Journal of Information Assurance and Security.
ISSN 1554-1010 Volume 6 (2011) pp. XXX–XXX

Dealing with the security behaviour of large scale systems

Marco Benini[1] **and Sabrina Sicari**[2]

[1]University of Leeds, Department of Pure Mathematics,
Woodhouse Lane, Leeds, LS2 9JT, UK
M.Benini@leeds.ac.uk

[2]Università degli Studi dell'Insubria, BICOM,
via Mazzini 5, IT-21100, Varese, Italy
sabrina.sicari@uninsubria.it

Abstract: **This paper describes a risk assessment method suited for large systems. In essence, the method has been introduced in previous works [5, 8, 9, 10, 27], where its properties have been analysed. In this paper, we develop and mathematically justify a variant of it which allows to divide a large system into overlapping subsystems, each one analysed by an expert. We will show that, independently from the division strategy, there is an effective way to combine the experts' assessments into a global picture, as far as a few and natural hypotheses on the metrics are satisfied.**
Keywords: Risk assessment, mathematical model, large systems

I. Introduction

Understanding the security behaviour of systems is a fundamental piece of knowledge to prevent malicious attacks. In this respect, risk assessment is a fundamental step to systematically analyse the security state of systems. For instance, risk assessment measures the possible negative impacts of an undesired event in a given system. The key words in the previous definition are 'measures' and 'possible': the word 'measures' suggests that an engineering method is needed to quantitatively evaluate the possible occurrences of an event; the word 'possible' suggests that what has to be measured is not a specific event but, instead, its ability to occur and, eventually, the consequences of its occurrence.

There are many ways to perform a risk assessment and many methods have been proposed: a succinct survey is presented in Section VIII. Each method has its own peculiarities that make it more apt to evaluate the risk of a specific class of threats or more suitable to some class of systems.

Roughly speaking, risk assessment methods can be divided into two main groups: the empirical methods, usually derived from a formalisation of best practices, and the theoretical ones, justified by a formal model of some sort. In everyday practice, the first group is preferred since its methods provide reasonable risk evaluations although on an empirical basis: usually, the outcomes of the application of these methods are hard to justify in a scientific sense because they are based on encoded experience.

On the contrary, the second group justifies its outcomes and provides an insight on the origin and the nature of the analysed risk sources: most of the times, these methods start from the evaluations by some experts and their goal is to combine and refine these initial assessments into an outcome not strictly depending on the experts' professional reliability .

We believe that a good risk assessment method should be both practical and theoretically sound, that is, it should justify its outcomes by a scientific argument. For these reasons, we proposed in [5] a risk assessment method that is based on a strict mathematical model: we have been able to prove some properties of the method that are considered useful in practice, like the independence from the metrics and the ability to combine evaluations from different experts as far as their metrics are compatible.

More specifically, during the last years we have analysed in depth the proposed method [27, 5] showing by a theoretical approach the main properties and the combination of expert evaluations [8, 9, 10], but now, trying to apply the method to real, large systems a new problem appears: how to efficiently handle the enormous number of vulnerabilities and dependencies among them.

In this paper, starting from the risk analysis method and related properties, we propose a simple but theoretically well founded solution which allows to apply our method also to large systems. The proposed approach divides a large system under analysis in subsystems which may overlap. So, a single expert may handle just a single subsystem at time.

We will prove, under suitable hypotheses, which are non-restrictive in practice, that a risk assessment of the whole system can be obtained by combining the assessments on subsystems, and that the resulting risk evaluation is unaffected by the division of the whole system into subparts.

In particular, we discuss three different procedures all leading to the same result, thus proving the correctness of their final outcomes. We will show that one of those procedures is computationally more efficient than the others, while another procedure provides a good balance between efficiency and extra-information which can be effectively used to better understand the risk analysis and, thus, to make wise security-

related decisions.
The paper is organised as follows: Section II introduces the notion of metric we adopt; Section III describes the risk assessment procedure in general, along with its main properties. The material in these sections is a summary of our previous findings [27, 5, 8, 9, 10], while what follows is novel.
In Section IV, we introduce the mathematical machinery which allows to derive the properties of the risk assessment method when considering subsystems: it amounts to prove a minimality result. Section V applies the results to large systems, describing two obvious procedures one may adopt to cope with division into subsystems. A third procedure is derived which is reasonably efficient and more suited to provide a better picture of the risk posture of the system.
In Section VI we will show the behaviour of the previously defined procedures on a simple but critical example. A discussion on the consequences of the analysis of the example is presented in Section VII. A survey of related literature concludes the paper.

II. Dealing with Metrics

A metric is a set of values, not necessarily numbers, used to measure an homogeneous class of observables. In the case of risk analysis, the values in a metric form an ordered set, and the relations are equality and less-than-or-equal ($\leq$). Formally, $\leq$ is required to be an order relation, i.e., reflexive ($x \leq x$), transitive ($x \leq y$ and $y \leq z$ implies $x \leq z$) and anti-symmetric ($x \leq y$ and $y \leq x$ implies $x = y$). The $\leq$ relation may be partial, that is, not defined for every pair x, y of values.
The order relation naturally generates a few operations implicitly assumed as given. These operations are the maximum and the minimum of a subset of values or, in the general case of partial or infinite orders, the least upper bound (*lub*, for short) and the greatest lower bound (*glb*).
Defining a metric just as a partially ordered set prevents the development of risk assessment procedures. In fact, risk is intended to be the worst outcome of an attack to a system, thus the need to calculate the lub of a set of values each one measuring the risk of a single outcome. Also, leveraging among risk evaluations requires to compute a glb.
But, to ensure that the lub and the glb always exist, the metric must form a complete lattice[1].
Definition II.1 *A* lattice *is a partial order $\langle \mathcal{O}, \leq \rangle$ such that every pair $x, y \in \mathcal{O}$ has a lub, denoted as $x \vee y$, and a glb, denoted as $x \wedge y$.*
A lattice is finite *if $\mathcal{O}$ is a finite set.*
Let $U \subseteq \mathcal{O}$ be non-empty, then $\bigvee U$ and $\bigwedge U$ are, respectively, the lub and the glb of the elements in U, if they exist.
A lattice is complete *if every non-empty subset $U \subseteq \mathcal{O}$ has a lub and a glb.*
In practice, we are mainly interested in finite lattices. In fact, the set of values used in a risk analysis is always finite: if numbers are used, they have a fixed amount of significant digits: if probabilities are used, only a few decimal places are considered; if qualifiers, like 'easy' or 'difficult', are used, there is a finite and fixed number of them.
Moreover, we are interested in lattices having two special values, $\bot$ and $\top$, denoting the impossibility to break the system and the immediate ability to abuse of a completely compromised system, respectively. In a metric based on numbers or probabilities, $\bot$ denotes the minimal value and $\top$ the maximal value, while, operating with qualifiers, we assume the existence of two appropriate values.
Definition II.2 *A lattice $\langle \mathcal{O}, \leq \rangle$ is* bounded *if there are two elements $\bot$ and $\top$ such that $\bot = \bigwedge \mathcal{O}$, i.e., every element is greater than $\bot$, and $\top = \bigvee \mathcal{O}$, i.e., every element is less than $\top$. In a bounded lattice, $\bigvee \emptyset = \bot$ and $\bigwedge \emptyset = \top$.*
Proposition II.3 *A finite lattice is bounded if and only if it is complete.*
Definition II.4 *A* metric *is a finite, bounded lattice.*

