Measuring Security Investment Benefit for COTS Based Systems - A Stakeholder Value Driven Approach

Yue Chen, Barry Boehm — *Center for Software Engineering*
Luke Sheppard — *Information Technology Services*
University of Southern California
Los Angeles, CA, 90089-0781, USA
{yuec, boehm, lshep}@usc.edu

Abstract

This paper presents the Threat Modeling method based on Attacking Path Analysis (T-MAP) which quantifies security threats by calculating the total severity weights of relevant attacking paths for Commercial Off The Shelf (COTS) based systems. Compared to existing approaches, T-MAP is sensitive to an organization's business value priorities and IT environment. It distills the technical details of thousands of relevant software vulnerabilities into management-friendly numbers at a high-level. In its initial usage in a large IT organization, T-MAP has demonstrated significant strength in prioritizing and estimating security investment effectiveness, as well as in evaluating the security performance of COTS systems. In the case study, we demonstrate the steps of using T-MAP to analyze the cost-effectiveness of how system patching, user account control and firewall can improve security. In addition, we introduce a software tool that automates the T-MAP.

Keywords

Software Economics, Security, COTS, Threat, Attack Path

1. Introduction

The recent legislation in computer security namely the Sarbanes-Oxley Act, HIPAA, and Gramm-Leach-Bliley have made security no longer optional to a wide range of organizations. As the trend of the usage of third party Commercial-Off-The-Shelf (COTS) and open source software continuously increases [4, 6, 28], COTS security has become a major concern for many large organizations whose daily business heavily rely upon a healthy IT infrastructure. Competing with often limited IT resources and the fast changing internet threats, the ability to prioritize security investment correctly and efficiently has become a critical success factor for every security manager.

The ability to estimate security benefit is a key driver to perform effective security economics analysis [8]. Unfortunately, the benefit estimation for security investment has been very difficult because of lack of historical data, lack of effective metrics, and the complex and sensitive nature of security [11]. Just as Butler pointed out, without tools and firm statistical data, many security managers currently have to make decisions based on their experience, judgment, and best knowledge [11].

This paper presents the novel Security Threat-Modeling method based on Attacking Path analysis (T-MAP) which quantifies security threats by calculating the total severity weights of attacking paths that are relevant to IT systems. The severity weights of attack paths are sensitive to an organization's business value priorities as well as the technical severity of vulnerabilities. The goal of T-MAP is to distill the ad hoc COTS-intensive-IT-system vulnerability details into an executive-friendly threat profile at high-level, and help security managers reason the cost-effectiveness of security practices using meaningful numbers. In addition, T-MAP provides a strong quantitative method to evaluate the security of COTS candidates for IT infrastructure as well. In addition, it is tool automated.

The Case Study We demonstrate the T-MAP process through a case study on a production server that communicates to several sensitive databases. In our T-MAP application, the security manager followed T-MAP method steps and provided the necessary inputs, including a balanced view on value propositions from all key stakeholders perspectives, value dependencies on the production server, COTS information on the production server, and most likely attackers. Then he manually prioritized the most vulnerable organizational values, most-likely attackers, and a sample set of vulnerabilities that are relevant to the production server based on his experience. His prioritizing results were then compared to the T-MAP output for method evaluation.

Paper Roadmap The rest of the paper is organized as follows: Section 2 introduces related work; Section 3 make assumptions that T-MAP is based upon; Section 4 presents the details of the T-MAP framework; Section 5 introduces the Tiramisu software tool which automates the T-MAP framework; Section 6 demonstrates the case study results of using T-MAP to analyze security cost-effectiveness; Section 7 presents the method evaluation results; and Section 8 summarizes the conclusions and identifies future work.

* *Manuscript submitted for exclusive review by the 29th International Conference on Software Engineering, Sept. 8, 2006.*

2. Related Work

Our work is primarily relevant to the research areas of Value Based Software Engineering, Security Economics, COTS System Vulnerability, and Attack Tree.

The initial theory of Value Based Software Engineering unveils the fact that successful security should be achieved at a level that makes all key stakeholders winners [5]. In previous work, Bodin et al, Butler, and Boehm demonstrated using the attribute-rating method in security/software investment evaluation [3, 10, 11]. Recently, Bodin et al proposed a promising approach that uses Analytical Hierarchy Process (AHP) invented by Saaty to evaluate security solution alternatives [10]. These works reveal that the nature of software security investment is a multi-criteria decision making problem that involves both quantitative and qualitative criteria. The Figure of Merit method has unique strength here because it provides a strong logical decision making framework that does not require accurate values of risk relevant probabilities and frequencies, which are usually impractical to obtain.

In the Security Economics sector, Gordan and Loeb presented an quantitative economic model to determine the optimal amount to invest in security to protect a given set of information [16]; Hoo proposed a risk management approach to answer the question "how much security is enough" [18]; Butler demonstrated using the *multiple-attribute risk assessment* in SAEM to reason the cost-benefit of security investments [11]; Cavusoglu et al proposed a quantitative model based on game theory to evaluate security investments, which can reflect the technical features of specific security systems such as the Intrusion Detection Systems [12]. Along with the classic IT risk management methodologies [1, 13, 19, 30, 41], these works established a promising conceptual framework to reason the cost-effectiveness of security investment. However, though, most of the work is still at a very high-level and lack of fine-grained consideration to specific COTS system vulnerability. In addition, the accurate value of many of the parameters used in these models such as probabilities, frequencies and size of loss are usually very difficult to estimate even for experienced security manager.

In the COTS Software Vulnerability sector, previous efforts have made it possible to look up the vulnerabilities of given COTS systems from comprehensive vulnerability libraries with the name, the version and the vendor of most COTS software [21]. Third party COTS and open source vulnerabilities have been studied and published near real-time by authority organizations such as NIST, MITRE, CERT, Cisco, FrSIRT, OVAL, Microsoft, and SANS. Comprehensive software vulnerability databases have been made publicly available [15, 14, 23, 26, 33, 38]; The Common Vulnerability Exposures (CVE) naming standard has effectively uniformed the vulnerability names across vulnerability reporting sources [19]. However, though, most existing COTS software vulnerability studies are surprisingly stakeholder/value neutral and have a strong technical focus. Although several metrics and ratings systems are available to evaluate the severity COTS software vulnerabilities [22, 24, 32, 36, 39], these ratings can only reflect the technical features of the vulnerability and can be misleading to security managers. For example, the vulnerability that can only compromises availability has a max rating of 3.3 out of 10 (low) in the current NIST Common Vulnerability Scoring System (CVSS) (i.e., the CVE-2006-3468), but for many production servers, availability can be mission-critical. In the case, the rating largely missed its initial goal of helping prioritize vulnerabilities.

Clearly, a significant gap exists between current works on economic security and the COTS software vulnerability. The Value Based Software Engineering philosophy and principles provides a general guideline to bridge the gap. Technically, our work is based upon the Schneier's *Attack Tree* concept. *Attack Tree*, also known as "And/Or Trees", is an excellent tool to enumerate and visualize attack scenarios [44]. It can be easily presented in XML and processed by software tools, but has the weakness of "no standard way of building, simulating and analyzing these trees", and "the tool is only as good as the analyst using it" [45]. We improve the *Attack Tree* by customizing it with the classic IT risk management framework [30], as presented in detail in Section 4.

