A Strategic Approach for Risk Analysis of Production Software Systems

Sumithra A
Research Scholar
Madurai Kamaraj University
Madurai, India

Ramaraj E
Technology Advisor
Madurai Kamaraj University
Madurai, India

Sree Ram Kumar T
Research Scholar
Madurai Kamaraj University
Madurai, India

ABSTRACT
Defects in production software can incur heavy damage to a business operation; yet most current approaches to software security assessment focus primarily on new code development. The paper aims at introducing a strategic approach for reducing the operational security risk. The familiar top-down structured development process used by internal development groups is totally inappropriate for risk analysis of production software systems. And generally the cost of finding and fixing a bug in a production system is regarded as too high. So there is an imperative necessity to focus on approaches tailored specifically for production software systems which is the one attempted here.

Keywords
Risk, Production Software System, Security Risk, Vulnerability, Software Components

1. INTRODUCTION
Insecure software is a major factor in internal/external fraud**.** This seemingly obvious observation is graphically borne out in a study that analyzed a sample of 167 customer data breaches in 2005.[1] Based on data provided by the Privacy Rights Clearinghouse,[2] the study classified each event according to attack method, attacker and vulnerability exploited. A conservative estimate showed that 49% of the events exploited software defects as shown in the below table. Theoretically we can mitigate half of the risk by removing software defects in existing applications. The question, which we will answer later, is how.

Table 1. Various Vulnerabilities

Vulnerability type	Total	Percentage
Accidental disclosure by email	5	3.0%
Human weakness of system users/operators	13	7.8%
Unprotected computers / backup media	67	40.1%
Software defects maliciously exploited	82	49.1%
Grand Total	167	100.0%

The Carnegie Mellon Software Engineering Institute (SEI) reports that 90 percent of *all* software vulnerabilities are due to well-known defect types (for example using a hard coded server password or writing temporary work files with world read privileges). All of the SANS Top 20 Internet Security vulnerabilities are the result of "poor coding, testing and sloppy software engineering" [3].

1.1 Do organizations really want to improve production software quality?
Let's examine commitment to quality at three levels in an organization: end-users, development managers and top executives. Users are conditioned to accept unreliable software on their desktop and development managers are inclined to accept faulty software as a tradeoff to meeting a development schedule. Executives, while committed to quality of their own products and services, do not find security breaches sufficient reason to become security leaders with their enterprise systems because:

a.They usually receive conflicting proposals for new information security initiatives with weak or missing financial justifications.

b.The recommended security initiatives often disrupt the business. [4]

1.2 How relevant are firewalls, anti-virus and anti-spyware to reducing operational risk?
IT security products are used to defend the organization rather than as a means of improving understanding and reducing operational risk. Today's defense in depth strategy is to deploy multiple tools at the network perimeter such as firewalls, intrusion prevention and malicious content filtering. The defense-focus is primarily on **outside-in** attacks, despite the fact that the majority of attacks on customer data and intellectual property are **inside out**. The notion of trusted systems inside a hard perimeter has practically disappeared with the proliferation of Web services, SSL VPN and convergence of application transport to HTTP.

A reactive tool such as a firewall cannot protect exploitation of production software defects and black-box application security that relies on checklists of vulnerabilities is no replacement for in-depth

understanding of specific source code vulnerabilities. We must conclude that traditional IT security products can do little to mitigate the risk due to vulnerabilities in buggy software.

2. COST EFFECTIVE DEFECT REDUCTION FOR PRODUCTION SOFTWARE

It is rare to see systematic defect reduction projects in production software running in the enterprise, apparently, if it were easy, everyone would be doing it. So what makes it so hard?

1.The familiar top-down structured development processes (including Extreme Programming) used by internal development groups are totally inappropriate for risk analysis of production software systems.

2.The cost of finding and fixing a bug in a production system is regarded as too high.[5]

3.The application developers and IT security teams don't usually talk to each other. The larger the organization, the more they lose when information gets lost in the cracks.

