

# Value Driven Security Threat Modeling Based on Attacking Path Analysis \*

Yue Chen, Barry W. Boehm

Luke Sheppard

Center for Systems and Software Engineering Information Service Division

University of Southern California

## Abstract

*This paper presents a quantitative threat modeling method, 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) 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. T-MAP can help system designers evaluate the security performance of COTS systems and analyze the effectiveness of security practices. In the case study, we demonstrate the steps of using T-MAP to analyze the cost-effectiveness of how system patching and upgrades can improve security. In addition, we introduce a software tool that automates the T-MAP.*

## 1. Introduction

The recent legislation in computer security namely the Sarbanes-Oxley Act, HIPAA, and Gramm-Leach-Bliley have made security no longer optional – executives now have to take legal responsibilities for security breaches [4]. Competing with limited IT budgets and fast changing internet threats, effective security threat modeling has been increasingly catching security managers' attention. A recent survey on 212 S&P 500 firms conducted by Gordon et al suggests that cost-effectiveness analysis is a promising basis for budgeting security investments [16].

Unfortunately, quantifying the effectiveness of security investments has been very difficult [6, 16, 17]. It requires effective knowledge integration of both the economic and technology sides of security as well as fine grained considerations of the organizational environment such as business value priorities and types of IT infrastructures.

## Related Works

An important branch of previous work has been focusing on the economic analysis of security.

Specifically, the classic risk-management methodologies have established solid conceptual frameworks and rich guidelines to help security managers to identify and cope with security threats [1, 3, 11, 15, 17, 18, 23]. Realizing that security is not a binary concept of either secure or not, pioneer researchers took a “security-economics” approach to answer the question “how much security is enough” [17]; Gordon and Loeb proposed a quantitative model for assessing the optimal amount to invest in information security from economic perspectives [6]; Butler demonstrated using the *multiple-attribute risk assessment* in SAEM to reason the cost-benefit of security investments [6]; Cavusoglu et al proposed a quantitative model based on game theory for security investments evaluation, which can reflect the features of security systems such as the false positives and false negatives ratios of Intrusion Detection Systems [7]. Observing that, by nature, security investment is a multi-criteria decision making problem involving both quantitative and qualitative criteria, Bodin, Gordon, and Loeb demonstrated using the ratings method variant of the Analytic Hierarchy Process to evaluate IT security investment in their recent study [5].

From the technical side, authority organizations such as CERT, Cisco, FrSIRT, OVAL, NIST, Microsoft, SANS, and XForce publish near real-time software vulnerabilities and fixes. Comprehensive software vulnerability databases have been made publicly available [9, 14, 20, 22, 24, 25, 28]; The latest data shows that the NIST National Vulnerability Database (NVD) has been reporting 20 software system vulnerabilities per day on average, and there are 17731 vulnerabilities reported in total by the time this paper is written [22]. The Common Vulnerability Exposures (CVE) naming standard has effectively uniformed the vulnerability names across vulnerability reporting sources [19]. Furthermore, several vulnerability severity metrics and ratings systems have been developed to prioritize vulnerabilities [10, 13, 21, 26, 30].

Unfortunately, most current approaches in security

economics still stay at a high-level and lack strong connections to the large volumes of fast-changing internet vulnerabilities and specific organization's IT environment. On the other hand, most current vulnerability severity studies have been focusing on the technical aspects of vulnerabilities and have been surprisingly business-value-neutral. Based on current technologies, it is still very difficult to measure the security of IT systems along with the cost-effectiveness of security practices such as patching and firewalls, as mentioned by our clients in an interview.

### Proposed Threat Modeling Method

This paper presents the novel Security Threat-Modeling method based on Attacking Path analysis (T-MAP), a quantitative threat modeling method, which quantifies security threats by calculating the total severity weights of attacking paths that are relevant to IT systems. The severity weights of attacking 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 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, most IT systems are integrated from Commercial-Off-The-Shelf (COTS) software. T-MAP provides a strong quantitative method to evaluate the security performance of COTS candidates for IT platforms as well.

### Paper Roadmap

Section 2 makes assumptions that the T-MAP framework is based upon; section 3 presents the details of the T-MAP framework; section 4 briefly introduces the Tiramisu software tool which automates the T-MAP framework; section 5 demonstrates using T-MAP to perform security cost-effectiveness analysis through a case study, and section 6 summarizes the conclusions and identifies future works.

