

# Measuring Security Investment Benefit for Off the Shelf Software Systems - A Stakeholder Value Driven Approach

**Yue Chen, Barry Boehm**

*Center for Systems and Software Engineering*

*University of Southern California  
Los Angeles, CA, 90089-0781, USA*

*{yuec, boehm, lshep}@usc.edu*

**Luke Sheppard**

*Information Technology Services*

## 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, and systematically establishes the traceability and consistency from management-level organizational value propositions to technical-level security threats and corresponding mitigation strategies. In its initial usage in a large IT organization, T-MAP has demonstrated promising 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

*Security Economics, Software Economics, Security, Risk Management, Value Based Software Engineering, COTS, Threat, Attack Graph, ROI*

## 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 [5, 10, 49], 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 [12]. Unfortunately, to date, 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 [16]. 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 [16].

This paper presents a stakeholder value driven approach, the 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. It provides a strong quantitative framework to evaluate the security of COTS candidates for IT infrastructure as well. In addition, we have developed a software tool, namely *Tiramisu*, which can largely automate this framework.

**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, limitations, and future work.

## 2. Related Work

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

In the initial theory of Value Based Software Engineering, Boehm et al unveiled the fact that “your enterprise will succeed if and only if it makes winners of your success-critical stakeholders”. The authors traced many difficulties and failures that caused because of technical-value mismatches in current software engineering practices, and pointed out “the fundamental goal of all good design and engineering is to create maximal value added for any given investment” [9, 12, 13]. Clearly, stakeholder values usually have strong influences on security investment decision making. Thus, the question “how much security is enough?” should be answered under system stakeholder value/utility context. T-MAP fully adopted this philosophy and explicitly injected stakeholder value propositions into security threat evaluation, attempting to achieve consistency and traceability from high level stakeholder value propositions to low level security technical strategies.

In previous work, Bodin et al, Butler, and Boehm demonstrated using the attribute-rating method in security/software investment evaluation [8, 15, 16]. Recently, Bodin et al proposed a promising approach that uses Analytical Hierarchy Process (AHP) invented by Saaty to evaluate security solution alternatives [15, 56]. These works reveal that the nature of software security investment is a multi-criteria decision making problem that involves both quantitative and qualitative criteria, and the criteria should reflect stakeholder value propositions. T-MAP leverages the strength of AHP to quantify the stakeholder value priorities into normalized numeric weights. Furthermore, Figure of Merit and AHP are used to evaluate security threats against stakeholder value criteria.

In the Security Economics sector, as one of the pioneer works, Anderson pointed out information security is also driven by economic incentives besides technical factors [71, 72]; Gordan and Loeb presented a quantitative economic model to determine the optimal amount to invest in security to protect a given set of information [28]; Hoo proposed a risk management approach to answer the question “how much security is enough” [30]; Butler demonstrated using the *multiple-attribute risk assessment* in SAEM to reason the cost-benefit of security investments [16]; 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 [17]. Also, Cavusoglu et al conducted an economic study on “how often to update software systems with necessary patches” [18]. August and Tunca studied the effect of user incentives on software security decision making by comparing four alternative patching policies [4]. All these works, along with the classic IT risk management methodologies [1, 22, 33, 52, 68], scoped a rich and sound theoretical framework to reason the cost-effectiveness of security investment. But, in general, the difficulties in estimating the probabilities, frequencies, size of losses and other similar parameters somehow constrained the practical usage of these approaches.

In the COTS Software Vulnerability sector, previous efforts have made it possible to look up the COTS software vulnerabilities from a comprehensive database from name, version and vendor [39]. Third party COTS and open source vulnerabilities have been published and discussed 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 [24, 26, 40, 46, 55, 65]; The Common Vulnerability Exposures (CVE) naming standard has effectively uniformed the vulnerability names across vulnerability reporting sources [24]. Security scanners are tools such as the Kuang tool [6], the Computer Oracle and Password System (COPS) [27], Nessus [44], SATAN [35], ISS [34], are developed to detect system security vulnerabilities and mis-configurations. However, most current vulnerability studies are stakeholder/value neutral. The current leading COTS software vulnerabilities rating systems [41, 42, 54, 62, 66] are strongly technical-focused. For example, as a representative leading approach, NIST Common Vulnerability Scoring System (CVSS) is a vulnerability ranking system that covered comprehensive technical details of vulnerabilities in depth. However, missing the value context, in some situations, the ratings can be very misleading to security managers. For example, a vulnerability that only compromises availability has a max rating of 3.3 out of 10 (low) in the current CVSS (i.e., the CVE-2006-3468), but for many production servers, availability can be mission-critical. In this case, the rating largely missed its initial goal of prioritizing vulnerabilities. In T-MAP, the technical evaluation on vulnerabilities is partially based on CVSS for its strength in vulnerability technical assessment. Furthermore, we systematically extended the evaluation to stakeholder value context, and the technical information of large volumes of vulnerabilities is distilled into management-friendly numeric measures with

automated software tool.

Another branch of related work is *Attack Tree*, which is also known as “And/Or Trees” [59]. The history of *Attack Tree* can be traced back to the *Fault Tree* developed by Bell Labs and the U.S. Air Force in 1960s [51]. The software related application of *Fault Tree* is also known as *Threat Tree*, which is an excellent tool for attack scenario analyses [2, 31]. In previous work, Sheyner and Wing established the first formal *Failure Scenario Graphs* based on Büchi Automata. They devised two algorithms to produce scenario graphs with tools implemented [57, 58, 70]. In order to improve the scalability problem that Sheyner’s approach encountered [48], Qu et al proposed a formal framework based on “dependency graph” and reduced the size of the graph to polynomial to the network size [45, 48]. Currently, a significant challenge to bring the attack tree analyses into practical is the difficulties in collecting accurate information such as the exploit pre-condition and post-condition of large volume of software vulnerabilities. In addition, other researchers criticized “there is no standard way of building, simulating and analyzing attack trees” and “the tool is only as good as the analyst using it” [25]. T-MAP employs a special type of Attack Tree, *the Structured Attack Tree*, to model the stakeholder value impact path from COTS vulnerability exploits. The key difference is that stakeholder value/utility is explicitly incorporated into the tree structure.

