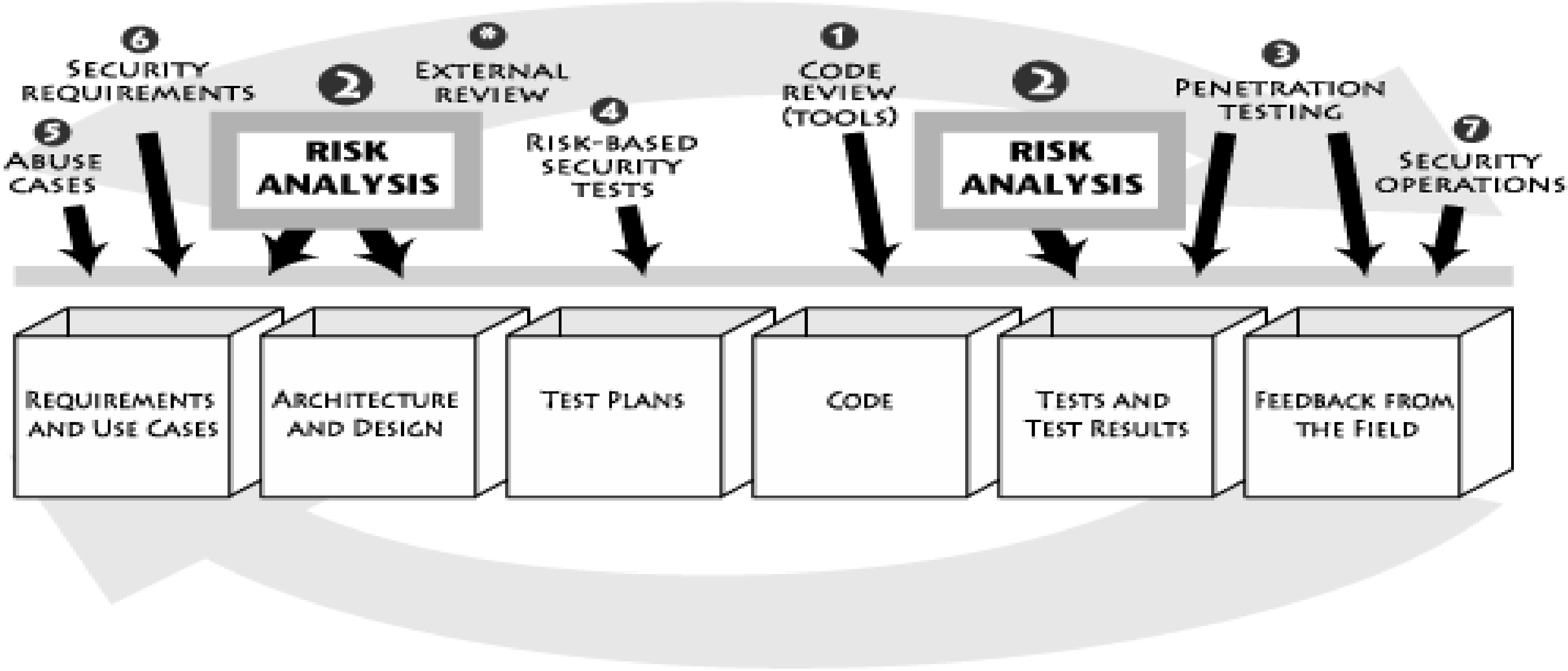
Architectural Risk Analysis

Architectural Risk Analysis



Risk analysis approach

1)Learn as much as possible about the target of analysis.

- Read and understand the specifications, architecture documents, and other design materials.
- Discuss and brainstorm about the target with a group.
- Determine system boundary and data sensitivity/criticality.
- Play with the software (if it exists in executable form).
- Study the code and other software artifacts (including the use of code analysis tools).
- Identify threats and agree on relevant sources of attack (e.g., will insiders be considered).

Risk analysis approach

2) Discuss security issues surrounding the software.

- Argue about how the product works and determine areas of disagreement or ambiguity.
- Identify possible vulnerabilities, sometimes making use of tools or lists of common vulnerabilities.
- Map out exploits and begin to discuss possible fixes.
- Gain understanding of current and planned security controls.