III. Modelling an attack

The goal of risk assessment is to determine the likelihood that the identifiable threats of a system will harm, weighting their occurrence with the damage they may cause. A risk assessment method is a procedure to define the risk of the occurrence of one or more threats.
The starting point is to consider a system as a composition of communicating black-box elements; a link (c_1, c_2) between the components c_1 and c_2 means that c_1 may directly influence c_2. Thus, the architecture of the system is modelled by the graph $\mathcal{A} = \langle C, L \rangle$ where C is the set of components and L is the set of links. Moreover, each component or link is assumed to be vulnerable: a vulnerability is a flaw or weakness in the design, implementation or management of a system or component that could be used to violate the security policy, as defined in [26].
The vulnerabilities are organised in a structure showing how they can be used to perform an attack, thought of as a goal to achieve. By recursively dividing each goal into sub-goals, a complex attack can be analysed. The resulting analysis provides a hierarchical plan to perform the attack. This approach is the one of *attack trees* [19, 23], a well-known and widely-adopted method to describe attacks as goals to threaten a system: the attacks are naturally represented in a tree structure, with the main goal as the root node and the different ways of achieving it as children. In turn, each internal node represents an intermediate goal. There are `and` nodes and `or` nodes, each one representing an immediate sub-goal of the father node: `or` nodes are alternative ways to achieve the father goal; `and` nodes represent the steps (ordered from left to right) toward the achievement of the father goal; the leaves of the tree represent the system vulnerabilities.
Thus, fixed a metric, the simplest risk assessment method using attack trees is described as follows:

1. The threats to the system under examination are modelled using attack trees and to each vulnerability v is associated a value $\varepsilon(v)$, called *exploitability*, that measures the difficulty to abuse of v and to perform a successful attack.

2. The risk associated to the threat under examination is computed by recursively aggregating the exploitabilities along the attack tree: the exploitability of an `or` subtree is the lub the exploitabilities of its children, and

[1] The theory of lattices is standard. Our treatment is limited to the definitions and the properties of interest in the paper. The interested reader is referred to [17] for a detailed presentation of the mathematical aspects.

the exploitability of an `and` sub-tree is the glb of the exploitabilities of its children.

The aggregated exploitability of the root node measures the feasibility of the attack. Since an attack tree is a finite object, the calculation of the aggregated exploitability terminates with the number of tree nodes as a bound to the number of steps.
The simple method just described assumes no further knowledge on the system than the exploitabilities of the leaves in the attack tree. This method is perfectly adequate when it is possible to evaluate each vulnerability in isolation, as if it does not interfere with the other vulnerabilities. Also, this method implicitly assumes that the experts trying to assess the risk of a system agree both on the possible attack strategies and on the evaluation of each vulnerability.

A. Modelling dependencies

In most cases, the vulnerabilities of a system are dependent to each other, that is, an attacker can use one of them to simplify the abuse of another one. Thus, there is a relation among the vulnerabilities that specifies how much easier becomes to abuse of the v vulnerability, broken every vulnerability in the set U. This relation is called a *dependency* between the ordered pair (U, v) and its weight, denoted[2] as $\varepsilon(v|U)$, measures the exploitability of v, given the abuse of U. The value $\varepsilon(v|U)$ is called the *conditional exploitability* of v given U.
Evidently, when the vulnerabilities are dependent one on the others, the previously defined simple risk assessment procedure is no more sound since the attack tree may not represent all the possible attacks allowing to achieve the root goal from the identified vulnerabilities and following the attack plan.
Given a pair of dependencies (U, u) and (V, v), we say that (U, u) is *stronger* than (V, v) if $u = v$, $U \subseteq V$ and $\varepsilon(u|U) \geq \varepsilon(v|V)$, meaning that it is convenient to abuse of (U, u) than (V, v) since less components have to be violated or the result is easier to obtain. It is worth noticing that $\varepsilon(v) = \varepsilon(v|\emptyset)$. So, $\varepsilon(v)$ must be less than $\varepsilon(v|V)$ for any non-empty V to make the (V, v) dependency significant.
The dependencies can be organised as an hypergraph $\mathcal{D} = \langle W, D\rangle$, where W is the set of all vulnerabilities and D is the set of dependencies. It is safe to assume that, for every $d \in D$, d is not stronger than any other dependency in D, since only the strongest dependencies may influence a risk evaluation[3]. Thus, the graph $\mathcal{D}$ is really an hypergraph, having arcs from sets of nodes to a node, but it is not a multigraph. Hence, the cardinality of D is bounded by $|W|2^{|W|}$.
Therefore, the simple risk assessment procedure in Section III can be extended to consider dependencies.

1. The threats to the system under examination are modelled using an attack tree, as before.

2. The dependencies among identified vulnerabilities are introduced considering also contextual, architectural and topological information. The dependencies are represented in the *dependency graph* $\mathcal{D} = \langle W, D\rangle$. Moreover, an exploitability value $\varepsilon(u|U)$ weights each dependency $(U, u) \in D$. The values $\varepsilon(u|\emptyset)$ are the initial exploitability of the vulnerabilities, as in the simple method.

3. The exploitability of each vulnerability v is calculated from its dependencies: initially, $\varepsilon_0(v) = \bot$ and then

$$\begin{aligned}\varepsilon_{i+1}(v) = \varepsilon_i(v) \vee \\ \vee \bigvee\{\varepsilon(v|V) \wedge \\ \wedge \textstyle\bigwedge_{w\in V} \varepsilon_i(w) \colon (V, v) \in D\}\end{aligned} \tag{1}$$

whose rationale is to update a value when it is convenient to use the dependency instead of the direct attack pattern. The function ε_i is said to be *final* when for all $j \geq i$, $\varepsilon_j = \varepsilon_i$.

4. The risk associated to the threat under examination is computed by recursively aggregating the final exploitabilities along the attack tree.

It is not immediately evident that there exists an index i such that ε_i is final, although it is clear that the metric must be a complete lattice to apply the method, since the updating rule applies lubs and glbs on subsets of values.
Proposition III.1 *For any initial evaluation ε_0, there is an index i such that ε_i is final.*
proof: See [9]. □

The method just described assumes no further knowledge on the system than the conditional exploitabilities of dependencies, which are fixed. This method is perfectly adequate when it is possible to evaluate each vulnerability in relation to its dependencies and the dependencies do not vary in time. As in the case of the simple risk assessment procedure, this method implicitly assumes that the experts trying to assess the risk on a system agree both on the possible attack strategies and on the evaluation of dependencies, in particular on vulnerabilities.