There are some other relevant systems. *Stellar* is a system that can construct security attack scenarios from IDS alerts and then prioritize the scenarios based on pre-defined rules and keywords. Its attack scenarios are derived from categorizing and grouping IDS alerts [14]. *Stellar* made a remarkable contribution in prioritizing the IDS alerts and security events based on the alert characteristics such as the source/dest IP addresses. Compared to T-MAP, *Stellar* is strongly technical focused and value-neutral. Perceivably, its prioritization output may largely depend on how good the rules are, but no explicit mechanism was found in *Stellar* to connect its rules to organizational value propositions. This may cause value-tech mismatches.

*Skybox* is another interesting commercial system that generates attack paths based on network topologies and identify critical vulnerabilities [61]. However, the information about it is very much limited and we are unaware of any scientific validation of this system. According to its whitepapers and a related patent, it has its own vulnerability database and uses a special network monitoring engine to generate attack paths [37, 61]. The security risk is calculated through multiplying the probability of a successful exploit of attack path with the size of loss of such an exploit. This approach has clear credits in automating risk assessment through attack tree analyses. However, based on the limited information available, its output may be just as good as the estimations of the probabilities, frequencies, and size of loss, which are usually very difficult to obtain [16]. In addition, we are unaware of how comprehensive/accurate its generated attack paths are, which also may have major influence on the system output. Other researchers reported it may have scalability issues [37].

### 3. Scope and Assumptions

It is important to point out that the scope of this paper focuses on the security threats caused by published COTS software vulnerabilities. As known, an organization’s IT infrastructure may involve rich components other than COTS software, for example, data, in-house developed software, network devices, printers, etc. These components may cause security risks as well. But, the COTS software such as operating systems, database servers, web services usually are the most fundamental and important components of IT infrastructures, which are also targeted by a large portion of security attacks.

Also, as a practical approach, T-MAP is largely constrained by the fact that existing vulnerability databases are still not sufficient to generate practical multi-step-exploit attack graph analyses because of the known difficulties, particularly the scalability issues and modeling the “pre-condition” and “post-condition” of large volumes of COTS vulnerability exploits. At current stage, we limit scope of T-MAP to one-step-exploits only.

In addition, we assume the vulnerability database is accurate, comprehensive and up-to-date. This assumption is reasonable based on our survey on existing COTS software vulnerability databases [24, 26, 40, 46, 55, 65].

### 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 Attack Path concept to characterize possible scenarios wherein an attacker can jeopardize organizational values. T-MAP calculates the severity weight of each *Attack Path* (attack 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 Attack Paths.

#### 4.1 Nature of The Structured Attack Graph

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 Hosts” stands for the IT servers and hosts in 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.

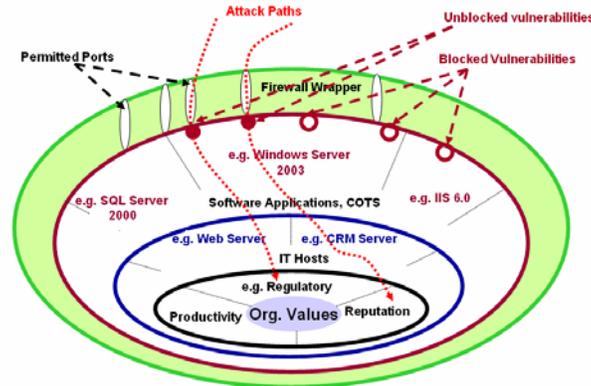


Figure 1 Nature of The Problem

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 (CIA) of the victim host, and result in further damages to stakeholder values.

We liken the COTS system security to defending a castle.

**Castle Defense Analog:** *measure the security of a castle by the value of treasures in the castle, the number of holes on the walls, as well as the size of the holes.*

Based on the analog above, as a summary, we attempt to measure security threat through the following steps: (1) identify key stakeholders and value propositions (the treasures in the castle); (2) establish a set of security evaluation criteria based on stakeholder value propositions; (3) enumerate and analyze attack paths based on a comprehensive COTS vulnerability database containing 27,400 vulnerability information (the holes); (4) evaluate the severity of each scenario in terms of numeric ratings against the evaluation criteria established in Step 2 (the size of the holes); (5) the security threat of each vulnerability is quantified with the total severity ratings of all attack paths that are relevant to this vulnerability; (6) system total threat is quantified with the total severity ratings of all attack paths. Step 3~6 are tool automated.

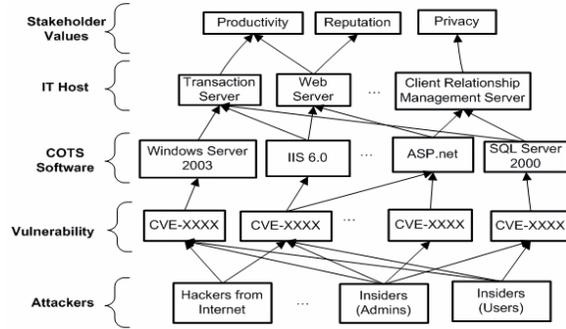
## 4.2 Modeling Attack Scenarios

T-MAP defines a formal framework to enumerate and evaluate attack scenarios based on attack graph analysis. We propose a *Structured Attack Graph* 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 [52, 59].

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 representing the same entity as illustrated in Figure 2. In practice, the *Structured Attack Graph* can be derived from analyzing the stakeholder value result chains [9], data flow of organizational IT operations [64], and the system use/abuse cases [69].