3. Assumptions

We limit the scope of our discussions on the security threats caused by published COTS system vulnerabilities, and do not cover the security threats from passive attacks such as Phishing. In addition, we assume the vulnerability database is accurate, comprehensive and up-to-date.

4. Value Driven Threat Modeling for CBS

The T-MAP framework is based upon the observations of: 1) the more security holes left open for an IT system, the less secure it is; 2) different IT servers might have different levels of importance in terms of supporting the business's core values; 3) the more vulnerabilities are exposed to motivated attackers, the more risk is involved. As a value driven approach, T-MAP uses the Attacking Path concept to characterize possible scenarios wherein an attacker can jeopardize organizational values. T-MAP calculates the severity weight of each *Attacking Path* (attacking scenario) based on not only the technical severity of the relevant system vulnerability, but also its value impacts. T-MAP then quantifies the IT system threat with the total weight of all possible Attacking Paths.

4.1 Nature of The Structured Attack Path

A typical scenario of how a successful attack can affect organizational values is illustrated in Figure 1. The core with a caption of "Org. Values" represents the key values that are important to an organization, for example, reputation and productivity; the second layer from inside to outside with a caption of "IT Infrastructure" stands for the computers and devices in the organization's IT infrastructure. The third layer with a caption of "Software Applications, COTS" stands for the commercial off the shelf software installed on the IT servers. The outer-most layer with a caption of "Firewall Wrapper" presents the Firewall or other types of security wrappers protecting the IT infrastructure. On the edge of the "Software Applications, COTS" layer, the small solid dots represent the software vulnerabilities that make attacks possible, while the small circles stands for the vulnerabilities that are made inaccessible to attackers by the firewall.

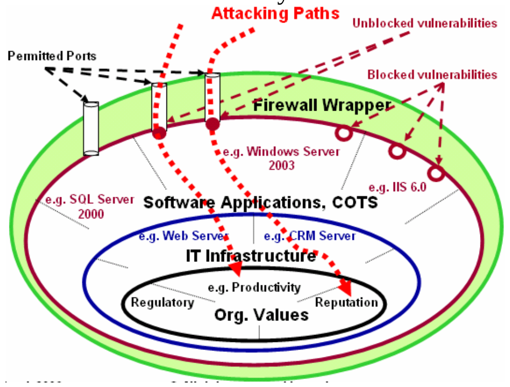


Figure 1 Nature of The Problem

In a typical successful attack, for example, the attacker may first find and get access to the victim host, then exploit the vulnerabilities of the COTS software installed on the victim host, thus compromise the confidentiality, integrity or availability of the victim host, and result in further damages to stakeholder values.

4.2 Modeling Attack Scenarios

T-MAP defines a formal framework to enumerate and evaluate attack scenarios based on attack graph analysis. We propose a structured *Attack Path* approach that incorporates the Schneier's attack tree and the classic IT risk management framework of attacker, asset, vulnerability, and impact to enumerate the possible attack scenarios [30, 44]. The new attack tree nodes are structured into five layers corresponding to each layer in Figure 1: the first layer nodes represent stakeholder values, for example, productivity, privacy, or reputation. The second layer nodes represent IT hosts that the stakeholder values rely upon. The third layer nodes represent the COTS software that is installed on the IT hosts. The fourth layer nodes represent software vulnerabilities that reside in the COTS software. The fifth layer nodes represent possible attackers, for example, insiders, external hackers, etc. The associations between fifth layer nodes and fourth layer nodes represent the attacker have adequate privilege and access to the vulnerabilities.

Under this framework, we found most nodes in the attack tree are "OR nodes" in real world. For example, if COTS software has multiple vulnerabilities, then the software is compromised if any of the vulnerability is exploited. Thus, the new attack tree can be simplified into a *Structured Attack Graph* by merging the nodes resenting the same entity as illustrated in Figure 2. In practice, the *Structured Attack Graph* can be derived from analyzing the stakeholder value result chains [5], data flow of organizational IT operations [37], and the system use/abuse cases [42].

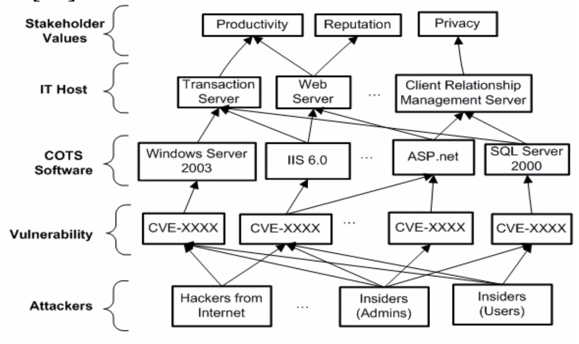


Figure 2 Structured Attack Graph

Definition 1: (Structured Attack Graph) A *Structured Attack Graph G=<Va, Vv, Vc, Vh, Vs, Eav, Evc, Ech, Ehs>* consists of five finite nonempty sets of vertices *Va, Vv, Vc, Vh, Vs* objects together with four (possibly empty) sets *Eav, Evc, Ech, Ehs* of ordered pairs of distinct vertices of *G* called edges. *Va, Vv, Vc, Vh,* and *Vs* represent the sets of attacker nodes, vulnerability nodes, COTS software nodes, IT host nodes, and Stakeholder Value nodes, respectively. *Eav=<a, v>* only contains ordered pairs of vertices where $a \in Va$ and $v \in Vv$; *Evc=<v, c>* only contains ordered pairs of vertices where $v \in Vv$ and $c \in Vc$; *Ech=<c, h>* only contains ordered pairs of vertices where $c \in Vc$ and $h \in Vh$; and *Ehs=<h, s>* only contains ordered pairs of vertices where $h \in Vh$ and $s \in Vs$;

Definition 2 (Structured Attack Path) For a given attack graph *G*, *A Structured Attack Path* is a 5-tuple *P=<A, V, C, H, S>*, where $A \in Va$, $V \in Vv$, $C \in Vc$, $H \in Vh$, $S \in Vs$, and the associations (edges) between elements from adjacent layers must exist. Clearly, *P* characterizes an attack scenario in terms of attacker, vulnerability, COTS software, IT Host, and Stakeholder Values, respectively.

Algorithm 1 enumerates all *Structured Attack Paths* for a given *Structured Attack Graph G*.

Algorithm 1 *List EnumerateAttackPath* (Structured_Attack_Graph *G*)

Return: The complete set of *Structured Attack Paths* associated with G

```
list attackPathList;
StructuredAttackPath p;
for(each sv in G.Vs) {    //s stands for stakeholder value
   // find the associated host to this value
   generate es1:=subset of G.Ehs where e1.s==sv, for ∀ e1 ∈ es1;
   for (each ehs in es1) {
      // find the COTS associated with this host
    generate es2:=subset of G.Ech where e2.h==ehs.h for ∀ e2 ∈ es2;
    for (each ech in es2) {
         // find vulnerability that affect the COTS software
         generate es3:=subset of G.Evc where e3.c == ech.c
                                                    for ∀ e3 ∈ es3;
        for(each evc in es3) {
           // Find attackers have privilege/access to the vulnerability
           generate es4:=subset of G.Eav where e4.v == evc.v
                                                    for ∀ e4 ∈ es4;
           for(each eav in es4) {
              p ← <eav.a, evc.v, ech.c, ehs.h, sv>;
              append p to attackPathList;
}}}}}
return attackPathList;
```

