Inf Syst E-Bus Manage (2012) 10:433–453
DOI 10.1007/s10257-011-0177-1

ORIGINAL ARTICLE

Defining and computing a value based cyber-security measure

Anis Ben Aissa · Robert K. Abercrombie · Frederick T. Sheldon · Ali Mili

Received: 31 July 2010 / Revised: 21 March 2011 / Accepted: 2 April 2011 /
Published online: 23 April 2011

Abstract In earlier work, we presented a value based measure of cybersecurity that quantifies the security of a system in concrete terms, specifically, in terms of how much each system stakeholder stands to lose (in dollars per hour of operation) as a result of security threats and system vulnerabilities; our metric varies according to the stakes that each stakeholder has in meeting each security requirement. In this paper, we discuss the specification and design of a system that collects, updates, and maintains all the information that pertains to estimating our cybersecurity measure, and offers stakeholders quantitative means to make security-related decisions.

A. B. Aissa
Faculty of Sciences of Tunisia, University of Tunis El Manar,
Tunis 2092, Tunisia
e-mail: anis_enit@yahoo.fr

R. K. Abercrombie · F. T. Sheldon
Oak Ridge National Laboratory, Oak Ridge, TN 37831, USA
e-mail: abercrombier@ornl.gov

F. T. Sheldon
e-mail: sheldonft@ornl.gov

A. Mili (✉)
College of Computing Sciences, New Jersey Institute of Technology,
Newark, NJ 07102-1982, USA
e-mail: mili@cis.njit.edu

Keywords Cyber security metrics · Risk management · Information security · Algorithms · Measurement · Performance · Design · Economics · Reliability · Experimentation · Security · Theory · Verification

1 Introduction

Recently, Ben Aissa et al. introduce a value-based metric that quantifies the security of a computing system by the statistical mean of the random variable that represents for each stakeholder, the amount of loss that results from security threats and system vulnerabilities (Ben-Aissa et al. 2010). This metric, which we call the Mean Failure Cost (**MFC**), varies by stakeholder, and takes into account the variance of the stakes that a stakeholder has in meeting each security requirement. In addition, it can be extended beyond security to capture other aspects of dependability, such as reliability, availability, safety, since it makes no distinction about what causes the potential loss/cost. Finally, it makes no distinction between reliability and safety, since it captures (over a continuum of costs) all potential losses, regardless of whether they stem from failing a high stake/low probability requirement (safety) or from a medium stake/medium probability requirement (reliability).

In this paper, we briefly introduce the metric of Mean Failure Cost, analyze it with respect to relevant standards, and then discuss its automation and its applications. In Sect. 2, we present the **MFC** as a cascade of linear models and discuss how the models are deployed in practice by collecting data from various relevant parties. In Sect. 3, we present the functional specifications of a tool that automates the calculation of the Mean Failure Cost for the stakeholders of a system, and supports the use of **MFC** for the quantitative decision-making. In Sect. 4, we analyze the proposed metric with respect to relevant standards, and in Sect. 5, we conclude by assessing our results and sketching directions of future research.

2 Mean failure cost: a cascade of linear models

2.1 The stakes matrix

We consider a system S and we let H_1, H_2, H_3, … H_k, be stakeholders of the system, that is, parties that have a stake in its operation. We let R_1, R_2, R_3, … R_n, be security requirements that we wish to impose on the system, and we let $ST_{i,j}$, for $1 \leq i \leq k$ and $1 \leq j \leq n$ be the stake that stakeholder H_i has in meeting security requirement R_j. In other words, whenever system S fails to meet requirement R_j, it causes stakeholder H_i a loss valued at $ST_{i,j}$. We let PR_j, for $1 \leq j \leq n$, be the probability that the system fails to meet security requirement R_j during a unit of operation time (say, 1 h), and we let $\mathbf{MFC}_i$ (Mean Failure Cost), for $1 \leq i \leq k$, be the random variable that represents the cost to stakeholder H_i that may result from a security failure.

We quantify this random variable in terms of financial loss per unit of operation time (e.g., \$/h); it represents the mean loss that the stakeholder may experience as a

result of a security failure. Under some assumptions of statistical independence, we find that the Mean Failure Cost for stakeholder H_i can be written as:

$$MFC_i = \sum_{1 \le j \le n} ST_{i,j} \times PR_j.$$

If we let **MFC** be the column-vector of size k that represents mean failure costs, let ST be the $k \times n$ matrix that represents stakes, and let PR be the column-vector of size n that represents probabilities of failing security requirements, then this can be written using the matrix product (°):

$$\mathbf{MFC} = \mathbf{ST} \circ \mathbf{PR}.$$

The Stakes matrix is filled, row by row, by the corresponding stakeholders is depicted in Fig. 1. As for **PR**, we discuss below how to generate it.

2.2 The dependency matrix

We consider the architecture of system S, and let C_1, C_2, C_3, … C_h, be the components of system S. Whether a particular security requirement is met or not may conceivably depend on which component of the system architecture is operational. If we assume that no more than one component of the architecture may fail at any time, and define the following events:

- E_i, $1 \le i \le h$, is the event: the operation of component C_i is affected due to a security breakdown.
- E_{h+1}: No component is affected.

