

# Risk reduction in legacy software systems

Danny Lieberman  
[dannyl@software.co.il](mailto:dannyl@software.co.il)

## ABSTRACT

Security breaches continue to garner headlines as conventional information security tools fail to halt attacks on customer data and intellectual property.

This article suggests that reduction of defects in enterprise legacy systems can be a highly effective approach for reducing operational risk. We introduce a risk analysis process that employs standard software vulnerability classifications and quantitative evaluation of how well removing defects reduces risk. We will show how this process meets the sizable challenges of improving legacy software quality.

The output of the process is financial justification for an effective risk mitigation plan. The plan includes the most cost-effective countermeasures that reduce the risk level to a minimum at a given capital and variable cost.

A real-life case study is included that illustrates the power of this practical, yet analytic method.

## THE PROBLEM: DEFECTIVE SOFTWARE IS INSECURE SOFTWARE

### Insecure software is a major factor in security breaches

This seemingly obvious observation is graphically borne out in a study that analyzed a sample of 167 privacy security breaches in 2005.<sup>1</sup> Based on data provided by the Privacy Rights Clearinghouse,<sup>2</sup> 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.

Aggregated vulnerability distribution by type		
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”.<sup>3</sup>

### Do organizations really want to improve legacy 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.<sup>4</sup>

<sup>1</sup> 2005 Breach Analysis, April 2006 <http://www.software.co.il/downloads/breachAnalysis2005.xls>

<sup>2</sup> Privacy Rights Clearinghouse, <http://www.privacyrights.org/>

<sup>3</sup> “Developing Secure Software, Noopur Davis, <http://www.softwaretechnews.com/stn8-2/noopur.html>

<sup>4</sup> “Top-down Security”, Alan Paller, <http://infosecuritymag.techtarget.com/articles/1999/paller.shtml>

### **Why are firewalls, anti-virus and anti-spyware having trouble keeping up?**

Security technology is seen as a means of defending the organization rather than as a means of improving understanding and reduction of 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 and at endpoints; removable device control and personal firewalls. The defense-focus is on **outside-in** attacks, despite the fact that the majority of attacks on customer data and intellectual property are **inside-out**.

Secure content management products can help control access, and regulate flows of sensitive data in order to prevent unauthorized disclosure. However, the process of classifying and monitoring enterprise content is highly complex and expensive, and as a result has not seen wide customer adoption.

## THE SOLUTION: COST EFFECTIVE DEFECT REDUCTION FOR LEGACY SOFTWARE

It is rare to see systematic defect reduction projects in production software, 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 legacy systems.
2. The cost of finding and fixing a bug in a production system is high.<sup>5</sup>
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 tenets:

1. Use a risk analysis process that is suitable for legacy systems.  
Collect data from all levels in the organization that touch the legacy 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.

