The ISC Int'l Journal of Information Security

ISeCure

July 2016, Volume 8, Number 2 (pp. 143–155)

http://www.isecure-journal.org

Quantitative Evaluation of Software Security: An Approach Based on UML/SecAM and Evidence Theory

Ali Sedaghatbaf [1], and Mohammad Abdollahi Azgomi [1,*]

[1] *School of Computer Engineering, Iran University of Science and Technology, Tehran, Iran*

ARTICLE INFO.

Article history:
Received: 2 February 2016
Revised: 18 June 2016
Accepted: 2 July 2016
Published Online: 10 July 2016

Keywords:
Software architecture, Security Evaluation, Uncertainty Quantification, Evidence Theory.

ABSTRACT

Quantitative and model-based prediction of security in the architecture design stage facilitates early detection of design faults hence reducing modification costs in subsequent stages of software life cycle. However, an important question arises with respect to the accuracy of input parameters. In practice, security parameters can rarely be estimated accurately due to the lack of sufficient knowledge. This inaccuracy is ignored in most of the existing evaluation methods. The aim of this paper is to explicitly consider parameter uncertainty in the software security evaluation process. In particular, we use the Dempster-Shafer theory of evidence to formulate the uncertainties in input parameters and determine their effects on output measures. In the proposed method, security attacks are expressed using UML diagrams (i.e., misuse case and mal-activity diagrams) and security parameters are specified using the SecAM profile. UML/SecAM models are then transformed into attack trees, which allow quantifying the probability of security breaches. The applicability of the method is validated by a case study on an online marketing system.

1 Introduction

Regarding the increasing complexity and variety of cyber-attacks, security is a significant concern for software systems nowadays. In order to determine whether security is improved or degraded, it has to be evaluated quantitatively. If this evaluation is performed in the early development stages, costly modifications in the latter development stages will be avoided.

In recent years, some model-based approaches have been proposed for quantitative evaluation of software security at the architecture design stage. However, their application is limited in practice. In particular, the appropriateness of the model and the accuracy of parameters values are two important issues that need to be addressed. On one hand, model assumptions may not hold in practice and on the other hand, assuring the accuracy of parameters values is difficult.

Model parameters are usually estimated based on the field data obtained by testing, historical data available from similar systems or reasonable guesses by security experts. However, these estimates are rarely accurate. This indicates that any prediction method based on point estimates is not appropriate, since any variation in the prediction results due to the uncertainty in input parameters is ignored. To avoid misleading results, explicit representation of the input uncertainty and its propagation to the output measures is necessary.

In the reliability domain, some analytic/simulation

* Corresponding author.
Email addresses: ali_sedaghat@comp.iust.ac.ir (A. Sedaghatbaf), azgomi@iust.ac.ir (M. Abdollahi Azgomi)
ISSN: 2008-2045

methods (e.g., [1–3]) have been introduced for modelling uncertainty in input parameters. However, their predominant assumption is that the probability distribution functions characterizing the uncertain parameters are known beforehand. This assumption is unrealistic regarding the insufficiency of statistical data, especially in the early development stages. Even if such data exist, we may not observe their stability from the statistical viewpoint [4].

Fuzzy theory is known as an alternative to probability theory for modelling uncertainty in cases that statistical data are scarce. For example, in [5] UML class diagrams are transformed to fuzzy fault trees to compute the security risk for various kinds of attacks. However, the framework of fuzzy theory needs to assume a certain type of possibility distribution for uncertain variables. This assumption is unreasonable in many cases [4]. Additionally, there is no general rule for generating suitable possibility distributions from statistical data. Furthermore, it is illustrated in [4, 6] that fuzzy variables are not suitable for representing uncertain quantities.

To successfully deal with the uncertainties in security parameters, the Dempster-Shafer theory of evidence (or evidence theory) is promising. Evidence theory proposes a set of rules for combining evidences from different sources and provides a measure of confidence that an uncertain variable takes a specific value. In particular, the uncertainty of an input parameter is expressed by multiple intervals each supplied by a separate source of information (e.g., an expert). To each interval a probabilistic value is assigned which indicates the belief degree that the actual value of the parameter belongs to that interval. Then, a combination rule (e.g., the Dempster's rule) is applied to the probabilities assigned to different intervals for their aggregation. The aggregated intervals are then propagated in the model so that the possible intervals for the output measure(s) can be determined.

This paper comprises the following contributions:

(1) Description of a UML-based methodology for quantitative evaluation of software security: UML as the de facto standard for software modelling is used for architecture description and representing the attacks threatening the system. Security parameters are specified through the SecAM profile [7]. This profile is an extension of MARTE, which is the most comprehensive UML profile supporting quantitative evaluation of software quality attributes.
(2) Presentation of an algorithm for transforming UML/SecAM models to attack trees, which are one of the most popular graphical security models [8].
(3) Using the evidence theory rules for representation and analysis of uncertainties in input parameters.
(4) Development of a software tool that automates the transformation and evaluation processes.
(5) Evaluating the security of an online marketing system using the proposed method.

In the sequel, we discuss about the related contributions in Section 2. Section 3 presents some background information about evidence theory and the SecAM profile. The proposed method is illustrated through an example in Section 4, and a case study evaluation is presented in Section 5. Finally, concluding remarks and the future works are discussed in Section 6.

2 Related Work

Software security analysis and uncertainty in quantitative evaluation of software quality attributes are two research areas related to the work presented in this paper.

From the security analysis perspective, the existing methods can be categorized into two groups [5]: (1) risk analysis based (e.g., [5, 9, 10]) and (2) dependability based (e.g., [11–13]). The former focus on evaluating the potential risk of different security threats by considering their likelihood and impact. The latter try to leverage the maturity of dependability modelling and evaluation techniques for quantifying security measures.

The AHP-based framework presented in [9] is an example of risk analysis approaches. The proposed risk analysis method consists of two stages. The first stage is dedicated to analyzing the risk of each security scenario. The risk of the whole system is then evaluated based on the AHP pair-wise comparison between scenarios. As another example, in [10] the object constraint language (OCL) is used to formalize attack scenarios and exposure metrics. Also, a software tool is presented which can be used to evaluate the specified metrics on software architecture models.

The dependability-based methods are inspired by the analogies between security and reliability. For example, Markov chains traditionally used for reliability analysis, are used in [14] for modelling software architectures and evaluating their security vulnerability index. In [13] each software component is modelled with a stochastic Petri Net (SPN). By analyzing the reachability graph of each SPN, the attack success probability is determined for each component. In order to evaluate the overall security of the system a hierarchical approach is proposed.

From the uncertainty analysis point of view, several works (e.g., [15–17]) include sensitivity analysis to de-

termine the effect of variations in input parameters on output measures. These analyses help us identify the critical parts of the system with respect to the quality attribute of interest. However, the results of these analyses may be misleading regarding the inaccuracy of input parameters.

Uncertainty can be controlled only by making it explicit [18]. Probability theory is practically used for explicit characterization of parameter uncertainty in software quality evaluation. For example, the Monte-Carlo method is used in [19–21] to propagate the uncertainty in input parameters to the output measures of software reliability/performance models. The principal assumption of this method is that the probability distributions characterizing the uncertain parameters are known beforehand. This assumption is unrealistic especially in the architecture design stage. To tackle this problem, some researchers (e.g., [22]) suggest using the Bayesian approach to incorporate historical data into prior distributions. Also, in order to consider both the historical data and the experts' knowledge, combination of the Bayesian approach and the maximum-entropy principle is suggested in [2].

