Risk Assessment in Practice: A Real Case Study

Marco Benini

Dipartimento di Informatica e Comunicazione,
Università degli Studi dell'Insubria,
IT-21100 via Mazzini 5, Varese, Italy

Sabrina Sicari

Dipartimento di Informatica e Comunicazione,
Università degli Studi dell'Insubria,
IT-21100 via Mazzini 5, Varese, Italy

Abstract

The aim of this work is to evaluate the risk of an external attack to the network of our Department in the University. Thus, this work wants to complement the results in [6] where a mathematical framework justifying our risk assessment method has been presented. Hence, this article describes a detailed account of our experience where the instruments, the techniques and the results are described and evaluated.

Key words: Risk assessment, case study, network security

1 Introduction

Risk assessment is the measuring of the possible negative impacts of an undesired event in a given system. The key words in the previous definition are "measuring" and "possible": the word "measuring" suggests that an engineering method is needed to quantitatively evaluate the possible occurrence 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 6. Each method has it own peculiarities that make it more or less apt to evaluate the risk of a specific

class of threats or that make it more suitable to be applied to certain types of systems. Roughly speaking, the 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 evaluation of some experts and their goal is to combine and refine these initial assessment to construct a final outcome that does not strictly depend on the professional reliability of the experts.

We believe that a good risk assessment method should be both practical and theoretically sound, that is, it should justify its outcomes by means of a scientific argument. For these reasons, we proposed in [4, 6] a risk assessment method that is based on a strict mathematical model: we have been able to prove a number of 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.

Therefore, the theoretical aspect of our proposed method has been established; the question whether the method is useful on a real case left open. Thus, in this work, we report our experience on the application of our method to the analysis of the security of the network of our Department. We will describe (Section 2) the architecture of the network and the view a potential attacker has of it. Then, we describe (Section 3) our method and we apply (Section 4) it step-by-step to the network: the details of the application are analysed and justified [1]. The results have been used to improve the security of our network both by identifying and eliminating the major sources of risk, and by introducing risk assessment as a standard practice in the network administration. In Section 5, we analyse the effects of the risk assessment both as a way to secure a network and as a best practice in the administration task.

As a consequence, the interested reader may apply our method to her/his system, following the guidelines deduced from our experience, and, in our hope, s/he will be convinced that the proposed method has both the required theoretical background that justifies the obtained results and the strength of a practical method conceived to apply to real situations.

[1] On purpose, some details will not be reported: being the account of a real risk assessment, the information that may be used to violate the security of the observed network has not been made explicit.

2 The environment

The experimental analysis of the risk assessment method has been conducted in the Dipartimento di Informatica e Comunicazione (DICOM) of the Università degli Studi dell'Insubria. The Università degli Studi dell'Insubria is a small university located in Varese and Como, two cities in the north of Italy. Because of its double location, the university has a complex wide-area network that supports the telecommunication needs of its structures (Faculties, Departments, research centres and offices). The DICOM is the department of Computer Science in Varese and it is linked to the campus network by a laser connection (10Mbit/s) and a leased line (2Mbit/s) for backup purposes.

The DICOM network supports the usual communication needs (email, web, ...) of its members and allows the access from outside to the services (web sites, software repositories, ...) related to the various research projects and to the teaching activities. Therefore, to enable the widest range of experimentation in networking, the connection from the DICOM to the external world is poorly protected. Consequently, when the network had been designed, there was an obvious need to secure the normal activities of the DICOM staff and the fundamental services, preserving at the same time the absolute freedom of the research related activities. This security requirement was further constrained by the cost in term of management effort since the DICOM technical staff counted just two system managers.

Hence, the DICOM network has been structured as a private network connected to a public zone linked to the university backbone and, then, to the Internet: the global picture is illustrated in Figure 1.

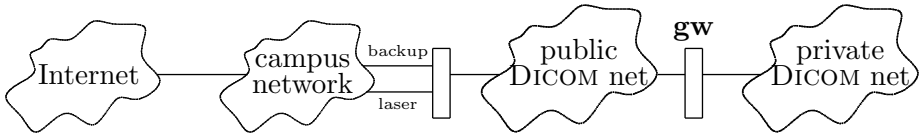

Fig. 1. The overall structure of the DICOM network

The public DICOM network hosts the unconstrained services to support the research projects. In contrast, the private DICOM network has been secured by several countermeasures to discourage the network attacks from outside and to prevent the access by intruders. In particular, the structure of the private network is depicted in Figure 2, where the **gw** gateway links the network to the public infrastructure, as shown in Figure 1.

Specifically, the components of the private network are:

- the **gw** gateway that masquerades and protects the private net, since it implements an IP stateful firewall [22] and since the whole network is hidden via the NAPT protocol [33];

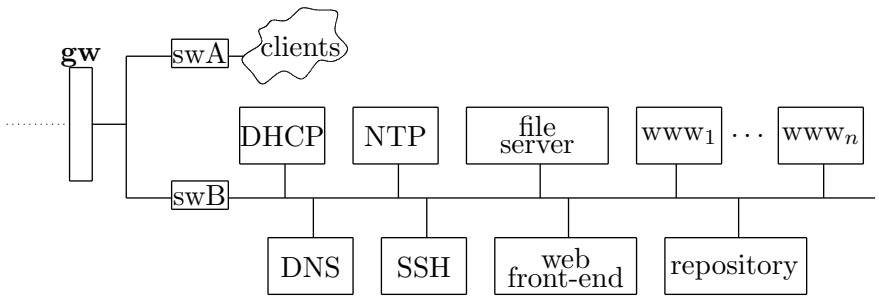

Fig. 2. The structure of the private DICOM network

- the **swA** switch interconnecting the physical LANs where the users' PCs reside;
- the cloud of the clients which contains the users' PCs and the public devices, e.g., the printers, for personal usage by the department staff;
- the **swB** switch that links to the external world the servers, both the structural, private service providers, like the DHCP server or the file server, and the public servers, providing stable (non research-oriented) services, accessible from outside, like the ssh server and the web front-end.

In addition, the various web servers ($\mathbf{www}_1$, ..., $\mathbf{www}_n$) are not directly accessible from outside, but they are used via the web front-end that acts like a proxy, to allow the uniform logging of web traffic. The repository is a CVS server [10], accessible from everywhere, which uses the SSH protocol [36] as its transport, thus ensuring encrypted information exchanges. Finally, the DNS is configured as a split name server [2] that resolves the addresses and the names differently if a query comes from the private network or from the outside world.

In the illustrated context, the risk assessment method has been applied to estimate the risk of an external attack, to evaluate the quality of the security countermeasures and, eventually, to improve them.

Therefore, the risk analysis is based on the view of the private network as seen from outside: this view, illustrated in Figure 3, is generated by the **gw** gateway via the NAPT protocol.