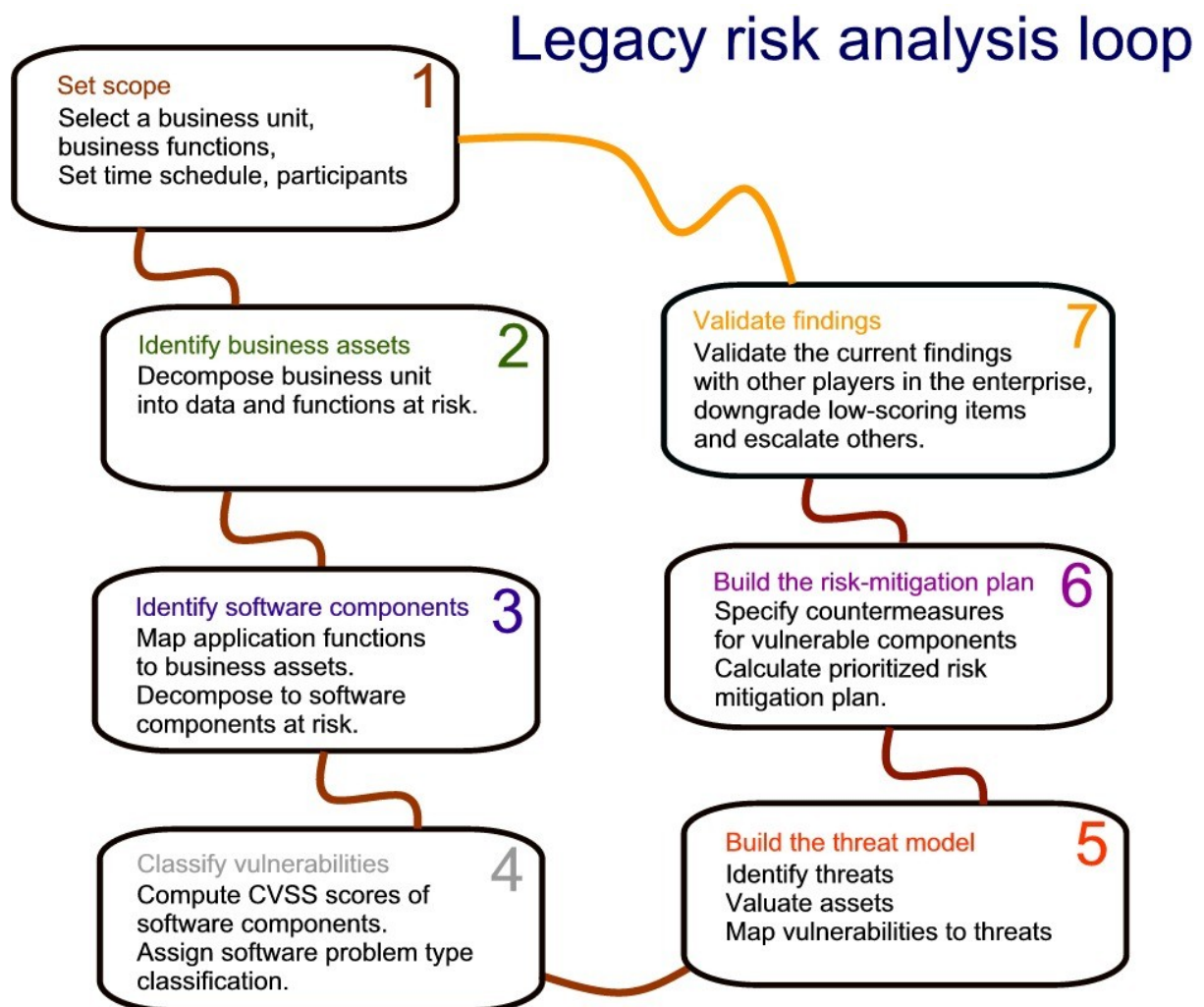
---

<sup>5</sup> "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

## RISK ANALYSIS PROCESS FOR LEGACY SYSTEMS (TENET #1)

### Overview

The risk analysis 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.



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.

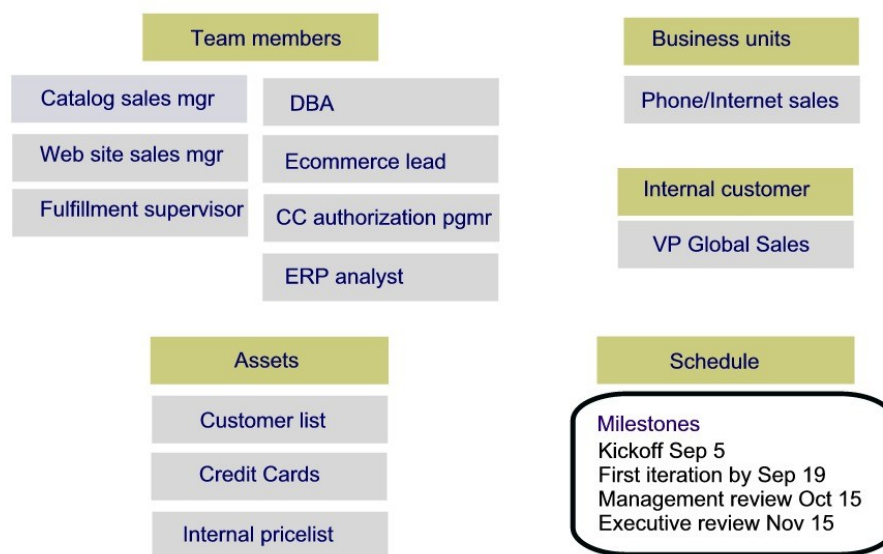
## 1. Set scope.