Given a set of complementary events E_1, E_2, E_3, … E_h, E_{h+1}, we know that the probability of an event F can be written in terms of conditional probabilities as:

Fig. 1 The stakes matrix

ST		**Requirements**							
		R_1	…R_j…						R_n
Stakeholders	H_1								
	…H_i…								
			Stake that stakeholders H_i has in meeting requirement R_j						
	H_m								

$$P(F) = \sum_{k=1}^{h+1} P(F|E_k) \times P(E_k).$$

We instantiate this formula with F being the event: the system fails with respect to some security requirement. To this effect, we let F_j denote the event that the system fails with respect to requirement R_j and we write (given that the probability of failure with respect to R_j is denoted by PR_j):

$$PR_j = \sum_{k=1}^{h+1} P\left(F_j|E_k\right) \times P(E_k).$$

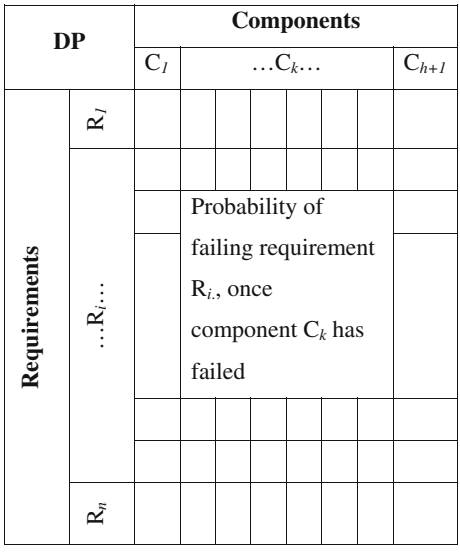
If

- we introduce the **DP** (Dependency) matrix, which has n rows and $h + 1$ columns, and where the entry at row j and column k is the probability that the system fails with respect to security requirement R_j given that component Ck has failed (or, for $k = h + 1$, that no component has failed),
- we introduce vector **PE** of size $h + 1$, such that PE_k is the probability of event E_k, then we can write

$$\mathbf{PR} = \mathbf{DP} \circ \mathbf{PE}.$$

Matrix **DP** can be derived by the system's architect, in light of the role that each component of the architecture plays to achieve each security requirement as depicted in Fig. 2. As for deriving vector **PE**, we discuss this matter in the next section.

Fig. 2 The dependency matrix



2.3 The impact matrix

Components of the architecture may fail to operate properly as a result of security breakdowns brought about by malicious activity. In order to continue the analysis, we must specify the catalog of threats that we are dealing with, in the same way that analysts of a system's reliability define a fault model as a catalog of faults that they are considering. To this effect, we consider the set of security threats that we are facing, and we let T_1, T_2, T_3, … T_p, represent the event that a cataloged threat has materialized during a unit of operational time, and we let T_{p+1}, be the event that no threat has materialized during a unit of operational time. Also, we let *PT* be the vector of size $p + 1$ such that

- PT_q, for $1 \leq q \leq p$, is the probability that threat T_q has materialized during a unitary period of operation (say, 1 h).
- PT_{p+1} is the probability that no threat has materialized during a unitary period of operation time.

Then, by virtue of the probabilistic identity cited above, we can write:

$$PE_k = \sum_{q=1}^{p+1} P\left(E_k | T_q\right) \times PT_q.$$

If

- we introduce the **IM** (Impact) matrix, which has $h + 1$ rows and $p + 1$ columns, and where the entry at row k and column q is the probability that component C_k fails given that threat q has materialized (or, for $q = p + 1$, that no threat has materialized),
- we introduce vector **PT** of size $p + 1$, such that PT_q is the probability of event T_q,

then we can write

$$\mathbf{PE} = \mathbf{IM} \circ \mathbf{PT}$$

Matrix **IM** can be derived by analyzing which threats affect which components, and assessing the likelihood of success of each threat, in light of perpetrator behavior and possible countermeasures as depicted in Fig. 3.

Vector **PT** can be derived from known perpetrator behavior, perpetrator models, known system vulnerabilities, etc. We refer to this vector as the Threat Configuration Vector or simply as the Threat Vector and is depicted in Fig. 4.

2.4 Summary of mean failure costs and the cascading of linear models

Given the stakes matrix **ST**, the dependability matrix **DP**, the impact matrix **IM** and the threat vector **PT**, we can derive the vector of mean failure costs (one entry per stakeholder) by the following formula:

$$\mathbf{MFC} = \mathbf{ST} \circ \mathbf{DP} \circ \mathbf{IM} \circ \mathbf{PT}.$$

Fig. 3 The impact matrix

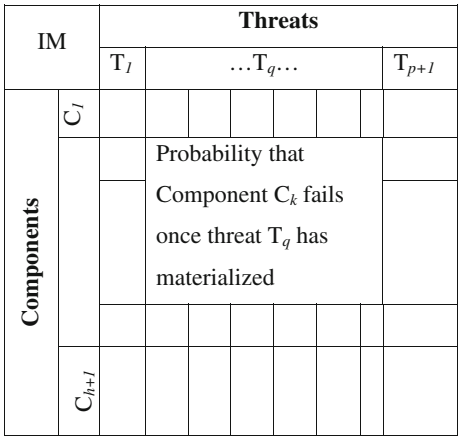
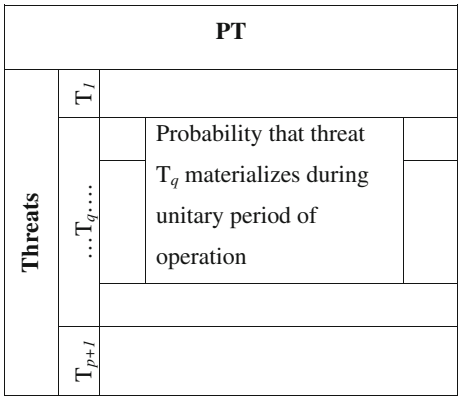


Fig. 4 The threat configuration vector



where matrix **ST** is derived collectively by the stakeholders, matrix **DP** is derived by the systems architect, matrix **IM** is derived by the security analyst from architectural information, and vector **PT** is derived by the security analyst from perpetrator models.