## 2. Assumptions

- The vulnerability database is accurate, comprehensive and up-to-date
- We limit the paper scope within the security threats for COTS software systems.

- The threats from passive attacks such as Phishing are not covered in the paper scope, but it could be an interesting future work

## 3. T-MAP Framework

The T-MAP framework is based upon the observation that the more security holes left open for an IT system, the less secure it is. Furthermore, different IT servers might have different levels of importance in terms of supporting the business's core values. 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 based on the total weight of all possible Attacking Paths.

### 3.1 Attacking Path Definition

An *Attacking Path* specifies an attacking scenario that results in compromising organization values in terms of:

- How the attacker gain access to the victim computer;
- Which vulnerability on which software on which computer can be taken advantage by attackers;
- How technical damages (i.e. confidentiality, availability, and integrity) can impact organization values (i.e. reputation, productivity, privacy).

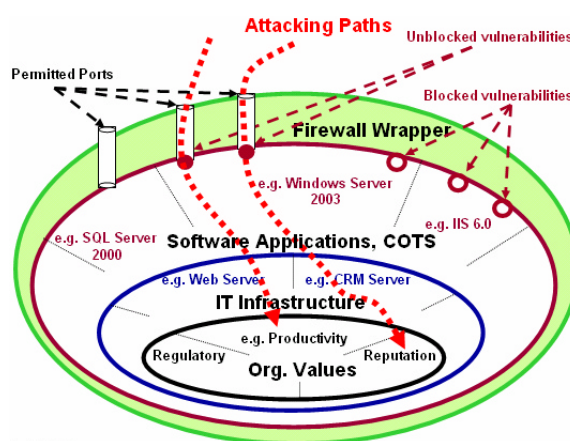


Figure 1 Attacking Path Concept Illustration

Figure 1 as shown above illustrates the attacking path concept: 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. Obviously, the organization’s value realization is dependent on its 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. In general, the vulnerabilities of the COTS software systems are important sources that cause security threats to the organization’s values. The outer-most layer with a caption of “Firewall Wrapper” presents the Firewall or other types of security wrappers protecting the IT infrastructure. The thick dots on the edge of the outside layer represent the software vulnerabilities that make attacks possible.

### Attacking Path UML Model

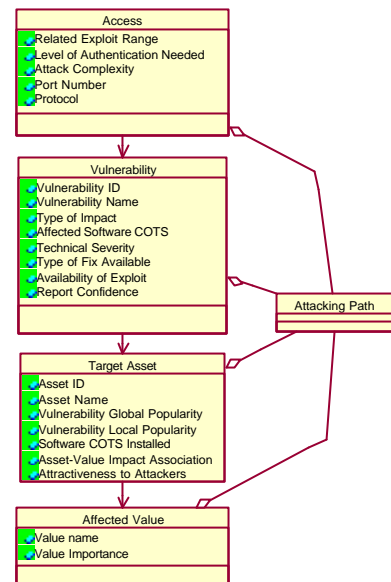
We use the UML class diagram in Figure 2 on the right to model the steps in an Attacking Path. Corresponding to each step, there are four classes involved:

- *Access*, specifying how an attacker enumerate the victim system, for example, by port scanning or sniffing the network communications;
- *Vulnerability*, specifying which vulnerability the attacker makes use of as well as the technical impact in terms of confidentiality, availability, and integrity;
- *Target Asset*, specifying which COTS software on which computer causes the vulnerability;
- *Affected Value*, specifying which organization value is impacted (i.e. reputation, productivity, privacy).

The selection of the class attributes are based on, but not limited to the emerging national standard Common Vulnerability Scoring System (CVSS), a joint effort across CERT/CC, Cisco, DHS/MITRE, eBay, Internet Security Systems, Microsoft, Qualys, and Symantec. In Figure 2, the descriptions of each T-MAP attributes are:

- *Related Exploit Range (AccessVector)*, characterizing if the vulnerability associated with the attacking path can be exploited remotely or locally [13];
- *Level of Authentication Needed (Authentication)*” which characterizes if the associated vulnerability requires valid user account on the victim computer [13];

- *Attack Complexity (Access Complexity)*, which characterizes if the associated vulnerability requires victim activities to enable such an attack, for example, opening an email attachment [13];