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 will be guided by 2-4 expert risk analysts that have good people skills and patience to work in a chaotic process.

The output of Set Scope is one page:

## Set Scope

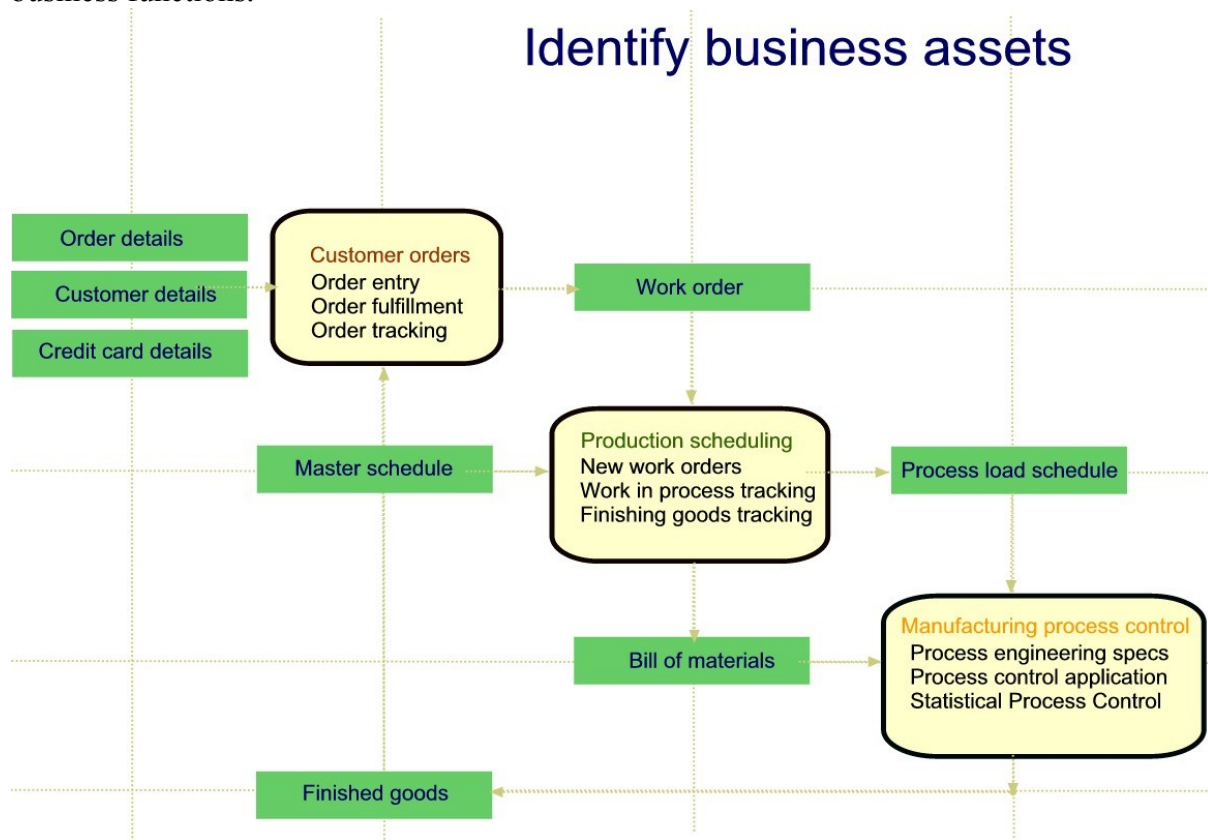


## 2. Identify business assets

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 (white boxes) are placed on a diagonal from top left to bottom right as shown in the below figure. Assets (colored in green) flow clockwise around the diagonal of business functions.