Specifically, the various clients in Figure 2 are dynamically mapped into the pool of virtual clients. The public servers are mapped to the public IP address of the gateway, thus the access to them is restricted to the public protocols they implement, e.g., the ssh server is limited to receive only ssh connections from outside. The other servers are not allowed to receive communications from outside, in fact, they have not even a public IP address.

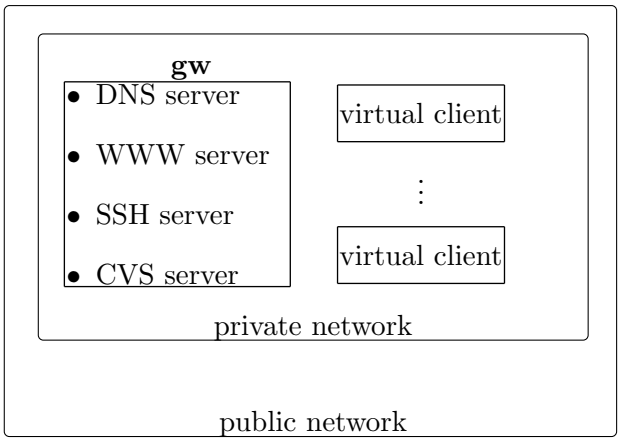

Fig. 3. The private DICOM network as seen from outside

3 The risk assessment method

In general, the goal of risk assessment is to quantitatively evaluate the risk of an undesired event in a given environment. In the present case, the environment is the private DICOM network described in Section 2. Therefore, the aim of this section is to define the notion of risk and to describe the method to assess it; the formalisation and the properties enjoyed by the risk assessment procedure are beyond the scope of this paper and the interested reader is referred to [4, 6].

In the context of the case study under development, the risk is a function on two parameters: the damage potential and the level of exploitability. The damage potential measures the ability to damage as the number of the affected users (both the staff members and the students) times the average number of days the affected service is unavailable; the level of exploitability measures the easiness to perform an attack, as analysed in [19].

The risk is measured by a procedure consisting of a sequence of four steps:

- **Step 1**: A threat to the system under examination is modelled by using an attack tree [29]: the root node represents the attack goal and, recursively, its children can be alternative subgoals, each one satisfying the father goal (`or` subtree) or partial subgoals, whose composition satisfies the father goal (`and` subtree).
- **Step 2**: The dependencies among the identified vulnerabilities are introduced: a vulnerability v depends on a vulnerability w if and only if v becomes easier to achieve when w is already exploited.

  In this step, a numerical index E, called *exploitability*, is associated to each vulnerability v, to measure the likelihood that v may be successfully used to break the security of the system. The evaluation of E can be rough: to apply the method, it suffices that the order of the indexes reflects the

relative difficulty of exploitation (see [6] for the details). Similarly, the dependencies are weighted on the same metric as vulnerabilities taking into account contextual, architectural and topological information.

- **Step 3**: The index E of each single vulnerability is updated according to its dependencies. This step is iterated until the values reach a fixed point: in [6] it has been proved that the iteration process converges in finite and bounded time.
- **Step 4**: The risk associated to the threat under examination is finally computed by recursively aggregating the exploitabilities along the attack tree. The exploitability of an `or` subtree is the maximum (easiest) value among its children, and the exploitability of an `and` subtree is the minimum (most difficult) value among its children. The aggregated exploitability of the root node measures the level of feasibility of the attack and is combined with the damage potential to assess the risk of the threat.

The dependencies identified in Step 2 influence the values of the exploitabilities of the various vulnerabilities. The exact evaluation of this influence is the core of Step 3 and of the whole risk assessment method. The formal presentation of the influence of the dependencies on the exploitability is as follows.

The network architecture is described as a graph $S = \langle C, L \rangle$ where C is the set of *components* and L is the set of *links* between components. The components and the links are exposed to the set of vulnerabilities V_C and V_L respectively: the notation $(v, c) \in V_C$ and $(v, l) \in V_L$ means that the component c or the link l is susceptible to be subverted thanks to the flaw v. The set of all vulnerabilities is $V = V_C \cup V_L$. To ease notation, $e(v) \in C \cup L$ denotes the unique element of S to which the vulnerability v applies.

Initially, during Step 2, a numerical measure of exploitability is assigned to every vulnerability; this value is called the *initial exploitability* of the vulnerability v in the system S and it is denoted as $E_0(v)$.

Similarly, the exploitability value of the vulnerability v given that the vulnerability u has already been exploited is denoted with $E(v|u)$. The dependencies among vulnerabilities are represented in the *vulnerability dependence graph* $G = \langle V, D \rangle$. There is an edge $(u, v) \in D$ if $E(v|u) > E_0(v)$, i.e., if it is easier to compromise the element $e(v)$ when one has already exploited u over $e(u)$.

The notion of exploitability is generalised by means of the $E\colon \mathbb{N} \times V \to M$ function that maps the vulnerabilities to M, a total ordered set of degrees of exploitability; in the present case, M is the metric $\{x \in \mathbb{N}\colon 0 \leq x \leq 10\}$ where 0 means "not exploitable at all" and 10 "immediate to compromise". To each node v in the vulnerability dependence graph G is associated the value $E(i, v)$, written as $E_i(v)$. Initially, each node v is labelled with the value $E_0(v)$. The conditional exploitabilities $E(v|u)$ are used to label the edges (u, v) of G.

To calculate the influence of dependencies during Step 3, the following formula is applied to each vulnerability v:

$$E_{i+1}(v) = \max(\{E_i(v)\} \cup \{\min(E(v|x), E_i(x)) \colon (x, v) \in D\}) \qquad (1)$$

The rationale is: at any iteration $i+1$, the $E_i(v)$ is updated considering if the vulnerability v becomes easier to exploit at time $i+1$ thanks to a dependency (x, v). Since the method is guaranteed to converge in bounded time [2], the Formula (1) is applied until a fixed point is reached, i.e., no dependency in the G graph can be used to easier the abuse of a vulnerability.

It should be remarked that complex vulnerabilities, spreading over more than one component or link, like architectural network flaws, can be managed by this method to a limited extent. Precisely, since the method focuses on single vulnerabilities and their interdependencies, a complex vulnerability v can be expressed either by a set of simpler, inter-related vulnerabilities that form an attack tree whose root node is v, or by structuring the dependence graph as a multi- or an iper-graph. The first approach, reduction to simple vulnerabilities, can be carried out within the limits of the presented method, while the second approach requires an extension to the risk assessment procedure which is beyond the scope of this paper. It should be noted that it is not always possible or convenient to use the above mentioned reduction.

4 Application to the DICOM network

The purpose of this work is to evaluate the risk associated to a security violation from the outside world in the private DICOM network.