We can meet these challenges in a cost-effective way by establishing three core principles:

1.Use a risk analysis process that is suitable for production software systems. Collect data from all levels in the organization that touch the production system and classify defects for risk mitigation according to standard vulnerability and problem types.

2.Provide executives with financial justification for defect reduction. Quantify the risk in terms of assets, software vulnerabilities, and the organization's current threats.

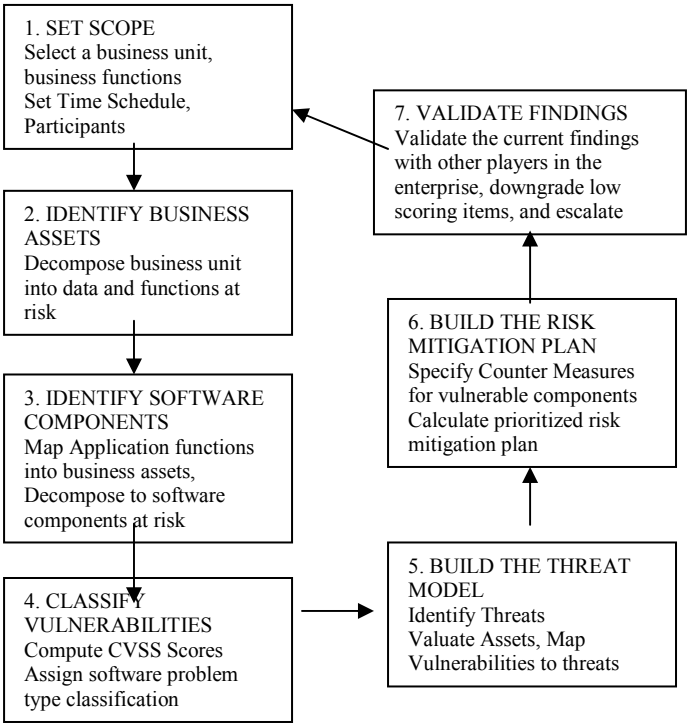
3.Require the development and IT security teams to start talking. Explicit communications between software developers and IT security can be facilitated by an online knowledge base and ticketing tool that provide an updated picture of well-known defects and security events.

This paper examines the first principle in more detail.

3. RISK ANALYSIS FOR PRODUCTION SOFTWARE

The process identifies, classifies and evaluates software vulnerabilities in order to recommend cost-effective countermeasures. The process is iterative and its steps can run independently, enabling any step to feed changes into previous steps even after partial results have been attained.