Fault tree is a formalism extensively used in reliability, safety and security domains. In order to handle uncertainties in fault tree analysis, the use of fuzzy logic is suggested in [23]. Fuzzy fault tree analysis has gained much attention in recent years and it is applied in several domains [24]. As an example in the software security domain, it is used in [5] to analyze the risk of software systems based on the security patterns they contains. Furthermore, it is used in [25] to determine the most critical components from the reliability point of view.

Using evidence theory for uncertainty modelling allows us to obtain the possible bounds for the system security, while there is no need to make assumptions about the probability/possibility distributions of input parameters. The possible large inaccuracy in the output results reflects the incompleteness of the input information, which may direct the search for additional information sources.

3 Preliminaries

3.1 The SecAM Profile

MARTE extends UML with a set of stereotypes and tags for modelling and analysis of real-time embedded systems. This profile makes it easy to specify performance-related information in UML models, and it is extensible to any quality attribute other than performance. For example, DAM is an extension of the MARTE profile for dependability-related annotations and SecAM is an extension of DAM for the security analysis domain.

Table 1. Some SecAM stereotypes and data types.

Stereotype	Description
secaAttackGenerator	A mechanism used to attack the system
secaStep	Action/state representing a securiyty threat/mechanism
secaLink	A secure communication link
secaSecurityDevice	A security hardware mechanism (e.g. firewall)
Data Type	**Description**
secaAttack	Descriptor for an attack (including type, objective and target host)
secaIntrusion	Descriptor fo an intrusion (including success probabilty and the related vulnerability/attack)
secaVulnerable	Descriptor for a vulnerability (incluing severity degree and lower-level vulnerabilities if any)
secaAntivirus	Descriptor for an antivirus (including the scanning mode)

In [12] an initial definition of the SecAM profile is presented, which covers only the resilience issues. This definition is then enhanced in [26] with a set of new stereotypes which allows to specify cryptographic and access control mechanisms. An overview of this profile is presented in Figure 1.

The SecAM profile is composed of two main packages: *SecAM_UML_Extensions*, which includes the definitions of SecAM's stereotypes, and *SecAM_Library* in which the data types associated to the stereotypes' tags are defined. The SecAM_UML_Extensions package consists of four sub-packages addressing different security issues. The *Resilience* sub-package addresses the security threats (i.e., vulnerabilities, attacks and intrusions) and their causal relationships, the *Cryptographic* package deals with ciphers and encryption algorithms, the *AccessControl* package supports the specification of access control policies and the *SecurityMechanism* package characterizes different kinds of hardware/software security solutions. An excerpt of the SecAM stereotypes and data types is provided in Table 1.

3.2 Dempster-Shafer Theory of Evidence

Evidence theory was originated by Arthur P. Dempster and developed by Glenn Shafer [27] for representing uncertainty. This theory starts by defining a frame of discernment Ω, which is a set of mutually exclusive propositions. Each proposition represents a hypothesis about the system. If we let P_Ω denote the power set of Ω, then the knowledge about each member p_i

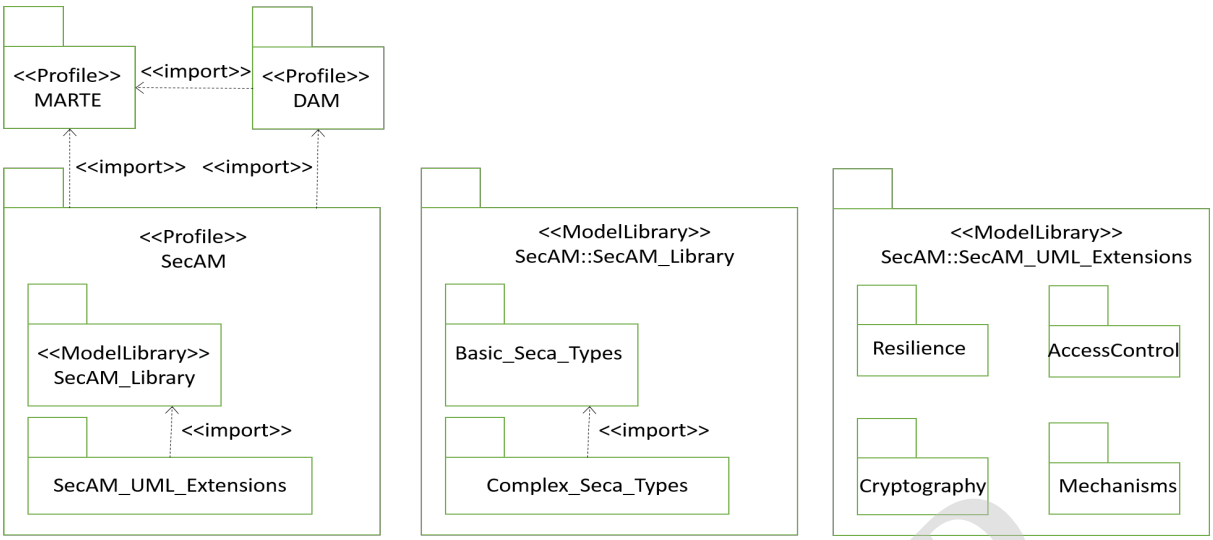

Figure 1. An overview of the SecAM profile.

of P_Ω is represented by a probabilistic value called *Basic Probability Assignment* (BPA). Hereafter, this knowledge is denoted by $m(p_i)$ which is subject to the following assumptions:

$$m : P_\Omega \to [0,1] \tag{1}$$

$$m(\varnothing) = 0 \tag{2}$$

$$\sum_{p_i \in P_\Omega} m(p_i) = 1 \tag{3}$$

If $m(p_i) > 0$, then p_i is called a focal element. Our total degree of belief in p_i is expressed with a belief-plausibility interval $[Bel(p_i), Pl(p_i)] \subseteq [0,1]$ where $Bel(p_i)$ and $Pl(p_i)$ are given as:

$$Bel(p_i) = \sum_{p_k \subseteq p_i} m(p_k) \tag{4}$$

$$Pl(p_i) = \sum_{p_k \cap p_i \neq \varnothing} m(p_k) \tag{5}$$

Evidence theory allows us to combine different sources of evidence i.e., if there are multiple estimates for p_i represented by $m_{i1}, \ldots, m_{ik}$, then several methods have been proposed to calculate the aggregation m_i of all the estimates. The most popular method is the Dempster's rule [28]. The precondition of this rule is that each estimate contains at least one focal element enclosing the true value of the event. Therefore, this rule cannot be used in situations where the conflict between evidences is considerable. Instead, in this paper the weighted mixture method [29] is used for aggregating the estimates of different experts. In this method a weight w_i is assigned to each estimate. This weight quantifies the degree of importance of that estimate. If all the estimates are equally important, then $w_i = 1$ for $i = 1, \ldots, k$. The weighted mixture of the estimates is calculated as follows:

$$m(p_i) = \frac{\sum_{j=1}^{k} w_j m_j(p_i)}{\sum_{j=1}^{k} w_j} \tag{6}$$

4 Methodology Description

The proposed methodology can be synthesized by four steps. The first step addresses the specification of attack scenarios using a combination of misuse case diagrams [30] and mal-activity diagrams [31]. The SecAM stereotypes and tags are utilized to annotate these diagrams with the information required for quantitative security analysis. Due to data scarcity, the exact values of some tags may be unknown. In the second step, the analyst consults security experts to estimate upper/lower bounds for uncertain parameters values. The experts' judgments are then combined and appended to the UML/SecAM model which is transformed to an attack tree in the third step. Finally, the obtained attack tree is analyzed to estimate the belief and plausibility measures for the security breach probability.