Figure 3 Enumerate Attack Paths

UML Model of Structured Attack Path

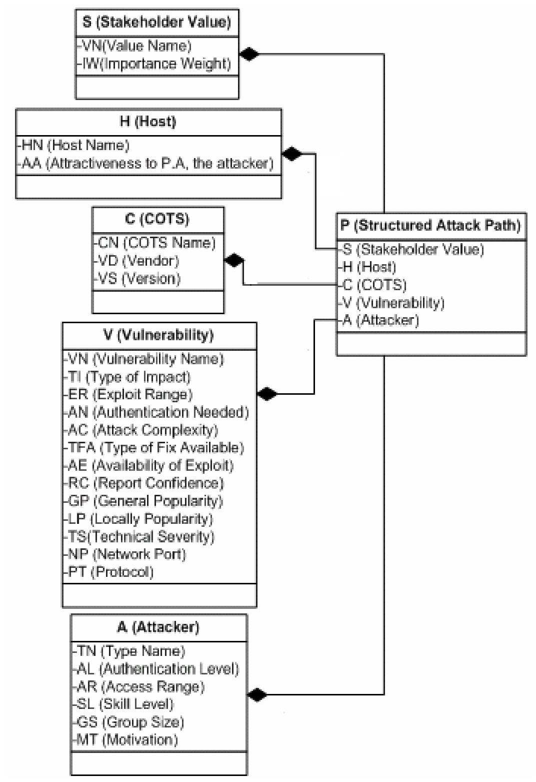


Figure 4 UML Model of A Structured Attack Path

In order to enable further fine-grained threat assessment of attack scenarios, T-MAP defines a set of threat-relevant attributes for each compositing elements of *P*, as illustrated in the Unified Modeling Language (UML) model of *P* [40] in Figure 4. Obviously, by definition, the class of *Structured Attack Path* has 1:1 composition associations with classes of *A*, *V*, *C*, *H*, *S*, respectively. These class attributes can be classified into three categories: probability-relevant, size-of-loss relevant and descriptive. The selection of the class attributes are primarily based on, but not limited to the NIST IT Risk Guide and the emerging national standard *CVSS* [24], a joint effort across CERT, Cisco, MITRE, eBay, ISS, Microsoft, Qualys, and Symantec. Assuming *P* is a Structured Attack Path in Structured Attack Graph *G*, the description of the attributes of *P* is summarized in Table 1. Because of the page limit, we elaborate some of the important attributes as follows.

Table 1 Attack Path Attribute Description (Partial)

S.VN	Affected stakeholder value. Clearly, $S \in G.Vs$;
S.IW	Level of importance of the stakeholder value *S.VN* in terms of a ratings value between 0 and 1. Details of obtaining the weight is presented in Section 4.3
H.HN	Affected host in the IT infrastructure. Clearly, $H \in G.Vh$;
H.AA	The estimated motivation of attacker *P.A* to attack host *P.H* terms of a ratings value between 0 and 1. Details of obtaining the weight is presented in Section 3.5
C.AP	If automated patching service is provided by vendor
V.VN	The Common Vulnerability Name (CVE) of the vulnerability that involved in this attack path [14]. Clearly, $V \in G.Vv$;
V.TI	The impact of the vulnerability *V* can cause on victim host *H* in terms of confidentiality, integrity, and availability
V.AC	Specifies if the associated vulnerability requires victim activities to enable such an attack, for example, opening an email attachment [24];
V.ER	Specifies if the vulnerability associated with the attacking path can be exploited remotely or locally [24];
V.AN	Specifies if the associated vulnerability requires valid user account on the victim computer [24];
V.TFA	Specifies if the fixes of the associated vulnerability are publicly available in terms of *Patch, Advisory, or None* [24];
V.GP	How popular the vulnerability is exploited in a general sense, for example, if listed as top 20 by authority organizations such as SANS.
V.NP	The network port(s) that the vulnerability uses, if there is any;
V.TS	The level of compromise to the victim host if exploited in terms of None, Partially, or Completely to confidentiality, integrity, or availability [24];
A.TN	The type of potential attackers, for example, insiders, hackers from internet, etc.
A.AL	The level of authentication that the attackers of *A.TN* have on host *H*.
A.R	If the attackers of A.TN have *Local* or *Remote* access to host *H*.
A.SL	The estimated skill level of the attacker in terms of *Low, Medium, High*
A.GS	The group size of this type of possible attackers
A.MT	The level of motivation of this type of attackers

These attributes are practically obtainable. We have established a vulnerability database that contains 18654 published COTS vulnerabilities affecting 31713 COTS software based on the comprehensive NIST National Vulnerability Database (NVD) [25]. It extends NVD by adding severity ratings, recommended solutions, and network port and protocol information from other sources such as Microsoft, SecurityFocus, Frsirt, SANS, and CERT. In addition, security managers usually are aware of the stakeholder value propositions, the value dependencies on the IT infrastructure, the installed COTS software, as well as the most-likely attackers.

4.3 Value Driven Evaluation of Attack Scenarios

As a descriptive model of attack scenarios, the adverse impact of an attack path can be characterized in terms of *confidentiality, integrity* and *availability* (CIA) [30]. However, CIA is a value neutral concept and does not reflect the utility function of stakeholders. In order to capture the stakeholders' value perception of the attack path impact, we firstly identify the system relevant key stakeholders/values. Then we establish a set of evaluation criteria based on the identified key stakeholder values. Finally, we evaluate the system CIA against each criterion.

Clearly, some of the stakeholder values can be quantified in terms of tangible units such as dollars. However, some of them are not, for example, the organizational reputation. Thus, we propose an evaluation method based on Figure of Merit method [3] (also known as AHP) to evaluate the impact severity of Attack Paths. Traditionally used in Decision Sciences as well as tradeoff analysis in System Engineering, the Figure of Merit method has unique strength in multiple criteria evaluation that involves quantitative and qualitative criteria. An example of applying AHP in security investment evaluation can be found in a recent work of Bodin et al [10].

Case Study Server X is an important development server that communicates with a sensitive USC database. Recently, a hacker crime against a USC database-driven web application was reported by Sci-Tech-Today on April 21, 2006 [35]. In order to improve system security, the security manager identified three alternative plans: 1) Patch server X and harden its Operating System; 2) build a firewall segment for server X to deny all access from remote non-USC IPs; 3) tighten user accounts. Given a budget around $6,000, the security manager is wondering what plan could produce the best cost-effectiveness for improving the security of *Server X*. Assuming the labor cost is $125 per hour, the estimated costs are:

Table 2 Security Cost Estimation of the Case Study

Plan	Practice	Cost
1	Apply software patch and work around (manually)	$250 per patch
2	Deploy and configure a firewall segment	$6,884
3	Tighten user account control (might affect other systems)	$750

Step 1: Identify Key Stakeholders/Values and Establish Evaluation Criteria

In the critical first step to evaluate the severity of attack paths, we identify stakeholder values that are dependent on IT system security. Furthermore, we establish a set of criteria to evaluate the value-criticality of attack paths.

For example, Table 3 summarizes the top-level USC stakeholders, stakeholder value propositions, and corresponding criteria weights that are relevant to Server X.