4.1 Step 1: construction of the attack tree

The vulnerabilities have been identified analysing the network by means of the Nessus Vulnerability Scanner v3.03 [22, 25], a well-known program equipped with several constantly updated modules each one testing the presence of

[2] Since the set of the exploitability values is finite and, at each step, either the fixed point is reached or the exploitability of at least one vulnerability is incremented, after enough steps, bounded by the number of vulnerabilities times the number of exploitability values, the fixed point is reached since no exploitability value can be incremented any further. Thus, the proof is an application of the well-known pigeon hole principle.

a particular threat [3]. At the moment, Nessus has more than 100.000 modules and it allows various degrees of personalisation in the scanning strategy. Summarising the outcomes, Nessus found eight vulnerabilities in the private DICOM network:

- V_1 is a "race condition" in the OpenSSH server that may allow an unauthenticated remote attacker to crash the service (denial of service attack) or to execute code on the affected host. Nessus warns that the successful exploitation of these issues requires the GSSAPI (Generic Security Services Application Programming Interface) [21] authentication to be enabled in the server.
- V_2 is an "information disclosure" vulnerability found in the OpenSSH server. This vulnerability allows to validate usernames and, thus, to perform a "brute force" attack to gain the users' passwords and then, to access the affected system.
- V_3 is another "information disclosure" vulnerability in the OpenSSH server. Specifically, the software allows GSSAPI credentials to be delegated to users who log in without the GSSAPI authentication if 'GSSAPIDelegateCredentials' is enabled. This vulnerability allows a remote hacker to access GSSAPI credentials and to use resources allocated to the original user bypassing the access control system.
- V_4 is an "assertion failure" vulnerability in the name server that enables a denial of service attack. This vulnerability is caused by an assertion error during the SIG query processing and it may cause a crash in a recursive server when many SIG Resource Records (RRset) are returned and in an authoritative server that is associated to a DNSSEC [15] with many SIG RRSet exchanges.
- V_5 is an "insist failure" vulnerability in the name server that allows to perform a denial of service attack. Synthetically, a burst of queries may cause a deadlock in the server.
- V_6 is the vulnerability enabling the download of the source code of several scripts on the web server; these scripts may contain sensible information. By appending various suffixes (e.g., ".old", ".bak", "~", etc...) to the names of the web pages, it is possible to download the source code of server-side scripts.
- V_7 is the possibility to use the TRACE and/or TRACK HTTP methods enabled in the web server. The TRACE and TRACK are HTTP methods

[3] A good security scanner tests for the presence of a large number of possible vulnerabilities. Nothing prevents to use any other instrument or way to individuate the potential vulnerabilities of a system. One can even introduce unknown vulnerabilities in the vulnerability dependence graph to model suspected threats: it suffices to add a new node, modelling the unknown vulnerability, and arcs to every other node, modelling the unknown influences. In this case, appropriate weights should be guessed.

used to debug web connections. Since these methods are subject to cross-site-scripting (XST) attacks when used in conjunction with some weaknesses in the browsers, an attacker may use this flaw to trick the legitimate web users and to steal their credentials.

- V_8 is a weakness caused by the adoption of an insecure cryptographic protocol in the SSH daemon.

Goal: Violation of the network security
1. Accessing the system
  1.1 As a superuser
    1.1.1 Climbing privileges
      1.1.1.1 Accessing as a user (see 1.2)
  1.2 As a user
    1.2.1 Shell code
      1.2.1.1 Race condition [V_1]
    1.2.2 Brute force [V_2]
    1.2.3 Password cracking [V_6 and V_8]
    1.2.4 Information disclosure [V_3 and V_6]
2. Disabling a service
  2.1 Denial of service (DoS)
    2.1.1 Assertion failure [V_4]
    2.1.2 Insist failure [V_5]
    2.1.3 Accessing the system as a user (see 1.2)
    2.1.4 Race condition [V_1]
    2.1.5 Accessing as a superuser (see 1.1)
  2.2 Distributed denial of service (DDoS) (see 2.1)
3. Gaining unauthorised information
  3.1 Accessing as a superuser (see 1.1)
  3.2 Accessing as a user (see 1.2)
  3.3 SQL injection
    3.3.1 Cross site tracing (XST)
      3.3.1.1 TRACE/TRACK HTTP methods enabled [V_7]
    3.3.2 Information disclosure [V_2, V_3 and V_8]

Fig. 4. The attack tree

Nessus weights the found vulnerabilities considering their absolute importance, i.e., the easiness to perform them and the level of mis-functioning they may generate in the affected system: these weighting values are used and discussed in the second step, see Section 4.2.

The possible attacks are formalised in the attack tree shown in Figure 4. Three main violations have been considered: the access to the unique (as visible from outside) server system by an unauthorised user; the stop of a delivered service; the collection of private and sensible system information. Each one of these goals of a possible intruder has been analysed to construct the attacks that may be performed to attain them: the analysis is based on the suggestions provided by Nessus as part of its vulnerability report. More specifically, the

attack tree has been constructed starting from the Nessus suggestions and then, by searching in the usual repositories of threat descriptions, the possible attacks have been identified and structured. Therefore, the first level of the attack tree (points 1, 2 and 3) has been defined in the very beginning as what was meant to be evaluated, while the structure of the subtrees has been identified "bottom-up", i.e., systematically reconstructing the possible attacks from the vulnerabilities as identified by the security scanner.

In the obtained attack tree, it is important to notice that the internal nodes correspond to attack techniques (named, but not explicated), while the leaves correspond to the identified vulnerabilities (shown in square brackets). To simplify the tree, we used references when a subtree occurs more than once, and, in particular, we considered as equivalent the DoS and the DDoS attack patterns (and, thus, their associated subtrees) since the actions are the same when performed on the network under analysis.

4.2 Step 2: the vulnerability dependence graph

It is clear that the identified vulnerabilities are not independent. In fact, the V_2, V_3 and V_8 vulnerabilities are dependent from each other since they allow an information disclosure: the exploitation of one of them simplifies the others since the intruder gains a deeper knowledge of the system.

The V_4 and V_5 vulnerabilities (weaknesses in the name server) depend on the V_1 race condition inside the signal handler of the OpenSSH server. Specifically, if the race condition is successfully used to gain a partial access to the system, it becomes easier to exploit V_4 and V_5 and to perform a DoS attack against the name server: it suffices to modify the data on the cracked system. Similarly, V_4 and V_5 also depend on the V_2 and V_8 information disclosures, that, exploited in conjunction with a brute force attack, allow a partial access to the system, thus allowing the application of the already described attack pattern.

The V_7 vulnerability (TRACE and TRACK methods) depends on V_6, the download of some scripts from the web server. These scripts may contain sensible information, as the credentials to access the system databases. This kind of information reduces the difficulty to exploit the V_7 vulnerability since the intruder gains the knowledge of what to trace in the web connections.