IV. Fixed points on metrics

In large systems, the previously remarked implicit assumptions are not valid in general. In fact, it is hard to find a group of experts on the whole system. Usually, experts are so on limited areas of the system. Thus, it makes sense to hire experts on different subsystems, and to make them to work together.
In this section, we want to develop the mathematical counterpart of this choice, namely when we are allowed to divide the risk assessment job among experts considering just smaller subsystems, without affecting the overall evaluation.
In this respect, we need to consider evaluations, i.e., functions from vulnerabilities to a fixed metric, and we need to develop some results about their mathematical structure. The aim of these results is to show that our method is minimal with respect to the structure of evaluations, which is the property we need to apply it naturally to large scale systems.
To simplify the treatment, we will assume that dependencies are binary, that is, each dependency (U, v) is such that $U = \{u\}$ for some vulnerability u. We write (u, v) instead of $(\{u\}, v)$.

[2]The chosen notation, $\varepsilon(v|U)$, resembles a conditional probability as $\varepsilon(v)$ resembles a probability. This is done on purpose to help intuition, although the ε function is not a probability measure.

[3]This is true because we perform a worst-case analysis; in an average-case analysis, all the dependencies must be considered.

Definition IV.1 *Let $\mathcal{M} = \langle M, \leq_{\mathcal{M}} \rangle$ be a metric and let $f, g: V \to M$ be functions from the set of vulnerabilities to the metric, then $f \leq g$ if and only if $f(x) \leq_{\mathcal{M}} g(x)$ for each $x \in V$.*

Being $\leq_{\mathcal{M}}$ a partial ordering, it follows that $\leq$ is a partial ordering, as well. Also, since $\mathcal{M}$ is a bounded lattice, $\mathcal{F} = \langle \{f: f: V \to M\}, \leq \rangle$ is a bounded lattice. Unfolding the definitions, it follows that

1. $(f \vee g)(x) = f(x) \vee g(x)$ for every $x \in V$;
2. $(f \wedge g)(x) = f(x) \wedge g(x)$ for every $x \in V$;
3. $\bot(x) = \bot$ and $\top(x) = \top$ for every $x \in V$.

So $\mathcal{F}$ is a metric. But we are mainly interested in another property of $\mathcal{F}$, namely, being a strict CPO.

Definition IV.2 *Given a partial order $\mathcal{O} = \langle O, \leq \rangle$, a sequence $\{e_i\}$ with $e_i \in O$ is said to be* monotone *if $e_1 \leq e_2 \leq \ldots \leq e_n \leq \ldots$ The* limit *of a monotone sequence $\{e_i\}$ is $\bigvee\{e_i\} = \bigvee_i e_i$, when it exists. Finally, $\mathcal{O}$ is said to be a* strict and complete partial order *(strict CPO, for short) when every monotone sequence has a limit and there is a minimal element in $\mathcal{O}$.*

Proposition IV.3 *$\mathcal{F}$ is a strict CPO.*

proof: Being a lattice, $\mathcal{F}$ has a minimal element $\bot$ and it is a partial order. So, let $\{e_i\}$ be a monotone sequence. But $\mathcal{F}$ is finite since $\mathcal{M}$ is so, thus $\{e_i\}$ forms a finite set of elements. Hence, there exists $\bigvee\{e_i\}$ in $\mathcal{F}$. □

Knowing that $\mathcal{F}$ is a strict CPO allows us to derive properties about the class of transformations from $\mathcal{F}$ to itself when they preserve limits.

Definition IV.4 *Let $\{e_i\}$ be a monotone sequence in $\mathcal{F}$ and let $\tau: \mathcal{F} \to \mathcal{F}$ be a transformation. Then, τ is said to be* monotone *if, whenever $x \leq y$ with $x, y \in \mathcal{F}$, $\tau(x) \leq \tau(y)$. Also, if τ is monotone, it is said to be* continuous *when $\bigvee\{\tau(e_i)\} = \tau(\bigvee\{e_i\})$.*

Theorem IV.5 (Minimal fixed point) *If $\tau \in \mathcal{F}$ is a continuous transformation, then there is a minimal $\mu \in \mathcal{F}$ such that $\mu = \tau(\mu)$.*

proof: The proof is standard, see [29], but we prefer to repeat it here since we are going to use it.

Consider the sequence $\{\tau^n(\bot)\}$ where $\tau^0(\bot) = \bot$ and $\tau^{n+1}(\bot) = \tau(\tau^n(\bot))$. Since τ is monotone, $\{\tau^n(\bot)\}$ is a monotone sequence in $\mathcal{F}$ and, since $\mathcal{F}$ is a strict CPO, there is $\mu = \bigvee\{\tau^n(\bot)\}$.

But $\tau(\mu) = \tau(\bigvee\{\tau^n(\bot)\}) = \bigvee\{\tau^{n+1}(\bot)\} = \mu$, so μ is a fixed point. Let λ be any other fixed point of τ: by induction on $i \in \mathbb{N}$, it follows that $\tau^i(\bot) \leq \lambda$ for any i, so $\mu \leq \lambda$. □

Corollary IV.6 *Let $f \in \mathcal{F}$ and let $S_f = \{g \in \mathcal{F}: f \leq g\}$. If $\tau: \mathcal{F} \to \mathcal{F}$ is a continuous transformation, then there is a minimal $\mu \in S_f$ such that $\mu = \tau(\mu)$.*

proof: It is immediate to verify that S_f is a strict CPO, so the theorem applies. □

Actually, the mathematical machinery developed so far is needed to analyse the risk assessment method, specifically the way we propagate dependencies. It turns out that the updating rule can be seen as a continuous transformation.

Definition IV.7 *Given $\mathcal{D} = \langle V, D \rangle$, a graph of dependencies in a risk assessment problem, the* fibre *on $v \in V$ (in $\mathcal{D}$) is $F(v) = \{(w, v) \in D: w \in V\}$. Let $f \in \mathcal{F}$, the* exploitability of the fibre *on v (in $\mathcal{D}$) with respect to f is*

$$E_F(v, f) = \bigvee_{(w,v) \in F(v)} (E(v|w) \wedge f(w)) \ .$$

Finally, $\xi: \mathcal{F} \to \mathcal{F}$ is defined by $\xi(f)(x) = f(x) \vee E_F(v, f)$.

Proposition IV.8 *If the metric $\mathcal{M}$ is a distributive lattice, then ξ is a continuous transformation.*

proof: Let $\{e_i\}$ be a monotone sequence in $\mathcal{F}$:

$$\begin{aligned} &\xi(\textstyle\bigvee\{e_i\})(x) = \\ &= (\textstyle\bigvee\{e_i\})(x) \vee \bigvee_{(w,x) \in D} (E(x|w) \wedge (\bigvee\{e_i\})(w)) = \\ &= \textstyle\bigvee\{e_i(x)\} \vee \bigvee_{(w,x) \in D} (E(x|w) \wedge \bigvee\{e_i(w)\}) \ , \end{aligned}$$

since the $\vee$ operation is pointwise.

So,

$$\begin{aligned} &\xi(\textstyle\bigvee\{e_i\})(x) = \\ &= \textstyle\bigvee\{e_i(x)\} \vee \bigvee_{(w,x) \in D} \bigvee\{E(x|w) \wedge e_i(w)\} \ , \end{aligned}$$

since $\mathcal{M}$ is a distributive lattice, i.e., for every $a, b, c \in \mathcal{M}$, $a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c)$.