Figure 1



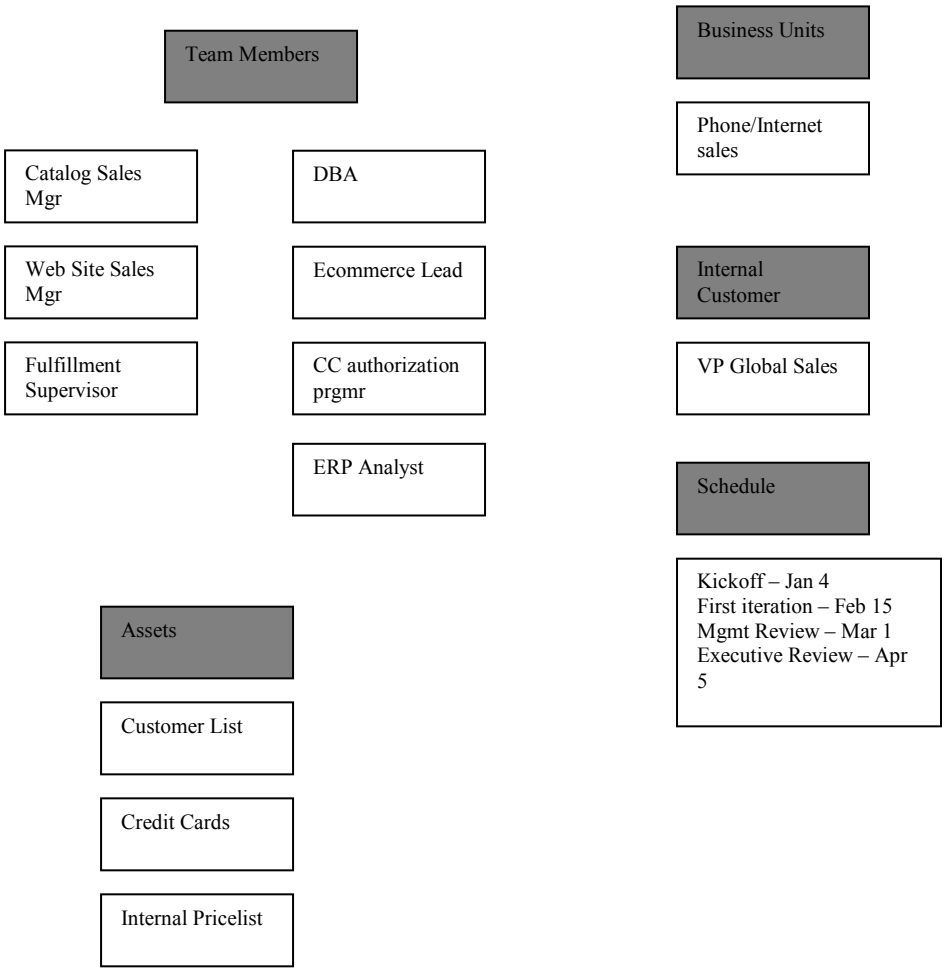
Continuous review of findings is key to success of the project. For example, an end-user may point-out fatal flows in an order entry form to the VP engineering during the Validate Findings step and influence the results in the Classify Vulnerabilities and Build the threat model steps.

3.1. Set scope

The first step is to determine scope of work in terms of business units and assets. Focus on a particular business unit and application functions will improve the ability to converge quickly. The process will also benefit from executive level sponsorship that will need to buy into implementation of the risk mitigation plan.

The team members are chosen at a preliminary planning meeting with the lead analyst and the project's sponsor. There will be 4-8 active participants with relevant knowledge of the business and the software. The team is guided by expert risk analysts that have good people skills and patience to work in a chaotic process.