The analysis of dependencies among vulnerabilities is complete: the identification of dependencies has been conducted by combining the single points in the detailed descriptions of the attacks identified in the previous step of the risk assessment procedure. For obvious security reasons (the case study is real) these descriptions have been omitted in this presentation. Whenever a point in the description of one attack is a (partial) gain in the description of another

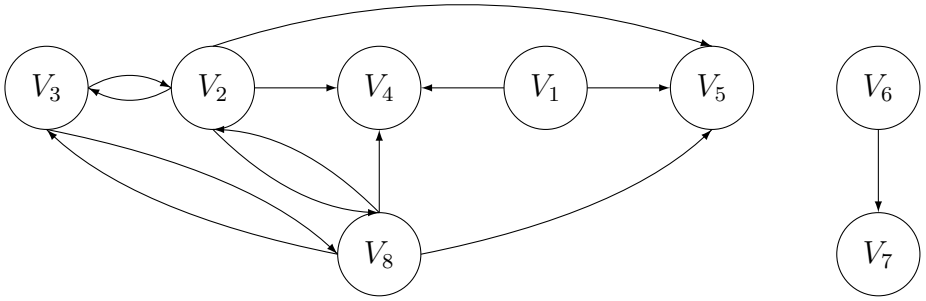

Fig. 5. The vulnerability dependence graph

attack, a dependency has been identified between the enabling vulnerabilities of the former and the latter attacks. Therefore, the vulnerability dependence graph can be drawn as shown in Figure 5.

Having a complete and reliable picture of the vulnerabilities of the system and their dependencies, as well as of the way to exploit them, either singularly or jointly, it is time to evaluate them. The weighting of the vulnerabilities uses a range from 0 to 10, where 0 means "impossible to violate" and 10 means "completely straightforward". The Nessus scanner indicates the risk factor and the CVSS (Common Vulnerability Scoring System [4]) base score [12] for every found vulnerability. These values have been modified to consider the context where the vulnerabilities have been found.

In particular, the number and the importance of the compromised services has been considered to modify the base values as reported by the Nessus scanner: in this respect, the DNS system is of primary importance, since its mis-function is propagated to all the other services, due to the internal structure of the network; the SSH server is considered as important because its abuse allows to gain control over the servers and, thus, to potentially multiply the effects of a successful attack; on the contrary, the attacks and the vulnerabilities requiring to guess the passwords have been considered as less critical because of the strict password policy employed in the DICOM system management. Hence, the base values as reported by the Nessus scanner have been uniformly mapped in the $0 \ldots 10$ range, then they have been incremented by 2 if affecting the DNS service, incremented by 1 if affecting the SSH server, and decremented by 1 if requiring to guess passwords.

The final result has been used to label the nodes in the dependence graph, as shown in Figure 6.

[4] The CVSS scores refer to the National Vulnerability Database [11]. The CVSS is a standard to define an open and universal method to evaluate the risk level of vulnerabilities. The level of risk of each vulnerability is measured by three types of metrics: base (taking into account only the intrinsic features), temporal and environmental conditions.

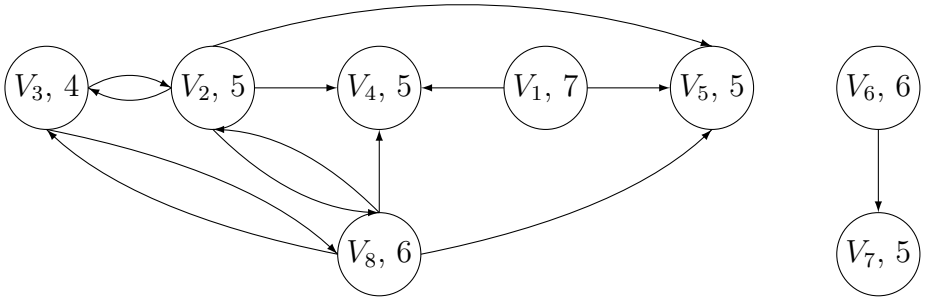

Fig. 6. The initial assessment of the single vulnerabilities

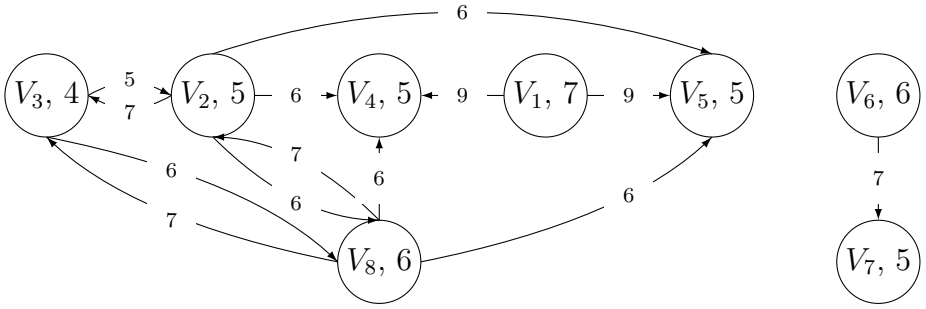

Fig. 7. The dependence graph with full labels

The degree of dependence among vulnerabilities is measured on the $0 \ldots 10$ range. The value represents how easier becomes to use a vulnerability to perform an attack, broken the preceding one in the dependence graph. Thus, the dependency (u, v) is weighted by applying an increment to the value of the v vulnerability that depends on the importance of u in the attack patterns when both u and v are used to compromise the system. The increment is calculated as if the already compromised vulnerability u has a value of 10 in the attack description and then, the value of the dependent vulnerability v is derived from the evaluation of the overall difficulty of the attack. In some cases, the influence of a dependency does not raise in a significant way the degree of exploitability of a vulnerability, as is the case of V_2 (exploitability = 5) when V_3 is achieved (if V_3 is exploited, V_2 raises to 5). In most cases, the raise is significant. The overall picture is reported in Figure 7, where, as explained before, the dependent exploitability values are used to label the arcs.