Hence[4],

$$\begin{aligned} &\xi(\textstyle\bigvee\{e_i\})(x) = \\ &= \textstyle\bigvee\{e_i(x)\} \vee \bigvee\{\bigvee_{(w,x) \in D} (E(x|w) \wedge e_i(w))\} = \\ &= \textstyle\bigvee\{e_i(x) \vee \bigvee_{(w,x) \in D} (E(x|w) \wedge e_i(w))\} \ , \end{aligned}$$

because $\vee$ is associative and commutative.

Thus, $\xi(\bigvee\{e_i\})(x) = \bigvee\{\xi(e_i)(x)\}$ for every x, that is, ξ is continuous. □

Theorem IV.9 *Let $\mathcal{M} = \langle M, \leq_{\mathcal{M}} \rangle$ be a metric which is also distributive. Given $E_0: V \to M$, there is a minimal function $E: V \to M$ such that*

1. *$E(v) = E(v) \vee E_F(v, E)$ for every $v \in V$;*
2. *$E_0 \leq E$.*

proof: The first requirement says that E must be a fixed point of the continuous transformation ξ, but Corollary IV.6 asserts that there is a minimal fixed point E in S_{E_0} for ξ, meeting both requirements at once. □

Corollary IV.10 *The minimal fixed point E in the previous theorem is the limit of the sequence $\{E_i\}$ where $E_{i+1}(x) = E_i(x) \vee \bigvee_{(w,x) \in D} (E(x|w) \wedge E_i(w))$.*

proof: Evident form the proof of Theorem IV.5. □

So, the final exploitability of the vulnerabilities in a risk assessment problem with binary dependencies can be characterised as an iterative formula, or as the minimal function satisfying the requirements of Theorem IV.9. These descriptions of E are equivalent when the metric is distributive.

It is uncommon to use non-distributive metrics, as every total order is distributive. Nevertheless, it is easy to decide whether a metric is distributive.

Theorem IV.11 *A lattice is distributive if and only if it does contain a sublattice isomorphic to a pentagon or a diamond (see [17]).*

[4]The assumption on having just binary dependencies is fundamental to perform the next step in the proof. To overcome the difficulties with general dependencies, we need an additional hypothesis, that, for every $W \subseteq V$, $\bigwedge_{w \in W} \bigvee\{e_i(w)\} = \bigvee\{\bigwedge_{w \in W} e_i(w)\}$.

V. Analysing large systems

Suppose to have a large system $\mathcal{A} = \langle C, L\rangle$ with a huge number of components and links. To determine the risks it is exposed to, a natural choice is to hire some security experts, each one specialised on some aspect of the system. Together, their knowledge covers the whole system, although none of them is able, alone, to derive a reliable risk analysis, because none of them has enough expertise and time to understand the whole system.

In previous works [8, 9, 10], we have shown how it is possible to combine evaluations of two or more experts, each one adopting his own metric. In those works, each expert provides an evaluation of the risks on the whole system, and the difficulty lies in comparing exploitabilities expressed in different metrics, i.e., distinct way of measuring.

In the scenario we are dealing with now, the problem is different: each expert works on a different system $\mathcal{A}_i = \langle C_i, L_i\rangle$, and, on the basis of our previous works, we can safely assume that all the experts share the same metric and the same attack tree. But the whole system $\mathcal{A} = \langle C, L\rangle$ is such that $C = \bigcup_i C_i$ and $L = \bigcup_i L_i$, and our question becomes how to combine the evaluations of the experts into a risk assessment for the whole $\mathcal{A}$ system. It is worth remarking that the assumption of sharing the same attack tree is much less committing than it appears at a first sight: since experts analyse different subsystems and since they have different fields of expertise, it is natural to sum their knowledge about the possible attacks on the system. As a guideline, one may think that each main-level goal can be split into a series of subgoals (becoming an `or` node), one for each subsystem. For example, if $\mathcal{A}$ is composed by the subsystems $\mathcal{A}_1$ and $\mathcal{A}_2$, and we want to analyse the risk of an intrusion, we can divide this goal into the two subgoals 'successful intrusion in $\mathcal{A}_1$' and 'successful intrusion in $\mathcal{A}_2$'.

Moreover, the assumption of sharing the same metric is reasonable because, as shown in [8, 9, 10], if each expert uses his own metric, we can combine them into a single, system-wide metric, as far as the experts' metric are compatible, a fairly natural requirement.

So, suppose we have n experts, each one analysing the subsystem $\mathcal{A}_i = \langle C_i, L_i\rangle$; also, let $\mathcal{A} = \langle C, L\rangle$ be the system under examination, and $C = \bigcup_i C_i$ and $L = \bigcup_i L_i$. Notice, that we don't assume that the $\mathcal{A}_i$ systems are independent: it may be the case that $C_i \cap C_j \neq \emptyset$ for i and j distinct. Also, let M be the common, distributive metric.

A necessary step in the risk assessment process is to identify the vulnerabilities and the dependencies among them (as in Section V, we assume to have just binary dependencies, since we need to apply Theorem IV.9). This step must be performed at the beginning of the process and it produces two outcomes: the attack tree and the dependency graph. In a large system, the production of these outcomes is a team work where experts add the knowledge of their subsystems with the one of their colleagues.

Then, the experts should assess the exploitabilities associated with the identified dependencies and vulnerabilities. It is unlikely that the experts will agree, as some of them cannot reliably evaluate the risks not directly in their expertise area. Also personal conflicts, misunderstandings, and different attitudes may prevent the definition of a common picture. In fact, this phase is better conducted by separating the experts, so that each one can work on his best in the specific subsystem where his expertise is fully recognised.

In principle, there are two distinct ways to proceed:

- either (*procedure L*, for local) each expert produces a complete assessment of the vulnerabilities and dependencies in his subsystem, obtaining their final exploitabilities using the method of Section III-A,
- or (*procedure G*, for global) each expert limits himself to assess the initial exploitabilities of all the vulnerabilities and dependencies in his subsystem.

In both procedures, the correct way to deal with the collected data is to map the values obtained from the experts in the dependency graph, to calculate the final exploitabilities of the vulnerabilities in the whole $\mathcal{A}$ system, and, then, to assess the risk using the attack tree, as described in Section II. But procedure L is more efficient than procedure G, since it parallelises part of the work.

To understand the correctness of procedure L, it suffices to show that it produces the same final result as procedure G. In fact, procedure G is nothing else than a direct, canonical application of the risk assessment method described in Subsection III-A. The only difference is that we may have more than one initial exploitability value for each vulnerability or dependency. In those cases, we just take the worst evaluation, i.e., the supremum among the values, or, in other words, the "strongest" evaluation which, as shown in Subsection III-A, is the only significant one.