Risk analysis approach

3) Determine probability of compromise.

- Map out attack scenarios for exploits of vulnerabilities.
- Balance controls against threat capacity to determine likelihood.

4) Perform impact analysis.

- Determine impacts on assets and business goals.
- Consider impacts on the security posture.

Risk analysis approach

5)Rank risks.

6)Develop a mitigation strategy.

- Recommend countermeasures to mitigate risks.

7)Report findings.

- Carefully describe the major and minor risks, with attention to impacts.
- Provide basic information regarding where to spend limited mitigation resources.

Risk Analysis in Practice

Two basic categories:

1. Commercial: STRIDE from Microsoft, Security Risk Management Guide, also from Microsoft, ACSM/SAR (Adaptive Countermeasure Selection Mechanism/Security Adequacy Review) from Sun
2. Standards-Based: ASSET (Automated Security Self-Evaluation Tool) from the National Institute on Standards and Technology (NIST) , OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) from SEI

Traditional Risk Analysis Terminology

- Asset: A system component, data, or even a complete system.
- Risk: The probability that an asset will suffer an event of a given negative impact.
- Threat: The actor or agent who is the source of danger.

Traditional Risk Analysis Terminology

- Vulnerability: In general, a vulnerability is a defect or weakness in system security procedures.
- Countermeasures or safeguards: Technical controls prescribed for an information system which, taken together, adequately protect the confidentiality, integrity, and availability of the system and its information.

Traditional Risk Analysis Terminology

- Probability: The likelihood that a given event will be triggered. Three simple buckets:
    1. High (H)
    2. Medium (M),
    3. Low (L).

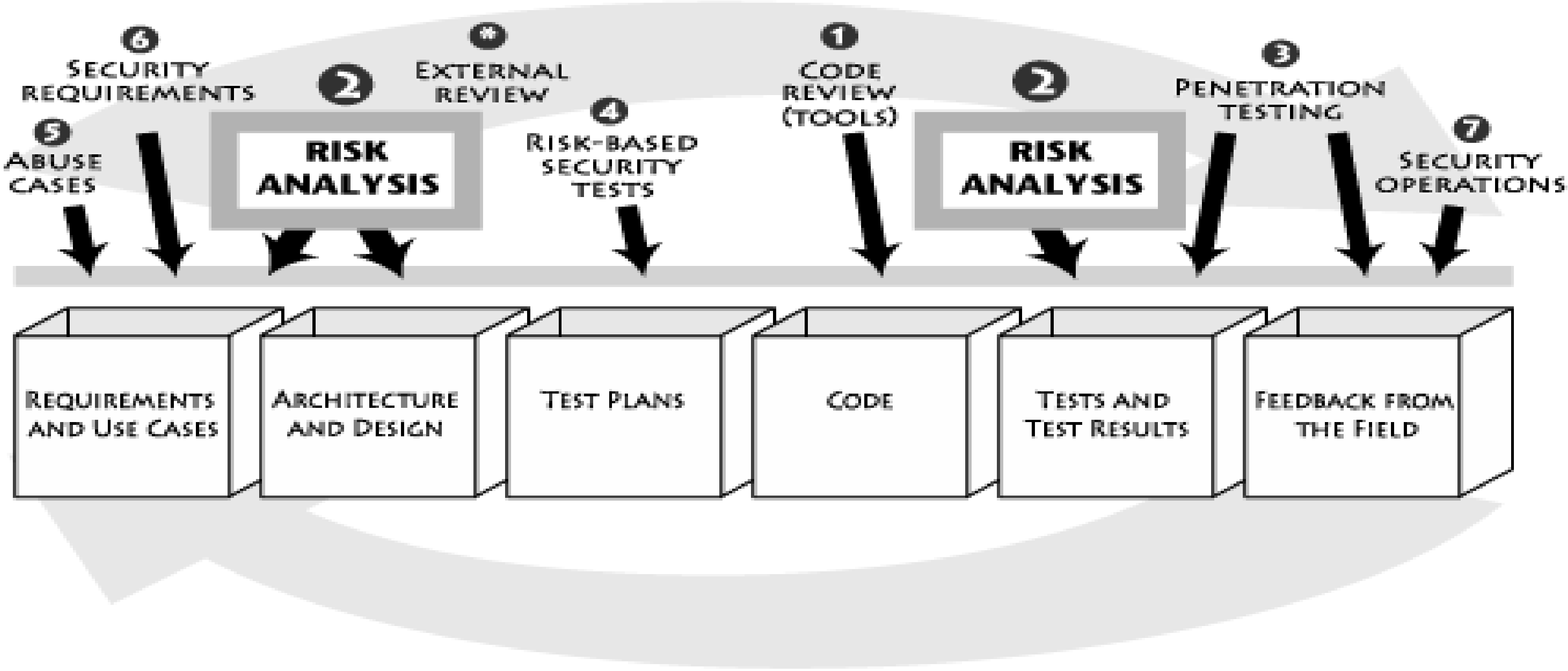
Threat Modeling versus Risk Analysis: Microsoft Redefines Terms