2.5 Applications

Once each stakeholder computes his/her mean failure cost, then each has the means to use it to make his/her own decisions. A sample of applications of the **MFC** includes:

- *Mean failure cost as a lower bound of hourly earnings.* Any stakeholder who is making less money per hour than his mean failure cost is in the wrong business and should bail out of the system. The mean overall gain of this stakeholder is statistically negative.

- *Mean failure cost as an upper bound of insurance premium.* The mean failure cost represents the upper bound that a stakeholder ought to agree to pay to an "insurance agent" to protect against the risk of security failures.
- *Mean failure cost as a basis for analyzing remedial measures*. If we are considering taking a measure to improve system security, how can we tell whether the measure is worthwhile, given that the deployment of the measure costs IC? We decompose this investment cost into the contribution that each stakeholder put forth towards deploying the security measure:

$$\mathrm{IC} = \mathrm{IC}_1 + \mathrm{IC}_2 + \mathrm{IC}_3 + \ldots + \mathrm{IC}_m.$$

Then each stakeholder can compute the Return on Investment (ROI) of the security measure in light of the following factors:

- His/her share of the investment cost,
- Investment parameters such as discount rate, investment cycle,
- The benefit gained from the security measure: this can be quantified by the **MFC** reduction that results from the security measure.

As to the question of how the investment cost gets divided between the stakeholders, we can cite two possible options (among many):

- In proportion to their respective **MFC** reductions,
- In such a way that all the stakeholders get the same ROI.

Whichever option is chosen, it is fair to consider that a security measure is worthwhile if and only if it yields a positive ROI for all stakeholders.

3 An automated tool for computing mean failure cost

We envision and have implemented a system that supports the archival of information pertaining to the **MFC**, as well as applications of the **MFC** metrics, such as: computing the return on investment for a given V&V action, or computing the return on investment for a given architectural enhancement (Ben-Aissa et al. 2010). Typically, functionality and its associated functional requirement tend to vary greatly between two requirement specifications, especially across different application domains. However, all applications need to specify levels of identification, authentication, authorization, integrity, privacy, etc. (Firesmith 2004). Conceptual models are emerging to apply a structure to collecting this data (Estevez et al. 2010). We are benefitting from this logic. Several distinct stakeholders intervene in the determination and entry of relevant information; hence a natural way to structure the requirements is by stakeholder. We identify individual stakeholders, then specify for each what functionalities that stakeholder expects from the system. We are writing the requirements specification of a system, whose purpose is to compute and use the **MFC** of systems of interest; in order to avoid confusion, we refer to the system we are specifying as *K* and to object systems that system K analyzes as *S*.

The following list presents the set of relevant stakeholders; for all intents and purposes, a stakeholder designates a role more than a person or a group of persons. For example, the same person or group of persons may act as more than one stakeholder.

- *The requirements engineer of system S*. This stakeholder is responsible for eliciting, organizing, and documenting the requirements of system S. In particular, he has the crucial task of structuring the requirements of system S for the purposes of the stakes matrix. The precision of the **MFC** calculations is dependent upon the decomposition of the system requirements into orthogonal (i.e. non-overlapping) components/clauses, a condition that is virtually impossible to meet in practice, but we maintain it as a criterion. Among the services that this stakeholder will expect from system K, we cite:
  - The ability to represent and store the clauses of the system requirement specification. Each clause can be represented by a name, a textual description, a graphic, or some appropriate mathematical notation.
  - The ability to modify a previously stored/structured requirements specification, with an automatic notification of all the stakeholders who must be informed (for example, all the users of the system who have previously entered cost information with respect to a decomposition must be informed if the decomposition has changed).
  - The ability to define default cost transfer options when a requirements specification is modified; for example, if a clause is simply renamed, then the cost information is automatically transferred; if a clause is split into two subclauses, then as a default we can divide associated failure costs in two; etc.

We envision and have implemented the following user interface for the requirements engineering team (Fig. 5).

- *The users of system S.* Each user is responsible for entering the failure costs that he/she associates with each component of the system requirements, as entered by the requirements engineer. Among the services that this stakeholder will expect from system K, we cite:
  - The ability to review clauses of the system (S) requirements specification and to associate failure costs to them, in terms of dollars.
  - The ability to address a question regarding a particular clause to the requirements engineer, requesting (for example) a clarification. The questions as well as the requirement engineer's reply remain on the public record attached to the clause (so that subsequent users can benefit from the clarification).
  - Computes and displays stakeholder **MFC**.
  - Computes and displays stakeholder ROI, for a variety of relevant decisions.
  - Approve or turn down a possible system wide change on the basis of own ROI/NPV value.
  - Performs what-if analyses, without impact on stored data.