Table 3 Case Study Stakeholder Value Propositions

Weight	Criteria	Stakeholders			Organizational
		S1*	S2*	S3*	Value Description
0.095	1. Productivity	+	+	++	Help faculties, students and staff create, acquire, and disseminate knowledge.
0.738	2. Regulation		+	++	Comply with applicable federal, state, and local laws
0.167	3. Privacy				Protect the privacy of software, files, and materials stored on or transmitted by university computer equipment
0.648	a. Student	+		++	
0.230	b. Faculty		++	++	
0.122	c. Staff			++	

* S1: Students; S2: Faculties; S3: ISD Staff +: relevant; ++: highly relevant

The weight values in the first column are determined through AHP [10]. For example, the calculation of top-level criteria weights is illustrated in Table 4. The number in each cell represents the value pair-wise relative importance: number of 1, 3, 5, 7, or 9 in row *i* and column *j* stands for that the stakeholder value in row *i* is equally, moderately, strongly, very strongly, and extremely strongly more important than the stakeholder value in column *j*, respectively. By definition, cell(i,j) equals 1/cell(j,i). In order to calculate weight, each cell is divided by the sum of its column, and then averaged by each row [10]. The results of the final averaged weight are listed in the bolded *Weight* column in Table 4. The sum of weights equals 1. Similar process is used to determine the weights for sub-criteria.

Table 4 Pair-wise Comparison across ITS Values

	Productivity	Regulation	Privacy	**Weight**
Productivity	1	1/7	1/2	**0.095**
Regulation	7	1	6	**0.738**
Privacy	2	0.2	1	**0.167**

Step 2: Evaluate Compromise Scenarios for Each Host

Given the adverse impact of an attack path can be specified in terms of CIA of the victim host [30], we evaluate the relative severity of loss of confidentiality, integrity and availability against the evaluation criteria established in step 1.

The example results of our case study are summarized in Table 5. Cell(*i*,*j*) stands for the evaluation score of compromise condition *j* against evaluation criteria *i*. The weights of each row are derived through AHP pair-wise comparison. The sum of the weights of each row equals 1. In our case study, since the stakeholder values are additive, we used *weight of sum* as the evaluation function. The confidentiality and integrity have high score because they might result in regulation violation that has a high penalty.

Thus, for a given Attack Path, we can look up its value impact score by *S.VN* (stakeholder value name) and *V.TI* (type of impact) in Table 5. Obviously, the score values reflect the stakeholder value priorities.

It is important to note that selecting an appropriate evaluation function based on the stakeholder values is a critical step toward generating meaningful evaluation scores. An example of an inappropriate evaluation function that could lead to wrong selection of technologies was presented in Boehm's previous work [3], Chapter 15.

Table 5 Security Breach Scenario Evaluation Score

Weight	Criteria	Server X		
		Confidentiality	Integrity	Availability
0.095	1. Productivity	0.12	0.65	0.23
0.738	2. Regulation	0.63	0.26	0.11
0.167	3. Privacy			
0.648	a. Student	0.78	0.16	0.07
0.230	b. Faculty	0.54	0.30	0.16
0.122	c. Staff	0.60	0.20	0.20
Evaluation Score		0.593	0.287	0.121

Step 3 Modeling Attackers

The attackers are another significant factor that drives the severity of security incidents [30]. T-MAP models attacker groups with attributes of skill level, group size, and the motivation, represented by *A.SL*, *A.GS* and *A.MT*, respectively. Often, the accurate values of these attributes are difficult to estimate, for example, the group size of hackers from internet. Thus, in order to compare and prioritize attacker groups, we employ AHP to determine the relative weights of these attributes.

In our case study, the security manager identified three top attacker groups: *AG1*: the skilled internet hackers who know how to escalate user privileges on the victim; *AG2*: less skilled internet hackers who don't know how to escalate user privileges; *AG3*: insiders who usually have user accounts. The relative ratings of the *A.GS*, *A.SL*, and *A.MT* for each group were calculated through AHP as summarized in Table 6.b. For example, the relative group size ratings (*A.GS*) are calculated through AHP pair-wise comparison as illustrated in Table 6.a.

Furthermore, not all vulnerabilities are exploitable for all attackers. For example, some vulnerability requires attackers have valid user account on the victim host, thus they are not vulnerable to hackers who do not have such privileges. Some vulnerability requires attackers have local access to the victim host, thus they are not vulnerable to hackers who do not have local access. T-MAP reflects these differences in Attack Path generation based on the values of *Authentication Level* and *Remote/Local Access*, which are represented by *A.AL* and *A.R*, respectively. The information of Attacker Groups of *AG1*, *AG2*, and *AG3* in our case study is summarized in Table 6.c as follows.

Table 6.a Pair-wise Comparison of Attacker Group Size

	AG1	*AG2*	*AG3*	**Weight**
AG1	1	1/9	1/3	**0.07**
AG2	9	1	7	**0.78**
AG3	3	1/7	1	**0.15**

Table 6.b Ratings of A.GS, A.SL, and A.MT

	Description	*AG1*	*AG2*	*AG3*
A.GS	Group Size	0.07	0.78	0.16
A.SL	Skill Level	0.63	0.26	0.11
A.MT	Motivation	0.72	0.22	0.06

Table 6.c Vulnerability Exposure to Attackers

	AG1	*AG2*	*AG3*
A.R (Access)	Remote	Remote	Local
A.AL (Authentication Level)	User	None	Admin

Step 4 Assess Difficulty Level of Exploiting Attack Path

We model the difficulty level of exploiting an attack path based on the pioneer work of CVSS [24]. The ratings of all relevant attributes are between 0 and 1. Because of page limit, only part of the table is shown for illustration:

Table 7 Vulnerability technical attribute ratings (Partial)

Attribute	**Rating**	**Rating Value**
V.ER Exploit Range (Access Vector) *	Remote	1.0
	Local	0.7
V.TFA Type of Fix Available (Remediation Level) *	Official Fix	0.85
	Temporal Fix	0.9
	Work around	0.95
	None	1.0
V.AN Authentication Needed (Authentication) *	Required	0.6
	Not Required	1.0
V.GP Vuln. General Popularity	Listed as Top	1.0
	Not Listed as Top	0.8
H.AA Attractiveness of Asset Computer	Attractive	1.0
	Neutral	0.8
	Not Attractive	0.7

*[Note] Fields marked with * are referenced from CVSS* [24]

4.4 T-MAP Weighting System

After all the *Attack Path* attribute-ratings determined through the above value driven evaluations, we establish the T-MAP weighting system to reason and quantify the threat associated with given COTS system. Furthermore, we propose a novel method to assess the effectiveness of common security practices such as patching and firewalls.

A. Weight of Attack Paths

T-MAP scores the severity of the Attack Path with a numeric weight. Based on the classic risk calculation formula: *Risk = Probability * Size of Loss*, we calculate the weight of each *Attack Path* by multiplying its threat relevant attributes ratings together.

Definition 3: ***Weight of Attack Path***. For given Attack Path *P*, define:

$$Weight(P) = \prod_i Rating(P.Attribute_i)$$

where $P.Attribute_i$ enumerates once each of the *Probability Relevant* and the *Size-of-Loss Relevant* attributes of *P*.

The *Probability Relevant Attributes* include *H.AA*, *V.ER*, *V.AN*, *V.AC*, *V.TFA*, *V.GP*, *A.GS*, *A.MT*, and *A.SL*; The *Size of Loss Relevant Attributes*, including *S.IW*, and *V.TS*; Other attributes are *Descriptive*, including *S.VN*, *H.HN*, *C.CN*, *V.VN*, *V.TI*, *A.TN*, *A.AL*, and *A.AR*. (*Attributes descriptions are in Table 1.*)

By nature, the weight serves as a heuristic or proxy of the risk associated with the Attack Path. It is a rating number ranges between 0 and 1.