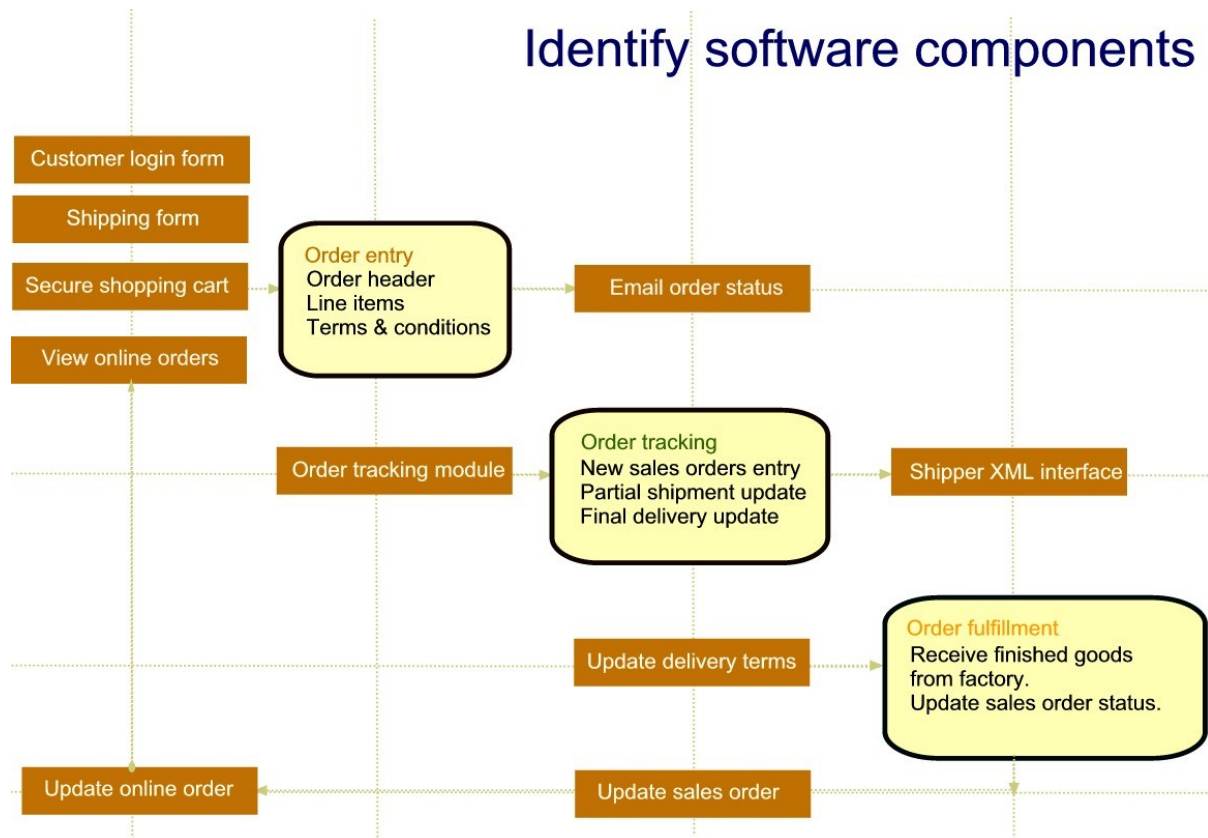
### 3. Identify software components

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.