STRIDE is an acronym for

- Spoofing,
- Tampering,
- Repudiation,
- Information disclosure,
- Denial of service, and
- Elevation of privilege.

Source: Official Link

Architectural Risk Analysis



Risk Analysis (Requirements And Use Cases)

- Missing Stakeholders
- Wrong Stakeholders
- Ambiguous Requirements
- Incomplete Requirements
- Conflicting Requirements
- Infeasible Requirements
- Unverifiable Requirements
- Undocumented Assumptions
- Invalid Assumptions
- Inadequate Validation

Risk Analysis (Architecture And Design)

- Design flaws account for 50% of security problems.
- Some requirements are not specified properly.
- Validation rules might be improper in requirement stage.
- Designer should know about tools and languages.
- Designer should be aware of known attacks.

Traditional Risk Calculation Approach

One classic risk-analysis method expresses risk as a financial loss, or annualized loss expectancy, based on the following equation:

$$ALE = SLE \times ARO$$

where SLE is the single loss expectancy and ARO is the annualized rate of occurrence.

For an example,

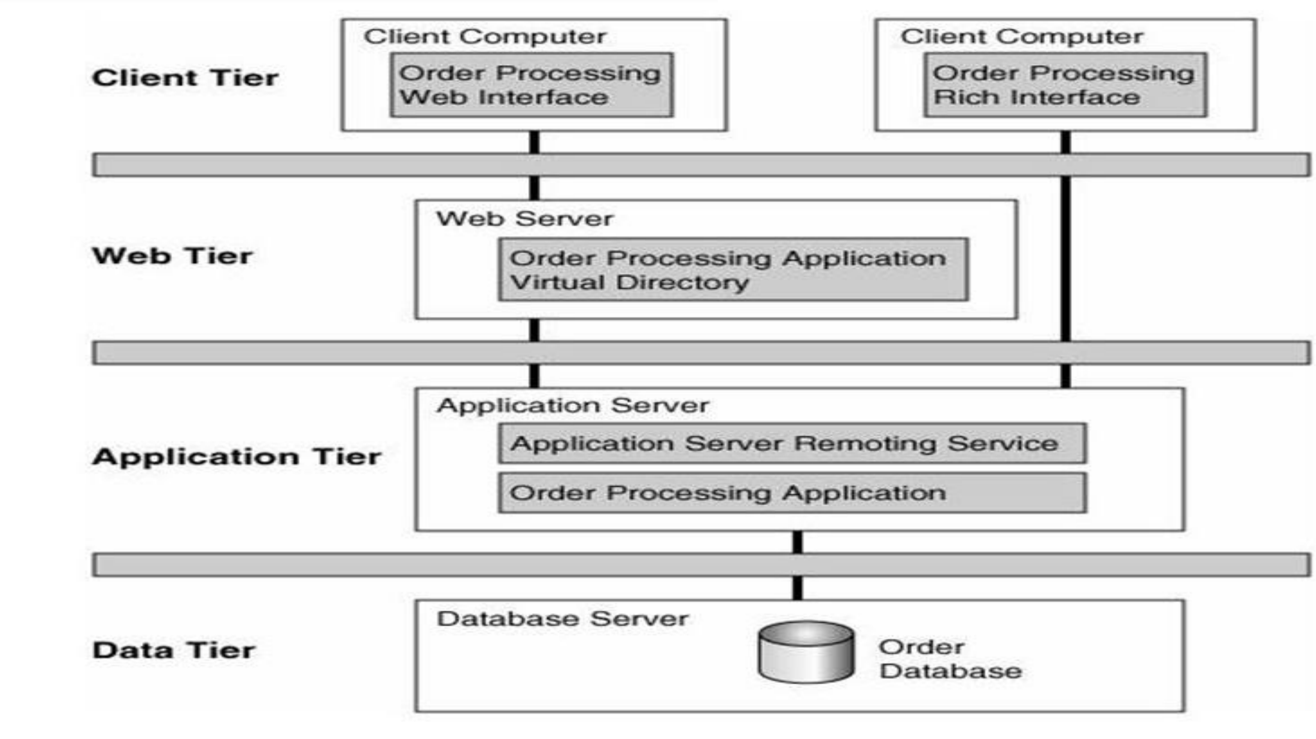
A event causes financial loss for ABC market. Let’s assign a cost of \$150 for any such event, so SLE = \$150. With an ARO of just 100 such events per year, the cost to the company (or ALE) will be \$15,000.