Perceivably, T-MAP can be extended to analyses more complex multi-step-exploit attack graph by chaining the exploit steps between the layers of “IT hosts”, “COTS Software”, and “Vulnerability” in the Structured Attack Graph, if the connect-ability between exploits is known.



**Figure 2** Structured Attack Graph

**Definition 1: (Structured Attack Graph)** A *Structured Attack Graph*  $G = \langle Va, Vv, Vc, Vh, Vs, Eav, Evc, Ech, Ehs \rangle$  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.  $Eav = \langle a, v \rangle$  only contains ordered pairs of vertices where  $a \in Va$  and  $v \in Vv$ ;  $Evc = \langle v, c \rangle$  only contains ordered pairs of vertices where  $v \in Vv$  and  $c \in Vc$ ;  $Ech = \langle c, h \rangle$  only contains ordered pairs of vertices where  $c \in Vc$  and  $h \in Vh$ ; and  $Ehs = \langle h, s \rangle$  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  $P = \langle A, V, C, H, S \rangle$ , 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** is a brute force algorithm that enumerates all *Structured Attack Paths* for a given *Structured Attack Graph*  $G$ .

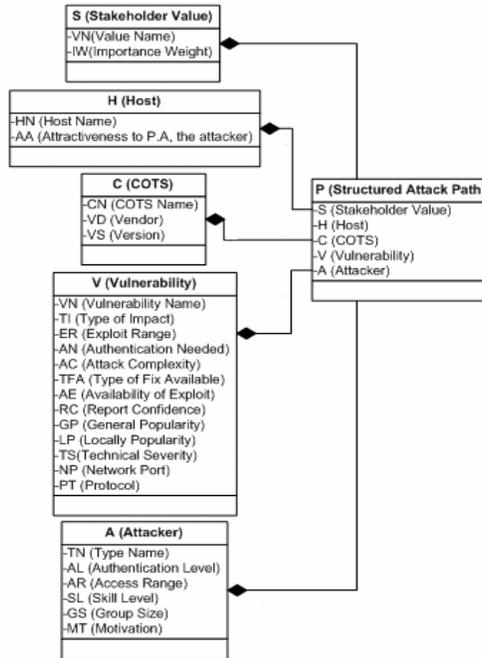
```

Algorithm 1 List EnumerateAttackPath (Structured_Attack_Graph G)
Return: The complete set of Structured Attack Paths associated with G
1: list attackPathList;
2: StructuredAttackPath p;
3: for(each sv in G.Vs) { //s stands for stakeholder value
4: // find the associated host to this value
5: generate es1:=subset of G.Ehs where e1.s==sv, for  $\forall e1 \in es1$ ;
6: for (each ehs in es1) {
7: // find the COTS associated with this host
8: generate es2:=subset of G.Ech where e2.h==ehs.h for  $\forall e2 \in es2$ ;
9: for (each ech in es2) {
10: // find vulnerability that affect the COTS software
11: generate es3:=subset of G.Evc where e3.c == ech.c
for  $\forall e3 \in es3$ ;
12: for(each evc in es3) {
13: // Find attackers have privilege/access to the vulnerability
14: generate es4:=subset of G.Eav where e4.v == evc.v
for  $\forall e4 \in es4$ ;
15: for(each eav in es4) {
16: p ← <eav.a, evc.v, ech.c, ehs.h, sv>;
17: append p to attackPathList;
18: }}}}
19: 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  $P$  [67] 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 [42], 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 [24]. 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 [42];
$V.ER$	Specifies if the vulnerability associated with the attacking path can be exploited remotely or locally [42];
$V.AN$	Specifies if the associated vulnerability requires valid user account on the victim computer [42];
$V.TFA$	Specifies if the fixes of the associated vulnerability are publicly available in terms of <i>Patch, Advisory, or None</i> [42];
$V.GP$	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 [42];
$A.TN$	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 $H$ .
$A.AR$	If the attackers of $A.TN$ have <i>Local</i> or <i>Remote</i> access to host $H$ .
$A.SL$	The estimated skill level of the attacker in terms of <i>Low, Medium, High</i>
$A.GS$	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 27,400 published COTS vulnerabilities affecting 31713 COTS software based on the comprehensive NIST National Vulnerability Database (NVD) [43]. 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 practice, the stakeholder value propositions, the value dependencies on the IT infrastructure, the installed COTS software, and the most-likely attackers can be obtained from security managers through structured interview.

### 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) [52]. 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 [8] and AHP [15, 56] 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 [15].

**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 [60]. 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 Value Description
		S1*	S2*	S3*	
0.095	1. Productivity	+	+	++	Help faculties, students and staff create, acquire, and disseminate knowledge.
0.738	2. Regulation		+	++	
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 [15]. 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 [15]. 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	<b>Weight</b>
Productivity	1	1/7	1/2	<b>0.095</b>
Regulation	7	1	6	<b>0.738</b>
Privacy	2	0.2	1	<b>0.167</b>

**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 [52], 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 [8], Chapter 15.

**Table 5** Security Breach Scenario Evaluation Score

<i>Weight</i>	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
<b>Evaluation Score</b>		0.593	0.287	0.121

**Step 3 Modeling Attackers**

The attackers are another significant factor that drives the severity of security incidents [52]. 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

	<i>AG1</i>	<i>AG2</i>	<i>AG3</i>	<b>Weight</b>
<i>AG1</i>	1	1/9	1/3	<b>0.07</b>
<i>AG2</i>	9	1	7	<b>0.78</b>
<i>AG3</i>	3	1/7	1	<b>0.15</b>

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

	Description	<i>AG1</i>	<i>AG2</i>	<i>AG3</i>
<i>A.GS</i>	Group Size	0.07	0.78	0.16
<i>A.SL</i>	Skill Level	0.63	0.26	0.11
<i>A.MT</i>	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 [42]. 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 alue
V.ER	Remote	1.0
Exploit Range (Access Vector) *	Local	0.7
V.TFA	Official Fix	0.85
Type of Fix Available (Remediation Level) *	Temporal Fix	0.9
	Work around	0.95
	None	1.0
V.AN	Required	0.6
Authentication Needed (Authentication) *	Not Required	1.0
V.GP	Listed as Top	1.0
Vuln. General Popularity	Not Listed as Top	0.8
H.AA	Attractive	1.0
Attractiveness of Asset	Neutral	0.8
Computer	Not Attractive	0.7

[Note] Fields marked with \* are referenced from CVSS [42]

#### 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 quantitative 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 Graph  $G$ , 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  $N$ .

---

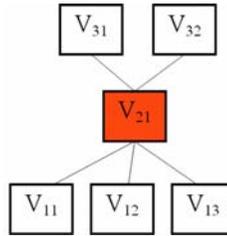
Therefore, the greater the value of the *ThreatKey* of a node, the more vulnerable that node is. For example, those vulnerabilities which have a higher *Threat Key* value should be patched at a higher priority than others. The COTS software associated with higher *ThreatKey* values is more vulnerable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

**A Fast Algorithm to Calculate ThreatKey**

By definition, the *ThreatKey* of a node is the sum of the weight of all attack paths that go through that node, but we found the calculation of the *ThreatKey* does not necessarily require emulating all the Attack Paths. We devised a fast algorithm with a time complexity of  $O(n)$  to calculate *ThreatKeys*, where  $n$  is the sum of total number of nodes of all layers in  $G$  and the total number of edges in  $G$ .

Specifically, for a given *Structured Attack Graph*  $G$ , the weight of each node can be calculated by multiplying the ratings of all its *Attack Path* relevant attributes together. So, the weight of a *Attack Path* can be computed by multiplying the weights of all nodes on that *Attack Path*. We observed that in a simplified *Structured Attack Graph* as shown in the following Figure, for example, if  $ThreatKey(V_{21}) = \sum Weight(AttackPath_i)$  then

$$\begin{aligned} ThreatKey(V_{21}) &= TopDownWeight * BottomUpWeight / W(V_{21}), \\ &\text{where} \\ BottomUpWeight &= W(V_{21}) * [W(V_{31}) + W(V_{32})] \\ TopDownWeight &= W(V_{21}) * [W(V_{11}) + W(V_{12}) + W(V_{13})] \end{aligned}$$



**Figure 5** ThreatKey Calculation for 3-layer-Attack-Graph

This calculation can be expanded to a standard *Structured Attack Graph*. For each node, all the attack paths that go through it form two trees, one above it and one below it, as illustrated in Figure 5. The *BottomUpWeight* represents the *ThreatKey* weight contributed by the upper tree; the *TopDownWeight* represents the *ThreatKey* weight contributed by the lower tree. Both *BottomUpWeight* and *TopDownWeight* of each node can be calculated recursively, as illustrated in pseudo-code in Algorithm 3 and 4. Algorithm 2 calculates the *ThreatKey* of each node. Furthermore, for prioritizing purpose, the nodes can be sorted in  $O(n \cdot \lg(n))$  time with quick sort or similar algorithms.

---

**Algorithm 2** *Void CalculateAttackPathSeverityWeight(Structured\_Attack\_Graph G)*

---

**Return:** A updated *Structured Attack Graph*  $G$  wherein the *ThreatKey* of Each Node calculated

---