We can formalise procedure G as follows: let $e_0^i(v)$ be the initial exploitability of the v vulnerability as assessed by the i expert on the subsystem $\mathcal{A}_i$; similarly we denote by $e^i(v|u)$ the exploitability of the (u, v) dependency as stated by the i expert in the context of the $\mathcal{A}_i$ subsystem. So, $e_0(v) = \bigvee_i e_0^i(v)$ is the initial exploitability of v in the whole $\mathcal{A}$ system, where $e^i(v) = \bot$ when v does not lie in $\mathcal{A}_i$; also, $e(v|u) = \bigvee_i e_i(v|u)$ is the exploitability of the (u, v) dependency in the $\mathcal{A}$ system, where $e^i(v|u) = \bot$ when (u, v) is not a dependency in $\mathcal{A}_i$. Hence, applying the method in Subsection III-A on $\mathcal{A}$, starting from e_0, we obtain G, the final exploitability evaluation of the whole system.

Analogously, procedure L operates as follows: with the same notation as above, each expert i calculates e_f^i, his final exploitability evaluation on the $\mathcal{A}_i$ subsystem, starting from e_0^i. Now, H, the final exploitability evaluation on the whole $\mathcal{A}$ system starting from $h_0 = \bigvee_i e_f^i$, where $e_f^i(v) = \bot$ when v is not a vulnerability in $\mathcal{A}_i$, is calculated in the usual way.

Thus, establishing the correctness of procedure L means to prove that $H = G$.

Clearly, $e_0^i \leq e_0$ for every i. Let G^i be the final exploitability evaluation starting from e_0^i in the whole $\mathcal{A}$ system, where each dependency (u, v) has $e(v|u)$ as its exploitability value. It is immediate to see that $e_f^i \leq G^i$ for every i, since the exploitabilities of the dependencies in $\mathcal{A}_i$ may be lower than those in $\mathcal{A}$. Also, $G^i \leq G$ for every i, since $e_0^i \leq e_0$. Thus, $e_f^i \leq G$ for each i, and $\bigvee_i G^i \leq G$ by definition of supremum. By Theorem IV.9, G is the minimal fixed point such that $e_0 \leq G$ and $G(v) = G(v) \vee E_F(v, G)$ for every $v \in V$. But $\bigvee_i e_0^i \leq \bigvee_i G^i$ because $e_0^i \leq G^i$ for every i. Also,

$\left(\bigvee_i G^i\right)(v) = \bigvee_i G^i(v)$ and

$$\begin{aligned}
&\bigvee_i G^i(v) \vee E_F(v, \bigvee_i G^i) = \\
&= \bigvee_i G^i(v) \vee \bigvee_{(w,v)\in D} \left(e(v|w) \wedge \bigvee_i G^i(w)\right) = \\
&= \bigvee_i G^i(v) \vee \bigvee_i \bigvee_{(w,v)\in D} \left(e(v|w) \wedge G^i(w)\right) = \\
&= \bigvee_i \left(G^i(v) \vee \bigvee_{(w,v)\in D} \left(e(v|w) \wedge G^i(w)\right)\right) = \\
&= \bigvee_i G^i(v) \ ,
\end{aligned}$$

because G^i is a fixed point. So, $\bigvee_i G^i$ is a fixed point, and by minimality of G, $G \leq \bigvee_i G^i$. But we already proved that $\bigvee_i G^i \leq G$, hence $G = \bigvee_i G^i$.
Evidently, $h_0 \leq G$ because $e^i_f \leq G$ for every i. Since $e^i_0 \leq e^i_f$, it holds that $e^i_0 \leq h_0$, and we can deduce that $G \leq H$.
By Theorem IV.9, we know that H is the minimal fixed point such that $h_0 \leq H$ and $H(v) = H(v) \vee E_F(v, H)$ for every $v \in V$. But $h_0 \leq G$ and $G(v) = G(v) \vee E_F(v, G)$, being G a fixed point. So $H \leq G$ by minimality, and, thus, $G = H$, proving the correctness of procedure L.
It is interesting to notice how an apparently more efficient procedure, *procedure E*, could be devised, as promised in the Introduction: in fact, forcing each expert to use the global values for the dependencies, i.e., $\bigvee_i e^i(v|u)$ instead of $e^i(v|u)$, and his values for the vulnerabilities, i.e., e^i_0, his final evaluation becomes exactly G^i when it is calculated on the $\mathcal{A}$ system. Thus, the global evaluation of exploitabilities for the whole system can be immediately computed as the supremum $\bigvee_i G^i$, as shown above. In this way, the propagation of exploitabilities in the dependency graph is completely parallelised among the experts.

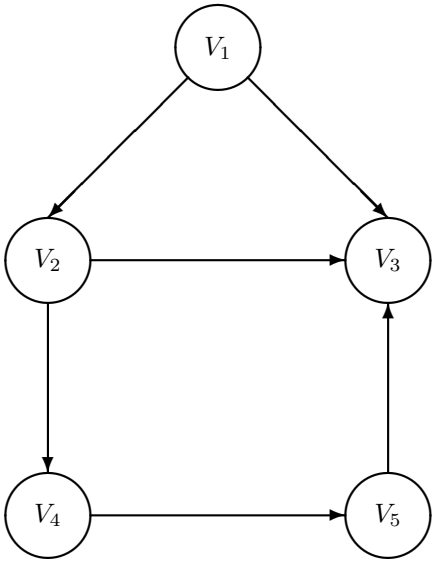

Figure. 1: The example system

VI. An illustrating example

Suppose to have a system whose dependency graph is the one in Figure 1, and let Alice, Bob and Cam be three security experts. The system is divided into three subsystems and each expert analyses his own subsytem, obtaining an initial evaluation on the metric $\bot = 0, \ldots, 9 = \top$, i.e., on the integer numbers from 0 to 9 ordered in the natural way. The subsystems and the experts' evaluations are depicted in Figures 2, 3 and 4. The experts's initial evaluations are denoted as e^A_0, e^B_0 and e^C_0, according to the first letter in the name.

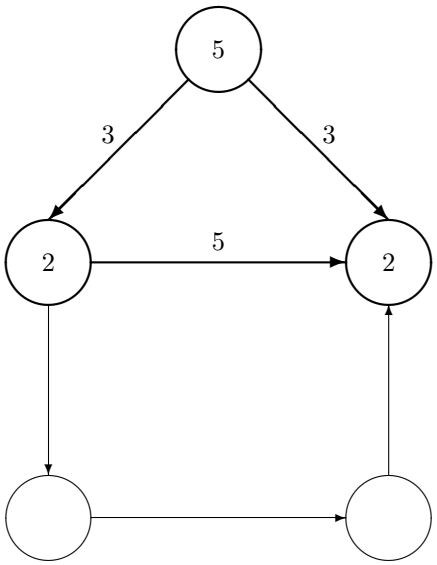

Figure. 2: Alice's initial evaluation

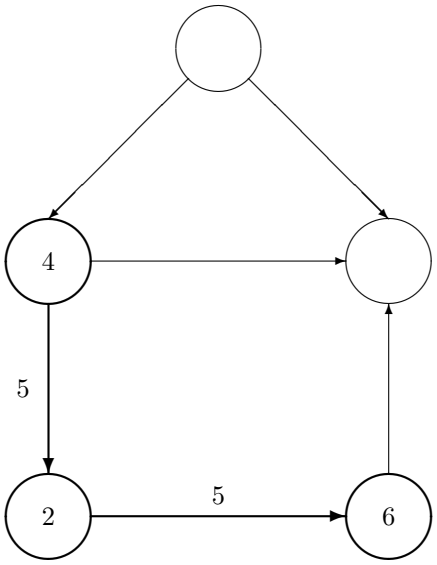

Figure. 3: Bob's initial evaluation

Following procedure G, we calculate $e_0 = e^A_0 \vee e^B_0 \vee e^C_0$, getting $e_0(V_1) = 5$, $e_0(V_2) = 4$, $e_0(V_3) = 2$, $e_0(V_4) = 2$, and $e_0(V_5) = 6$; analogously, $e(V_2|V_1) = 3$, $e(V_3|V_1) = 3$, $e(V_3|V_2) = 6$, $e(V_4|V_2) = 5$, $e(V_5, V_4) = 5$, and $e(V_3|V_5) = 4$. So we can compute G, the final exploitability evaluation, as follows:

	V_1	V_2	V_3	V_4	V_5
e_0	5	4	2	2	6
e_1	5	4	4	4	6
e_2	5	4	4	4	6
G	5	4	4	4	6

(2)

If we apply procedure L, each expert evaluates the final ex-

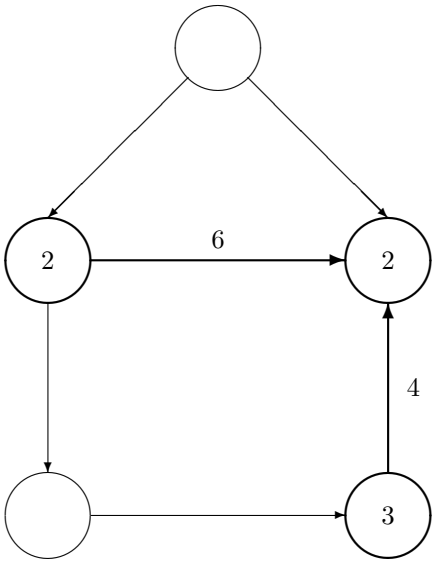

Figure. 4: Cam's initial evaluation

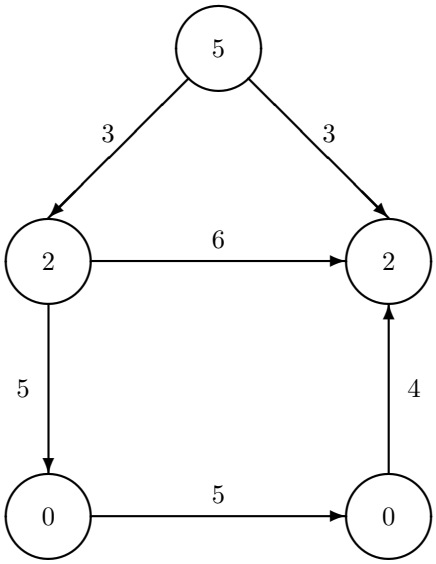

Figure. 5: Alice applying procedure E

ploitability evaluation on his subsystem:

	V_1	V_2	V_3	V_4	V_5
e_0^A	5	2	2	0	0
e_1^A	5	3	3	0	0
e_2^A	5	3	3	0	0
e_f^A	5	3	3	0	0

(3)

	V_1	V_2	V_3	V_4	V_5
e_0^B	0	4	0	2	6
e_1^B	0	4	0	4	6
e_2^B	0	4	0	4	6
e_f^B	0	4	0	4	6

(4)

	V_1	V_2	V_3	V_4	V_5
e_0^C	0	2	2	0	3
e_1^C	0	2	3	0	3
e_2^C	0	2	3	0	3
e_f^C	0	2	3	0	3

(5)

Then, posing $e(u|v)$ as before and $h_0 = e_f^A \vee e_f^B \vee e_f^C$, we can calculate the final exploitability evaluation for the whole system:

	V_1	V_2	V_3	V_4	V_5
h_0	5	4	3	4	6
h_1	5	4	4	4	6
h_2	5	4	4	4	6
G	5	4	4	4	6

(6)

As expected, the final result is equal to procedure G's.
When applying procedure E, each expert calculates the final exploitability evaluation on the whole system starting from his initial evaluation, i.e., the method of Subsection III-A is applied to the graphs in Figures 5, 6 and 7, obtaining:

	V_1	V_2	V_3	V_4	V_5
e_0^A	5	2	2	0	0
G_1^A	5	3	3	2	0
G_2^A	5	3	3	3	2
G_3^A	5	3	3	3	3
G_4^A	5	3	3	3	3
G^A	5	3	3	3	3

(7)

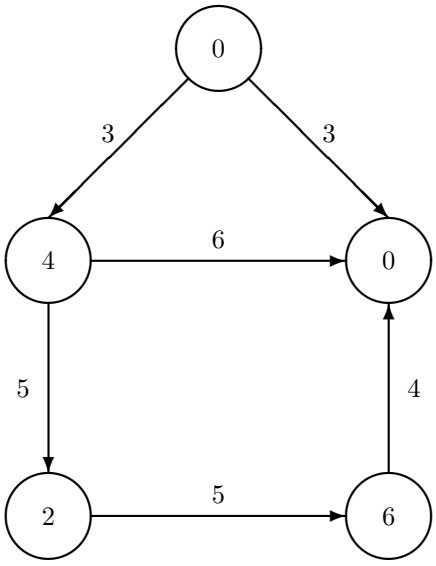

Figure. 6: Bob applying procedure E

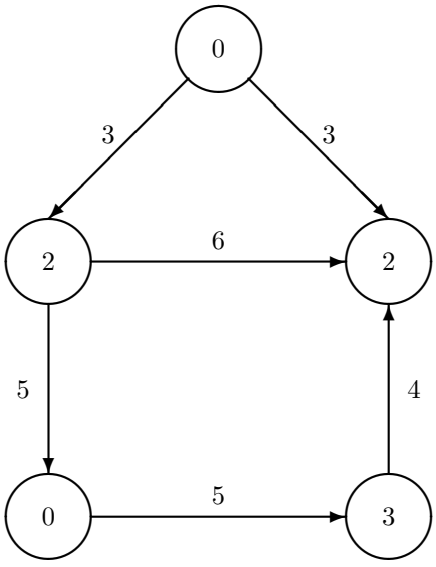

Figure. 7: Cam applying procedure E

	V_1	V_2	V_3	V_4	V_5
e_0^B	0	4	0	2	6
G_1^B	0	4	4	4	6
G_2^B	0	4	4	4	6
G^B	0	4	4	4	6

(8)

	V_1	V_2	V_3	V_4	V_5
e_0^C	0	2	2	0	3
G_1^C	0	2	3	2	3
G_2^C	0	2	3	2	3
G^C	0	2	3	2	3

(9)

The final exploitability of the whole system is $G = G^A \vee G^B \vee G^C$, that is,

	V_1	V_2	V_3	V_4	V_5
G	5	4	4	4	6

(10)

as expected.