Figure 2

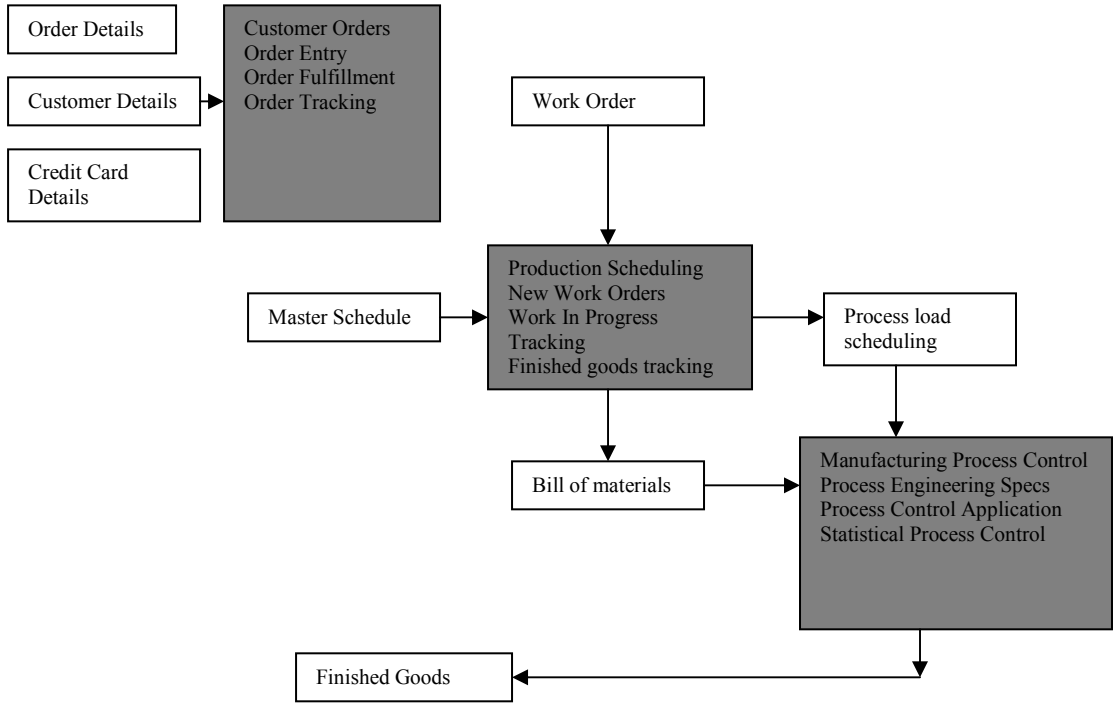


3.2 Identify business assets

In step 2, the team identifies operational business functions and their key assets:

This part of the process can be done using wall-charts as shown in the below figure. The graphic format helps the team visualize the scope of assets and estimate potential impact of threats on assets. Business functions (shaded boxes) are placed on a diagonal from top left to bottom right as shown in the below figure. Assets flow clockwise around the diagonal of business functions.

Figure 3



3.3 Identify software components

After identifying business functions in Identify business assets, the team now identifies software components (but doesn't assess vulnerabilities) using two sub-steps:

a.Identify application functions that serve the business function

b.Decompose application functions to software components

In order to help build a consistent, reasonably high-level view of the system, this part of the process can be done using wall-charts as shown in the below figure. Application functions (shaded boxes) are placed on a diagonal from top left to bottom right as shown in the below figure. Decomposed components flow clockwise around the diagonal of application functions.