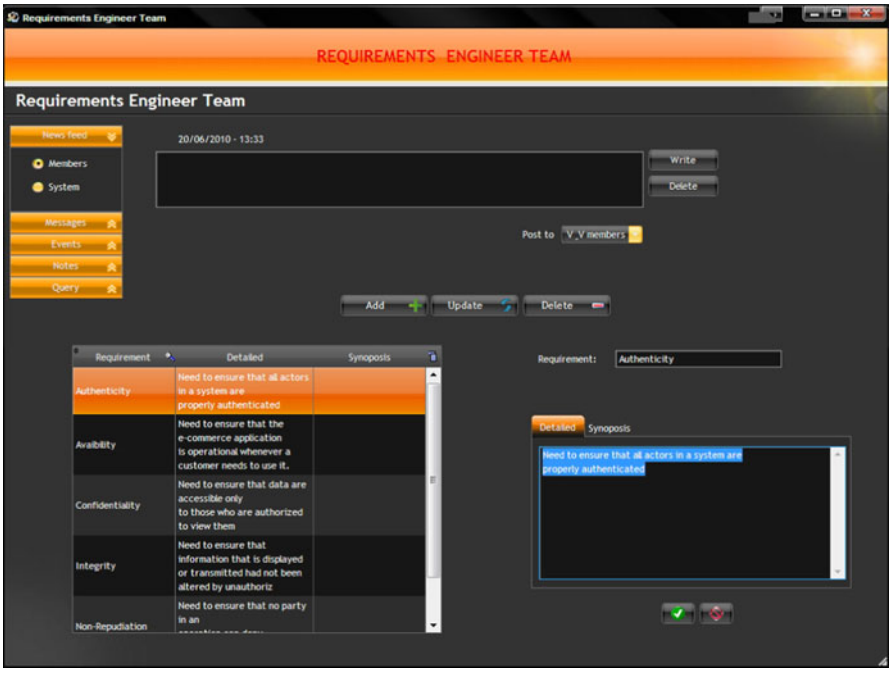


Fig. 5 Requirements engineering team interface

We envision and have implemented the following user interface for the users/stakeholders of system S (Fig. 6).

- *The architect of system S*. The architect of system S designs the architecture of the system based on the requirements entered by the requirements engineer and on any other architecture-relevant information such as non functional attributes; also, he/she is responsible for filling the dependency matrix by entering, for each clause R of the requirements specification and for each component C of the architecture, the probability that system S fails to satisfy clause R given that component C has failed. Note that a component, in our understanding here, may refer to a component or a connector, in the usual sense of architectural description languages (Garlan et al. 2009; Garlan et al. 2000). Among the services that this stakeholder will expect from system K, we cite:
  - The ability to record the architecture of system S, by a combination of text, hypertext, graphics, mathematical formulas, all appropriately interlinked.
  - The ability to query the requirements engineer about clauses of the requirements specification; the architect's query and the requirements engineer's reply both remain in the public record, for future reference.
  - The ability to help the architect estimate conditional probabilities of failure, using a catalog of architectural patterns.
  - The ability to modify the architecture, and to notify all relevant stakeholders (e.g. the V&V team) of the modification.

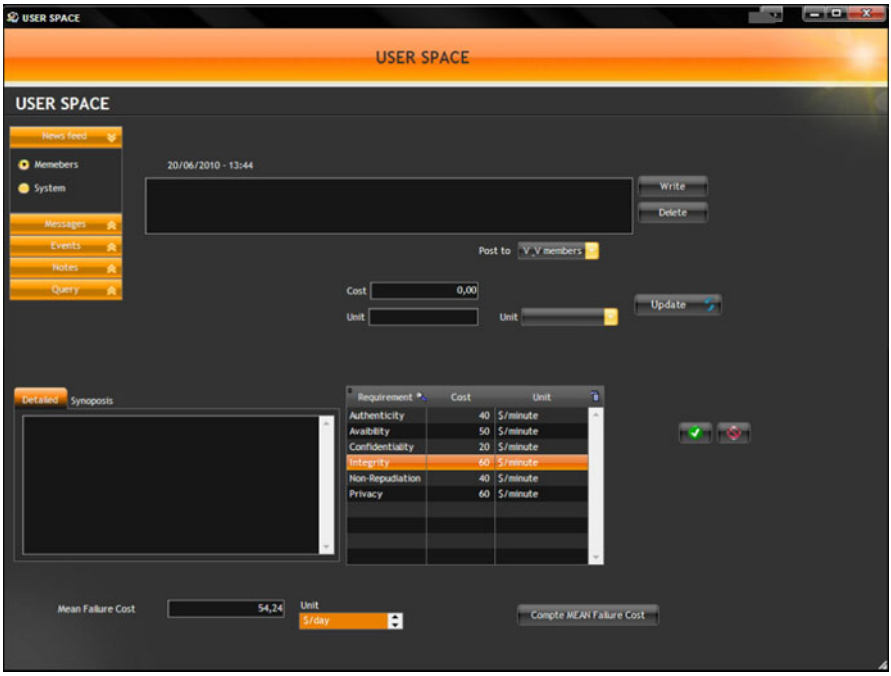


Fig. 6 Users/stakeholders interface

We envision the following user interface for the system architecture team (Fig. 7).

- *The verification and validation team*. Given a catalog of security threats, the Verification and Validation team is able to determine which components are likely to be affected by each threat, and with what probability each component may be affected once the threat has materialized. This team is, in effect, responsible for filling the impact matrix; they can enter the required information once the security team has entered the threat vector and the architecture team has entered the architecture of the system. Among the services that this stakeholder will expect from system K, we cite:
  - The ability to store and modify entries of the impact matrix.
  - The ability to support communication to the system architect, with notification of all relevant parties.
  - Assistance with filling entries of the impact matrix, using common patterns.

We envision the following user interface for the verification and validation team (Fig. 8).