4.1 UML/SecAM Model Construction

In this step, the attacks that may target the system are identified and expressed using a misuse case diagram (MCD). MCDs extend UML use case diagrams with unwanted use cases that cause harm to some stakeholder. The scenario of each attack is modelled by a mal-activity diagram (MAD), which extends the UML activity diagram to allow describing malicious behaviors. For illustration, a simple UML/SecAM model is depicted in Figure 2. Two attack scenarios are described by this model. The SecAM annotations indicate that the objective of the first attack is making the system unavailable to its intended users, while by the second attack the attacker intends to run an

arbitrary code on the system. The *secaStep* stereotype allows distinguishing normal steps from security threats/means. For example, *Action5* and *Action7* are identified as vulnerable steps that may lead to intrusions if compromised by attackers.

As it is clear from the UML/SecAM model, the security analyst is uncertain about the exact values of some tags. For example, he/she is not sure about the exact occurrence probability of each of the two attacks. To complete the UML/SecAM model, he/she should ask security experts to provide their own judgments about each parameter. This step is discussed in the next section.

4.2 Parameter Estimation

In this step, the analyst asks the experts to specify the interval that they believe the real value of each uncertain parameter can lie in. A possible scenario is to collect historical data on the target system or similar systems and suggest a set of intervals to the experts asking their degree of belief for each interval.

Since more than one expert may be asked to provide estimations, it is necessary to aggregate their judgments. For example, assume that $[a_1, a_2]$ and $[b_1, b_2]$ with BPAs $m_1$and m_2, are the intervals supplied by two experts for an input parameter. At this point, the intervals are aggregated using the weighted mixture rule described in Section 3.2. For illustration, the estimated intervals for the occurrence probabilities of the two attacks in Figure 2(a) and their aggregations are provided in Table 2.In these estimations, the judgments of Expert1 are assumed more reliable, hence their weight is 2, while the weight of the Expert2's judgments is 1.

The results of applying the aggregation rule to all judgments are then appended to the UML/SecAM model. As an example, considering the aggregation results in Table 2, the MCD in Figure 2(a) is modified as it is shown in Figure 3.

4.3 Attack Tree Generation

In this step, an attack tree is derived from the UML/SecAM model. Attack tree is an AND-OR tree structure typically used for modelling cyber-attacks. In this formalism, attacker's main goal is specified as the root of the tree, and it is refined into sub-goals conjunctively or disjunctively. The refinement process may be repeated several times until the reached sub-goals represent basic attack actions.

Algorithm 1 elaborates the derivation procedure. The root of the tree denotes a security breach in the system. This node is connected to the second-layer nodes through an OR gate. Each node in the second layer corresponds to a misuse case (i.e., attack) in the MCD. Each misuse case node has three children: two leaf nodes parameterized with the occurrence and success probabilities of the attack, and an AND-rooted sub-tree which represents the scenario leading to the attack success (i.e., an intrusion).

Algorithm 1 Attack tree generation algorithm

```
1: sys_node ← new ORNode
2: for each misuse case muc in MCD do
3:     muc_node ← new ANDNode
4:     sys_node.addChild(muc_node)
5:     occ_node ← new LeafNode
6:     sys_node.addChild(occ_node)
7:     let mad be the MAD describing the behavior of muc
8:     sce_node ← new ANDNode
9:     muc_node.addChild(sce_node)
10:    let p be the path from the initial node to the intrusion
       node in mad
11:    for each secaStep s in p do
12:        st_node ← new LeafNode
13:        sce_node.addChil(st_node)
14:    end for
15:    suc_node ← new LeafNode
16:    muc_node.addchild(suc_node)
17: end for
```

The attack tree derived from the UML/SecAM model in Figure 2, is depicted in Figure 4. The root node of this tree has two children *attack1* and *attack2* corresponding to the two misuse cases in Figure 2(a). The parameters assigned to the leaf nodes correspond to the probabilistic values assigned to the SecAM tags in the UML/SecAM model. For example, the parameter assigned to the node *occ_attack1* is equivalent to the *occurenceProb* tag in the SecAM annotation of the *attack1* misuse case.

4.4 Security Evaluation

In order to determine the probability of a security breach in the system under analysis, the probabilistic value assigned to the root node of the attack tree must be calculated based on the values assigned to the leaf nodes. The mathematical formula equivalent to the attack tree generated in the previous step is as follows:

$$p_{sb} = 1 - \prod_{j=1}^{n}(1 - op_j sp_j \prod_{k=1}^{m} p_k) \qquad (7)$$

where p_{sb} denotes the probability of security breach, op_j (sp_j) is the occurrence (success) probability of attack j, p_k is the execution probability of step k, and n is the number of attacks identified for the system.

Regardless of the accuracy of (7), if considerable uncertainty exists in the input parameters (as usually does), then a significant uncertainty exists in the value