**Figure 2** Attacking Path Class Diagram

- *Port Number*, which characterizes the TCP/IP port(s) that the vulnerability uses, if there is any;
- *Protocol*, which characterizes the relevant network protocol that the vulnerability uses ;
- *Vulnerability ID*, which is the CVE name of the vulnerability [19];
- *Type of Impact*, which specifies what types of impacts the vulnerability could cause in terms of integrity, confidentiality, and/or availability;
- *Affected COTS Software*, which specifies the software that causes the vulnerability;
- *Technical Severity*, which characterizes the technical severity of such an attack in terms of None, Partially, or Completely to confidentiality, integrity, or availability [13];
- *Type of Fix Available (RemediationLevel)* which characterizes if the fixes of the associated vulnerability are publicly available [13];
- *Availability of Exploit (Exploitability)* which characterizes if there are exploit code or tools publicly available [13];
- *Level of Verification that Vulnerability Exists (Report Confidence)* which characterizes the

credibility of the report of the vulnerability [13];

- *Asset Name*, the name of the IT computer asset;
- *Vulnerability Global Popularity*, which characterizes that how popular the vulnerability is in a general sense. This factor is rated by if the vulnerability has been listed as top threats by authority organizations such as SANS and CERTS;
- *Vulnerability Local Popularity*, the experience at USC ISD shows that some of the vulnerabilities have been significantly more frequently exploited than others even these vulnerabilities are not listed as top by authority organizations. T-MAP uses this driver to help security managers differentiate the “locally hot” vulnerabilities from others;
- *COTS Software Installed*, which specifies what are the affected COTS software installed on the associated IT system;
- *Asset-Value Impact Association*, which specifies if security breach happens to an IT computer asset, then what business value(s) could be affected and how much it will be affected;
- *Attractiveness of Asset Computer*, which characterizes the attractiveness of the involved IT server to attackers;
- *Value Name*, which specifies the organization value that is affected by the attacking path;
- *Value Importance*, which specifies how important the value is to the organization, because organizations might have value priorities.

### Attribute Ratings

The ratings of all attributes are between 0 and 1. The attributes rating criteria and values are listed in Table 1:

**Table 1** T-MAP Severity Attribute Ratings

Attribute	Rating	Rating Value
Related Exploit Range (Access Vector) *	Remote	1.0
	Local	0.7
Availability of Exploit (Exploitability) *	Unproven	0.85
	proof-of-concept	0.9
	Functional	0.95
	High (Functional and delivered by virus or worms)	1.0

**Table 1** T-MAP Severity Attribute Ratings (Continued)

Attribute	Rating	Rating Value
Type of Fix Available (Remediation Level) *	Official Fix	0.87
	Temporal Fix	0.9
	Work around	0.95
	None	1.0
Report Confidence *	Unconfirmed	0.9
	Uncorroborated	0.95
	Confirmed	1.0
Attack Complexity (Access Complexity) *	Required	0.8
	Not Required	1.0
Level of Authentication Needed (Authentication) *	Required	0.6
	Not Required	1.0
Vulnerability General Popularity	Listed as Top	1.0
	Not Listed as Top	0.8
Vulnerability Local Popularity	Frequently exploited	1.0
	Got exploited, but not frequently	0.9
	Never got exploited before	0.7
Attractiveness of Asset Computer	Attractive	1.0
	Neutral	0.8
	Not Attractive	0.7
Technical Severity *	Complete	1.0
	Partial	0.7
	None	0
Asset-Value Impacts	Complete	1.0
	High	0.9
	Medium	0.6
	Low	0.3
	None	0
Value Importance	High	1.0
	Medium	0.6
	Low	0.3
	None	0

[Note] Fields marked with \* are referenced from CVSS [13]

As an important note, the values of the ratings in Table 1 are designed based on the authors’ experiences. Other security managers can further customize the rating values to better fit their specific organizational environment.

### 3.2 T-MAP Weighting System

Not all attacking paths are equal. Some attacking paths are easier for attackers to exploit than others. Some have more severe business value impacts than others. And some are more attractive to attackers than others.