- *The security team*. The main responsibility of this team is to fill the threat vector. Among the services that this stakeholder will expect from system K, we cite:

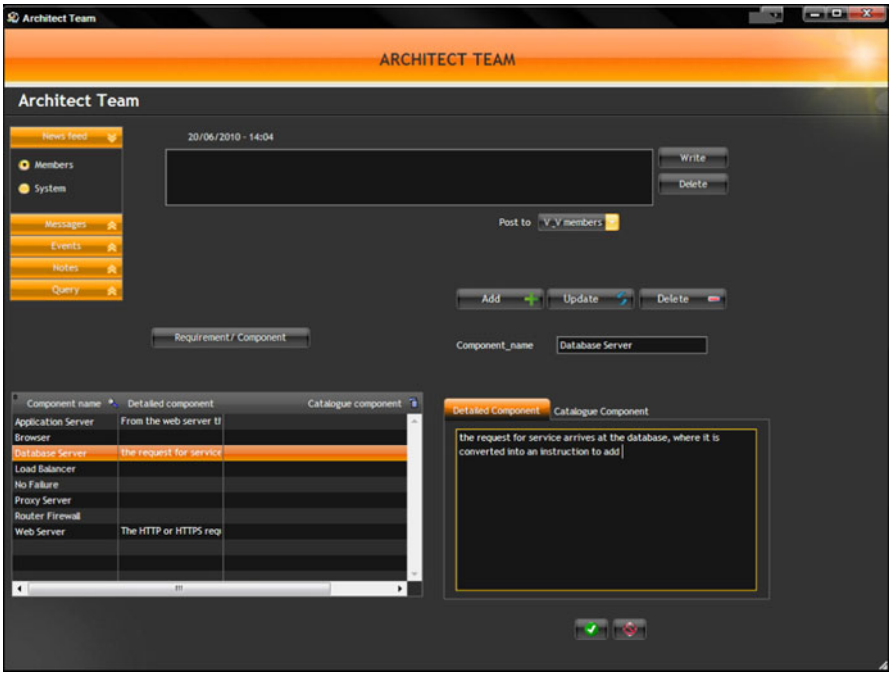


Fig. 7 Architect team interface

- Maintain an active vector of credible threats, possibly from an exhaustive list.
- Assign them occurrence probabilities.
- Assist the security team in computing these probabilities, using some common patterns (e.g. common modular redundancy schemes).

We envision the following user interface for the security team (Fig. 9).

- *The system administration team*. This team maintains information about global ROI (versus stakeholder ROI). Gives and monitors login and access privileges.
  - Subscribes/un-subscribes stakeholders.
  - Manages login information, access rights.
  - Initiates an ROI calculation.
  - Enters system wide factors.
  - Mediates negotiations between stakeholders.
  - Moderates cross postings between stakeholders.
  - Approves system-wide modifications.
  - Displays **MFC** data and ROI/NPV data for all users.
  - Perform What-If analyses on ROI data.

We envision the following user interface for the system administration team (Fig. 10).

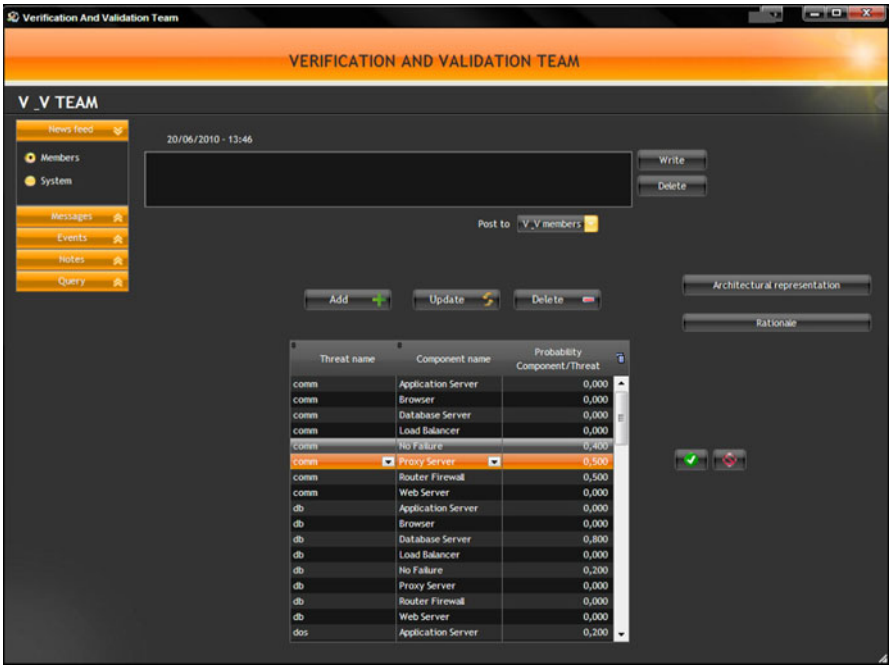


Fig. 8 Verification and validation team interface

- *General requirements*. The system should maintain, for each user, a log of required information, and an indication of when this information has been completed.

We envision and have implemented the following user interface for this function (Fig. 11).

3.1 Application of the automated tool for computing the mean failure cost

We have applied this tool to the sample e-commerce system discussed in (Ben-Aissa et al. 2010). In this sample e-commerce system, we recognize four stakeholders, namely: the customer, the merchant, the technical intermediary, and the financial intermediary. We briefly review the stakes that they have in meeting the security requirements, as these determine the corresponding values in the stakes matrix:

- *The customer*. The stakes that the customer has in the secure operation of the system include: the loss of confidential information, which the customer may provide during the ecommerce transaction; transaction failure; identity theft.
- *The merchant*. The stakes that the merchant has in the secure operation of the system include: the loss of business that may result from failing the availability requirement; the loss of customer loyalty that may result from failing the availability requirement; the loss of customer loyalty that may result from