4.3 Step 3: calculating the dependencies

The propagation of the dependencies by means of the formula (1) over the dependency graph constructed in the previous step yields to the results shown in Table 1. At the second iteration, the exploitability values reach their fixed point and, thus, the labelling of the dependence graph becomes stable. There-

Table 1
The calculation of the effects of the dependencies

$$
\begin{array}{lll}
E_1(V_1) = \max(\{7\} \cup \emptyset) & = 7 & = 7 \\
E_1(V_2) = \max(\{5\} \cup \{\min(5,4), \min(7,6)\}) & = \max(5,4,6) & = 6 \\
E_1(V_3) = \max(\{4\} \cup \{\min(7,5), \min(7,6)\}) & = \max(4,5,6) & = 6 \\
E_1(V_4) = \max(\{5\} \cup \{\min(9,7), \min(6,5), \min(6,6)\}) & = \max(5,7,5,6) & = 7 \\
E_1(V_5) = \max(\{5\} \cup \{\min(9,7), \min(6,5), \min(6,6)\}) & = \max(5,7,5,6) & = 7 \\
E_1(V_6) = \max(\{6\} \cup \emptyset) & = 6 & = 6 \\
E_1(V_7) = \max(\{5\} \cup \{\min(7,6)\}) & = \max(5,6) & = 6 \\
E_1(V_8) = \max(\{6\} \cup \{\min(6,5), \min(6,4)\}) & = \max(6,5,4) & = 6 \\
 & & \\
E_2(V_1) = \max(\{7\} \cup \emptyset) & = 7 & = 7 \\
E_2(V_2) = \max(\{6\} \cup \{\min(5,6), \min(7,6)\}) & = \max(6,5,6) & = 6 \\
E_2(V_3) = \max(\{6\} \cup \{\min(7,6), \min(7,6)\}) & = \max(6,6,6) & = 6 \\
E_2(V_4) = \max(\{7\} \cup \{\min(9,7), \min(6,6), \min(6,6)\}) & = \max(7,7,6,6) & = 7 \\
E_2(V_5) = \max(\{7\} \cup \{\min(9,7), \min(6,6), \min(6,6)\}) & = \max(7,7,6,6) & = 7 \\
E_2(V_6) = \max(\{6\} \cup \emptyset) & = 6 & = 6 \\
E_2(V_7) = \max(\{6\} \cup \{\min(7,6)\}) & = \max(6,6) & = 6 \\
E_2(V_8) = \max(\{6\} \cup \{\min(6,6), \min(6,6)\}) & = \max(6,6,6) & = 6
\end{array}
$$

fore, the final exploitabilities of the various vulnerabilities are:

$$
\begin{array}{llll}
E(V_1) = 7 & E(V_2) = 6 & E(V_3) = 6 & E(V_4) = 7 \\
E(V_5) = 7 & E(V_6) = 6 & E(V_7) = 6 & E(V_8) = 6 \ .
\end{array}
$$

4.4 Step 4: assessing the risk

As said in Section 3, the risk is a function of the damage potential and the exploitability; the damage potential has been defined in the context of the present case study as the number of affected users multiplied by the average number of days the compromised service does not operate. Since the number of days lies in the range $0 \ldots 10$ because of the internal organisation of the technical staff together with the disposal of backups for both the hardware and the software, and since the number of potential "interesting" users is less than 1000 (roughly the number of students using the DICOM services), the damage potential is a number in the interval 0 to 10000.

In addition, since the exploitabilities lie in the range $0 \ldots 10$ and a good psychological range to measure the risk is the interval $0 \ldots 10$, the risk is calculated by the function

$$r(d, e) = d \cdot e \cdot 1/10000 \ ,$$

where d is the damage potential and e the exploitability. The rationale of the formula is that the damage potential is weighted by the exploitability and, then, normalised to the desired range of values.

At this point, it is possible to aggregate the exploitability values along the attack tree to evaluate the risk connected to each node as defined in Section 3. The damage potential has been evaluated as follows:

- if a threat damages a service available to the students, then the number of affected users is considered to be 1000, the student population;
- if a threat is confined to the department, the affected users are the staff members, thus the population is 50;
- an evident intrusion, immediately visible to everyone using the system, requires two days on average to be fixed since, during the weekends, the network is unattended;
- a non-evident threat, reported in the system logs but not immediate to deduce from the usual system behaviour, requires four days on average to be discovered and fixed, since the network is well monitored;
- an hidden threat, involving unauthorised manipulation of data, like SQL injection, may require ten days to be discovered and removed; in fact, it requires code fixing by the service developers, who are not usually part of the technical staff deputed to manage the network.

The damage potential of the internal nodes of the attack tree has been calculated as the maximum of the affected population in a subtree multiplied by the maximum of the days in a subtree (not necessarily the same), hence getting the final result depicted in Figure 8 which is the same as Figure 4 augmented with the risk evaluation. Summarising the outcomes, the risk analysis on the interesting nodes shows that:

- the three main attack vectors, accessing the system, disabling a service and gaining unauthorised, have risks values 0.35, 7 and 0.35, respectively, thus the major risk is on disabling a service;
- the major risk sources, i.e., the vulnerabilities generating the maximal risk factors, are V_4 and V_5, two weak points in the code of the name server;
- the stemming risk source in the "accessing the system" subtree is given by V_1, a bug in the code of the SSH server.

Goal: Violation of the network security ($e = 7$, $d = 10000$, $r = 7$)
1. Accessing the system ($e = 7$, $d = 500$, $r = 0.35$)
  1.1 As a superuser ($e = 7$, $d = 500$, $r = 0.35$)
    1.1.1 Climbing privileges ($e = 7$, $d = 500$, $r = 0.35$)
      1.1.1.1 Accessing as a user ($e = 7$, $d = 500$, $r = 0.35$)
  1.2 As a user ($e = 7$, $d = 500$, $r = 0.35$)
    1.2.1 Shell code ($e = 7$, $d = 500$, $r = 0.35$)
      1.2.1.1 Race condition [V_1] ($e = 7$, $d = 500$, $r = 0.35$)
    1.2.2 Brute force [V_2] ($e = 6$, $d = 150$, $r = 0.09$)
    1.2.3 Password cracking [V_6 and V_8] ($e = 6$, $d = 200$, $r = 0.12$)
    1.2.4 Information disclosure [V_3 and V_6] ($e = 6$, $d = 500$, $r = 0.3$)
2. Disabling a service ($e = 7$, $d = 10000$, $r = 7$)
  2.1 Denial of service (DoS) ($e = 7$, $d = 3000$, $r = 2.1$)
    2.1.1 Assertion failure [V_4] ($e = 7$, $d = 3000$, $r = 2.1$)
    2.1.2 Insist failure [V_5] ($e = 7$, $d = 3000$, $r = 2.1$)
    2.1.3 Accessing the system as a user ($e = 7$, $d = 500$, $r = 0.35$)
    2.1.4 Race condition [V_1] ($e = 7$, $d = 500$, $r = 0.35$)
    2.1.5 Accessing as a superuser ($e = 7$, $d = 500$, $r = 0.35$)
  2.2 Distributed denial of service (DDoS) ($e = 7$, $d = 3000$, $r = 2.1$)
3. Gaining unauthorised information ($e = 7$, $d = 500$, $r = 0.35$)
  3.1 Accessing as a superuser ($e = 7$, $d = 500$, $r = 0.35$)
  3.2 Accessing as a user ($e = 7$, $d = 500$, $r = 0.35$)
  3.3 SQL injection ($e = 6$, $d = 500$, $r = 0.3$)
    3.3.1 Cross site tracing (XST) ($e = 6$, $d = 200$, $r = 0.12$)
      3.3.1.1 TRACE/TRACK HTTP methods enabled [V_7]
        ($e = 6$, $d = 200$, $r = 0.12$)
    3.3.2 Information disclosure [V_2, V_3 and V_8] ($e = 6$, $d = 500$, $r = 0.3$)

Fig. 8. The attack tree and the associated risks

5 Evaluation and impact

The purpose to perform a risk assessment is to adopt the most effective countermeasures to contrast the attackers and to limit the possible damages to a system. In the described case study, the first and simplest actions that reduce the risks and, thus, to raise the security level of the private DICOM network are:

- to update the name server to remove or, at least, to make more difficult to misuse the V_4 and V_5 vulnerabilities.
- to update the SSH server to remove the V_1 vulnerability.