T-MAP differentiates Attacking Paths by calculating their severity weights from attacking path attribute ratings. The weight calculation is inspired by the classic risk calculation formula:

$$Risk = Probability * Size\ of\ Loss.$$

We first categorize the attacking path attributes based on their relevance to the *Probability* of such an attack to be successful, the *Size of Loss* and the *Descriptive*. The classification of Attacking Path attributes are summarized in Table 2 as follows:

**Table 2** Attacking Path Threat Severity Relevant Attributes

RELEVANCE	CLASS	ATTRIBUTES
Probability Relevant Attributes	Access	<i>Related Exploited Range</i>
		<i>Level of Authentication Needed</i>
		<i>Attack Complexity</i>
	Vulnerability	<i>Vulnerability Global Popularity</i>
		<i>Vulnerability Local Popularity</i>
		<i>Availability of Exploit</i>
		<i>Type of Fix Available</i>
Target Asset	<i>Report Confidence</i>	
Size-of-Loss Relevant Attributes	Vulnerability	<i>Technical Severity</i>
	Target Asset	<i>Asset-Value Impact Association</i>
	Affected Value	<i>Value Importance</i>
Descriptive	Access	<i>Protocol / Port Number</i>
	Vulnerability	<i>Vulnerability ID</i>
		<i>Vulnerability Name</i>
		<i>Type of Impact</i>
	Target Asset	<i>Affected COTS Software</i>
		<i>Asset ID</i>
		<i>Asset Name</i>
	Value	<i>Software COTS Installed</i>
	Value	<i>Value Name</i>

As shown in Table 2, the *Probability* relevant attributes affect how likely such an attack is to happen. It largely depends on how easy it is for attackers to launch such an attack and how attractive the victim system is to attackers.

The *Size-of-Loss* relevant attributes characterize how much the loss will be if such an attack happens. It largely depends on the type of the security breaches and how the attacking scenario will impact organizational key values.

Then we calculate the threat severity weight of each attacking path by multiplying the ratings of all the *Probability* relevant and the *Size-of-Loss* relevant attributes. We define:

$$Weight_{AttackingPath} = \prod_{i=1}^{12} Rating(SeverityAttributes_i)$$

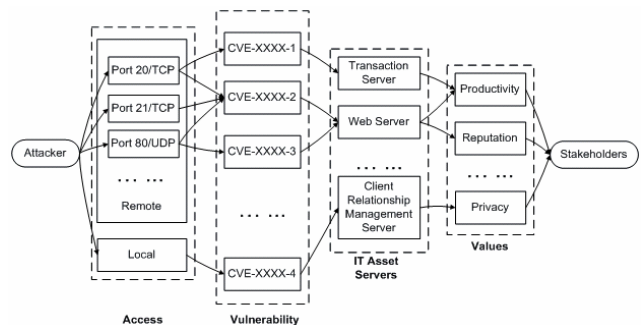
Clearly, the *weight* is a rating number between 0 and 1 because the rating values of all attributes are between 0 and 1. By the nature of the definition, it is a quantified risk heuristic of the attacking path.

### 3.3 Threat Modeling by Attacking Path Analysis

Obviously, there might be many attacking paths affecting an organization's values. In each step on the attacking path, the attacker might have multiple choices. For example, when exploiting system vulnerability, the attacker might be able to steal information, and/or change data, and/or shutdown the system, thus generating multiple possible attacking scenarios.

#### The Overall Threat

We use a graph as shown in Figure 3 to illustrate the possible attacking scenarios. Each attacking path starts from the "Attacker" node in the left and ends at the "Stakeholders" node in the right, representing a possible attacking scenario. Besides the "Attacker" and "Stakeholder" nodes, there are four groups of other nodes: the "Access", "Vulnerability", "IT Asset Computers", and "Values", corresponding to the four key steps in an attacking path as described in section 3.1. Using the path finding algorithm for the graph, we can calculate a complete set of the possible attacking paths from the attackers to the organization's stakeholders.