```
1:   for(each v in G.Va U G.Vv U G.Vc U G.Vh U G.Vs) { //multiply the severity drivers of each vertices together as the
                                     initial severity weight
2:       v.initialThreatKey = [] v.Attribute;
3:   }
4:   for(each v in G.Va) { //The BottomUpWeights for all nodes are calculated in the recursive calculation
5:       v.bottomUpThreatKey = CalculateBottomUpWeight(G, v);
6:   }
7:   for(each v in G.Vs) { //The TopDownWeights for all nodes are calculated in the recursive calculation
8:       v.topDownThreatKey = CalculateTopDownWeight(G, v);
9:   }
10:  for(each v in G.Va U G.Vv U G.Vc U G.Vh U G.Vs) { //multiply the severity drivers of each vertices together as the
                                     initial severity weight
11:      v.threatKey = v.bottomUpThreatKey * v.topDownThreatKey / v.initialThreatKey;
12:  }
13:  return;
```

---

---

**Algorithm 3** float *CalculateBottomUpWeight*(Structured\_Attack\_Graph  $G$ , Vertice  $v$ )

---

**Return:** A updated *Structured Attack Graph*  $G$  with temp results toward getting ThreakKey

---

```

1:   v.bottomUpThreatKey = 0;
2:   if ( $v \in G.Vs$ ) { //reach the top layer, the values in the graph
3:       return v.initialThreatKey;
4:   } else {
5:       for(each  $v'$  where  $v'$  (is in the adjacent upper layer of  $v$ ) and
6:           ( $v$  is associated with  $v'$ ) ) {
7:           v.bottomUpThreatKey += v.initialThreatKey * CalculateBottomUpWeight( $G, v'$ );
8:       }
9:       return v.bottomUpThreatKey;
10:  }
```

---



---

**Algorithm 4** float *CalculateTopDownWeight*(Structured\_Attack\_Graph  $G$ , Vertice  $v$ )

---

**Return:** A updated *Structured Attack Graph*  $G$  with temp results toward getting ThreakKey

---