Limitations of Traditional Approaches

- In the case of a Web server providing a company's face to the world, a Web site defacement might be difficult to quantify as a financial loss.
- Traditional risk analysis techniques do not necessarily provide an easy guide of all potential vulnerabilities and threats to be concerned about at a component level.

Source: Risk analysis in software design

A Basic Risk Analysis Approach


Client Tier
Client Computer
Order Processing Web Interface
Client Computer
Order Processing Rich Interface
Web Tier
Web Server
Order Processing Application Virtual Directory
Application Tier
Application Server
Application Server Remoting Service
Order Processing Application
Data Tier
Database Server
Order Database

A Basic Risk Analysis Approach

During the risk analysis process one should consider…

- The threats who are likely to attack our system.
- The risks present in each tier's environment.
- The kinds of vulnerabilities that might exist in each component, as well as the data flow.
- The business impact of such technical risks, were they to be realized.
- The probability of such a risk being realized.
- Any feasible countermeasures that could be implemented at each tier.

Touchpoint Process: Architectural Risk Analysis

- A risk analysis should be carried out only once a reasonable, big-picture overview of the system has been established.
- Thus the first step of the process shown in the figure is to build a one-page overview of the system under analysis. Sometimes a one-page big picture exists, but more often it does not.
- The one-page overview can be developed through a process of artifact analysis coupled with interviews.

Touchpoint Process: Architectural Risk Analysis

Three critical steps (or subprocesses) make up the heart of this architectural risk analysis approach