**Figure 3** Graph View of Attacking Paths

The pseudo-code of the Attacking Path calculation algorithm is described in Figure 4 as follows:

---

**Algorithm 1** *GenerateAttackingPath(VulnDB, AssetDB, ValueList, AttackingPathList)*

---

```

1:  for (each Value in ValueList) {
2:      search AssetDB for Host that are associated with this value;
3:      for(each Host in the search results in line 2) {
4:          search AssetDB for Software installed on the Host;
5:          for (each Software in the search results in line 4) {
6:              search VulnDB for Vulnerability whose Affected Software
7:                  equals Software;
8:              for(each Vulnerability in the search results in line 6) {
9:                  list out all possible Access method of the Vulnerability;
10:                 for (each Access obtained in line 9) {
11:                     AttackingPath ←
12:                         <Access|Vulnerability|Software|Host|Value>;
13:                     Calculate the SeverityWeight of AttackingPath;
14:                     Append AttackingPath to AttackingPathList;
15:                 }
16:             }
17:         }
18:     }
19: return AttackingPathList;

```

---

**Figure 4** Attacking Path Calculation

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

$$TotalThreat = \sum_i Weight(AttackingPath_i),$$

where  $i$  varies from 1 to the total number of attacking paths and  $AttackingPath_i$  represents the  $i$ th Attacking Path associated to the organization.

### Threat Key of Node

In Figure 3, each node in the graph is associated with a sub-set of attacking paths that pass through that node. By comparing the total weight of all attacking paths passing through the node, we can further determine which nodes are more vulnerable than others.

We define the *Threat Key* of a node as the total weight of all the attacking paths that pass through that node:

$$ThreatKey_{Node_j} = \sum_i Weight(AttackingPathGoThroughNode_j),$$

where  $i$  varies from 1 to the total number of attacking paths that go through node  $j$ .

For example, the *Threat Key* of Port 80 in Figure 3 is the total weight of all the four attacking paths that pass through Port 80:

AP1 = {Port 80, CVE-XXXX-2, Web Server, Productivity}

AP2 = {Port 80, CVE-XXXX-3, Web Server, Productivity}

AP3 = {Port 80, CVE-XXXX-2, Web Server, Reputation}

AP4 = {Port 80, CVE-XXXX-3, Web Server, Reputation}

Then,  $ThreatKey_{Port\ 80} = \sum_i Weight_{AttackingPathGoThroughPort\ 80}$

Therefore, the greater the value of the *Threat Key* of a node, the more vulnerable that node is. For example, in firewall configurations, opening those ports associated with higher *Threat Key* values is more dangerous than opening those ports which have lower values. Those vulnerabilities which have a higher *Threat Key* value should be patched at a higher priority than others. And those computers that have higher *Threat Key* values cause more threats to the organization values than others.

### 3.4 Measuring 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 attacking paths and nodes that this security practice can suppress. 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.

So, we define the effectiveness of certain security practice as:

$$Effectiveness(SecurityPractice)$$

$$= 1 - TotalThreat(AfterPractice)/TotalThreat(BeforePractice)$$

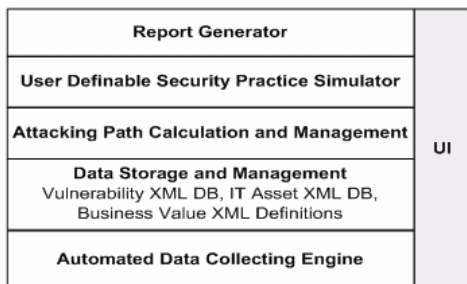
which represents the percentage of total threat weight that has been suppressed because of taking that security practice. Therefore, once the cost of a security practice is known, security managers can further calculate the cost-effectiveness of the security practice.

### 3.5 Comparison with CVSS

- 1) 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.
- 2) T-MAP is an organization value centric approach. It focuses on organization value impacts from system vulnerabilities. In comparison, CVSS is value-neutral and strongly focuses on the technical aspects of vulnerabilities

## 4. T-MAP Implementation - Tiramisu

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.



**Figure 5** Tiramisu Architecture

The *Tiramisu* adopts a 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.

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.

## 5. Case Study: Cost-Effectiveness of Patching

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 [29]. To enhance system security and minimize the chance of another compromise of sensitive data, the system administrator has been considering a software upgrade from MS SQL Server v2000 to v2005. We summarize the software changes in Table 3:

**Table 3** Software Comparison: Current and Upgraded

Current ( <i>All software patched to latest</i> )	Upgraded	Notes
Windows Server 2003	Windows Server 2003	No change
<b>MS SQL Server 2000</b>	<b>MS SQL Server 2005</b>	<b>Changed</b>
IIS 6.0, SP2	IIS 6.0, SP2	No change
Symantec Antivirus	Symantec Antivirus	No change
HP Openview	HP Openview	No change
<b>Python 2.4</b>		<b>No longer needed</b>

With limited budget and other competing tasks, the security administrator has been considering two alternative plans to improve security:

The *Alternative 1* is to upgrade the SQL Server to v2005 as well as perform automated patching, but no manual fixes for those vulnerabilities that cannot be automatically fixed. The *Alternative 2* is to upgrade the SQL Server, as well as apply both automated patches and all existing manual fixes.

