithms application to optimal batch plant design', *Chemical Engineering and Processing: Process Intensification*, Vol. 47, No. 3, pp.420–434, DOI: 10.1016/j.cep.2007.01.020.

Rapid7 (2019) *Top Rated Vulnerability Scanner: Rapid7 InsightVM* [online] https://www.rapid7.com/products/insightvm/ (accessed 13 June 2019).

Scarfone, K., Souppaya, M., Cody, A. and Orebaugh, A. (2008) *Special Publication 800-115 Technical Guide to Information Security Testing and Assessment*, National Institute of Standards and Technology (NIST), Gaithersburg, MD 20899-8930, September.

SWASCAN (2019) *SWASCAN – The First Cloud Cyber Security Platform* [online] https://www.swascan.com/it/ (accessed 13 June 2019).

Teixeira, A., Pérez, D., Sandberg, H. and Johansson, K.H. (2012) 'Attack models and scenarios for networked control systems', in *HiCoNS'12 Proceedings of the 1st International Conference on High Confidence Networked Systems*, Beijing, China, ACM, pp.55–64, DOI: 10.1145/2185505.2185515.

Zhang, X., Wuwong, N., Li, H. and Zhang, X. (2010) 'Information security risk management framework for the cloud computing environments', in *2010 10th IEEE International Conference on Computer and Information Technology*, Bradford, UK, DOI: 10.1109/CIT.2010.501.

Notes

1 https://www.dhs.gov/.

2 https://vulndb.cyberriskanalytics.com/.