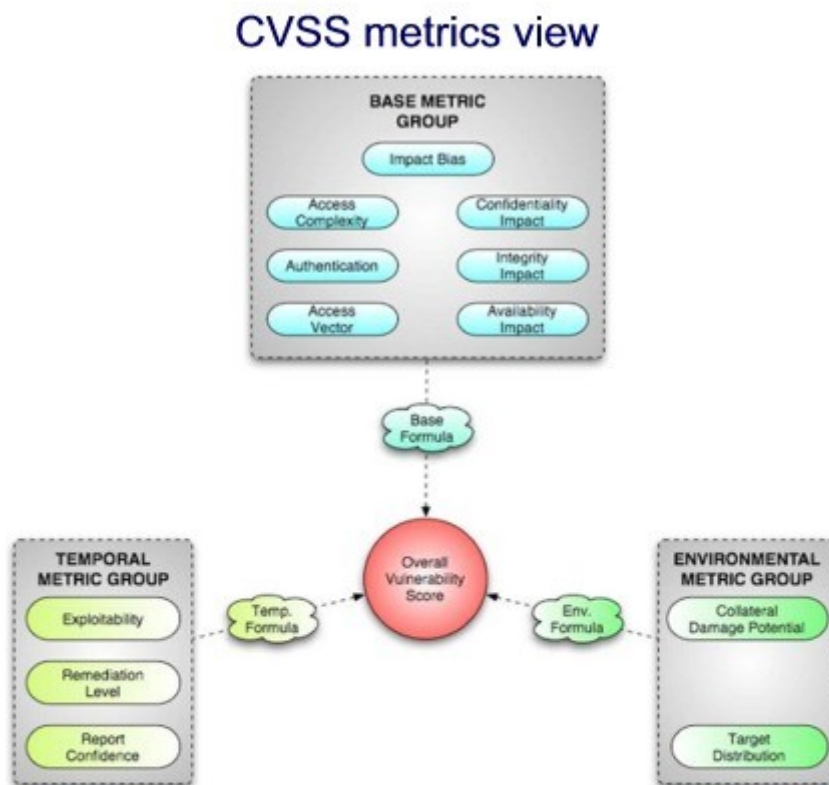
Application functions (white boxes) are placed on a diagonal from top left to bottom right as shown in the below figure. Decomposed components (colored in tan) flow clockwise around the diagonal of application functions.



#### 4. Classify the software vulnerabilities.

CVSS<sup>6</sup> 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<sup>7</sup> 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.



#### Problem type category (example)

Use of hard-coded password

Severity: High

Likelihood of exploit: Very high

Avoidance and mitigation

Utilize a "first login" mode,  
which requires the user to  
enter a unique strong password.

<sup>6</sup> 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.