One of the major considerations here is that not all software vulnerabilities are fixable through automatic vendor patching services. Some vulnerability can only be fixed manually such as changing software configurations or disabling services, thus much more labor-intensive. Some other vulnerability just cannot be fixed at all.

Now, we demonstrate using the T-MAP method to help the security administrator analyze the cost-effectiveness of each plan through the following steps:

**Step 1 - Determine the Security-Value impact of Server X.** As a development server, its major security concern focuses on the confidentiality and integrity, because successful hackers might be able to steal sensitive information if they compromise the server confidentiality. In addition, they might use this server as a base to launch further attacks against other computers. Table 4 summarizes the security-value impact ratings in different security breach scenarios:

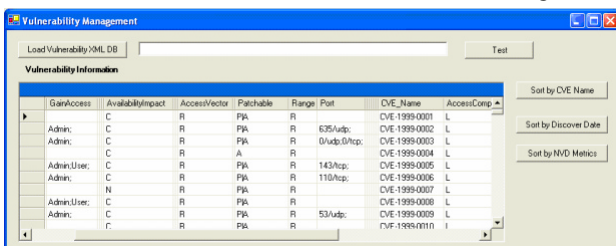
**Table 4** Case Study Security-Value Impact Ratings

		Key Organization Values		
		Reputation	Regulatory	Productivity
Security	Confidentiality	High	Complete	None
	Integrity	High	Complete	Low
	Availability	None	None	Medium

The ratings in Table 4 will be translated into numbers by looking up the item “Asset-Value Impact” in Table 1 for Attacking Path weight calculation.

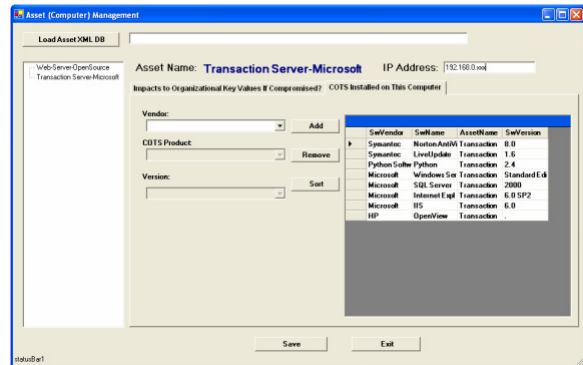
**Step 2 – Calculate the threat severity of current status.**

Step 2.1 Load all 17,731 vulnerabilities published by NIST from 1999 to 2006 into the *Tiramisu* tool (figure 6).



**Figure 6** Screenshot of Load Vulnerability Database

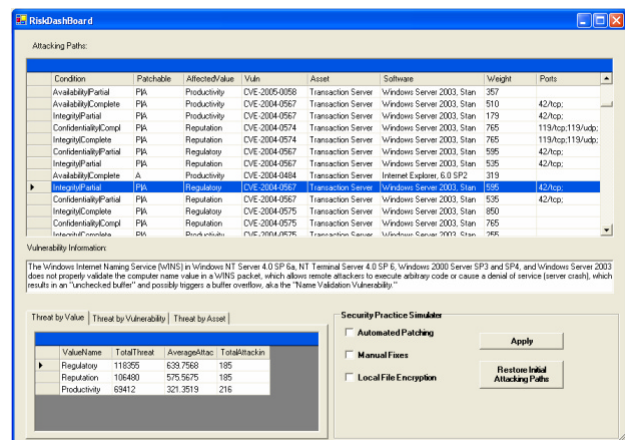
Step 2.2 Input the software installed on current Server X into the tool. (Figure 7)



**Figure 7** Screenshot of Input Installed Software

Step 2.3 Calculate the attacking paths and associated severity weights of the current system without any patching (Figure 8). The output shows that there are a total of 586 attacking paths with a total weight of 294.3 for the current system if no patches are applied. Because currently the automated patching has been running, we remove all the attacking paths whose associated vulnerabilities have automated vendor patches available from the current attacking path set.