The proposed solutions are rapid and inexpensive to implement and they address the main sources of risk as deduced from the analysis conducted in Section 4.4.

However, if the exploitability level of the vulnerabilities V_1, V_4, and V_5 is reduced or even nullified, the values of the other system vulnerabilities remain the same. In the depicted scenario, it is reasonable to insert some countermeasures to fortify the other vulnerabilities and to improve the overall security level.

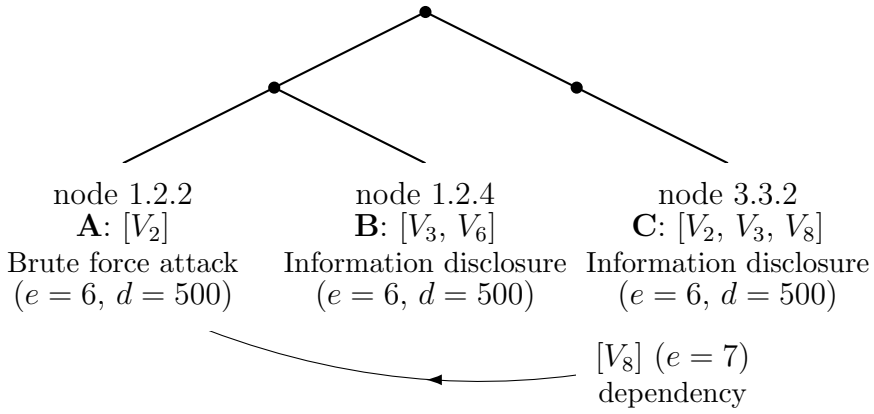

Fig. 9. The fragment of the attack tree related to GSSAPI support

Therefore, aside the software updating of the SSH server, it sounds reasonable to remove the GSSAPI support to mitigate the V_2 and V_3 vulnerabilities. Looking at the attack tree and at the dependence graph, summarised in Figure 9, the effect of this countermeasure is limited to the A, B and C nodes. In the A node, the countermeasure lowers the initial exploitability value, but the final value does not change thanks to the dependency on V_8; in the B node, the countermeasure is ineffective since the V_6 vulnerability has still its negative influence; in the C node, the final exploitability value of V_8 remains unchanged, thus making the countermeasure ineffective. Hence, in contrast to what one may expect, the removal of the GSSAPI support is not beneficial, although not dangerous: the reason is the combination of this countermeasure with the other system vulnerabilities. This example of countermeasure has been selected to show how, in practice, an "obviously good" security solution may reveal itself as completely ineffective.

In fact, the countermeasures that have been adopted to secure the private Dicom network are the upgrade of the SSH and the name server, the removal of the GSSAPI support together with the removal of the SSH1 support (that gave raise to the V_8 vulnerability) and, finally, the disabling of the TRACE/TRACK HTTP methods (the source of the V_7 vulnerability) in the web front-end along with an accurate cleaning of the web trees (greatly mitigating the V_6 vulnerability): this combined set of countermeasures reduces the risk of the 1,2 and 3 nodes in Figure 8 to far less than 1, which has been judged as an acceptable threshold. In fact, this threshold value is justified since it almost forces the elimination (exploitability less than 1) of the worst vulnerabilities and attacks, the ones difficult to detect (ten or more days to manage) and affecting a large population (the students).

Therefore, the evaluation of the sources of risk, and their removal or mitigation in conjunction with the study of the effect of dependencies, allowed to greatly improve the overall security of the DICOM network in a mostly inexpensive way.

The impact of the risk assessment procedure on the system management activity of the DICOM network is worth a few comments. In fact, after the experimental phase, whose outcomes are reported in this work, the risk assessment has become the main instrument to direct the security policy of the Department: it has proved to be effective, simple to use and flexible. Moreover, it is used to measure the effectiveness of the design choices of the new net services by means of the simulation of their placement in the DICOM network architecture along with a rough estimation of their exploitability values and their damage potentials. In this way, the risk assessment procedure becomes not only an instrument to analyse the security posture of the network, but also a precious tool to predict the evolution of the network infrastructure, thus aiding both its design and management.

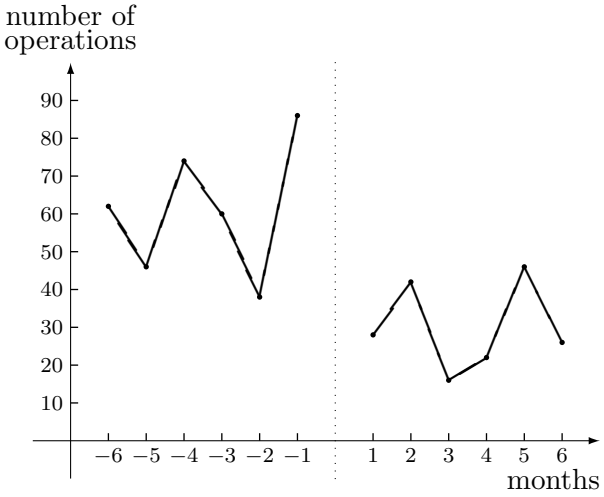

Fig. 10. Frequency of security related operations

In practice, a few measures of the impact have been taken: they do not possess the quality of a real statistical study and the observation time has been too limited to consider the results as conclusive, but, still, the derived consequences may have a general interest. In particular, the frequency of the security related operations, the number of hours per month dedicated to the network administration and the number of hours per month dedicated to the risk assessment activity (as part of the usual network administration) have been traced for six months: the results are shown in Figures 10 and 11. The negative months represent the period before the introduction of the risk assessment and the dotted vertical line indicates its adoption. In Figure 11, the upper curve on the right measures the overall management effort, while

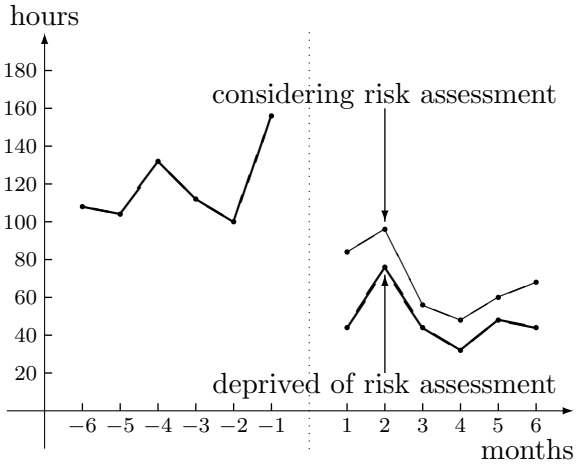

Fig. 11. Hours per month dedicated to the network management

the lower one measures the management effort deprived of the risk analysis maintenance.