```

1:   v.topDownThreatKey = 0;
2:   if ( $v \in G.Va$ ) { //reach the bottom layer, the attackers in the graph
3:       return v.initialThreatKey;
4:   } else {
5:       for(each  $v'$  where  $v'$  (is in the adjacent lower layer of  $v$ ) and
6:           ( $v$  is associated with  $v'$ ) ) {
7:           v.topDownThreatKey += v.initialThreatKey * CalculateTopDownWeight( $G, v'$ );
8:       }
9:       return v.topDownThreatKey;
10:  }
```

---

#### 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. A *Suppressing Matrix* as illustrated in Table 9 demonstrates how some of the popular security practices may affect the *Structured Attack Graph*.

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  $SP$ ,

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

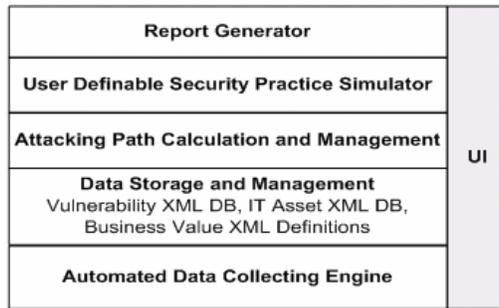

---

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

### 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 6. 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 27,400 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.



**Figure 6** 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.

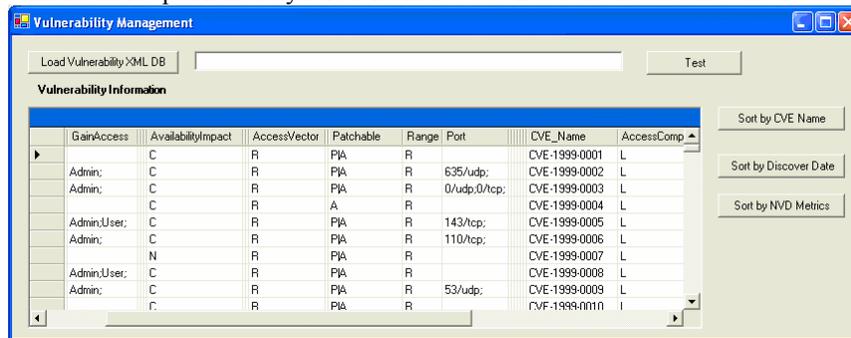
The “User Definable Security Practice Simulator” layer allows user to specify what kind of attacking paths can be blocked by certain security practices. This feature of calculating the total weight of attacking paths that are suppressed by certain security practices helps security managers estimate the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

## 6. Case Study Results

We 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 the following steps. The results are the snapshot at the time of Sept. 2006 when the case study was conducted.

### Step 6 Load Vulnerability Information

In this step, 17,731 vulnerabilities published by NIST from 1999 to 2006 are loaded into the *Tiramisu* tool.



**Figure 7** 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 8). 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 [24], associated with *ThreatKey* values of 1.539, 1.157 and 0.984, respectively.

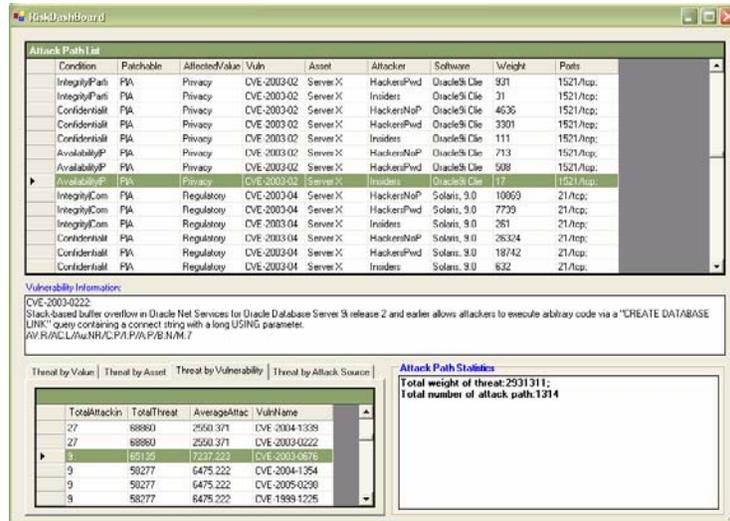


Figure 8 Screenshot of Attack Path Calculation Results

### 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.	Data Encryption	Backup Server	Digital Signature
V.TFA	Official Fix		80%				
	Temporal Fix		10%				
	Work around		10%				
	None						
V.AN	Required			100%			
V.TI	Confidentiality				95%		
	Integrity						100%
	Availability					80%	
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 9 as follows.

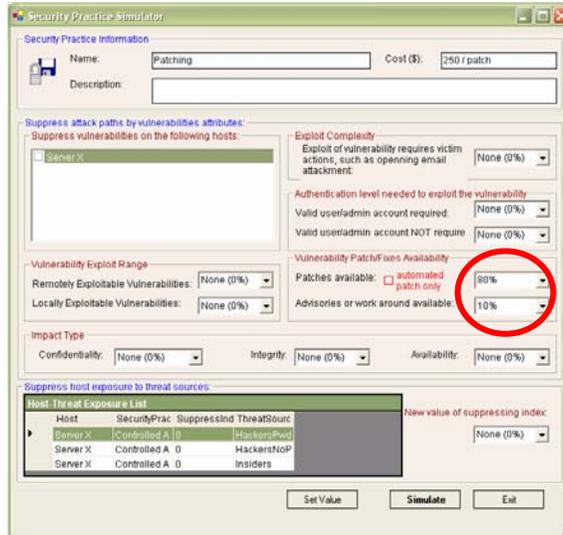


Figure 9 Screenshot of Security Practice Simulator

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  $106 \times 80\%$  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 10. Assuming the vulnerabilities that have higher *ThreatKey* values are fixed first, clearly, the security manager should first tighten the user account on Server X, and then patch the top 22 vulnerabilities to achieve the best *Effectiveness* while stay in the budget. According to the *Tiramisu* output, the investment *Effectiveness* of this plan is 87.6%.



Figure 10 Optimal Security Practice Plan

It is important to point out that Figure 10 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.

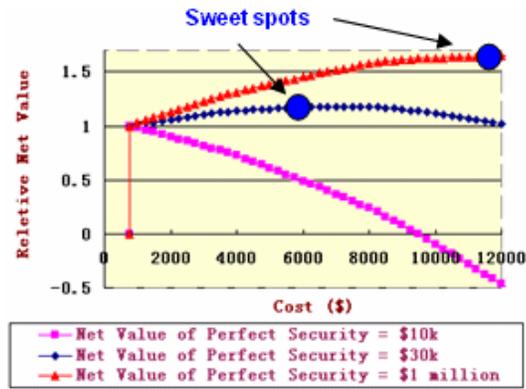


Figure 11 Sweet Spots of Investment

## 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 were compared to the Tiramisu rankings as illustrated in Figure 12.

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 *dterm* 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 10 was convincing to the security manager and matched his perception on the *Server X* security context.

Also, Figure 10 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 risk assessment. The top vulnerabilities identified by the tool roughly met the security manager’s experience except for a few outlier vulnerabilities that are associated with disabled services.

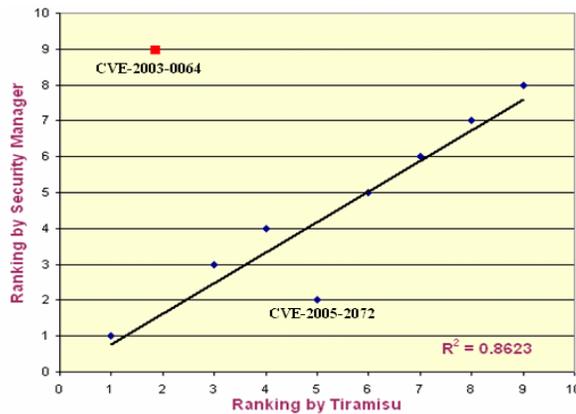


Figure 12 Human-Tiramisu Vulnerability Rankings Comparison

## 8. Conclusions, Limitations and Future Work

T-MAP is a novel stakeholder value centric security threat modeling approach devised in light of a large body of previous works across Value Based Software Engineering, Security Economics, COTS Vulnerability Studies, and Attack Graph. It defines a formal framework that evaluates security scenarios under stakeholder/value context, and enables distilling the technical details of thousands of published software vulnerabilities into executive-friendly language at a high-level. Compared to current value-neutral approaches, T-MAP systematically establishes the traceability and consistency from management-level organizational value propositions to technical-level security threats and corresponding mitigation strategies. Moreover, in our case study, T-MAP well captured the stakeholder value priorities through AHP pair-wise comparisons/analyses and injected the value priorities in attack scenario evaluation. Compared to traditional security risk management approaches, T-MAP does not require accurate values of probabilities, frequencies, and size of loss which are