The calculation results from the *Tiramisu* tool shows that there are 107 attacking paths left with a total severity weight of 62.53 for the current status. The top three vulnerabilities for this plan are CVE-2005-2388, CVE-2000-0119, and CVE-2004-0867 [22], associated with *ThreatKey* values of 3.29, 3.1 and 3.1, respectively. To date, according to securityfocus.com, CVE-2005-2388 does not have any fix yet. CVE-2000-0119 and CVE-2004-0867 have only manual fixes available.



**Figure 8** Attacking Path Calculation – Results

### Step 3 - Calculate the Effectiveness of Alternative 1

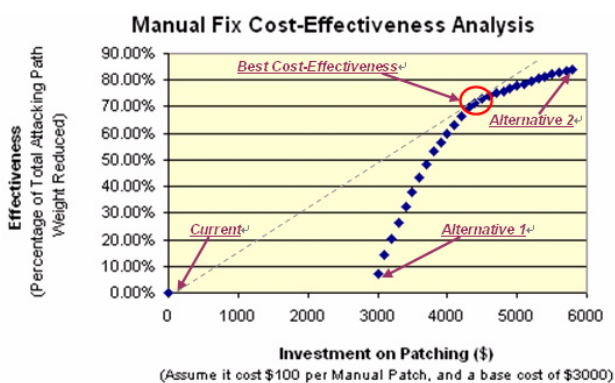
For *Alternative 1*, we first modify the software installation from SQL Server v2000 to v2005. We then repeat *Step 2.1* to *Step 2.3* to calculate the initial Attacking Path set of the upgraded system, and remove those attacking paths whose associated vulnerabilities have automated vendor patches available. Our results show that there are 101 attacking paths left open with a total severity weight of 58.1. The top three vulnerabilities for this plan are still CVE-2005-2388, CVE-2000-0119, and CVE-2004-0867 associated with *ThreatKey* values of 3.29, 3.1 and 3.1, respectively [22]. Comparing to the current status, the security effectiveness is:

$$\text{Effectiveness} = 1 - 58.1/62.53 = 7.1\%$$

Assuming the automated patching cost is \$1,000 for the system down time because of patching, the SQL Server upgrading cost is \$2,000, the cost-effectiveness is  $7.1\% / (\$1000 + \$2,000)$  equals 2.36% of improvement per thousand dollars.

### Step 4 - Calculate the Effectiveness of Alternative 2

Because the *Alternative 2* plans to apply all manual fixes, we further remove those attacking paths from the results in Step 3, if the vulnerability in the Attacking Path has manual work arounds or fixes available. The *Tiramisu* output suggests there are 28 manual fixes need to be applied to the upgraded system. Our clients estimate that it costs \$100 to apply each manual fix. Plus the automated patching cost of \$1,000 for the system down time and the SQL Server upgrade costs of \$2,000, the total cost will be \$5,800 for *Alternative 2*.



**Figure 9** Cost-Effectiveness of Manual Fixes

Furthermore, Figure 9 plots the cost-effectiveness of applying different number of manual fixes. The X axis

represents the *Cost*, which increases linearly along with the number of manual fixes. The Y axis represents the *Effectiveness*. Assuming the vulnerabilities that have high *ThreatKey* values are fixed first, the plot shows that the *Effectiveness* increases along with the number of manual fixes applied. The high-payoff point is around applying the top 15 manual fixes based on the automated patching.

The *Alternative 2* is located at the very right end of the curve. It has 17 attacking paths left open with a total severity weight of 10.1. The top three vulnerabilities are CVE-2005-2388, CVE-2005-2089, and CVE-2005-1184 associated with *ThreatKey* values of 3.29, 2.2 and 0.6, respectively [22]. Comparing to the current status, the *Effectiveness* is  $1 - 10.1/62.53$  equals 83.85%. The cost-effectiveness is  $83.85\% / (\$2,000 + \$1,000 + 28 * \$100)$  equals 14.46% of improvement per thousand dollars.

Clearly, the *Alternative 2* achieves the best security. However, to achieve the best cost-effectiveness, the administrator should apply the top 15 vulnerabilities that have the highest *ThreatKey* values. The follow up evaluation on the top vulnerabilities met common sense.