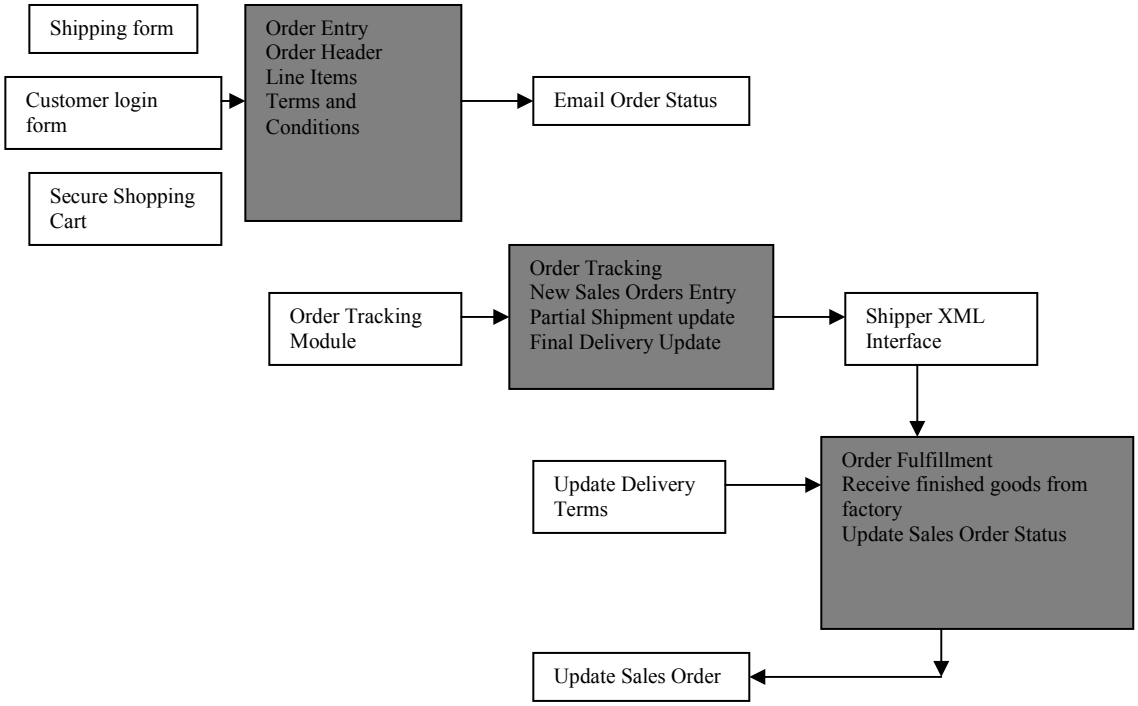


Figure 4

3.4. Classify the software vulnerabilities.

CVSS[6] scores are computed for each component identified in the Identify software components step. In addition to the CVSS score, we collect an additional field, the CLASP [7] problem type category, for example "**Use of hard-coded password"**.

The knowledge base supporting the process contains a baseline of classified software vulnerabilities and evolves over time as the team classifies new vulnerabilities. Various source code scanners may also be used in this step, for example – FindBugs to find problems in Java source code.