usually very difficult to estimate in practice. An  $O(n)$  algorithm was devised to calculate the associated threat weight (*ThreatKey*) in *Structured Attack Graph*, which considerably increased the T-MAP scalability to large volume of hosts, software, and vulnerabilities. This scalability is achieved at the tradeoff on limiting the analyses scope to single-step-exploits. The security threat evaluation process has been automated by the *Tiramisu* tool, which greatly reduces the necessary human effort involved. So far, we are unaware of any other academic attempt that takes a stakeholder value centric approach to model security threats, automate the process, and perform similar security economic analyses as T-MAP.

This paper also reports a case study in which we used T-MAP to help our client prioritize possible security practices under a tight budget. In the case study, T-MAP demonstrated significant strength in prioritizing security practice and estimating the effectiveness of security practices. 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. The case study output indicates about 30% of the vulnerabilities caused 80% of the risks, which moderately matches the well known empirical “20/80 rule” in security risk assessment. The top vulnerabilities identified by the tool basically met the security manager’s experience except for a few outliers. 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 these value propositions in the automated security threat evaluation. However, since the tool as shown in Figure 12 generates some vulnerability overestimates and underestimates, it is still important for managers to balance T-MAP recommendations with those based on expert judgment.

These conclusions are specific to the case study we have conducted based on a single-host database system. However, though, the method demonstrated some perceivable potential even for big organizations like USC 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.”

### **Limitations and Mitigations**

At current stage, T-MAP has several none-trivial limitations. First, constrained by the fact that the existing vulnerability databases are still not sufficient to support multi-step-exploit attack graph modeling, our discussion on T-MAP is limited on single-step-exploit only to keep the method practical. But, as a bottom-line, a successful multi-step-exploit can be treated as a chain of several successful single-step-exploits. Thus, the failure of any of the single-step-exploits can make the multi-step-exploit unsuccessful. In this sense, considerably, T-MAP still can capture a large portion of the overall threat even after making the bold “one-step-exploit” simplifications. Moreover, perceivably, T-MAP has some potential to be extended to integrate more complex multi-step-exploit attack graph by chaining the steps between the layers of “IT hosts”, “COTS Software”, and “Vulnerability” in the Structured Attack Graph, if the connect-ability of precondition and post conditions between exploits was known.

Second, as a practical approach, the T-MAP method quantifies security threats all based on reported vulnerabilities of software, thus is not sensitive to unpublished vulnerabilities. As known, 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” [3]. In this sense, the T-MAP method can still capture the major part of the security threats for COTS software systems.

Third, many organizations have their own in-house developed software. For the security vulnerabilities reside in in-house software, if the vulnerability is known, it can be handled by adding its information to the T-MAP vulnerability database as a COTS vulnerability entry. If the vulnerability is unknown, it falls into the limitations of “unknown vulnerabilities” as discussed above.

Finally, 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.

### **Future Work**

Currently, we are in the process to further validate T-MAP with more empirical case studies. As future works, we plan to re-consider T-MAP from a system engineering and economics perspective, and extend the T-MAP to handle multi-step-exploits. Meanwhile, we will continue growing and refining the vulnerability database that T-MAP relies upon.