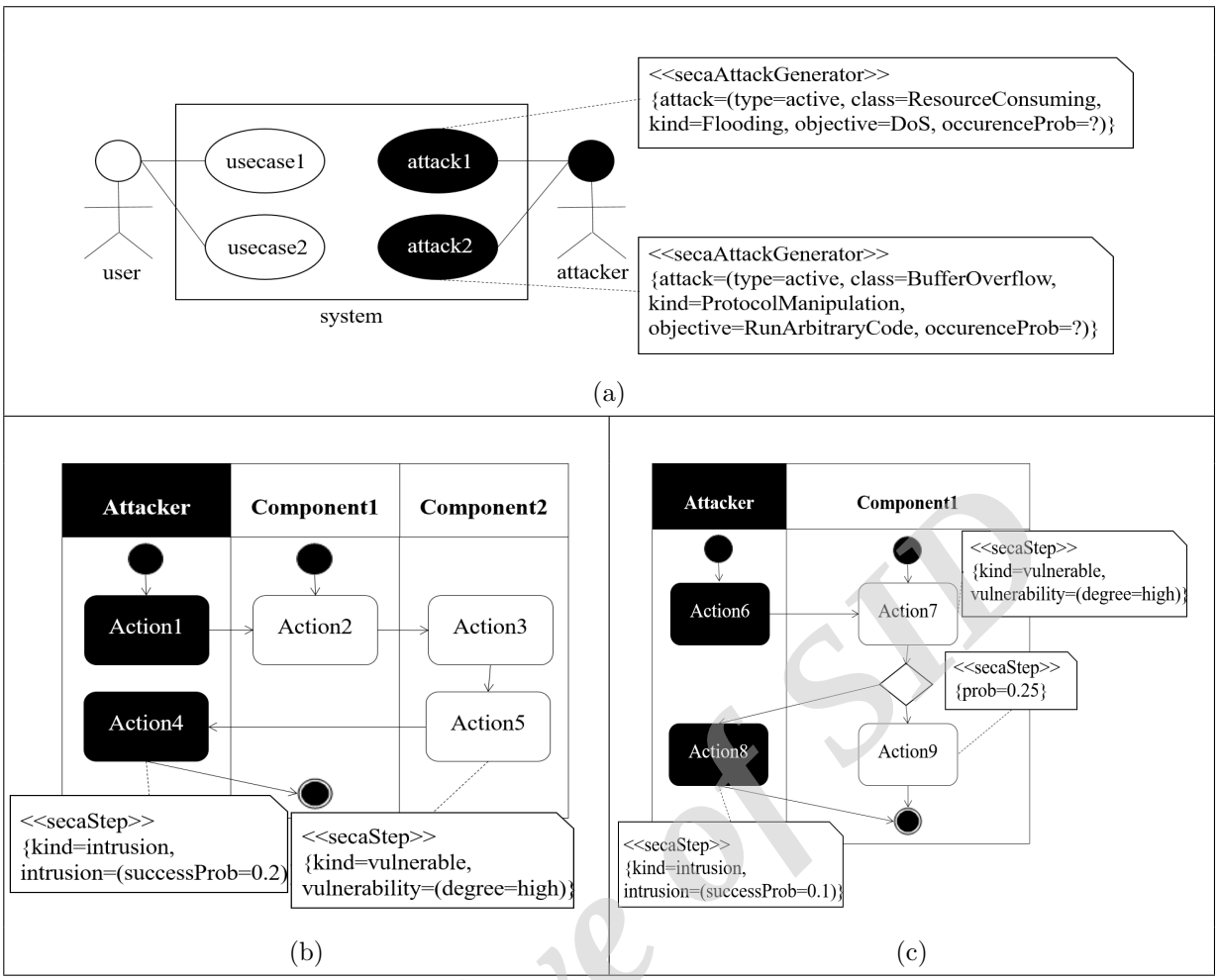

Figure 2. A simple UML model: (a) MCD, (b) MAD of $attack_1$, (c) MAD of $attack_2$

Table 2. Judgment aggregation for UML/SecAM model in Figure 2.

Parameter	Source	Initial Intervals	Initial PBAs	Final Intervals	Final PBAs
$attack_1.occurrenceProb$	Expert1(w=2)	[0.02, 0.04]	0.6	[0.02, 0.04]	0.4
		[0.04, 0.06]	0.4	[0.04, 0.06]	0.26666
	Expert2(w=1)	[0.03, 0.07]	1	[0.03, 0.07]	0.33333
$attack_2.occurrenceProb$	Expert1(w=2)	[0.06, 0.08]	0.9	[0.06, 0.08]	0.6
		[0.04, 0.06]	0.1	[0.04, 0.06]	0.06666
	Expert2(w=1)	[0.05, 0.07]	1	[0.05, 0.07]	0.33333

estimated for breach probability. Therefore, computing the point estimate of security based on the point estimate of the input parameters is not appropriate.

In order to reflect the uncertainties in the input values on p_{sb}, the aggregated judgments for the uncertain parameters must be propagated from the leaf nodes to the root of the attack tree. The method proposed here is based on using ordinary arithmetic operations. In particular, if the intervals $[a_1, a_2]$ and $[b_1, b_2]$ are associated with two nodes A and B, then the interval of $AND(A, B)$ is $[a_1, a_2].[b_1, b_2] = [a_1 b_1, a_2 b_2]$ and the interval of $OR(A, B)$ is $1 - (1 - [a_1, b_1])(1 - [a_2, b_2])$.

After propagating the intervals to the root node, the analyst is faced with multiple intervals for the probability of security breach. For example, according to Table 2, there are two uncertain parameters for the UML/SecAM model in Figure 2, and three aggregated intervals are associated to each of them. This means that nine different intervals will be calculated for the root node of the attack tree in Figure 4. These intervals are presented in Table 3. More than the interval themselves, the analyst may be interested to

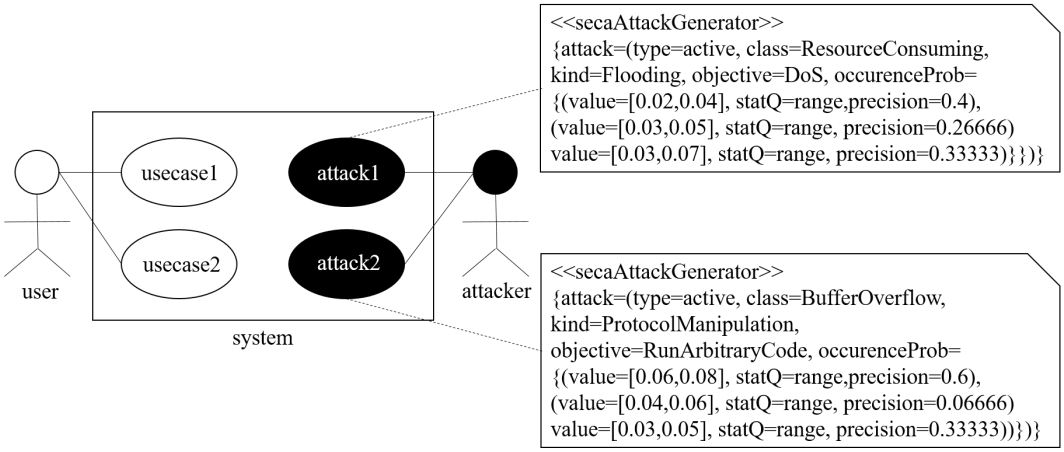

Figure 3. Adding the aggregated judgments to the MCD of Figure 2.

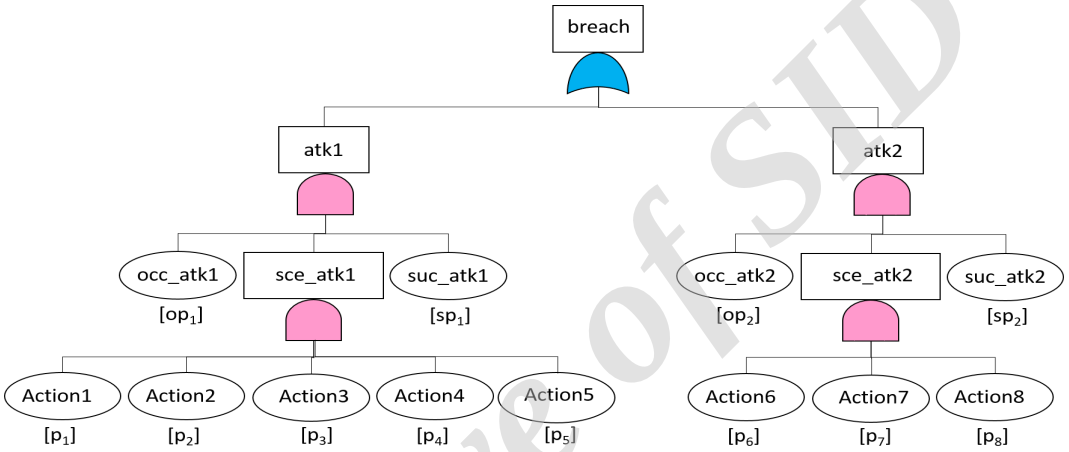

Figure 4. The attack tree generated from the UML/SecAM model of Figure 2.