In the example, procedure G requires 3 iteration steps, procedure L requires 6 steps (3 for the longest evaluation among e_f^A, e_f^B and e_f^C, plus 3 steps for combining them), and procedure E requires 5 steps (the longest evaluation, i.e., G^A).

It is worth noticing how the example shows that procedure E is more efficient than procedure L, as one would expect, while procedure G requires less iteration steps than procedure E, which is unexpected. In fact, the example has been chosen on purpose to illustrate this behaviour.

The longest path without cycles in the system's dependency graph, see Figure 1, is $V_1 \to V_2 \to V_4 \to V_5 \to V_3$, so a bad value in V_1 may take up to 4 steps to propagate to V_3, as it happens during the evaluation of G_A. And, of course, one more step is required to stabilise the evaluation.

So, on the average, we should expect that the evaluations of the various G^i will take a larger number of iteration steps than the evaluations of the corresponding e_f^i, since each G^i is calculated on a larger graph, with a potentially wider longest path without cycles. Of course, the evaluation of each G^i has the same order of magnitude as the calculation of G via procedure G, because the same graph is processed, although procedure G benefits from the 'speedup' given by starting from fewer $\perp$ values in e_0.

Thus, we can conclude that on a huge graph, as it is the general case of risk assessment on large systems, procedure E is slightly more efficient than procedure L, because the local evaluation of e_f^i will be faster than the corresponding G^i, but combining the various e_f^i in the final G has the same complexity as procedure G, so it loses the initial advantage most of the times.

If a large graph is evaluated following either procedure G or procedure L, the number of steps needed to reach the final evaluation is comparable and close to the length of the longest path without cycles in the graph. In fact, only a peculiar combination of the shape of the system graph and of the initial evaluations, as we did in the example, provides a sensible gap between the length of the computations of the two procedures. In those cases, procedure G is sligthly better, as in the example, because of the 'speedup' effect it can benefit of, having fewer $\perp$ values in its starting point.

VII. Effective information versus efficiency

It may be somewhat surprising that procedure G is more efficient than procedure E: in fact, completely parallelising the workload of computing the risk of the whole system should be, by common sense, more efficient than computing the same result through a strictly sequential procedure. But, as the example in the previous section has shown, this is not true and for a good reason, as we suggested in the end.

But, still, we claim that the overall amount of information that is generated in the computation of procedure E is more useful than the one provided by procedure G, making procedure E a better way to perform the risk assessment of a complex system.

The intermediate results in the computation of a fixed point following the algorithm in Subsection III-A are not very significant: their meaning is the propagation of the initial values through a chain of dependencies which is long as most as the iteration step being performed. This piece of information in progress is rather technical and not immediately useful.

In fact, the useful knowledge is the worst path, i.e., the chain of dependencies that has increased the exploitability value of a vulnerability to its final result. This piece of information is difficult to retrieve in the case of procedure L, which 'breaks' the chains when switching from the local subsystems to the global one, while it is easy in procedures G and E.

But procedure E provides this piece of information in a cleaner way: not only we are able to know the worst path, but also to which expert it pertains. For example, if we look at V_4 in the example of the previous section, it is immediate to see that the final value comes from Bob's evaluation and that the worst path is $V_2 \to V_4$. In fact, in the evaluation (8) we see where V_4 changes its value, and recalculating $G_1^B(V_4)$, we see that it is due to the exploitability of V_2 propagated along the (V_2, V_4) dependency. Analogously, the same path can be tracked in the calculation (2), which represents the application of procedure G, but it requires a further look at the subsystems to see that the evaluation comes from Bob.

The worst path is useful to understand where countermeasures should be applied: it is useless to apply a countermeasure which mitigates the exploitability of the V_4 vulnerability since the worst path tells us that V_2 will still have its bad influence. It is much more reasonable to mitigate V_2, as it affects also V_4. Evidently, the best security expert to ask for a good countermeasure to V_2 is Bob since he evaluated V_2 in such a way that V_4 gets a high value.

So, procedure E is more effective than procedure G or L when worst path analysis has to be performed because it allows to immediately individuate the experts to ask for globally effective countermeasures.

But there is another way in which procedure E is superior to procedures G and L. In fact, procedure E allows to decide what is the critical subsystem among the ones the whole system is divided into. By 'critical' we mean the subsystem which has the widest influence on the whole system, the one that, potentially, may spread its exploitabilities on the largest part of the system through the net of dependencies.

Individuating the critical subsystem is important because it is the subsystem which is convenient to attack in order to produce the maximal influence on the whole system: a successful attack on a critical subsystem makes the whole sys-

tem more vulnerable in a diffuse, general way. So, a critical subsystem is a natural candidate for strong protection, independently from the intrinsic value of its assets.
Also, critical subsystems are responsible for lower-bounding the risk assessment: their exploitability values influence the whole system, or a large part of it, so whatever countermeasure is applied on the system vulnerabilities, it cannot mitigate their exploitabilities beyond the values induced by the critical subsystem.
It is worth noticing that critical subsystems are especially important in the case of large and complex systems because they are exactly the parts not allowing to divide the system in almost independent pieces, thus they are the primary source of the complexity of the risk assessment task.
Evidently, procedure G does not help individuating the critical subsystems: its rationale is to combine the experts' initial evaluations and then to calculate the final exploitability evaluation; the piece of information about how the system is divided into subsystem is lost in the very beginning. Also, procedure L does not help as well, since it operates by calculating exploitability values inside each subsystem, and only after they are merged in the whole system, the interaction among subsystems is taken into account. So, again, when the final exploitability evaluation of the system is calculated, the way each single subsystem influences the whole, is lost.
Differently, procedure E calculates exactly what needed: it operates on the whole system, but each expert only calculates the spread of his initial evaluation, which is limited to his subsystem. In the example of the previous section, the calculation (7) clearly reveals that Alice's subsystem is critical: the final evaluation G^A has non-minimal values in all the vulnerabilities, while the initial vector has two $\perp$ values out of five.
In general, a good indication of the presence of a critical subsystem is the length of the calculation performed on it, like the computing of G^A from e_0^A above; in fact, it takes time to propagate the values in the system along dependencies when a large number of potentially distant vulnerabilities are influenced, which happens exactly when we deal with a critical subsystem. In this respect, it is worth comparing the calculation (7) with (8) and (9).
Hence, ironically, the source of inefficiency of procedure E with respect to procedure G is what makes it perfectly effective to individuate critical subsystems!