## Acknowledgments

This research has been sponsored by the NSF Grant CCF-0438931 and the Affiliates of the USC Center for Systems and Software Engineering. We are grateful to Dr. Ming-deh Huang, Dr. Neno Medvidovic, Dr. Clifford Neuman, and Dr. Bert Steece from USC for their very helpful discussions and insights on our research. We express special thanks to Mr. Tyler Moore from Cambridge University and the anonymous WEIS reviewers for their valuable help and comments in revising this paper. We also highly appreciate the helpful insights and discussions with Dr. Rick Kazman and Dr. Dan Port from University of Hawaii, Mr. Robert C. Seacord from CMU, and Ms. Elizabeth Fong from NIST.

## References

- [1] AS/NZS 4360:1999 Australian Standard: Risk Management. Standards Australia (1999)
- [2] E.J. Amoroso, "Fundamentals of Computer Security", Prentice Hall, 1994
- [3] A. Arora, R. Krishnan, A. Nandkumar, R. Telang and Y. Yang, Impact of Vulnerability Disclosure and Patch Availability - An Empirical Analysis, WEIS, 2004
- [4] T.W. August, T. Tunca, "Network Software Security and User Incentives", Management Science, Vol. 52, No. 11, November 2006, pp. 1703-1720
- [5] V. Basili and B. Boehm, "COTS Based System Top 10 List", Computer, Vol.34, no. 5, 2001, pp.91-93
- [6] R. Baldwin, "Rule based analysis of computer security", Technical Report TR-401, MIT LCS Lab, 1988 (The Kuang Tool)
- [7] H. Berghel, D. village: The two sides of ROI: return on investment vs. risk of incarceration, Communications of the ACM, April 2005
- [8] B. Boehm, Software Engineering Economics, Prentice Hall PTR, ISBN 0-13-822122-7, Pp223-242, 1981
- [9] 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.
- [10] 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
- [11] B. Boehm, L. Huang, A. Jain, R. Madachy, The Nature of Information System Dependability – A Stakeholder/Value Approach, Technical Report, 2004
- [12] B. Boehm, K. Sullivan, Software Economics: A Roadmap, The Future of Software Engineering, ACM 2000, pp 319-343
- [13] B. Boehm, R. Ross, Theory-W Software Project Management: Principles and Examples. IEEE Transactions Software Engineering, 1989, pp 902-916.
- [14] S. Boyer, O. Dain, R. Cunningham, "Stellar: A Fusion System for Scenario Construction and Security Risk Assessment", Information Assurance, 2005, Proceedings, Third IEEE International Workshop on, March 2005
- [15] L. D. Bodin, L. A. Gordon, M. P. Loeb, Evaluating Information Security Investment Using the Analytic Hierarchy Process, Communications of The ACM, February 2005
- [16] S. A. Butler, Software evaluation: Security attribute evaluation method: a cost-benefit approach, Proceedings of the 24th International Conference on Software Engineering, May 2002
- [17] H. Cavusoglu, B. Mishra, S. Raghunathan, A model for evaluating IT security investments, Communications of the ACM, July 2004
- [18] Huseyin Cavusoglu, Hasan Cavusoglu, Jun Zhang, "Economics of Security Patch Management", The Fifth Workshop on the Economics of Information Security (WEIS 2006), University of Cambridge, England, June 2006
- [19] Y. Chen, B. Boehm, L. Sheppard, "Value Driven Security Threat Modeling Based on Attack Path Analysis", 40<sup>th</sup> Hawaii International Conference on System Sciences, Big Island, Hawaii, January 2007
- [20] Y. Chen, B. Boehm, L. Sheppard, "Measuring Security Investment Benefit for COTS Based Systems", CSSE Technical Reports #2006-609, Sept. 2006
- [21] Y. Chen, B. Boehm, L. Sheppard, "Stakeholder Value Driven Threat Modeling for Off The Shelf Based Systems", 29<sup>th</sup> International Conference on Software Engineering (ICSE), Doctoral Symposium, Minneapolis, May 2007
- [22] Centers for Medicare and Medicaid Services, CMS Information Security Risk Assessment Methodology, v1.1, 2002
- [23] <http://www.cert.org/octave>
- [24] Common Vulnerability and Exposures, MITRE Corporation, <http://cve.mitre.org/>
- [25] G.C. Dalton II et al, Analyzing Attack Trees using Generalized Stochastic Petri Nets, Proceedings. of IEEE Workshop on Information Assurance, NY, 2006
- [26] FiSIRT Security Advisories, <http://www.frsirt.org/english>
- [27] D. Farmer and E. H. Spafford. The cops security checker system. Technical Report CSDTR-993, Purdue University, September 1991
- [28] Gordon, L., and Loeb, M., The economics of information security investment. *ACM Trans. Inf. Syst. Sec.* 5, 4 (2002), 438–457
- [29] L. A. Gordon, M. P. Loeb, Budgeting process for information security expenditures, Communications of The ACM, January 2006
- [30] Hoo, K.J.S. How much is enough? A risk management approach to computer security. Ph.D. Dissertation, Stanford University, 2000
- [31] Michael Howard, David LeBlanc, Writing Secure Code, Microsoft Press, 2002, ISBN 0-7356-1722-8, Chapter 4, pp 69-124
- [32] M. Howard, J. Pincus, J.M. Wing, "Measuring Relative Attack Surfaces", Proceedings of Workshop on Advanced Developments in Software and Systems Security, Taipei, December 2003
- [33] ISO 15288, Systems Engineering–System Life Cycle Processes, 2002