know whether the probability of security breach is lower than a given threshold p_{th} or not. The *belief* and *plausibility* measures of the evidence theory are the means to determine the minimum and maximum believes about this fact. Based on the intervals I_i presented in Table 1, the belief and plausibility trends for the UML/SecAM model in Figure 2 are reported in Figure 5. As an example, assuming that the threshold is 0.013, the belief and plausibility measures for the event $p_{sb} \leq 0.013$ are determined as follows:

$$Bel([0, 0.013]) = \sum_{I_i \subset [0,0.013]} PBA(I_i) = I_2 + I_3 = 0.159996$$

$$Pl([0, 0.013]) = \sum_{I_i \cap [0,0.013] \neq \varnothing} PBA(I_i) = 1.0$$

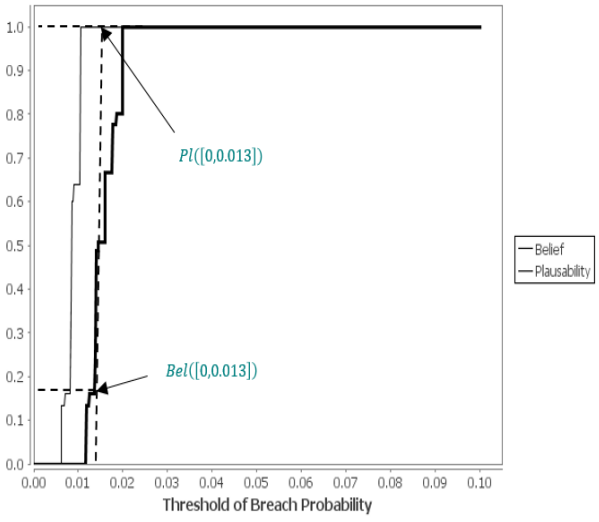

Figure 5. Belief and plausibility trends for the example system

5 Case Study

As a case study, we examined an online marketing server [32] which allows customers submit purchase requests to a business firm via the internet. The users of this application are (1) the website administrator who manages the product list, (2) customers seeking for products, (3) the warehouse clerk who is respon-

Table 3. Probability intervals for the UML/SecAM model of Figure 2.

Aggregated Judgment for $attack_1$		Aggregated Judgment for $attack_2$		Aggregated Judgment for the Root Node	
Interval	*PBA*	*Interval*	*PBA*	*Interval*	*PBA*
[0.02, 0.04]	0.40000	[0.06, 0.08]	0.60000	[0.008482, 0.013952]	0.240000
[0.02, 0.04]	0.40000	[0.04, 0.06]	0.06666	[0.006988, 0.012464]	0.026664
[0.02, 0.04]	0.40000	[0.05,0.07]	0.33333	[0.006241, 0.01172]	0.133332
[0.04, 0.06]	0.26666	[0.06, 0.08]	0.60000	[0.010473, 0.01594]	0.159996
[0.04, 0.06]	0.26666	[0.04, 0.06]	0.06666	[0.008982, 0.014455]	0.017775
[0.04, 0.06]	0.26666	[0.05,0.07]	0.33333	[0.0082365, 0.0137125]	0.088885
[0.03, 0.07]	0.33333	[0.06, 0.08]	0.60000	[0.010473, 0.019916]	0.199998
[0.03, 0.07]	0.33333	[0.04, 0.06]	0.06666	[0.008982, 0.018437]	0.022219
[0.03, 0.07]	0.33333	[0.05,0.07]	0.33333	[0.0082365, 0.0176975]	0.111108

sible for shipping customers' orders and (4) hackers who wish to cause malicious damage to the system.

5.1 UML/SecAM Model

As depicted in Figure 6(a), a customer initiates the marketing process by requesting a new purchase. The server checks service availability on request receipt. If the service is available, the login page is loaded. Otherwise, an error message is sent to the customer. As the server does not scan the incoming requests, it is vulnerable to spoofing-based DoS attacks. This enables hackers to compromise the availability of the server by sending excessive requests. The MCD in Figure 6(c) indicates that the marketing system is also vulnerable to SQL injection attacks. The attack process is presented by a MAD in Figure 6(b). As the registration data is not validated, a hacker can send malicious SQL commands to modify customers' data and compromise the system integrity. Furthermore, the vulnerability to stored XSS lets hackers execute malicious scripts on the system to gather sensitive information (e.g. credit card numbers) and compromise the system confidentiality (see Figure 6(d)).

The SecAM annotations in Figure 6 indicate uncertainties in the occurrence/success probabilities of the attacks and the execution probabilities of some software activities. To handle these uncertainties, we consulted two security experts. The judgments of these experts and their aggregations are reported in Table 4.

5.2 Security Evaluation

In order to analyze the security of the marketing system, first we define (a subset of) the SecAM profile using the Papyrus tool [33]. This tool is a plugin developed for the Eclipse IDE which facilitates UML model development. Papyrus supports MARTE stereotypes and tags. Also, the DAM profile has been implemented recently [34]. However, as the best of our knowledge, there exists no publicly available implementation of the SecAM profile. After implementing the SecAM profile, we created the UML/SecAM model of the marketing server using the Papyrus tool.

After appending the aggregated judgments to the UML/SecAM model, we submitted its XMI representation to the SQME tool [35]. This tool is an Eclipse plugin we have developed to facilitate quantitative evaluation of software quality attributes. SQME is capable of transforming the UML/SecAM models to attack trees based on the algorithm presented in section 4.3.

The attack tree generated by the SQME tool is depicted in Figure 7. This tree must be analyzed 2187 times to determine the possible value intervals for the breach probability. This is because the UML/SecAM model in Figure 6 includes seven uncertain parameters and three aggregated judgments for each parameter,