VIII. Related works

In literature there are many attempts to face the risk assessment problem; some of them define systematic approaches while others provide more ad-hoc methods to evaluate the likelihood of (a class of) violations. Even though the application of risk evaluation methodologies has been widely discussed and analysed, see, e.g., [12, 1, 18, 28], among information security experts there appears to be no agreement regarding the best or the most appropriate method to assess the possibility of computer incidents [24].
In particular, it is of interest Baskerville's description [3] of the evolution of various ad-hoc methods to measure risk that sometimes could be combined to improve the accuracy of the security evaluation.
On the side of systematic approaches, S. Evans et al. [14] present a system security engineering method to discover system vulnerabilities and to determine what countermeasures are best suited to deal with them: the paradigm of this work is *analysing information systems through an adversary's eyes*.
Differently, [22] provides a probabilistic model that measures security risks. It is possible to calculate risk starting from hybrid values of a quantitative and/or qualitative nature.
With respect to the previous works, our approach, starting from its initial definition in [27], has been based on the structured evaluation of single vulnerabilities along with their mutual dependencies. In this respect, the results in [14] are similar to ours, although they do not propose a formal method based on mathematical arguments. In fact, the distinctive aspect of our work with respect to the discussed ones is the mathematical formalisation of the risk assessment method to derive its characterising properties. Also, the use of hybrid values in [22] resembles our approach to metrics considered as algebraic structures, even though, we do not map them down to probabilistic estimates.
There are more formalised approaches in literature, employing a graph-based representation of systems and their vulnerabilities, that provide methods whose properties are, at least partially, mathematically analysed. Among those approaches, of prominent interest are those based on attack graphs [21, 25], where state-transition diagrams are used to model complex attack patterns. In particular, [21] proposes the use of attack graphs to automate the step of hardening a network against a multi-step intrusions. The proposed security solution is expressed as an adjustable network configuration rather than a set of countermeasures to possible exploits. Similarly, [20] divides a system into sub-domains and each sub-domain could be characterised by vulnerabilities. Applying probability theory and graph transformations [20] evaluates the possibility that a insecurity flow exploits some vulnerability to penetrate into the system. The extreme consequence of this family of approaches is to use model-checking techniques to simulate attacks, like in [25].
In this respect, our approach is simpler both in the method and in its formalisation. Despite its simplicity, our results are stronger on the mathematical side and some experimentation [4, 6, 7] make evident the practical value of the method in real-world situations. In fact, we use the attack tree model [19, 23] to evaluate the security threats combining them with the dependency graph, a formalisation of a piece of experts' knowledge. This combination is part of the subject of our mathematical analysis, and being a richer structure than the simple attack trees, we are able to derive stronger properties for our method [9, 10].
On a rather different comparison line, the software component paradigm in software engineering has received a great deal of interest from both industry and academia since it allows the reusability of components and a natural approach to distributed programming. A software component is independently developed and delivered as an autonomous unit that can be combined to become part of a lager application. Despite its evident benefits, the component interdependence is often ignored or overlooked [11], leading to incorrect or imprecise models. To avoid this problem, complete models should be specified taking into account

system interconnections. In agreement with this point of view [11, 13, 14, 22, 24] present models for assessing security risks considering interdependence between components. Particularly, [11] uses techniques for automating and enhancing risk assessment studies of technological processes using qualitative models. A set of fundamental parameters and primitive functions are defined for the domain from which the system behaviour is derived, detecting some interesting interdependencies among components.
Similarly, [13] defines a model based on security policies and individual risks. The model makes possible to evaluate if the risk associated to each transaction is acceptable. The evaluation of risk also takes into account context information.
Independently from their application areas, the risk assessment methods have a core weakness: the use of subjective metrics. In fact, in the scientific community the main criticism to these methods is about the fact that values are assigned on the basis of personal knowledge and experience. In extreme cases, these assessment are regarded as *random* values, making the total risk evaluation process to be considered as a *guess*.
It is a fact that the evaluation metric behind exploitability deeply influences the risk evaluation. But, at least in our treatment, what matters is the *structure* of the metric rather than its absolute value.
Generalising, in many field of ICT there is the need to define an objective metric. In the abstract, a metric is defined as [2] the instrument to compare and to measure a quantity or a quality of an observable.
Our treatment of metrics follows the work of N. Fenton, in particular [15]. In agreement with him, we consider measurement as the process by which values are assigned to attributes of entities, in our case to the exploitability of a vulnerability. So, even though there is no widely recognised way to assess risks and to evaluate the induced damages, there are various approaches that provide methods by which the risk evaluation becomes more systematic.
In particular, Sharp et al. [24] developed a scheme for probabilistic evaluation of the impact of the security threats and proposed a risk management system with the goal of assessing the expected damages due to attacks in terms of their economical costs. Z. Dwaikat et al. [13] defined security requirements for transactions and provided mechanisms to measure likelihood of violation of these requirements.
Looking toward risk assessment as a decision support tool, Fenton [16] proposed the use of Bayesian networks. He distinguishes between certain and uncertain criteria and points out the power of Bayesian networks to reason about uncertainty. Differently, our approach toward objective risk assessment is based on the abstraction over values, thus what matters in our treatment is the *structure* of the metrics. Hence, objectivity is gained by considering values in the metric not as *absolute measures of risk*, but, instead, as *relative evaluations*. Therefore, in agreement with [11, 14, 16, 22], the information computed by our model can be used as a decision support.

IX. Conclusions

In this paper, we introduced a risk assessment procedure which is suited for analysing large systems.
Namely, a large system is divided into a number of subsystems, maybe overlapping, each one examined by an expert. By collecting and properly organising the knowledge of the various experts, we have shown that it is possible to efficiently combine their evaluations into a global risk assessment for the whole system. An effective combination is possible only under the hypotheses of Section II, which are structural, i.e., they apply on the metrics employed by the experts and not to the specific nature of the analysed system. It is worth remarking that the rather complex mathematical treatment we used in Subsection IV is needed to justify the risk assessment procedure when concurrently applied by the experts, but it is not used to compute the risk evaluations. Hence, the risk assessment method retains its simple computational pattern which enabled for the application to real systems, see, e.g., [4, 6, 7].
We defined and analysed three different procedures for assessing the risk in large systems. We proved that one of them, procedure G, is computationally efficient, but another one, procedure E, is more interesting since it is reasonably efficient on large systems, but the performed calculations can be used to extract useful information about the security posture of the system, hence enriching the risk analysis.
In particular, we have focused on worst chain of dependencies, allowing to individuate the distant nodes that affect a vulnerability of interest. In this respect, we have shown how procedure E not only can be used to individuate them, but also to name the expert who is the best choice for suggesting a countermeasure.
Also, we defined the notion of 'critical subsystem' as the one affecting the largest part of the system via propagation over dependencies. In this respect, we have shown that procedure E, differently from the other two procedures, can individuate the critical subsystems among the division operated on the whole system without performing other calculations, as this piece of information can be directly extracted from the inspection of the computation's steps.

Acknowledgments

The first author of this research was supported by a Marie Curie Intra European Fellowship (grant n. PIEF-GA-2010-271926, *Predicative Theories and Grothendieck Toposes*) within the 7th European Community Framework Programme.

References

[1] C. Alberts, A. Dorofee,J. Stevens, and C. Woody, "Introduction to the Octave approach," Oct. 2003. [Online]. `http://www.cert.org/octave/approach_intro.pdf`

[2] S. Arshad, M. Shoaib, and A. Shah, "Web metrics: The way of improvement of quality of non web-based systems," in *Proc. of the Intl. Conf. on Software Engineering Research and Practice*, H. R. Arabnia and H. Reza, Eds., vol. 2. Las Vegas, NV, USA: CSREA Press, 2006, pp. 489–495.

[3] R. Baskerville, "Information system security design methods: Implications for information systems devel-