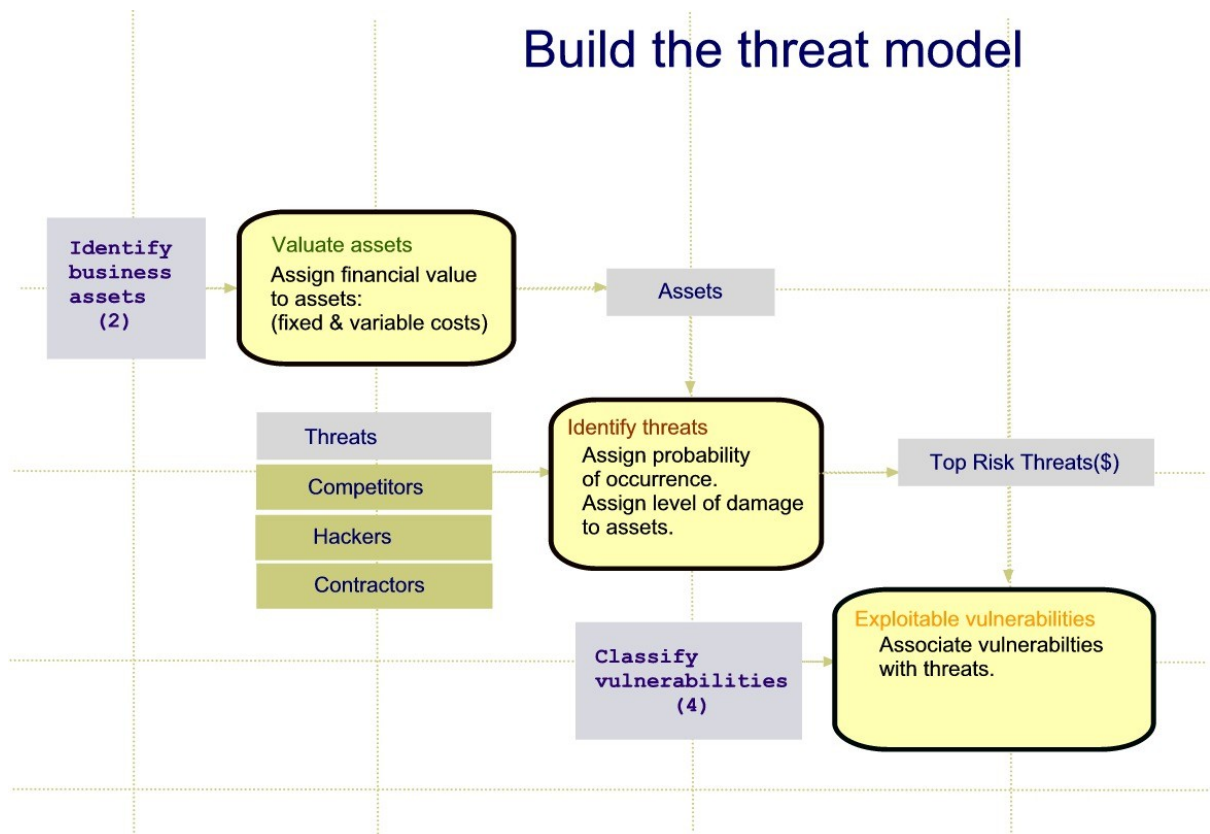
<sup>7</sup> CLASP (Comprehensive, Lightweight Application Security Process), <http://www.owasp.org/index.php/CLASP>

All contents are copyright © 2006 Software IL. All rights reserved.