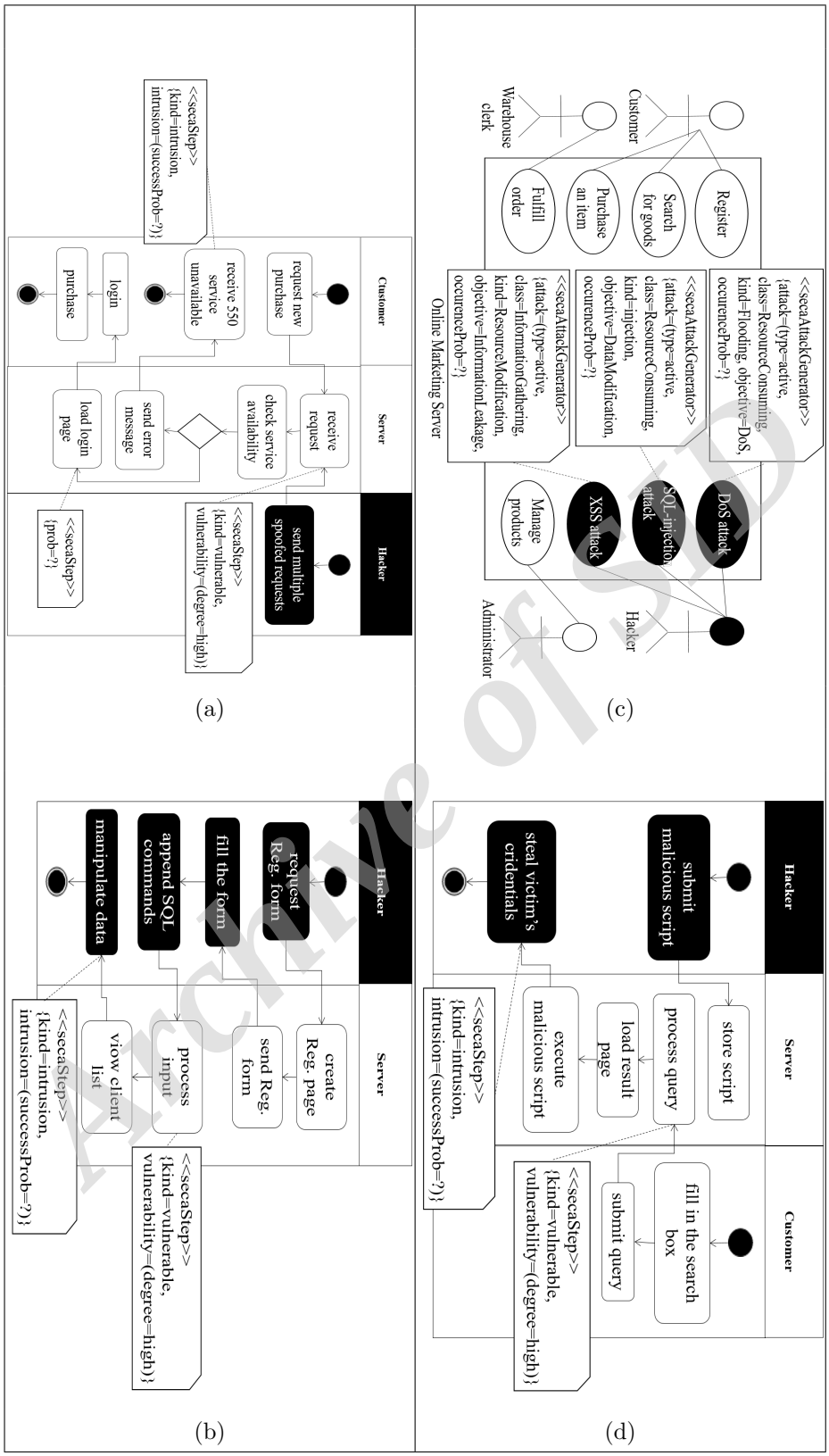

Figure 6. UML/SecAM model of the marketing system: (a) MAD of *DoS attack*, (b) MAD of *SQL injection attack*, (c) MCD, (d) MAD of *XSS attack*

Table 4. Judgment aggregation for the uncertain parameters of the UML/SecAM model in Figure 6.

Parameter	Judgments by Expert1		Judgments by Expert2		Aggregated Judgments	
	Interval	*PBA*	*Interval*	*PBA*	*Interval*	*PBA*
DoS.ocurProb	[0.05, 0.07]	0.9	[0.04, 0.06]	1	[0.05, 0.07]	0.45
	[0, 0.05]	0.1			[0, 0.05]	0.05
					[0.04, 0.06]	0.5
DoS.succProb	[0.2, 0.4]	0.6	[0.1, 0.3]	1	[0.2, 0.4]	0.3
	[0, 0.2]	0.4			[0, 0.2]	0.2
					[0.1, 0.3]	0.5
loadPage.prob	[0.1, 0.5]	0.8	[0.2, 0.4]	1	[0.1, 0.5]	0.4
	[0, 0.1]	0.2			[0, 0.1]	0.1
					[0.2, 0.4]	0.5
SQL.ocurProb	[0.01, 0.09]	0.9	[0.02, 0.08]	1	[0.01, 0.09]	0.45
	[0, 0.01]	0.1			[0, 0.01]	0.5
					[0.02, 0.08]	0.5
SQL.succProb	[0.2, 0.6]	0.9	[0.1, 0.5]	1	[0.01, 0.09]	0.45
	[0, 0.2]	0.1			[0, 0.2]	0.05
					[0.1, 0.5]	0.5
XSS.ocurProb	[0.02, 0.06]	0.8	[0.02, 0.08]	1	[0.02, 0.06]	0.4
	[0, 0.02]	0.2			[0, 0.02]	0.1
					[0.02, 0.08]	0.5
XSS.succProb	[0.2, 0.6]	0.9	[0.1, 0.6]	1	[0.2, 0.6]	0.45
	[0, 0.2]	0.1			[0, 0.2]	0.05
					[0.1, 0.6]	0.5

which leads to 3^7 interval combinations. Obviously, these combinations are not reported in this paper. However, belief and plausibility of the event $p_{sb} \leq p_{th}$ are shown in Figure 8.

5.3 Sensitivity Analysis

The analysis results in Figure 8 indicate significant differences between the values of belief and plausibility measures if the threshold value lies in the interval $[0.005, 0.05]$. This difference implies high degree of uncertainty in the input parameters. In many cases, it may not be possible to reduce uncertainty in all input parameters. However, we can utilize sensitivity analysis to determine uncertainty in which parameter has the most effect on the inaccuracy of the output results. In [36] several measures have been defined that can be used to measure the amount of uncertainty for a frame of discernment. For the marketing system, we used the generalized Hartley (GH) measure which is defined as follows:

$$GH(m) = \sum_{g \in G} m(g) log_2 |g| \qquad (8)$$

where G denotes the set of focal elements associated with m.

At first, we evaluated the GH measure for the output estimates generated by the SQME tool. Then, we replaced the aggregated intervals of an arbitrary uncertain parameter with a constant value, and analyzed the attack tree model again. This process was repeated for other uncertain parameters, and the new GH values were compared with the initial ones. The comparison results are presented in Figure 9. Accordingly, reducing uncertainty in *loadPage.prob* has the most effect on the accuracy of the security evaluation results.

5.4 Architecture Improvement

The main purpose of quantitative evaluation of security in the architecture design stage is early and cost-effective detection and removal of design faults that allow attackers compromise the system. For example, adding proper input validators to the registration form would prevent SQL injection attacks to the marketing system. This modification leads to the removal of the second subtree from the attack tree presented