## 6. Conclusions and Future Works

This paper presents the novel T-MAP threat modeling method that measures software system security with total weights of Attacking Paths. It demonstrates significant strength in measuring software system security and the effectiveness of security practices. It helps:

- IT system architects evaluate the security performance of COTS systems
  - Executives perform cost-effective analysis on security practices and investments;
  - Security administrators identify key vulnerabilities based on an organization's value preferences;
- Compared to existing approaches, it is:
- Sensitive to an organization's value priorities;
  - Distills the technical details of thousands of published software vulnerabilities into executive-friendly numbers at a high-level
  - Sensitive to an organization's IT environment;
  - Automated by software tool, thus greatly reduces the necessary human effort involved

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

However, 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/upgrading”, “change configurations”, “block access”, and “no solutions found” with an 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 .

## Acknowledgments

We are grateful to Mr. Christopher Backstrand for his discussions on the case study presented in this paper. The research has been supported by NSF ITR Grant CCR0086078 and the USC-CSE Affiliates.

## 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, Value-based software engineering, ACM SIGSOFT Software Engineering Notes, v.28 n.2, March 2003
- [4] H. Berghel, D. village: The two sides of ROI: return on investment vs. risk of incarceration, Communications of the ACM, April 2005
- [5] L. D. Bodin, L. A. Gordon, M. P. Loeb, Evaluating Information Security Investment Using the Analytic Hierarchy Process, Communications of The ACM, February 2005
- [6] S. A. Butler, Technical papers: Software evaluation: Security attribute evaluation method: a cost-benefit approach, Proceedings of the 24th International Conference on Software Engineering, May 2002
- [7] H. Cavusoglu, B. Mishra, S. Raghunathan, A model for evaluating IT security investments, Communications of the ACM, July 2004
- [8] ISO IS 15408, The Common Criteria for Information Technology Security Evaluation (CC) version 2.1, 1999
- [9] US-CERT Vulnerability Database, US Computer Emergency Readiness Team, <http://www.kb.cert.org/vuls/> (current 06/2006)
- [10] CERT Vulnerability Metrics, CERT®, <http://www.kb.cert.org/vuls/html/fieldhelp#metric>
- [11] Centers for Medicare and Medicaid Services, CMS Information Security Risk Assessment Methodology, v1.1, 2002
- [12] A UML Specification Language targeting Security Risk Assessment, <http://coras.sourceforge.net/> (current 06/2006)
- [13] M. Schiffman, Common Vulnerability Scoring System (CVSS), <http://www.first.org/cvss/> (current 06/2006)
- [14] FiSIRT Security Advisories, <http://www.frsirt.org/english>
- [15] US. General Accounting Office, Information Security Risk Assessment: Practices of Leading Organizations, 1999
- [16] L. A. Gordon, M. P. Loeb, Budgeting process for information security expenditures, Communications of The ACM, January 2006
- [17] Hoo, K. S, “How Much Is Enough? A Risk Management Approach to Computer Security” , Workshop on Economics and Information Security, UC Berkeley, CA, 2000
- [18] ISO 15288, Systems Engineering – System Life Cycle Processes, 2002
- [19] Common Vulnerability and Exposures, MITRE Corporation, <http://cve.mitre.org/> (current 06/2006)
- [20] Microsoft Security Bulletin, Microsoft Corporation, <http://www.microsoft.com/technet/security/bulletin/>
- [21] Microsoft Security Alert Severity Matrix, MS PSS Team <http://www.microsoft.com/technet/security/alerts/matrix.msp>
- [22] National Vulnerability Database, NIST, <http://nvd.nist.gov/> [23] G. Stoneburner, A. Goguen, A. Feringa, Risk Management Guide for IT Systems, NIST Special Publication 800-30, 2002
- [24] Open Vulnerability Assess Language, MITRE Corporation, <http://oval.mitre.org/oval/>, (current 06/2006)
- [25] SANS Top 20 Most Critical Vulnerabilities, SANS,

<http://www.sans.org/top20> (current 06/2006)

[26] SANS Critical Vulnerability Analysis Priority Ratings

[28] <http://www.securityfocus.com/>

[29] 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)

[30] Symantec Threat Severity Assessment,

<http://www.symantec.com/avcenter/threat.severity.html>