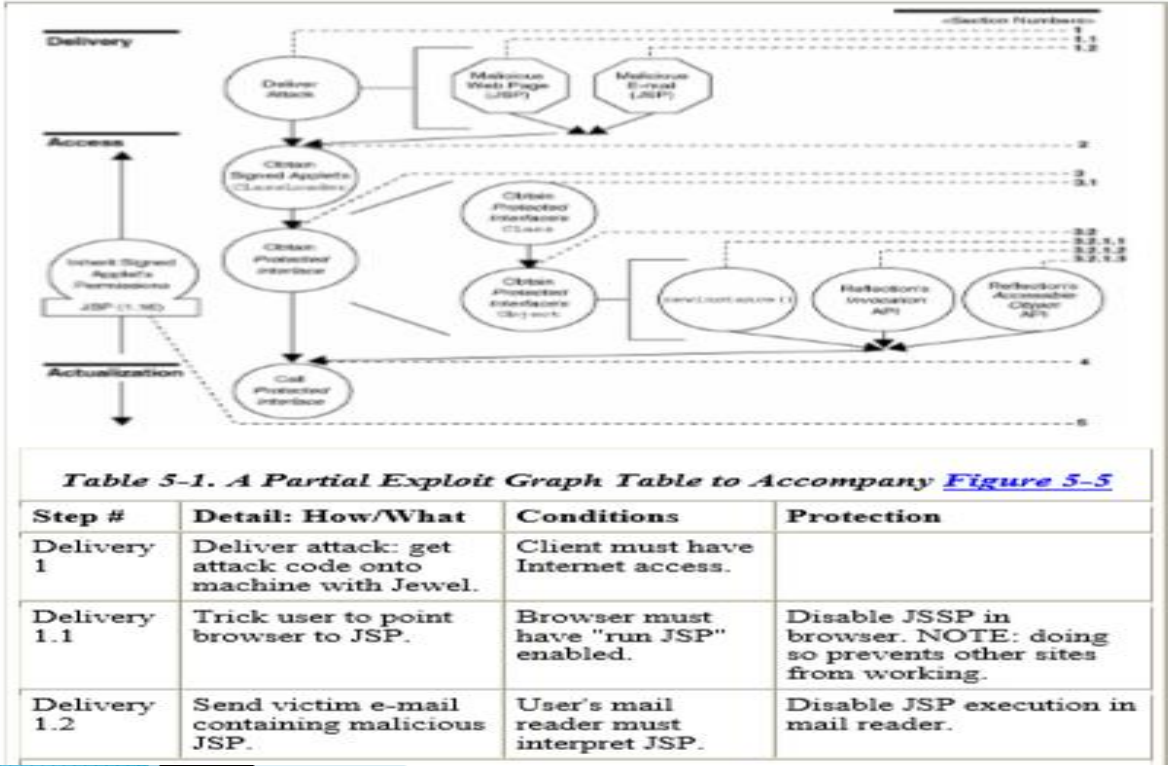
- Attack resistance analysis
- Ambiguity analysis
- Weakness analysis

Attack Resistance Analysis

Four steps are involved in this subprocess

- Identify general flaws using secure design literature and checklists (e.g., cycling through the Spoofing, Tampering, ... categories from STRIDE). A knowledge base of historical risks is particularly useful in this activity.
- Map attack patterns using either the results of abuse case development or a list of attack patterns.
- Identify risks in the architecture based on the use of checklists.
- Understand and demonstrate the viability of these known attacks (using something like exploit graphs; see the Exploit Graphs box).

Exploit Graphs box



<Section Numbers>
Delivery
Access
Actualization
Deliver Attack
Malicious Web Page (JSP)
Malicious E-mail (JSP)
Obtain Signed Applets (ClassLoader)
Obtain Protected Interface
Obtain Protected Interfaces Class
Obtain Protected Interfaces Object
Inherit Signed Applet's Permissions
JSP (1.1d)
Reflection's Invocation API
Reflection's Accessible Object API
Call Protected Interface
1
1.1
1.2
2
3
3.1
3.2
3.2.1.1
3.2.1.2
3.2.1.3
4
5

Table 5-1. A Partial Exploit Graph Table to Accompany Figure 5-5

Step #	Detail: How/What	Conditions	Protection
Delivery 1	Deliver attack: get attack code onto machine with Jewel.	Client must have Internet access.	
Delivery 1.1	Trick user to point browser to JSP.	Browser must have "run JSP" enabled.	Disable JSSP in browser. NOTE: doing so prevents other sites from working.
Delivery 1.2	Send victim e-mail containing malicious JSP.	User's mail reader must interpret JSP.	Disable JSP execution in mail reader.

Ambiguity Analysis

- Ambiguity analysis helps to uncover ambiguity and inconsistency
- Ambiguity analysis is the subprocess capturing the creative activity required to discover new risks
- This process, by definition, requires at least two analysts (the more the merrier) and some amount of experience
- this subprocess works best when carried out by a team of very experienced analysts

Weakness Analysis

- Weakness analysis is a subprocess aimed at understanding the impact of external software dependencies.
- It can be happened in Frameworks, network topology
- Example flaws
  - Debug interfaces
  - Unused (but privileged) product "features“
  - Interposition attacks—DLLs, library paths, client spoofing

Architectural Risk Analysis Is a Necessity