B. Total Threat of IT System

Under the assumption that the more attacking scenarios (or attacking paths) that exist, the more an organization's core values vulnerable, we use the total weight of all attacking paths to quantify the overall security threat to stakeholder values.

Definition 4: ***Total Threat.***

For given Structured Attack Graph *G*, define:

$$TotalThreat(G) = \sum_i Weight(AttackPath_i),$$

where *i* varies from 1 to the total number of attacking paths of *G* and $AttackPath_i$ represents the i-th *Attack Path* of *G*.

C. ThreatKey

As illustrated in Figure 2 in Section 4.2, each node of the *Structured Attack Graph* is associated with a set of attack paths that pass through that node. Thus, by comparing the total weight of all attacking paths passing through the node, we can determine those entities that are most vulnerable.

Definition 5: ***ThreatKey.***

For a given node *N* in a Structured Attack Grraph *G*, define:

$$ThreatKey(N) = \sum_i Weight(AttackPath_i),$$

where *i* varies from 1 to the total number of attacking paths that go through node *N*, and $AttackPath_i$ enumerates all the Attack Paths that go through *N*.

Therefore, the greater the value of the *ThreatKey* of a node, the more vulnerable that node is. For example, those vulnerabilities which have a higher *Threat Key* value should be patched at a higher priority than others. The COTS software associated with higher *ThreatKey* values is more vulnerable than those that have lower values. And the attackers that have higher *ThreatKey* values are more dangerous than others.

D. *Effectiveness* of Security Practices

Under T-MAP framework, the goal of many security practices can be understood as to block a certain set of Attacking Paths from the existing Attacking Path set. For example, Firewalls are to block those Attacking Paths that pass through controlled network ports. Enforcing physical access to important computers is done to block those *Attacking Paths* that require local access. Patching software vulnerabilities is done to block those *Attacking Paths* that are caused by the vulnerabilities.

In this sense, the effect of a security practice can be simulated by removing the corresponding attack paths and nodes that this security practice can suppress from the graph. For example, the effect of vulnerability patching can be simulated by removing all Attacking Paths that have vulnerability patches available from the Attacking Path set that is before applying patches.

Definition 6: *Effectiveness of Security Practices*

For a given security practice *SP*,

Effectiveness(*SP*)=1-*TotalThreat*(*AfterSP*)/*TotalThreat*(*BeforeSP*)

It represents the percentage of the total threat weight (before the security practice) that has been suppressed because of taking that security practice.

4.5 Comparison with CVSS

1) T-MAP is an organization value centric approach. It reflects stakeholder value priorities in assessing COTS system vulnerabilities. In comparison, CVSS is value-neutral and has a strongly technical focuses
2) T-MAP Weighting System distills the ad-hoc technical information of thousands of published software vulnerabilities into an executive-friendly high-level view of threats in terms of total weight Attacking Paths. In comparison, CVSS focuses on the severity of individual vulnerabilities.

5. Tiramisu Tool

We have implemented an automated T-MAP software test-bed with a project code of *Tiramisu* at USC. The *Tiramisu* takes three inputs: the general vulnerability information, an organization's IT infrastructure information, and how an organization's business values depend on its IT infrastructure. It calculates a complete list of associated Attacking Paths, and outputs the overall threats in terms of the total weights of Attacking Paths.

The *Tiramisu* employs layered software architecture as illustrated in Figure 5. From the bottom to the top, the "Automated Data Collecting Engine" collects the latest published vulnerability reports from CERT/CC, NIST, SANS, SecurityFocus, Symantec, and Microsoft websites automatically, formats and populates the data into the second layer "Data Storage and Management". Currently our database contains information on 17731 vulnerabilities that have been published on NIST since 1999 with extended information such as associated network ports, recommended solutions (by CERT, NIST, SecurityFocus respectively), if listed as top vulnerabilities by SANS, and so forth.

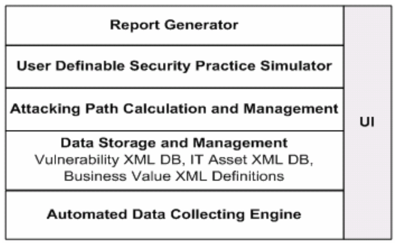

Report Generator
User Definable Security Practice Simulator
Attacking Path Calculation and Management
Data Storage and Management
Vulnerability XML DB, IT Asset XML DB, Business Value XML Definitions
Automated Data Collecting Engine
UI

Figure 5 Tiramisu Architecture

The "Data Storage and Management" layer includes an XML database implementing the ontology of vulnerabilities, IT asset computers, organizations' key business values, and the relationships among these entities. Through the GUI the users can easily specify the types of operating systems and COTS software installed on their IT servers, and specify how organizations' core business values can be affected by security breaches such as compromises of confidentiality, integrity, and/or availability on different IT servers.

The "Attacking Path Calculation and Management" layer consumes the data provided in the "Data Storage and Management" layer to generate a complete list of Attacking Paths, and calculates the severity weights of Attacking Paths based on user input.

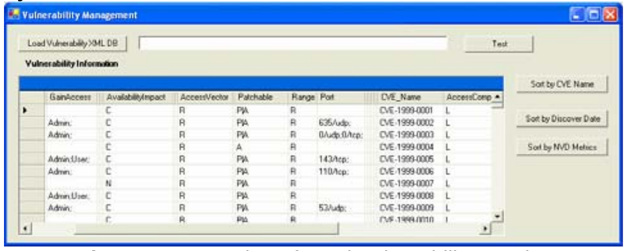
The "User Definable Security Practice Simulator" layer allows user to specify what kind of attacking paths can be blocked by certain security practices. This feature of calculating the total weight of attacking paths that are suppressed by certain security practices helps security managers estimate the effectiveness of different security practices.

6. Case Study Results

We have assessed the stakeholder/value impacts of possible compromise scenarios of server X as well as and identified/evaluated the major sources of possible attackers in section 4.3. With the T-MAP weighting system established, we continue our case study in following steps.

Step 6 Load Vulnerability Information

As next step, we load all 17,731 vulnerabilities published by NIST from 1999 to 2006 into the *Tiramisu* tool.



GainAccess	AvailabilityImpact	AccessVector	Patchable	Range	Port	CVE_Name	AccessComp
	C	R	P/A	R		CVE-1999-0001	L
Admin;	C	R	P/A	R	635/udp;	CVE-1999-0002	L
Admin;	C	R	P/A	R	0/udp;0/tcp;	CVE-1999-0003	L
	C	R	A	R		CVE-1999-0004	L
Admin;User;	C	R	P/A	R	143/tcp;	CVE-1999-0005	L
Admin;	C	R	P/A	R	110/tcp;	CVE-1999-0006	L
	N	R	P/A	R		CVE-1999-0007	L
Admin;User;	C	R	P/A	R		CVE-1999-0008	L
Admin;	C	R	P/A	R	53/udp;	CVE-1999-0009	L
	C	R	P/A	R		CVE-1999-0010	L

Figure 6 Screenshot of Load Vulnerability Database

Step 7 Determine the COTS Software Installed

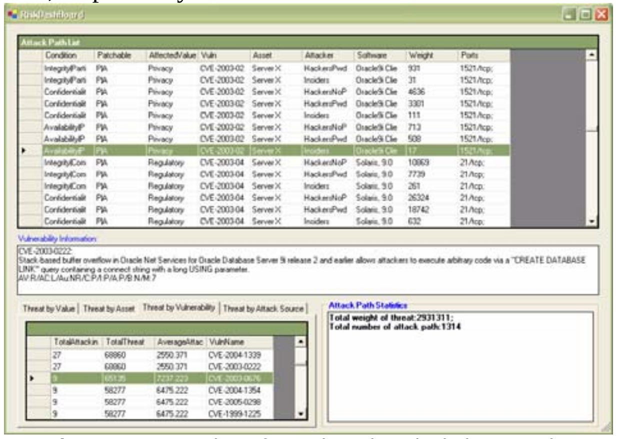
The COTS software that is installed on Server X is summarized in Table 8. Tiramisu takes this information as input.