## 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

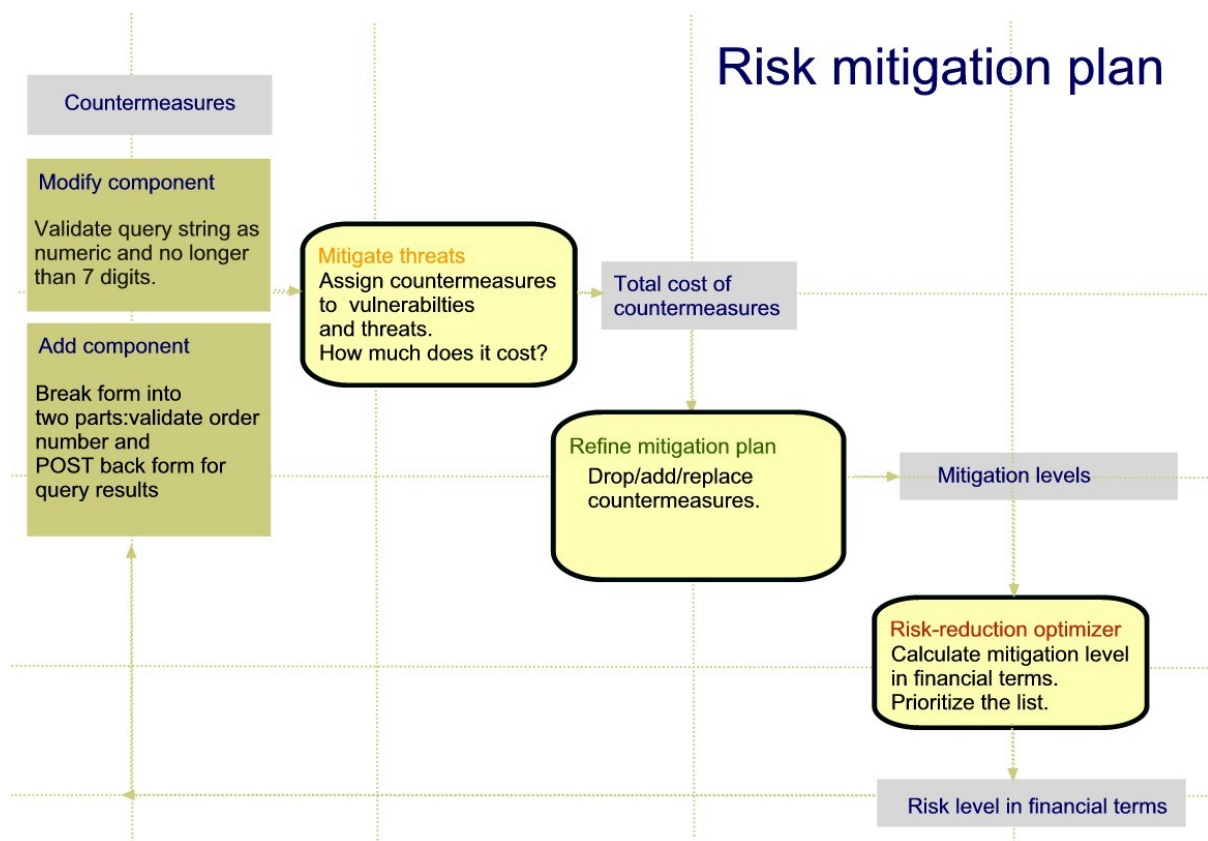


## 6. Build the risk-mitigation plan

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.



## 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`.

## QUANTITATIVE EVALUATION AND FINANCIAL JUSTIFICATION (TENET #2)

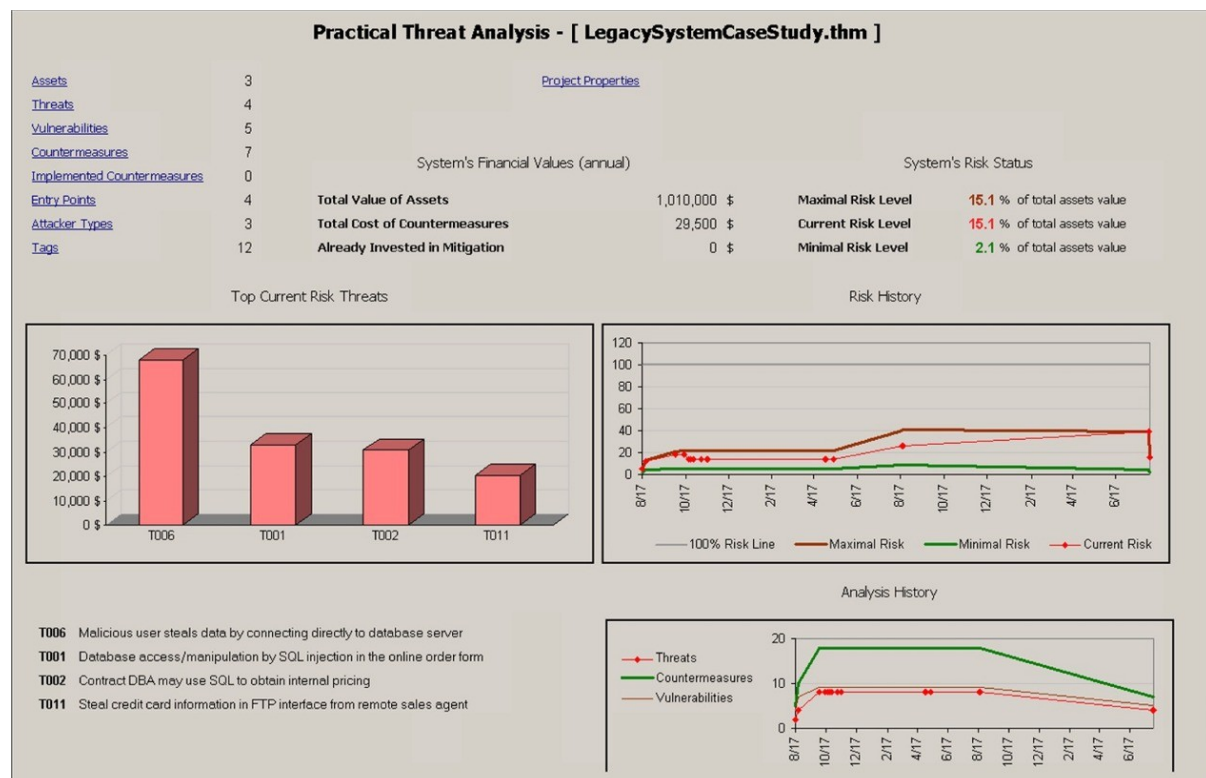
The output of the process provides executives with financial justification for an effective risk mitigation plan.