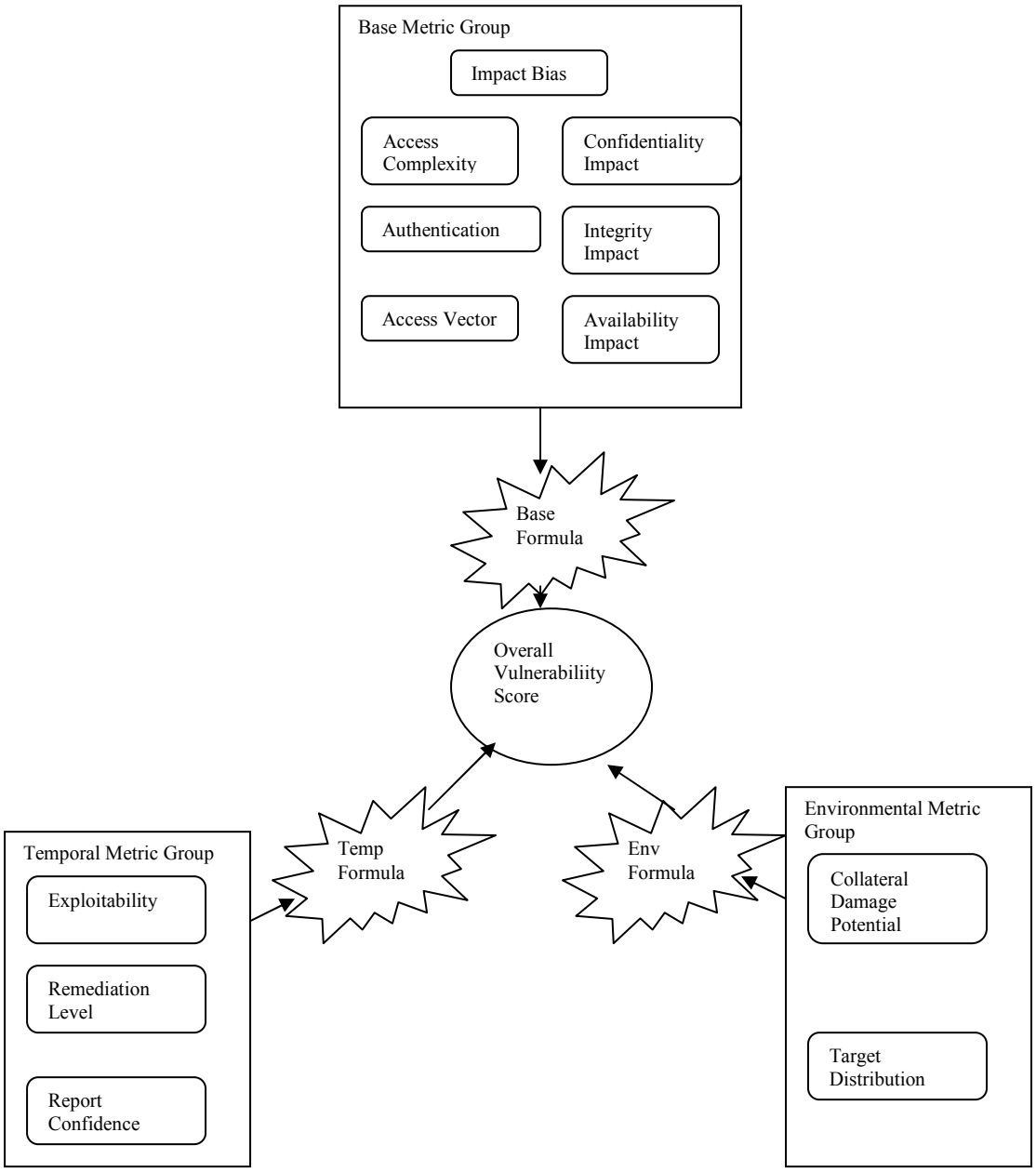


Figure 5

3.5. Build the threat model

The team now populates the PTA (Practical Threat Analysis) threat model.

Assets collected in the Identify business assets step are assigned a financial value.

Threats are named and classified as to their probability of occurrence and damage levels. Vulnerabilities that were collected in the Classify the vulnerabilities step are associated with threats

3.6 Build the risk-mitigation plan

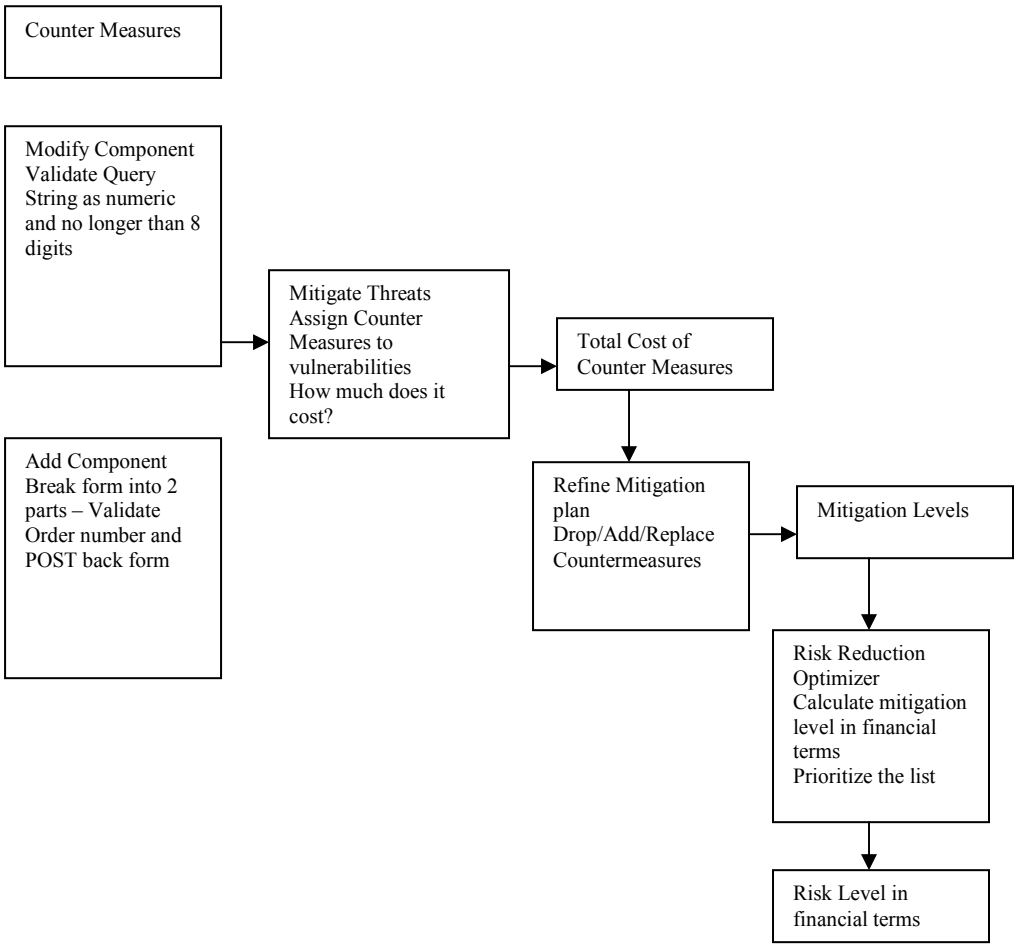
In step 6, the team specifies countermeasures for vulnerabilities found in the software components and records them in the PTA data model. While the best countermeasure for a problem is fixing it, in reality there may not be documentation and the programmers who wrote the code are probably in some other job. This means that other means may be required, such as code wrappers or application proxies. The possible types of countermeasures are Retain, Modify and Add as seen in the below figure:

●Retain the existing component (leave the defects in place) or,

●Modify the component (fix the defect or put in a workaround) or,
●Add components (for example call the Global LDAP directory to authenticate on-line users instead of using a proprietary customer table).

●Each countermeasure is assigned a cost and mitigation level. The cost may be a combination of fixed and variable cost in order to describe a one time cost of fixing a problem and ongoing maintenance cost.

Figure 6



3.7. Validate findings

This extremely important step validates the current findings with expert/relevant players in the enterprise. The objective is to use all means at the disposal of the team to qualify components and vulnerabilities as to **where** (they are in the system), **which** (assets are involved), **what** (they do now and in the past), **why** (they perform the way they do) and **when** (a component is initialized and activated). Conceptually, no limits are placed on what questions can be asked. Users may downgrade low-risk software components and escalate others for priority attention. They may add or remove assets from the model and argue parameters such as probability, asset value, estimated damage etc. For example, a server-side order confirmation script that sends email to the customer may have received a low CVSS score in Classify the software vulnerabilities. The team can simply decide to eliminate that vulnerability from the list during Validate findings.

4. Conclusion and Future Work

In this paper, we presented a strategic approach for risk management in production software systems. Attention was paid to the "risk analysis" phase of this process. More work has to be done with regard to the second and third phases – viz. providing financial justification to executives and requiring the IT and security team to talk to each other. Considering the tremendous impact such security risks can have on organizations, the effort required is justifiable.

5. References

[1] 2005 Breach Analysis, April 2006 http://www.software.co.il/downloads/breachAnalysis2005.xls

[2] Privacy Rights Clearinghouse, http://www.privacyrights.org/

[3] Developing Secure Software, Noopur Davis, http://www.softwaretechnews.com/stn8-2/noopur.html

[4] Top-down Security", Alan Paller, http://infosecuritymag.techtarget.com/articles/1999/paller.shtml

[5] In production, it's often 100 times more expensive than finding and fixing the bug during requirements and design phase". Barry Boehm, Victor R. Basili, IEE Computer, 34(1): 135-137, 2001

[6] CVSS (Common Vulnerability Scoring System) is a standard way to convey vulnerability severity and help determine urgency and priority of response, http://www.first.org/cvss/intro/ Vendors such as Cisco, Symantec and Skype use CVSS to score their own application vulnerabilities.

[7] CLASP (Comprehensive, Lightweight Application Security Process), http://www.owasp.org/index.php/CLASP