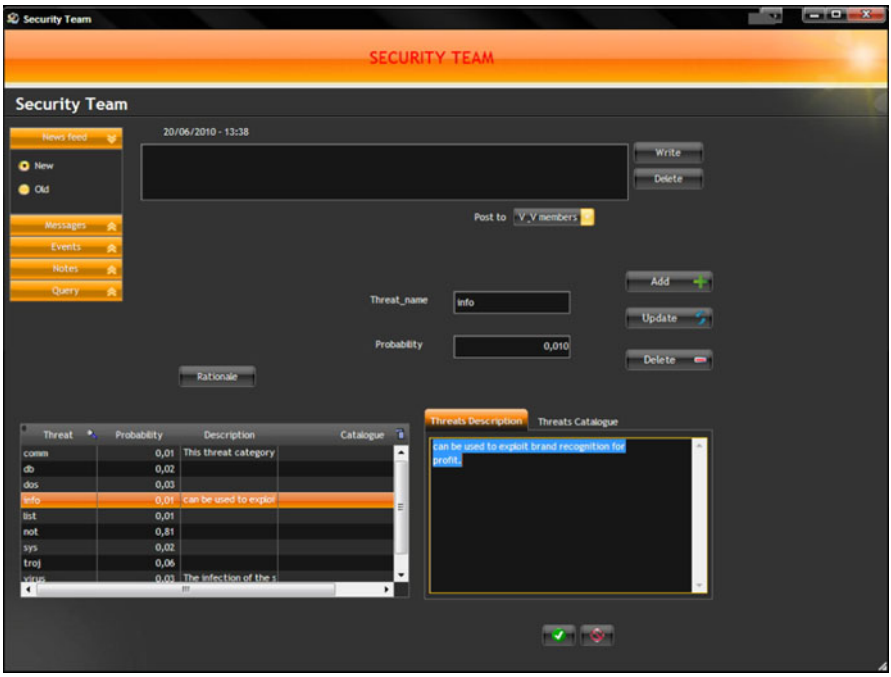


Fig. 9 Security team interface

failing the confidentiality or the privacy requirements; the loss of business that may result from failing the integrity requirement, etc.

- *The technical intermediary*. The stakes that the technical intermediary has in the secure operation of the system include: the loss of business from the merchant; the loss of reputation for good service, which may result in lost corporate value.
- *The financial intermediary*. The stakes that the financial intermediary has in the secure operation of the system include: financial losses that result from malicious activities by customers; the loss of business from the merchant; the loss of reputation for good service, which may result in lost corporate value.

Thus, we summarize the four stakeholders as:

- The customer, who buys from the e-commerce site,
- The merchant, who owns/operates the e-commerce site,
- The technical intermediary, who provides the technical infrastructure for the site,
- The financial intermediary, who mediates the financial transactions for the merchant.

Using data entered by the various partners in this situation, we find the following vector of mean failure costs (Table 1).

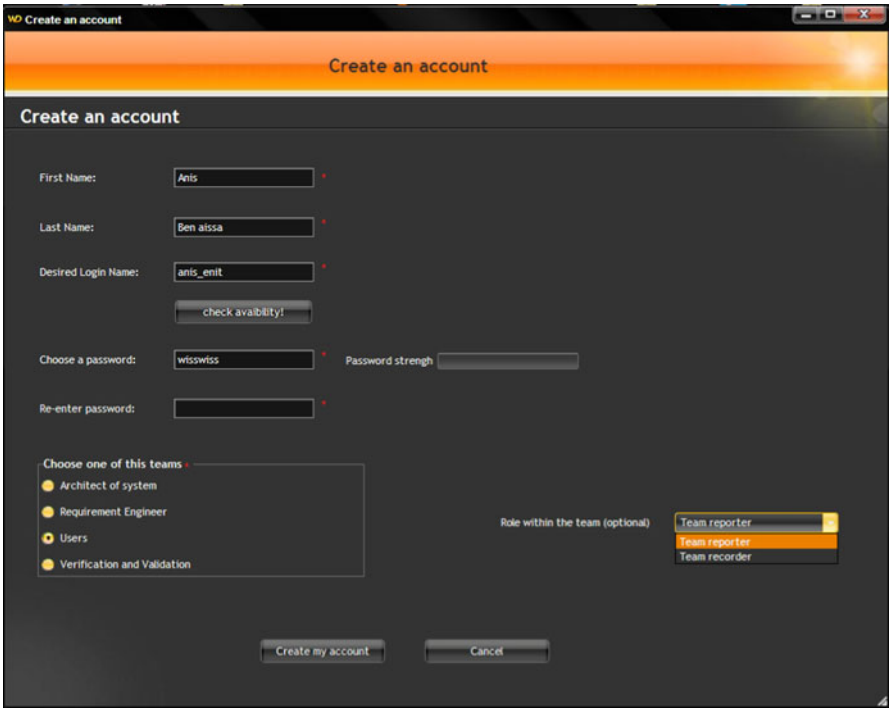


Fig. 10 Create an account interface

The current security configuration (estimated in light of current threats, vulnerabilities, counter-measures, etc.) yields a mean failure cost of $8.11 per hour for the customer, $112.97 per hour for the merchant, $31.17 per hour for the technical intermediary, and $54.24 per hour for the financial intermediary. We assume that the customer is charged this amount through higher prices levied by the merchant. Among the applications of this data, we mention:

- The merchant had better be making more than $112.97 per hour on this e-commerce site; if not, his mean operational benefit is negative.
- The merchant ought not to pay more than $112.97 per hour of insurance premium if he wants to protect against security risks.
- If system administrators are considering an upgrade in the security configuration, then the merchant can assess whether he is interested by computing the ROI of this upgrade on the basis of what share of the cost of the upgrade he is bearing, and how much reduction in **MFC** the upgrade will yield.