Table 8 COTS Software Installed on Server X

Host	COTS Name	Version	Vendor
Server X	Solaris	9.0	SUN
	Directory Server	5.2.x	SUN
	Oracle9i DBMS	9.2.0.4	Oracle
	Oracle9i Client	9.2.0.2	Oracle

Step 8 Calculate Attack Paths

This step calculates the initial attack paths and associated severity weights of the current system without any security protection (Figure 7). The output shows that there are a total of 1314 attack paths with a total weight of 29.311. In addition, there are 106 COTS vulnerabilities reside in the current system. The top 3 vulnerabilities are CVE-2002-0858, CVE-2003-0722, and CVE-2004-1351 [22], associated with *ThreatKey* values of 1.539, 1.157 and 0.984, respectively.



Condition	Patchable	AffectedValue	Vuln	Asset	Attacker	Software	Weight	Ports
IntegrityParti	P/A	Privacy	CVE-2003-02	Server X	HackersPwd	Oracle9i Clie	931	1521/tcp;
IntegrityParti	P/A	Privacy	CVE-2003-02	Server X	Insiders	Oracle9i Clie	31	1521/tcp;
Confidentialit	P/A	Privacy	CVE-2003-02	Server X	HackersNoP	Oracle9i Clie	4636	1521/tcp;
Confidentialit	P/A	Privacy	CVE-2003-02	Server X	HackersPwd	Oracle9i Clie	3301	1521/tcp;
Confidentialit	P/A	Privacy	CVE-2003-02	Server X	Insiders	Oracle9i Clie	111	1521/tcp;
AvailabilityP	P/A	Privacy	CVE-2003-02	Server X	HackersNoP	Oracle9i Clie	713	1521/tcp;
AvailabilityP	P/A	Privacy	CVE-2003-02	Server X	HackersPwd	Oracle9i Clie	508	1521/tcp;
AvailabilityP	P/A	Privacy	CVE-2003-02	Server X	Insiders	Oracle9i Clie	17	1521/tcp;
IntegrityCon	P/A	Regulatory	CVE-2003-04	Server X	HackersNoP	Solaris, 9.0	10869	21/tcp;
IntegrityCon	P/A	Regulatory	CVE-2003-04	Server X	HackersPwd	Solaris, 9.0	7739	21/tcp;
IntegrityCon	P/A	Regulatory	CVE-2003-04	Server X	Insiders	Solaris, 9.0	261	21/tcp;
Confidentialit	P/A	Regulatory	CVE-2003-04	Server X	HackersNoP	Solaris, 9.0	26324	21/tcp;
Confidentialit	P/A	Regulatory	CVE-2003-04	Server X	HackersPwd	Solaris, 9.0	18742	21/tcp;
Confidentialit	P/A	Regulatory	CVE-2003-04	Server X	Insiders	Solaris, 9.0	632	21/tcp;

CVE-2003-0222:
Stack-based buffer overflow in Oracle Net Services for Oracle Database Server 9i release 2 and earlier allows attackers to execute arbitrary code via a "CREATE DATABASE LINK" query containing a connect string with a long USING parameter.
AV:R/AC:L/Au:NR/C:P/I:P/A:P/B:N/M:7

TotalAttackin	TotalThreat	AverageAttac	VulnName
27	68860	2550.371	CVE-2004-1339
27	68860	2550.371	CVE-2003-0222
9	65135	7237.222	CVE-2003-0676
9	58277	6475.222	CVE-2004-1354
9	58277	6475.222	CVE-2005-0298
9	58277	6475.222	CVE-1999-1225

Total weight of threat:2931311;
Total number of attack path:1314

Figure 7 Screenshot of Attack Path Calculation Results

Step 9 Determine What Type of Attack Path Can Be Suppressed by Each Security Practice

We use a *Suppressing Matrix* as shown in Table 9 to summarize the effects of the effectiveness of each alternative security investment plan.

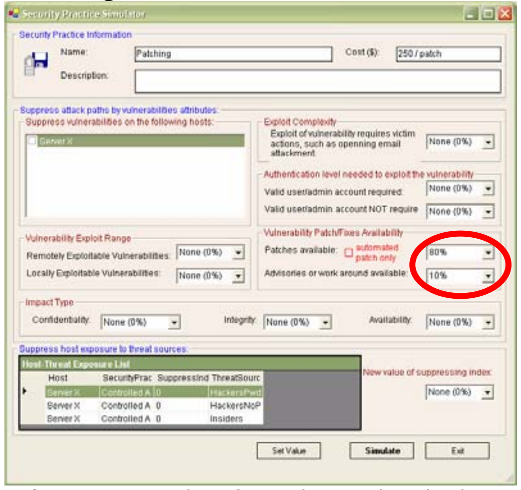
Table 9 Suppressing Matrix* (Partial)

Attack Path Attributes	Properties	Security Investment Plans		
		Firewall	Patch	User Ctrl.
V.TFA	Official Fix		80%	
	Temporal Fix		10%	
	Work around		10%	
	None			
V.AN	Required			100%
V.TI	Confidentiality			
	Integrity			
	Availability			
AG1	-	90%		10%
AG2	-	90%		90%
AG3	-			100%

* Attribute Definitions are in Table 1. AG1-3 are attackers, see Section 4.3

The first column represents the relevant Attack Path attributes. The second column is the possible ratings of each attribute. The percentage *p* in table cell at row *i* and column *j* stands for the security practice of column *j* can suppress *p* percent of the threat of those attack paths that have the attribute rating represented by row *i*. The suppressing percentages should be justified by experienced security managers.

For example, in our case study, our client estimated that 90% of the threats from internet hackers (Attack Group *AG1* and *AG2*) can be suppressed by firewall Thus the cell representing remote attackers are marked with 90% in column 1. The column of "Patch" represents that if the plan is to patch COTS software, in practice, only about 80% of the patches are applicable because of system compatibility and stability considerations. The column of "Acct. Ctrl." represents that by enforcing strict user account control, it can almost avoid all the attacks that need valid user account on *Server X*. In addition, by periodically changing system password and restricting system access, it can suppress almost all insiders, about 90% of the total threat from unskilled hackers; however, can only suppress about 10% of the skilled hackers. The *Tiramisu* tool screenshot of evaluating the effectiveness of patching *Server X* is illustrated in Figure 8 as follows.