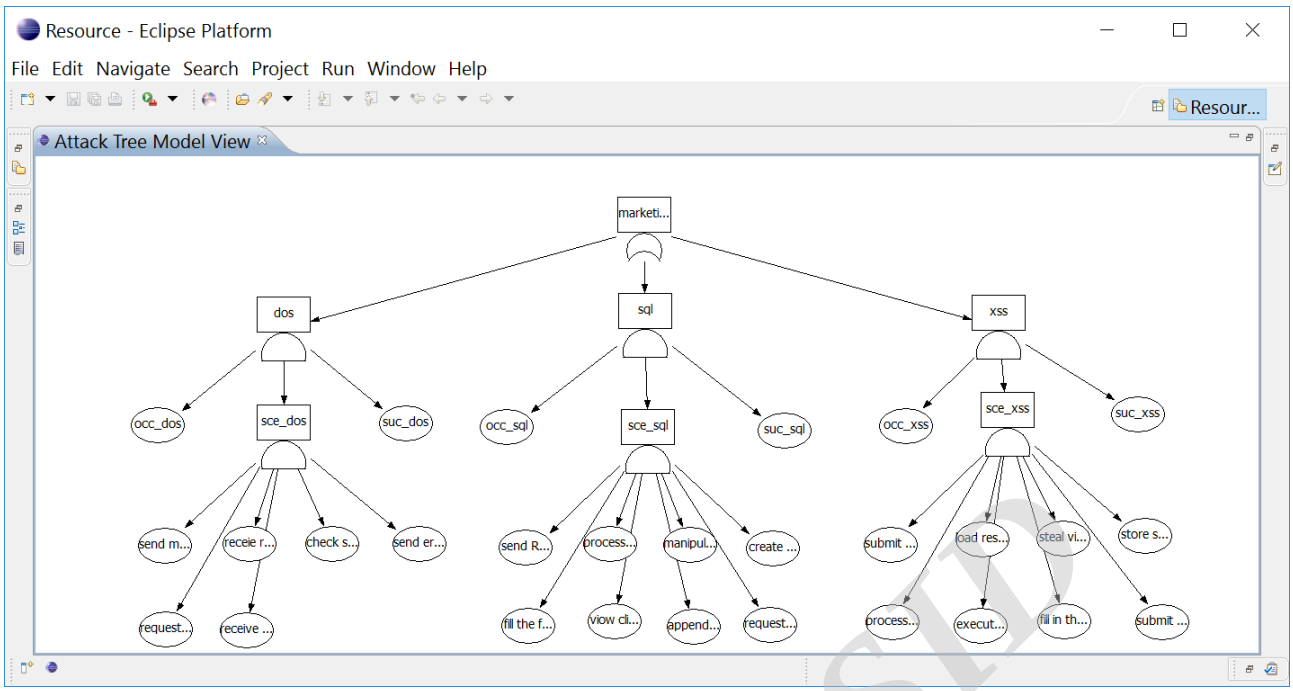

Figure 7. The attack tree of the marketing system.

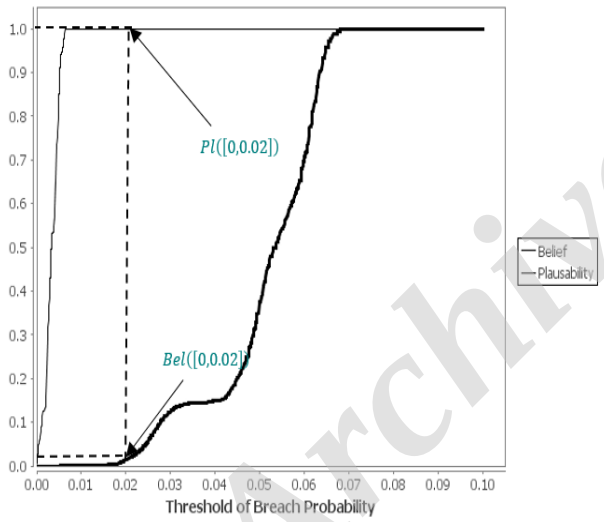

Figure 8. Belief and plausibility of $p_s b \leq p_t h$ for the marketing system.

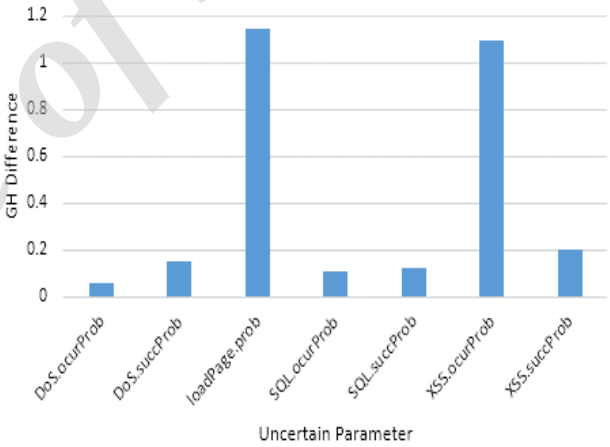

Figure 9. Sensitivity analysis results for the marketing system.

in Figure 9. The result of analyzing the new tree is depicted in Figure 10. This figure clearly indicates the improvement in security. For example, considering 0.02 as the threshold value, the belief-plausibility interval for the initial tree is approximately $[0.01, 1]$ (see the chart in Figure 8), which implies that the breach probability is most probably more than the threshold value. Whereas, the chart in Figure 10 indicates with high certainty that security breach occurs with a probability less than the threshold value.

It is necessary to note that due to the conflicts between some quality attributes, any architecture improvement with respect to one attribute may degrade another one. For example, security conflicts with performance, which means that the performance overhead of some security mechanisms may prevent architects from employing them.

6 Concluding Remarks

Regarding the necessity of analyzing the security of software system in the preliminary development stages, in this paper we introduced a new security evaluation method for software architectures. A significant benefit of early evaluation is reduction of modification costs in latter development stages. However, this benefit comes with the cost of the considerable inaccuracy in input parameters which may lead to misleading pre-

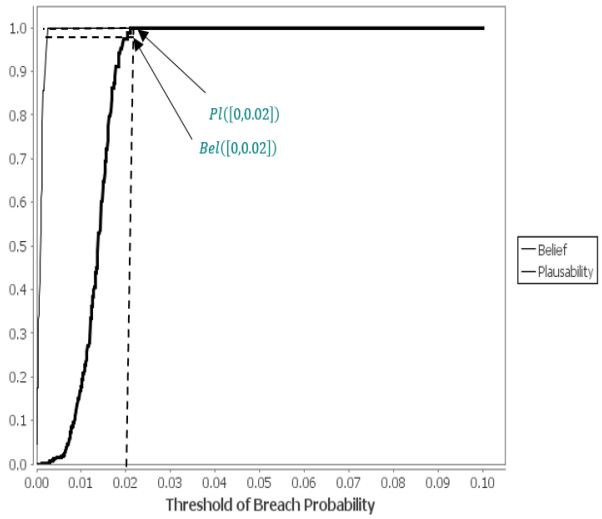

Figure 10. Security analysis results after modifying the architecture of the marketing system.

dictions. The best way to control this uncertainty is to make it explicit. To this end we used the Dempster-Shafer theory of evidence.

Not only the proposed method is easy to understand, but also the estimated security bounds provide useful information for objective decision making. However, the evaluation method is fairly simple and ignores vulnerability dependencies and propagations. Extension of this work to more sophisticated security evaluation methods will be considered in future.

References

[1] Amita Devaraj et al. Uncertainty propagation in analytic availability models. In *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, pages 121–130. IEEE, 2010.

[2] Yuan-Shun Dai et al. Uncertainty analysis in software reliability modeling by bayesian analysis with maximum-entropy principle. *IEEE Transactions on Software Engineering*, 33(11):781–795, 2007.

[3] Indika Meedeniya et al. Evaluating probabilistic models with uncertain model parameters. *Software & Systems Modeling*, 13(4):1395–1415, 2014.

[4] Lev V Utkin et al. Imprecise reliability: an introductory overview. In *Computational intelligence in reliability engineering*, pages 261–306. Springer, 2007.

[5] Spyros T Halkidis et al. Architectural risk analysis of software systems based on security patterns. *IEEE Transactions on Dependable and Secure Computing*, 5(3):129–142, 2008.

[6] Baoding Liu. Why is there a need for uncertainty theory. *Journal of Uncertain Systems*, 6(1):3–10, 2012.

[7] Simona Bernardi et al. A dependability profile within marte. *Software & Systems Modeling*, 10 (3):313–336, 2011.

[8] Barbara Kordy et al. Dag-based attack and defense modeling: Dont miss the forest for the attack trees. *Computer science review*, 13:1–38, 2014.

[9] Abdulaziz Alkussayer et al. Security risk analysis of software architecture based on ahp. In *Networked Computing (INC), 2011 The 7th International Conference on*, pages 60–67. IEEE, 2011.

[10] Mohamed Almorsy et al. Automated software architecture security risk analysis using formalized signatures. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 662–671. IEEE Press, 2013.