4 An analytical assessment of mean failure cost

Boehm et al. (Boehm et al. 2004) discuss the nature of information system dependability and highlight the variability of system dependability according to

Fig. 11 Login interface

Table 1 Vector of mean failure cost

Stakeholders	**MFC** ($/h)
Customer	8.11
Merchant	112.97
Technical intermediary	31.17
Financial intermediary	54.24

stakeholders; the dependency patterns of this model are subsequently analyzed in (Wu et al. 2010). We analyze the **MFC** model by discussing whether, how and to what extent it addresses the issues raised by (Boehm et al. 2004) with regards to the Stakeholder/Value definition of System Dependability. These five issues will be discussed in turn, below.

4.1 Variability within stakeholder classes

In (Boehm et al. 2004), stakeholders are divided into the following classes, on the assumption that each class corresponds to a distinct stakes profile:

- Information suppliers,
- Systems dependents,
- Information brokers,
- Information consumers, who are divided into two categories:
  - Mission critical and
  - Non-mission critical,
- System controllers,
- Developers,
- Maintainers,
- Administrators, and
- Acquirers.

This approach was further expanded and used in 16 real-client class projects, a graduate software engineering course (Wu et al. 2010). Their interpretation of the top-level stakeholder/value dependency matrix follows (Table 2):

Our stakes matrix is very similar to this matrix, except for the following details:

- The stakes matrix is not dependent on a classification of the stakeholders; we can have a row for each distinct stakeholder.
- The entries are real numbers rather than values on a discrete scale.
- The entries are expressed in dollars.

The stakes matrix is filled out by stakeholders, each filling out his/her row with respect to key security requirements (Rocha et al. 2005; Sekaran 2007; Sawma and Probert 2003). As an illustration (Table 3), consider the stakes matrix corresponding to the example of the e-commerce application to which we alluded earlier (Ben-Aissa et al. 2010):

Each stakeholder places a premium on satisfying each security requirement, commensurate with how much he stands to lose if the requirement is violated.

4.2 Variability within stakes values

In (Wu et al. 2010), as adapted by (Boehm et al. 2004), conducted a set of tests in the 16 client-class projects using real stakeholder and persons playing the role of real stakeholder. The objective of this study is to identify stakeholder/value dependence patterns. In six projects the average correlation values between real clients and role-play clients is 0.53. This result show that the choice of the discrete scale in stakes matrix give us:

- An Approximate values.
- We cannot estimate the cost of failure that a warring choice in rated scale can cause for the stakeholders.

To know the gravity of a bad choice in rating stakes values, we have used the example of e-commerce in (Ben-Aissa et al. 2010). Further analysis of the Table 3 shows the exact values of failure cost and the wrong failure in Table 4.

Table 2 Failure cost variability

Dependability attributes	Stake holder								
	Info suppliers	System dependents	Info brokers	Information consumers		System controllers	Developers	Maintainers	Administrators	Acquirers
				Mission						
				Critical	Uncrit					
Protection										
Safety		**		**		**				
Security	*	**	**	**		**				
Privacy	**		**	*	*					
Robustness										
Reliability		*	*	**		**		*	*	
Availability		*	*	**		**		*	*	
Survivability		*	*	**		**		*	*	
Quality of service										
Performance			**	**	*	**		*	*	
Accuracy, consistency	**		**	**	*	**			*	
Usability	*		*	**	**	**			*	
Evolvability			*	**	*	*		**	*	**
Interoperability			**						*	**
Correctness							*			**
Cost							*			**
Schedule			*	**	*	*	**			**
Resuability							**	*		*

Previously published (Wu et al. 2010) as modified from (Boehm et al. 2004)

() Insignificant or indirect

* Significant

** Critical

Table 3 Stakes (ST) matrix: cost of failing a security requirement stakes in $/h

ST		Security requirements					
		Confidentiality	Integrity	Availability	Non-repudiation	Authenticity	Privacy
Stake-holders	Customer	10	5	3	4	5	12
	Merchant	120	70	110	110	105	6
	Tech Int	20	20	20	20	30	20
	Fin Int	20	60	40	40	40	60

Previously published (Ben-Aissa et al. 2010)

Table 4 Failure cost variability

	Exact failure cost $/h	Wrong failure cost $/h
Customer	40	21.20
Merchant	551	292.03
Technical intermediary	150	79.50
Financial intermediary	270	143.10

Using the matrix of top-level stakeholder/Value metric in (Wu et al. 2010) we can just say:

- We have a highest correlation 0.53.

But if we use the Stake Matrix in (Ben-Aissa et al. 2010).

- We can give the exact values of the wrong failure cost.

4.3 Variability with Operational Context

The **MFC** model makes no distinction between normal operational contexts and exceptional operational contexts; they are specified by requirement clauses and each stakeholder puts a price/premium on each relevant clause. For this same reason, the **MFC** model makes no distinction between reliability and safety, because it makes no explicit (arbitrary) distinction between low stake clauses (reliability) and high stakes clauses (safety).

4.4 Variability with Maslow Need Hierarchy

Maslow's theory of human needs provides that needs are ranked hierarchically, and are addressed/fulfilled in a specific order, from the bottom of hierarchy going up (Maslow 1943; Wikipedia® 2010; Simons et al. 1987). This theory also provides that the layers of the hierarchy are not independent, in the sense that fulfilling a high priority need may lower its priority. This theory may be relevant to our discussion in

the sense that by assigning premiums to requirements clauses, a stakeholder is essentially defining a hierarchy of needs. As it is currently modeled, the **MFC** infrastructure does not reflect the interactions between layers of need, since the Stakes matrix is fixed. It could model layer interactions if the following features were provided:

- First, a way to control whether a particular requirement clause is satisfied, and the extent to which it is satisfied.
- Second a way for the stakeholder to assign an evolving premium to a requirement clause, dependent on the extent to which the clause is satisfied.