Security Practice Simulator
Security Practice Information
Suppress attack paths by vulnerabilities attributes:
Suppress vulnerabilities on the following hosts:
Exploit Complexity
Authentication level needed to exploit the vulnerability
Vulnerability Exploit Range
Vulnerability Patch/Fixes Availability
Impact Type
Suppress host exposure to threat sources:
Host-Threat Exposure List
Set Value
Simulate
Exit

Name: Patching
Cost ($): 250 / patch
Description:

Server X

Exploit of vulnerability requires victim actions, such as openning email attackment: None (0%)

Valid user/admin account required: None (0%)
Valid user/admin account NOT require: None (0%)

Remotely Exploitable Vulnerabilities: None (0%)
Locally Exploitable Vulnerabilities: None (0%)

Patches available: ☐ automated patch only: 80%
Advisories or work around available: 10%

Confidentiality: None (0%)
Integrity: None (0%)
Availability: None (0%)

Host	SecurityPrac	SuppressInd	ThreatSourc
Server X	Controlled A	0	HackersPwd
Server X	Controlled A	0	HackersNoP
Server X	Controlled A	0	Insiders

New value of suppressing index: None (0%)

Figure 8 Screenshot of Security Practice Simulator

The *Tiramisu* assessment outputs for each security practice alternatives are summarized in Table 13. The columns of "Cost", "Total AP Weight", and "Effectiv." represent the cost, the total severity weight of all remaining attack paths, and the *Effectiveness* of each alternative, respectively. We found for Server X, the vulnerability of CVE-2005-1495, 2002-2089, 1999-0795, and 1999-0568 do not have patch or workaround available by the time the paper is written. Assuming the top 80% of the vulnerability has applicable patches, there is about 106*80% equals 85 applicable patches. Thus the cost of applying patches increases linearly along with the number of patches to apply at a cost of $250 per patch.

In order to determine the optimal plan for a budget around $6,000, we plotted the relationship between investment *Effectiveness* and the number of patches applied, as shown in Figure 9. Assuming the vulnerabilities that have higher *ThreatKey* values are fixed first, clearly, the plan of tightening user account as well as patching at the same time achieves the best *Effectiveness* of 87.6% at this budget. According to *Tiramisu* output, there are 22 patches need to be applied total.

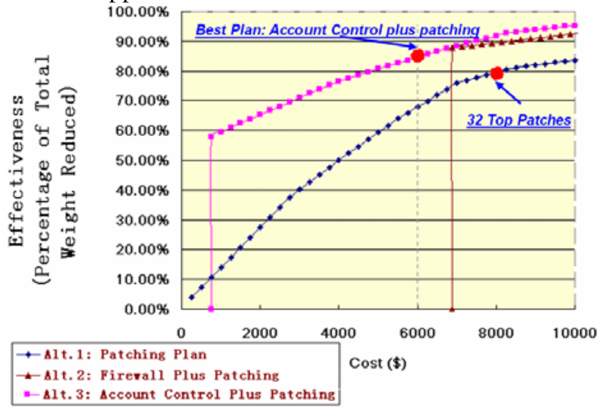


Figure 9 Optimal Security Practice Plan

It is important to note that Figure 9 does not directly answer the question what will be the optimal amount to invest. As the percentage measure of the total threat weight reduced by certain security practice, *Effectiveness* does not reflect the absolute value of security investment benefit, thus cannot be directly used for Net Value analysis.

Under the assumption that *Net Value*(n) = *A***Effectiveness*(n) – *Cost(n)*, where *A* is the net value of perfect security and *n* is the number of patches to apply, we plotted the *Relative Net Value (RVN)* analysis for Alt.2 for our case study, where *RVN*(n)=*NV*(n)/*NV*(0). The sweet spots is achieved at *n* where $\frac{dRVN(n)}{dn} = 0$. The trend meets the common sense that the more valuable perfect security, the higher optimal amount to invest should be.

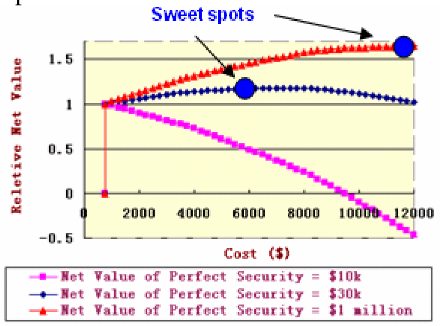


Figure 10 Sweet Spots of Investment

7. Method Evaluation

An experiment was conducted to test how well T-MAP method captures a security manager's priorities in security assessment. We sampled a set of *Server X* vulnerabilities randomly. Then the security manager was asked to rank the sample vulnerabilities manually into the order his perception of how severe it is. His rankings ware compared to the Tiramisu rankings as illustrated in Figure 11.

Excluding the irrelevant outlier of CVE-2003-0064, the regression shows an *R* square value of 0.86 indicating a strong fit. The security manager ranked CVE-2003-0064 as the least because the relevant application *dtterm* is not enabled on *Server X* at all; he ranked the CVE-2005-2072 a little higher because a program is running as *setuid* with root privilege on *Server X*, thus involves more security risks. In similar experiments on the subjects of "organization values under most threat" and "top attackers", *Tiramisu* generated priority orders well matching the security manager's rankings as well. The concluded optimal plan in Figure 9 was convincing to the security manager and matched his perception on the *Server X* security context.

Also, Figure 9 shows that for our case study, the top 30.2% vulnerabilities (32 out of 106) caused about 80% of the total threat. This result moderately matches the well known "20/80 rule" in security.

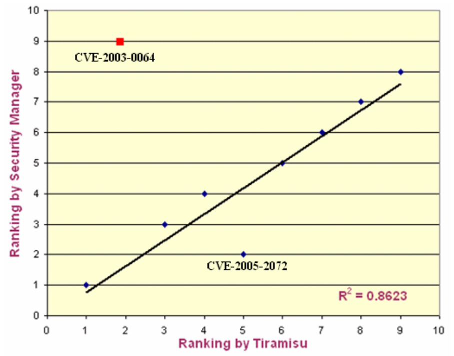


Figure 11 Human-Tiramisu Vulnerability Rankings Comparison

8. Conclusions and Future Work

This paper reports a case study in which we used the novel T-MAP threat modeling method to help our client prioritize possible security practices under a tight budget.

As a stakeholder value centric approach, the T-MAP defines a formal framework to measure COTS system security based on *Attack Path* weights. In our case study, it demonstrated significant strength in prioritizing security practice and estimating the effectiveness of security practices. In addition, it distills the technical details of thousands of published software vulnerabilities into executive-friendly numbers at a high-level. Furthermore, this process can be automated by the *Tiramisu* tool, thus greatly reduces the necessary human effort involved.

In our experiment of identifying top COTS software vulnerabilities and the organization values that are under most threat, the T-MAP generated similar priorities to the one ranked by the security manager manually, with a high correlation R square value of 0.86. We can conclude that at least in this instance T-MAP considerably well captured the human perceptions on stakeholder values through its method steps, and reflected the differences in the automated security evaluation process.

These conclusions are specific to the case study we have conducted. However, though, the method demonstrated considerable potentials even for big organizations whose daily business heavily relies upon a healthy IT security. For example, it can help

- Executives using cost-effectiveness analyses for their security practices and investments
- Security administrators identifying key vulnerability based on organizational value preferences
- IT system architects evaluating the security performance of COTS systems

Our clients commented that T-MAP was, "a valuable way of quantifying the very difficult tradeoffs that we have to make everyday."