[11] Zahra Aghajani et al. Security evaluation of an intrusion tolerant web service architecture using stochastic activity networks. In *International Conference on Information Security and Assurance*, pages 260–269. Springer, 2009.

[12] Ricardo J Rodríguez et al. Modelling and analysing resilience as a security issue within uml. In *Proceedings of the 2nd International Workshop on Software Engineering for Resilient Systems*, pages 42–51. ACM, 2010.

[13] Nianhua Yang et al. Modeling and quantitatively predicting software security based on stochastic petri nets. *Mathematical and Computer Modelling*, 55(1):102–112, 2012.

[14] Vibhu Saujanya Sharma et al. Quantifying software performance, reliability and security: An architecture-based approach. *Journal of Systems and Software*, 80(4):493–509, 2007.

[15] Gaogao Yan et al. Formal throughput and response time analysis of marte models. In *International Conference on Formal Engineering Methods*, pages 430–445. Springer, 2014.

[16] Murray Woodside et al. Transformation challenges: from software models to performance models. *Software & Systems Modeling*, 13(4):1529–1552, 2014.

[17] Thanh-Trung Pham et al. Reliability prediction for component-based software systems with architectural-level fault tolerance mechanisms. In *Availability, Reliability and Security (ARES), 2013 Eighth International Conference on*, pages 11–20. IEEE, 2013.

[18] Marco Autili et al. Eagle: Engineering software in the ubiquitous globe by leveraging uncertainty. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 488–491. ACM, 2011.

[19] Indika Meedeniya et al. Architecture-based reliability evaluation under uncertainty. In *Pro-*

ceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS, pages 85–94. ACM, 2011.

[20] Faramarz Khosravi et al. Uncertainty-aware reliability analysis and optimization. In *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 97–102. IEEE, 2015.

[21] Catia Trubiani et al. Model-based performance analysis of software architectures under uncertainty. In *Proceedings of the 9th international ACM Sigsoft conference on Quality of software architectures*, pages 69–78. ACM, 2013.

[22] William S Jewell. Bayesian extensions to a basic model of software reliability. *IEEE Transactions on Software engineering*, (12):1465–1471, 1985.

[23] PV Suresh et al. Uncertainty in fault tree analysis: a fuzzy approach. *Fuzzy sets and Systems*, 83(2): 135–141, 1996.

[24] YA Mahmood et al. Fuzzy fault tree analysis: A review of concept and application. *International Journal of System Assurance Engineering and Management*, 4(1):19–32, 2013.

[25] DK Mohanta et al. Importance and uncertainty analysis in software reliability assessment of computer relay. *Proceedings of the Institution of Mechanical Engineers, Part O: Journal of Risk and Reliability*, 225(1):50–61, 2011.

[26] Ricardo J Rodríguez et al. Modelling security of critical infrastructures: a survivability assessment. *The Computer Journal*, 58(10):2313–2327, 2015.

[27] Glenn Shafer et al. *A mathematical theory of evidence*, volume 1. Princeton university press Princeton, 1976.

[28] Kari Sentz et al. *Combination of evidence in Dempster-Shafer theory*, volume 4015. Citeseer, 2002.

[29] Scott Ferson et al. Dependence in probabilistic modeling, dempster-shafer theory, and probability bounds analysis. *Sandia National Laboratories, Report No. SAND2004-3072*, 2004.

[30] Guttorm Sindre et al. Eliciting security requirements with misuse cases. *Requirements engineering*, 10(1):34–44, 2005.

[31] Guttorm Sindre. Mal-activity diagrams for capturing attacks on business processes. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 355–366. Springer, 2007.

[32] Mohammad Jabed Morshed Chowdhury. Towards security risk-oriented mal activity diagram. *International Journal of Computer Applications*, 56(10), 2012.

[33] Sébastien Gérard et al. 19 papyrus: A uml2 tool for domain-specific language modeling. In *Model-Based Engineering of Embedded Real-Time Systems*, pages 361–368. Springer, 2010.

[34] Abel Gmez et al. Dice profiles, 2015. http://dice-project.github.io/DICE-Profiles.

[35] Ali Sedaghatbaf. Sqme tool, 2015. http://twcl.iust.ac.ir/projects/sqme.html.

[36] George J Klir. *Uncertainty and information: foundations of generalized information theory*. John Wiley & Sons, 2005.

Ali Sedaghatbaf obtained B.S. and M.Sc. degrees in computer engineering (software) in 2009 and 2011, respectively from school of computer engineering, Iran University of Science and Technology (IUST), Tehran, Iran. Since 2013, he has been with the Trustworthy Computing Laboraratory in IUST as a Ph.D. student in software engineering. His current research interests include software architecture, model-driven engineering and cyber security.

Mohammad Abdollahi Azgomi obtained B.S., M.S. and Ph.D. degrees in computer engineering in 1991, 1996 and 2005, respectively from department of computer engineering, Sharif University of Technology, Tehran, Iran. His research interests include modelling and evaluation of security, formal methods, privacy and trust, and dependable and secure software development. Dr. Abdollahi Azgomi is currently an associate professor and director of Trustworthy Computing Laboratory (TwCL) at school of computer engineering, Iran University of Science and Technology, Tehran, Iran.

The ISC Int'l Journal of Information Security
ISeCure

July 2016, Volume 8, Number 2 (p. 167)
http://www.isecure-journal.org

Persian Abstract

ارزیابی کمی امنیت نرم‌افزار: روشی مبتنی بر UML/SecAM و نظریه شواهد

علی صداقت‌باف[1] و محمد عبداللهی ازگمی[2]

[1] دانشکده مهندسی کامپیوتر، دانشگاه علم و صنعت ایران، تهران، ایران

ارزیابی کمّی و مدل-مبنای امنیت نرم‌افزار در مرحله طراحی معماری امکان تشخیص زودهنگام خطاهای طراحی و در نتیجه کاهش هزینه‌های تغییر در مراحل بعدی چرخه حیات نرم‌افزار را فراهم می‌کند. در عین حال، دقت پارامترهای ورودی مدل‌ها در مراحل اولیه توسعه چالش‌برانگیز است. در حقیقت، به دلیل عدم وجود دانش کافی، تخمین دقیق پارامترهای امنیتی به ندرت امکان‌پذیر است. این موضوع در اغلب روش‌های ارزیابی موجود نادیده گرفته شده است. هدف از این مقاله توجه صریح به عدم قطعیت پارامترهای ورودی در فرایند ارزیابی امنیت است. به عبارت دقیق‌تر، ما از نظریه شواهد به منظور توصیف عدم قطعیت در پارامترهای ورودی و سنجش اثر آن بر معیارهای امنیتی استفاده می‌کنیم. در روش پیشنهادی از نمودارهای UML به منظور توصیف حملات امنیتی و از نمایه SecAM به منظور توصیف پارامترهای امنیتی استفاده می‌شود. همچنین، به منظور ارزیابی احتمال رخنه امنیتی، مدل‌های UML/SecAM به درخت‌های حمله تبدیل می‌شوند. در ضمن، به منظور بررسی کاربردپذیری روش پیشنهادی یک مطالعه موردی برروی یک سامانه خرید و فروش اینترنتی انجام گرفته و نتایج آن گزارش شده است.

واژه‌های کلیدی: معماری نرم‌افزار، ارزیابی امنیت، کمّی‌سازی عدم قطعیت، نظریه شواهد.