Neither of these two features is straightforward.

4.5 Orthogonality of requirements clauses

We are mindful of the fact that the requirement clauses that represent the columns of the Stakes matrix (and the rows of the Dependency matrix) are not orthogonal. As a result, the formula of **MFC**, which computes the weighted sum of the costs associated with the requirements clauses, may provide an upper bound of the mean failure cost, rather than the exact mean. Our model makes three distinct contributions to this discussion:

- First, it gives meaning to the concept of orthogonality: Because our requirements are elements of a lattice (the refinement lattice), orthogonality can be defined in terms of lattice operations, by the condition that the meet of two elements is the universal lower bound of the lattice.
- Second, it provides an exact formula for mean failure cost when requirement clauses are orthogonal. The difficulty of non-orthogonal requirement clauses is that the associated joint costs get counted multiple times; when they are orthogonal, then by definition there are no joint costs.
- Third, the lattice provides a framework for decomposing arbitrary requirements clauses into elementary terms that are orthogonal. While lattice theory provides for this possibility, the difficulty we have not resolved yet is how costs are decomposed to parallel the decomposition of the requirements; this is currently under investigation.

5 Relation between security measures and security impacts

The **MFC** formula

$$\mathbf{MFC} = \mathbf{ST} \circ \mathbf{DP} \circ \mathbf{IM} \circ \mathbf{PT}$$

maps a threat configuration (**PT**) onto a vector of mean failure costs (**MFC**). When a security measure is deployed, its impact can be measured by considering how it affects the threat configuration (say, **PT**' instead of **PT**) and thereby how it affects (hopefully reduce) the **MFC** vector (**MFC**' instead of **MFC**). In (Ben-Aissa et al. 2010), we have used the **MFC** differential as a measure of the effectiveness of the

security measure at hand. This measure can, in turn, be used to support the following decisions:

- The system manager can determine whether a security measure is worthwhile by matching its deployment cost against its benefit, quantified in terms of reduced **MFC** (and represented in monetary terms). The decision can, in fact, be modeled as a return on investment decision and quantified by ROI functions.
- The system manager can also use the **MFC** reduction of each stakeholder as a basis for distributing the investment cost of the security measure on the various system stakeholders; in (Ben-Aissa et al. 2010), we have discussed alternative ways to do this.
- The individual stakeholders can use the cost sharing formula to assess how much the security measure costs them, and use the **MFC** reductions to quantify their respective gains from the security measure; using this information, they can then compute their ROI and determine whether the security measure benefits them individually.

6 Conclusion

In this paper, we discuss a security metric that measures the security of a system in value/stakeholder terms. This metric can be characterized by the following premises:

- It is not dependent exclusively on the system, but depends on system users/stakeholders as well.
- It takes into account the heterogeneity of security requirements, and the fact that different stakeholders place different premiums on different requirements,
- It takes into account the heterogeneity of security measures, and the fact that a security measure may give a higher probability of satisfying one requirement than another may.

We have discussed how this metric can be used for concrete quantitative decision making, and presented an evolving tool that allows system users and stakeholders to store ongoing relevant information, and to derive relevant decision making information.

References

Ben-Aissa A, Abercrombie RK, Sheldon FT, Mili A (2010) Quantifying security threats and their potential impact: a case study. innovation in systems and software engineering. p 13. doi: 10.1007/s11334-010-0123-2

Boehm B, Huang LG, Jain A, Madachy R (2004) The nature of system dependability: a stakeholder/value approach. Technical report USC-CSSE-2004–520. University of Southern California, Centre for Systems and Software Research, California

Estevez E, Fillottrani P, Janowski T (2010) Information sharing in government—conceptual model for policy formulation. Paper presented at the 10th European conference on e-government, Limerick, Ireland, 17–18 June 2010

Firesmith D (2004) Specifying reusable security requirements. J Object Technol 3(1):61–75

Garlan D, Monroe RT, Wile D (2000) Acme: architectural descriptions of component-based systems. In: Leavens GT, Sitaraman M (eds) Foundations of component-based systems. Cambridge University Press, Cambridge, pp 47–68

Garlan D, Schmerl B, Cheng S-W (2009) Software architecture-based self-adaptation. In: Mieso Denko LYaYZ (ed) Autonomic computing and networking, vol 21. Springer, US. pp 31–55. doi: 10.1007/978-0-387-89828-5_2. ISBN 978-0-387-89827-8

Maslow AH (1943) A theory of human motivation. Origin Pub Psychol Rev 50:370–396

Rocha SVD, Abdelounahab Z, Freire E (2005) Requirement elicitation based on goals with security and privacy policies in electronic commerce. In: Anais do WER05—Workshop em Engenharia de Requisitos. pp 63–74

Sawma VD, Probert RL (2003) E-commerce authenication: an effective countermeasures design model. In: Paper presented at the ICEIS 2003, proceedings of the 5th international conference on enterprise information systems, Angers, France, 22–26 April 2003

Sekaran KC (2007) Requirements driven multiple view paradigm for developing security architecture. PWASET Proc World Acad Sci Eng Technol 25:156–159

Simons JA, Irwin DB, Drinnien BA (1987) Psychology—the search for understanding. West Publishing Company, New York

Wikipedia® (2010) Maslow's hierarchy of needs. http://en.wikipedia.org/wiki/Maslow%27s_hierarchy_of_needs. Accessed 19 July 2010

Wu D, Li Q, He M, Boehm B, Yang Y, Koolmanojwong S (2010) Analysis of stakeholder/value dependency patterns and process implications: a controlled experiment. Proceedings of the 43rd Hawaii international conference on system sciences (HICSS-43). doi:10.1109/HICSS.2010.60