However, since the tool as shown in Figure 11 generates some vulnerability overestimates and underestimates, it is important for managers to balance its recommendations with those based on expert judgment.

Also, the T-MAP method requires comprehensive, accurate and up-to-date vulnerability information. Currently we are experimenting using web search and AI text reading technologies in growing our vulnerability database based on the NIST National Vulnerability Database. For example, our automated data-collecting engine can classify the "vulnerability solution" on securityfocus.com into four categories of "patching", "change configurations", "block access", and "no solutions found" with accuracy of 87.3% according to our latest testing results. We believe further updates and improvements in the vulnerability database are critical and valuable.

In addition, as an empirical approach, the T-MAP method quantifies security threats all based on reported vulnerabilities of software, thus is not sensitive to unpublished vulnerabilities. Fortunately, the impact from published vulnerabilities is much less significant than from published ones. An empirical study conducted by Arora shows that the average attacks per host per day jumped from 0.31 to 5.45 after vulnerability status changed from "secret" to "published" [2]. In this sense, the T-MAP method can still capture the major part of the security threats for COTS software systems.

Acknowledgement

This research has been sponsored by the NSF Science of Design Program and the Affiliates of the USC Center for Software Engineering.

References

[1] AS/NZS 4360:1999 Australian Standard: Risk Management. Standards Australia (1999)

[2] A. Arora, R. Krishnan, A. Nandkumar , R. Telang and Y. Yang, Impact of Vulnerability Disclosure and Patch Availability - An Empirical Analysis, WEIS, 2004

[3] B. Boehm, Software Engineering Economics, Prentice Hall PTR, ISBN 0-13-822122-7, Pp223-242, 1981

[4] V. Basili and B. Boehm, "COTS Based System Top 10 List", Computer, Vol.34, no. 5, 2001, pp.91-93

[5] B. Boehm and A. Jain, "An Initial Theory of Value-Based Software Engineering" in S. Biffl, A. Aurum, B. Boehm, H. Erdogmus, P. Gruenbacher (eds.), Value-Based Software Engineering, Springer Verlag, 2005.

[6] B. Boehm, D.Port, Y. Yang, J. Bhuta, "Not All CBS Are Created Equally", Proceedings of Second International Conference on COTS Based Software Systems, ICCBSS 2003

[7] B. Boehm, L. Huang, A. Jain, R. Madachy, The Nature of Information System Dependability – A Stakeholder/Value Approach, Technical Report, 2004

[8] B. Boehm, K. Sullivan, Software Economics: A Roadmap, The Future of Software Engineering, ACM 2000, pp 319-343

[9] H. Berghel, D. village: The two sides of ROI: return on investment vs. risk of incarceration, Communications of the ACM, April 2005

[10] L. D. Bodin, L. A. Gordon, M. P. Loeb, Evaluating Information Security Investment Using the Analytic Hierarchy Process, Communications of The ACM, February 2005

[11] S. A. Butler, Software evaluation: Security attribute evaluation method: a cost-benefit approach, Proceedings of the 24th International Conference on Software Engineering, May 2002

[12] H. Cavusoglu, B. Mishra, S. Raghunathan, A model for evaluating IT security investments, Communications of the ACM, July 2004

[13] Centers for Medicare and Medicaid Services, CMS Information Security Risk Assessment Methodology, v1.1, 2002

[14] Common Vulnerability and Exposures, MITRE Corporation, http://cve.mitre.org/ (current 09/2006)

[15] FiSIRT Security Advisories, http://www.frsirt.org/english

[16] Gordon, L., and Loeb, M. The economics of information security investment. *ACM Trans, Inf. Syst. Sec. 5,* 4 (2002), 438–457

[17] L. A. Gordon, M. P. Loeb, Budgeting process for information security expenditures, Communications of The ACM, January 2006

[18] Hoo, K.J.S. How much is enough? A risk management approach to computer security. Ph.D. Dissertation, Stanford University, 2000.

[19] ISO 15288, Systems Engineering–System Life Cycle Processes, 2002

[20] ISO IS 15408, The Common Criteria for Information Technology Security Evaluation (CC) version 2.1, 1999

[21] R. A. Martin, Managing Vulnerabilities in Your Commercial-Off-The-Shelf (COTS) Systems Using An Industry Standards Effort, IEEE, 2002

[22] Microsoft Security Bulletin, Microsoft Corporation, http://www.microsoft.com/technet/security/bulletin/

[23] Microsoft Security Alert Severity Matrix, MS PSS Team http://www.microsoft.com/technet/security/alerts/matrix.mspx

[24] M. Schiffman, Common Vulnerability Scoring System (CVSS), http://www.first.org/cvss/ (current 09/2006)

[25] National Vulnerability Database, NIST, http://nvd.nist.gov/

[26] Open Vulnerability Assess Language, MITRE Corporation, http://oval.mitre.org/oval/, (current 09/2006)

[27] D. Port and Z.H. Chen, "Assessing COTS Assessment: How much is Enough?" Proc. 3rd Int'l Conf. COTS Based Software Systems (ICCBSS 04, LNCS 2959, Springer-Verlag, 2004, pp 183-198

[28] D. Reifer et al., "COTS Based Systems: Twelve Lessons Learned", Proc. 4rd Int'l Conf. COTS-Based Software Systems (ICCBSS 04), LNCS 2959, Springer-Verlag, 2004, pp.137-145

[29] D. Reifer, Making the Software Business Case, Addison Wesley, 2002

[30] G. Stoneburner, A. Goguen, A. Feringa, Risk Management Guide for IT Systems, NIST Special Publication 800-30, 2002

[31] SANS Top 20 Most Critical Vulnerabilities, SANS, http://www.sans.org/top20 (current 09/2006)

[32] SANS Critical Vulnerability Analysis Priority Ratings

[33] http://www.securityfocus.com/

[34] T.L. Saaty, *The Analytic Hierarchy Process*. McGraw-Hill, NY, 1980.

[35] Man Charged with Hacking USC Database, http://www.sci-tech-today.com/story.xhtml?story_id=11200CI6KD4W

[36] Symantec Threat Severity Assessment, http://www.symantec.com/avcenter/threat.severity.html

[37] Michael Howard, David LeBlanc, Writing Secure Code, Microsoft Press, 2002, ISBN 0-7356-1722-8, Chapter 4, pp 69-124

[38] US-CERT Vulnerability Database, US Computer Emergency Readiness Team, http://www.kb.cert.org/vuls/ (current 09/2006)

[39] US-CERT Vulnerability Metrics, US Computer Emergency Readiness Team, http://www.kb.cert.org/vuls/html/fieldhelp#metric

[40] http://www.uml.org/

[41] US. General Accounting Office, Information Security Risk Assessment: Practices of Leading Organizations, 1999

[42] D. Verdon, G McGraw, Risk analysis in software design, Security & Privacy Magazine, IEEE Volume 2, Issue 4, Jul-Aug 2004 Page(s):79 - 84

[43] Y.Yang, J. Bhuta, D. N. Port, B. Boehm, Value-Based Processes for COTS-Based Applications, IEEE Software, July-August 2005

[44] B. Schneier. Attack trees: Modeling security threats. Dr. Dobb's Journal, December 1999.

[45] G.C. Dalton II et al, Analyzing Attack Trees using Generalized Stochastic Petri Nets, Proceddings. of IEEE Workshop on Information Assurance, NY, 2006