The graphs allow to deduce that the risk assessment and the consequent strengthening of the network security reduced the occurrence of the attacks, as clearly reported in Figure 10; moreover, the technical staff reduced its working time, mainly due to the improvement in the network security, and thus, to the reduced frequency of attacks. The overhead due to the need to constantly update the risk analysis developed insofar has been evidenced by the double function in Figure 11. At the moment, this overhead is largely compensated by the improvements in the network security: there is still no enough data to ascertain if this compensation remains in the long term.

6 Related work

Even though the application of risk evaluation methods has been widely discussed and analysed, see, e.g., [1,13,20,32], among information security experts there appears to be no agreement regarding the best or the most appropriate method to assess the probability of computer incidents [30].

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. In particular, we have found of interest Baskerville's description [5] 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. [16] 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, [28] provides a probabilistic quantitative model that measures the security risk.

In comparison, our approach, starting from its initial definition in [4], has been based on the structured evaluation of single vulnerabilities along with their mutual dependencies. In this respect, the results in [16] are similar to ours, although they do not propose a formal methodology 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 in order to derive its characterising properties.

Furthermore, there are approaches employing a graph-based representation of systems and their vulnerabilities, that provide methodologies whose properties are, at least partially, mathematically analysed. Among those approaches, of prominent interest are those based on attack graphs [26, 31], where state-transition diagrams are used to model complex attack patterns. In particular, [26] proposes the use of attack graphs to automate the step of hardening a network against a multi-step intrusion. The proposed security solution is expressed as an adjustable network configuration rather than a set of countermeasures to possible exploits.

Specifically, [24] divides a system into sub-domains and each sub-domain could be characterised by vulnerabilities. Applying probability theory and graph transformations, [24] evaluates the possibility that a malicious data 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 [31].

In this respect, our approach is simpler both as a method and in its formalisation. Despite its simplicity, our results are stronger on the mathematical side and this application to a real-case study makes evident the practical value of the method in real-world situations. In fact, we use the attack tree model [23,29] to evaluate the security threats combining them with the dependence graph, a formalisation of a piece of experts' knowledge. This combination is 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.

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 [9], leading to incorrect or imprecise models. In order to avoid this problem, complete models should be specified taking into account system interconnections. In agreement with this point of view [9,14,16,28,30] present models for assessing security risks taking into account interdependence between components.

Particularly, [9] 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 a number of interesting interdependencies among components. Similarly, [14] defines a model based on security policies and individual risks. The model allows to evaluate if the risk associated to each transaction is acceptable. The evaluation of risk also takes into account contextual information.

With respect to this family of risk assessment methodologies, whose goal is to evaluate the likelihood of a failure in the design of a complex software system, rather than to assess the risk of a malicious intrusion into a telecommunication network, our method appears to be an ad-hoc method. In fact, it has been conceived to analyse the security of a computer network, and, although it can be used in the analysis of information system designs, and, therefore, it may be compared with methodologies in this area, its origin is quite evident.

As a matter of fact, independently from their application areas, the risk assessment methodologies have a core weakness: the use of subjective metrics. In fact, in the scientific community the main criticism to these methodologies is about the fact that values, assigned on the basis of a personal knowledge and experience are regarded as *random*, 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. In fact, a metric is defined [3] as 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 [17].

In agreement with him, we consider measurement as the process by which numbers or symbols are assigned to attributes of entities, in our case to the exploitability of a vulnerability. Therefore, even though there is no widely recognised way to assess risks and to evaluate the induced damages, there are

various approaches that provide methodologies by which the risk evaluation becomes more systematic.

In particular, Sharp et al. [30] develop 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. Similarly, Z. Dwaikat et al. [14] define security requirements for transactions and provide mechanisms to measure likelihood of violation of these requirements.

Looking toward risk assessment as a decision support tool, Fenton [18] proposes 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 of risks*. Therefore, in agreement with [9, 16, 18, 28], the information computed by our model can be used as a decision support.

Summarising, risk assessment methods have been widely studied and we tried to compare our approach to the most representative methods of the various classes of paradigms. Nevertheless, few in-depth case studies are present in literature.

In our opinion, a complete and real case study is the natural complement to the description of an abstract method of risk analysis: real-world systems and networks are complex and difficult to analyse, but they provide a unique opportunity to demonstrate how a risk assessment can be conducted. Moreover, applying an abstract method to a real case is greatly simplified when a strong guideline is given, and a complete case study offers such a guideline.

In this respect, there are documented applications of the Coras approach [27], mainly the tele-cardiology service operated in Crete [34,35]. Also, there are reports of the application of Octave [1] to e-health applications and the Arpanet network. Unfortunately, these applications demonstrate the successful realisation of the cited approaches, but their reports are not detailed enough to serve as practical guidelines to the application of the corresponding methods in analogous real-world situations.

Therefore, the chosen risk assessment method produces the best result when it is applied in an environment where the experts can easily give a *relative* judgement on the severity of potential vulnerabilities, where the dependencies among vulnerabilities are locally clear, but globally obscure because of

the complexity of the architectural level. Both requirements are completely fulfilled by the network infrastructure of a Department of Computer Science.

7 Conclusion

This article reports the risk analysis of the private DICOM network supposing that the attack comes from the outside. In our opinion, there are a number of reasons why this account can be of interest: it represents a real case study; it shows how to apply in practice an abstract method; it gives some hints on the measurable benefits of adopting a risk assessment method as a standard practice in the administration of a network.

In these respects, this article generalises and specialises our previous results [4, 6–8] since it shows how an objective judgement of the initial values can be obtained by means of a security scanner whose outcomes are made contextual in a predicable way and, at the same time, this report tries to measure, although to a limited extent, the consequences of our analysis, in particular, how the adoption of risk analysis has become a standard practice in the administration of our network and what benefits occurred because of this.

In conclusion, we have shown on a concrete, real case how to apply a formal risk assessment method: our belief is that the same pattern we followed can be replicated in similar situations. Moreover, we have begun to analyse the positive results of our effort and they appear to be promising: honestly, these measures do not possess the width and the depth of a strong statistical analysis but, in any case, they have been presented and should be regarded as an indication of a trend. In fact, we strongly believe that the adoption of a formal risk assessment method in the standard practice of system and network administration produces more secure networks, less incidents, at a reasonable cost and we tried to convey this idea to the reader: our measures show that the increment in the management work induced by the need to maintain a valid risk model is largely compensated by the decrement due to the reduced incidence of security problems both in their occurrence and in their impact.