After specifying countermeasures, the PTA risk-reduction optimization algorithm produces a prioritized list of the countermeasures in financial terms. The risk-reduction algorithm combines the most cost-effective countermeasures to reduce the overall system risk level to a minimum at a total fixed and variable cost.

### Customer case study

We analyzed a business unit with systems for online order processing, make-to-order production and customer credit payment processing. Three key assets were identified: customer records, credit card details and internal pricelists. As can be seen in the below financial analysis, the risk to the assets can be reduced from 15% to 2% at a cost of \$29,500 with seven selected countermeasures. Risk is reduced to a minimum by proactive defect-reduction at a cost of less than 3 percent of the asset value. The cost is an order of magnitude less than acquisition of a proprietary system for preventing leakage of credit cards and privacy information.

You can see the [detailed risk mitigation plan](#) here. Click here to [Download the PTA data model](#) from the study and click here to [Download PTA](#) for a free copy of the software that will enable you to run this data model and build your own models using our study as an example.



### **EXPLICIT COMMUNICATIONS BETWEEN DEVELOPERS AND SECURITY (TENET #3)**

The first order of business is having people talk to each other and argue the issues. By publishing CVSS scores and countermeasure costs, the developer and security teams can be confident that they can respond to a particular type of event in a consistent fashion.

We have found in our practice with clients, that an online collaboration tool (such as Issue-Tracker) helps give the company a clear, updated picture of well-known defects and security events. The collaboration tool supports continuous risk analysis and management with two main applications: a knowledge base and issue tracker:

1. The database of standard CVSS scores for components and CLASP problem types classifications is always available for the entire organization. Users can add new entities and modify scores as the business environment changes.
2. The issue tracker provides:
  - a. A consistent thread of requests, changes and open action items during the risk analysis process and in particular in the `Validate findings` step
  - b. Updated implementation status of countermeasures.
  - c. Unlike email, issues cannot get lost or be ignored!

## **AFTERWARDS: SUSTAINING CONTINUOUS RISK REDUCTION**

### **Training a team that can sustain quality**

While the process itself has educational benefit, there is no question that the quantity and complexity of legacy systems in a large organization requires skills for continuous risk analysis and defect reduction.

To meet this objective, we propose two courses: The first - "*Risk analysis for the legacy*" is a five day course that trains analysts and qualifies them to run a defect reduction process as described in this paper. An advanced course will qualify analysts, who have conducted three risk analysis projects, to teach the "*Risk analysis for the legacy*" course and coach others.

### **Improving best practices in the software development life cycle**

The risk analysts supporting the process, together with the knowledge base are a significant resource for any organization that wants to evaluate the economic feasibility of a defect reduction program and improve best practices in the software development life cycle:

- 1.How to reduce avoidable rework
- 2.How to reliably identify fault-prone modules in a company's particular operation
- 3.How to identify modules with the most impact on system reliability and downtime
- 4.Develop sustaining metrics for defect reduction
- 5.Train application programmers in best security practices and help them see themselves as part of an integrated company-wide commitment to quality software.
- 6.Help the organization choose and implement disciplined practices such as Watts Humphrey's PSP (Personal Software Process) and TSP (Team Software Process) that can have high ROI in defect reduction in new software development.