- [34] <http://xforce.iss.net/>
- [35] <http://www.ibiblio.org/pub/packages/security/Satan-for-Linux/>
- [36] ISO IS 15408, The Common Criteria for Information Technology Security Evaluation (CC) version 2.1, 1999

- [37] R. Lippmann and K. Ingols. An annotated review of past papers on attack graphs. Technical report, MIT Lincoln Laboratory, March 2005
- [38] P.K. Manadhata, J.M. Wing, "An Attack Surface Metric", USENIX Security Workshop on Security Metrics (MetriCon), Vancouver, BC, August 2006
- [39] R. A. Martin, Managing Vulnerabilities in Your Commercial-Off-The-Shelf (COTS) Systems Using An Industry Standards Effort, IEEE, 2002
- [40] Microsoft Security Bulletin, Microsoft Corporation, <http://www.microsoft.com/technet/security/bulletin/>
- [41] Microsoft Security Alert Severity Matrix, MS PSS Team <http://www.microsoft.com/technet/security/alerts/matrix.aspx>
- [42] M. Schiffman, Common Vulnerability Scoring System (CVSS), <http://www.first.org/cvss/>
- [43] National Vulnerability Database, NIST, <http://nvd.nist.gov/>
- [44] <http://www.nessus.org/>
- [45] S. Noel, S. Jajodia, B. O'Berry, and M. Jacobs., Efficient minimum-cost network hardening via exploit dependency graphs. In 19<sup>th</sup> Annual Computer Security Applications Conference (ACSAC), December 2003
- [46] Open Vulnerability Assess Language, MITRE Corporation, <http://oval.mitre.org/oval/>
- [47] D. Port and Z.H. Chen, "Assessing COTS Assessment: How much is Enough?" Proc. 3<sup>rd</sup> Int'l Conf. COTS Based Software Systems (ICCBSS 04, LNCS 2959, Springer-Verlag, 2004, pp 183-198
- [48] Xinming Ou, Wayne F. Boyer, and Miles A. McQueen, A scalable approach to attack graph generation, In 13th ACM Conference on Computer and Communications Security (CCS 2006), Alexandria, VA, U.S.A., October 2006
- [49] D. Reifer et al., "COTS Based Systems: Twelve Lessons Learned", Proc. 4<sup>rd</sup> Int'l Conf. COTS-Based Software Systems (ICCBSS 04), LNCS 2959, Springer-Verlag, 2004, pp.137-145
- [50] D. Reifer, Making the Software Business Case, Addison Wesley, 2002
- [51] N. H. Roberts, WE Vesely, D. F. Haasl, and F. F. Goldberg. Fault Tree Handbook. Systems Reliability Research Office of U.S. Nuclear Regulatory Commission, Washington DC, 20555, January 1981
- [52] G. Stoneburner, A. Goguen, A. Feringa, Risk Management Guide for IT Systems, NIST Special Publication 800-30, 2002
- [53] SANS Top 20 Most Critical Vulnerabilities, SANS, <http://www.sans.org/top20>
- [54] SANS Critical Vulnerability Analysis Priority Ratings
- [55] <http://www.securityfocus.com/>
- [56] T.L. Saaty, The Analytic Hierarchy Process. McGraw-Hill, NY, 1980
- [57] O. Sheyner, "Scenario Graphs and Attack Graphs", Carnegie Mellon University, April 2004. PhD Thesis
- [58] O. Sheyner, J. Haines, S. Jha, R. Lippmann, and J.M. Wing, "Automated Generation and Analysis of Attack Graphs", Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2002
- [59] B. Schneier. Attack trees: Modeling security threats. Dr. Dobb's Journal, December 1999
- [60] Man Charged with Hacking USC Database, [http://www.sci-tech-today.com/story.xhtml?story\\_id=11200CI6KD4W](http://www.sci-tech-today.com/story.xhtml?story_id=11200CI6KD4W)
- [61] <http://www.skybox.com/>
- [62] Symantec Threat Severity Assessment, <http://www.symantec.com/avcenter/threat.severity.html>
- [63] S. Templeton, K. Levitt, "A requires/provides model for computer attacks", Proceedings of the 2000 New Security Paradigmes Workshop (NSPW'00), Cork, Ireland, 2000
- [64] Torr, P., Demystifying the threat modeling process Security & Privacy Magazine, IEEE Volume 3, Issue 5, Sept.-Oct. 2005 Page(s):66 – 70
- [65] US-CERT Vulnerability Database, US Computer Emergency Readiness Team, <http://www.kb.cert.org/vuls/>
- [66] US-CERT Vulnerability Metrics, US Computer Emergency Readiness Team, <http://www.kb.cert.org/vuls/html/fieldhelp#metric>
- [67] <http://www.uml.org/>
- [68] US. General Accounting Office, Information Security Risk Assessment: Practices of Leading Organizations, 1999
- [69] D. Verdon, G McGraw, Risk analysis in software design, Security & Privacy Magazine, IEEE Volume 2, Issue 4, Jul-Aug 2004 Page(s):79 – 84
- [70] J.M. Wing, "Scenario Graphs Applied to Security", Proceedings of Workshop on Verification of Infinite State Systems with Applications to Security, Timisoara, Romania, March 2005. Summary Paper
- [71] Ross Anderson, "Why information security is hard – an economic perspective", In 17<sup>th</sup> Annual Computer Security Applications Conference (2001), pp. 358–365
- [72] Ross Anderson and Tyler Moore, "The Economics of Information Security", Science 314 (5799), pp.610-613, October 27, 2006
- [73] Schechter, S.E., "Toward econometric models of the security risk from remote attacks, Security & Privacy Magazine", IEEE, Volume 3, Issue 1, Jan-Feb 2005 Page(s):40 – 44