References

[1] C. Alberts, A. Dorofee, J. Stevens, and C. Woody. Introduction to the Octave approach, October 2003.

[2] P. Albitz and C. Liu. *DNS and BIND*. O'Reilly, 5th edition, May 2006.

[3] S. Arshad, M. Shoaib, and A. Shah. Web metrics: The way of improvement of quality of non web-based systems. In H. R. Arabnia and H. Reza, editors, *Proceedings of the International Conference on Software Engineering Research and Practice*, volume 2, pages 489–495. CSREA Press, 2006.

[4] D. Balzarotti, M. Monga, and S. Sicari. Assessing the risk of using vulnerable components. In D. Gollmann, F. Massacci, and A. Yautsiukhin, editors, *Quality of Protection — Security Measurements and Metrics*, volume 23 of *Advances in Information Security*, pages 65–78. Springer-Verlag, New York, NY, USA, June 2006.

[5] R. Baskerville. Information system security design methods: Implications for information systems development. *ACM Computing Survey*, 25(4):375–412, 1993.

[6] M. Benini and S. Sicari. A mathematical framework for risk assessment. In H. Labiod and M. Badra, editors, *New Technologies, Mobility and Security*, Signals and Communication, pages 459–469. Springer-Verlag, May 2007.

[7] M. Benini and S. Sicari. Risk assessment: Intercepting VoIP calls. In *Proceedings of the V-IPSI 2007 Venice Conference*, March 2007.

[8] M. Benini and S. Sicari. Assessing the risk to intercept VoIP calls. *Journal of Computer Networks*, to appear, 2008.

[9] G. Biswas, K.A. Debelak, and K. Kawamura. Application of qualitative modelling to knowledge-based risk assessment studies. In M. Ali, editor, *Proceedings of the Second International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 1, pages 92–101. ACM Press, New York, NY, USA, 1989.

[10] `http://www.nongnu.org/cvs`.

[11] `http://nvd.nist.gov`.

[12] `http://www.first.org/cvss`.

[13] F. den Braber, T. Dimitrakos, B.A. Gran, M.S. Lund, K. Stølen, and J.Ø. Aagedal. The CORAS methodology: Model-based risk management using UML and UP. In L. Favre, editor, *UML and the Unified Process*, pages 332–357. IRM Press, 2003.

[14] Z. Dwaikat and F. Parisi-Presicce. Risky trust: Risk-based analysis of software system. In *Proceedings of the 2005 Workshop on Software Engineering for Secure Systems — Building Trustworthy Applications*, pages 1–7. ACM Press, New York, NY, USA, 2005.

[15] D. Eastlake. RFC 2535: Domain name system security extensions, March 1999.

[16] S. Evans, D. Heinbuch, E.Kyle, J. Piorkowski, and J.Wallener. Risk-based system security engineering: Stopping attacks with intention. *IEEE Security & Privacy Magazine*, 2(6):59–62, 2004.

[17] N. Fenton. Software measurement: A necessary scientific basis. *IEEE Transactions on Software Engineering*, 20(3):199–206, 1994.

[18] N. Fenton and M. Neil. Making decisions: Bayesian nets and mcda. *Knowledge-Based Systems*, 14(7):307–325, November 2001.

[19] M. Howard and D. Leblanc. *Writing Secure Code*. Microsoft Press, 2003.

[20] B. Jenkins. Risk analysis helps establish a good security posture; risk management keeps it that way, 1998. White paper.

[21] J. Linn. RFC 2743: Generic security service application program interface version 2, update 1, January 2000.

[22] C. May, M. Baker, D. Gabbard, T. Good, G. Grimes, M. Holmgren, R. Nolan, R. Nowak, and S. Pennline. Advanced information assurance handbook. Technical Report CMU/SEI-2004-HB-001, CERT/CC Training and Education Center, March 2004.

[23] A.P. Moore and R.J. Ellison. Survivability through intrusion-aware design. Technical Report 2001-TN-001, CERT Coordination Center, 2001.

[24] I.S. Moskowitz and M.H. Kang. An insecurity flow model. In *Proceedings of the 1997 Workshop on New Security Paradigms*, pages 61–74, ACM Press, New York, NY, USA, 1997.

[25] `http://www.nessus.org`.

[26] S. Noel, S. Jajoidia, B. O'Berry, and M. Jacobs. Efficient minimum-cost network hardening via exploit dependency graphs. In *Proceedings of 19th Annual Computer Security Applications Conference*, pages 86–95. IEEE Computer Society, 2003.

[27] D. Raptis, T. Dimitrakos, B.A. Gran, and K. Stølen. The CORAS approach for model-based risk analysis applied to the e-commerce domain. In *Proceedings of Communication and Multimedia Security*, pages 169–181. Kluwer, 2002.

[28] M. Sahinoglu. Security meter: A practical decision-tree model to quantify risk. *IEEE Security & Privacy*, 3(3):18–24, May/June 2005.

[29] B. Schneier. Attack trees. *Dr. Dobb's Journal*, 24(12):21–29, December 1999.

[30] G.P. Sharp, P.H. Enslow, S.B. Navathe, and F. Farahmand. Managing vulnerabilities of information system to security incidents. In *Proceedings of the 5th International Conference on Electronic Commerce*, pages 348–354. ACM Press, New York, NY, USA, 2003.

[31] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing. Automated generation and analysis of attack graphs. In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*, pages 273–284. IEEE Computer Society, Washington, DC, USA, 2002.

[32] T. Siu. Risk-eye for the IT security guy, February 2004.

[33] P. Srisuresh and M. Holdrege. RFC 2663: IP network address translator (NAT) terminology and considerations, August 1999.

[34] Y. Stamatiou, E. Skipenes, E. Henriksen, N. Stathiakis, A. Sikianakis, E. Charalambous, N. Antonakis, K. Stølen, F. den Braber, M.S. Lund, K. Papadaki, and G. Valvis. The CORAS approach for model-based risk management applied to a telemedicine service. In *Proceedings of Medical Informatics Europe*, pages 206–211. IOS Press, 2003.

[35] N. Stathiakis, C. Chronaki, E. Skipenes, E. Henriksen, E. Charalambous, A. Sykianakis, G. Vrouchos, N. Antonakis, M. Tsiknakis, and S. Orphanoudakis. Risk assessment of a cardiology eHealth service in HYGEIAnet. In *Proceedings of Computers in Cardiology*, pages 201–204. IEEE, 2003.

[36] T. Ylonen and C. Lonvick. RFC4251: The secure shell (SSH) protocol architecture, January 2006.