Risk Analysis for Secure Software Design

Dr. Natarajan Meghanathan
Associate Professor
Department of Computer Science
Jackson State University
E-mail: natarajan.meghanathan@jsums.edu

Terminologies

- Vulnerability – Is a weakness in the security system, in procedures, design or implementation, that might be exploited to cause loss or harm.
- Threat – A threat to a computing system is a set of circumstances that has the potential to cause loss or harm.
- Attack – An attack on the system is the execution of the threat by exploiting the vulnerabilities of the system.
- Risk – The possibility of an attack to occur
- Control – A control is an action, device, procedure, or technique that removes or reduces a vulnerability.
- A threat is blocked by control of a vulnerability.

Terminologies

- Asset – The object of protection efforts. Example: a system component, data or even a complete system
- Likelihood – The probability that a given event will be triggered. This quantity is often expressed as a percentage (it is tough to calculate one) or categories (e.g., High – H, Medium – M, Low – L)
- Impact – The impact on the organization using the software, if the risk is to be realized. The impact could be monetary, or tied to reputation, or may result from the breach of a law, regulation or contract. It is critical to quantify the impact (e.g., categories: High, Medium and Low)
- Risk Exposure – A measure of the overall impact to the organization and an indication of the extent to which the risk has to be mitigated. Risk Exposure = likelihood * impact.
- A risk indicator is a sign that the risk is materializing, an objective, measurable event that can be monitored and measured by the analyst to determine the status of a risk over time.

- Various factors determine the calculation for risk – the ease of executing an attack, the motivation and resources of an attacker, the existence of vulnerabilities in a system, and the cost or impact in a particular business context.

5/7 Stages of Risk Assessment

- Qualitative Model: The risk assessment methodology encompasses seven fundamental activity stages, listed below:
  - **Application characterization**
  - **Asset Identification**
  - **Architectural vulnerability assessment**
  - **Threat analysis**
  - **Risk likelihood determination**
  - **Risk impact determination**
  - **Risk mitigation**
- Quantitative Model (5 stages):
  - **Application characterization**
  - **Asset Identification**
  - **Architectural vulnerability assessment**
  - **Threat analysis**
  - **Risk Quantification**
    - **DREAD risk quantification model**
    - **Annualized loss expectancy (ALE) model**

1. Application Characterization

- Application characterization – defining the scope of the software system architecture, documented as artifacts. The goal is to produce document(s) that depict the vital relationships between the critical parts of the system.
  - For an application in the initiation or design phase, the artifacts are the design or requirements documents
  - For an application under development, the artifacts are the system design documents, security rules, attributes and plan
  - For an application that has been fielded, artifacts include data on the software in its production environment – system configuration, connectivity

2. Asset Identification

- Very often, software uses information assets that are critical for the business.
- Three different sub-stages:
  - Identify the information assets (e.g., databases, credentials like user id and password, audit records, financial info, intellectual property and etc) that must be protected
  - Each information asset may have one or more properties (e.g., confidentiality, auditability, integrity and availability) that must be maintained
  - Impact to the business if a property on an asset is not maintained.
- Often assets can be identified through a thorough understanding of the software and its working.
  - Example: The availability of the “Customer Accounts Database” to match the caller ID with the customer record is critical to reduce the caller waiting time as well as the aggregate call volume of the customer service rep.
- Given the information assets, one should be able to consider what software modules manipulate those assets.

Desirable Properties for an Asset

- Confidentiality – Preserving authorized restrictions on access and disclosure, including means for protecting personal privacy and proprietary information

- Integrity – Guarding against improper information modification or destruction, and includes ensuring

- Information non-repudiation - Actions of an entity are to be traced back uniquely to that entity

- Authenticity - Verifying that users are who they say they are and that each input arriving at the system came from a trusted source

- Availability – Ensuring timely and reliable access to and use of information.

3. Architectural Vulnerability Assessment

- Three types:
  - **Known vulnerability analysis**
  - **Ambiguity analysis**
  - **Underlying platform vulnerability analysis**
- **Known vulnerability analysis** – analyze the artifacts obtained for asset identification and review them with respect to the known bad practices or known good principles for confidentiality, integrity and availability.
  - **For example, identify system components that operate at an elevated privilege and low privilege and review the boundaries between these two areas and the kinds of communications across the boundaries.**
- **Ambiguity analysis**
  - **Ambiguity and/or disagreement is a rich source of vulnerabilities when it exists between requirements/specifications and development.**
  - **The artifacts for both requirements/specifications (use cases, user stories) should be compared with the development artifacts (code, API docs).**
    - For example, a web application may have a requirement that if an administrator locks an account, the user can no longer login while the account remains locked. The requirements should also clearly specify what happens to user accounts that were currently logged in when the administrator locked the accounts? Are these users forced to log out and can their sessions be active until they log out themselves? The implementation should be done accordingly

Architectural Vulnerability Assessment

- Underlying platform vulnerability analysis
  - This phase includes an analysis of the vulnerabilities associated with the application's execution environment (e.g., OS, network, platform vulnerabilities and vulnerabilities resulting from the interaction of the software components with the environment).
  - The goal would be to develop a list of application/environment vulnerabilities that may be accidentally triggered or intentionally exploited
- Vulnerability Classes
  - **incomplete parameter validation:** input parameters not validated for type, format, and acceptable values
  - **inconsistent parameter validation:** input validation does not follow consistent scheme
  - **implicit sharing of privileged/confidential data:** resources are not appropriately segregated
  - **asynchronous validation/inadequate serialization:** vulnerabilities resulting from concurrency, sequencing of events as in message queue systems
  - **inadequate identification/authentication/authorization:** access control vulnerabilities
  - **violable prohibition/limit:** lack of enforcement on resource limitations, such as buffer overflows
  - **exploitable logic error:** program logic errors enabling circumvention of access control

4. STRIDE Threat Model

- The different threats can be grouped into one of the following categories that represent the kinds of exploit (or motivation of the attacker) that are used. Each category affects one or more properties of an information asset that needs to be protected.
  - **Spoofing identity**: Using someone else' authentication information (such as username and password) to illegally get access
  - **Tampering with data**: involves the malicious modification of data (e.g., database injection attacks, hacking the network communication).
  - **Repudiation:** Users deny performing an action which cannot be traced otherwise (e.g., users perform an illegal operation in a system that lacks the ability to trace the prohibited operations).
  - **Information disclosure**: Exposure of information to individuals who are not supposed to have access to it.
  - **Denial of service**: DoS attacks deny service to valid users—for example, by making a Web server temporarily unavailable or unusable.
  - **Elevation of privilege:** an unprivileged user gains privileged access and thereby has sufficient access to compromise or destroy the entire system.

STRIDE vs. Security Feature

- Spoofing → Authentication
- Tampering → Integrity
- Repudiation → Non-repudiation
- Information Disclosure → Confidentiality
- Denial of Service → Availability
- Elevation of Privilege → Authorization

Network and Host Threats

- Network Threats
  - Information gathering
  - Sniffing
  - Spoofing
  - Session hijacking
  - Denial of service
- Host Threats
  - Viruses, Trojan horses, and worms
  - Footprinting
  - Password cracking
  - Denial of service
  - Arbitrary code execution
  - Unauthorized access

Attack Patterns

- Attack patterns (blue print of an exploit) help developers to get a solid understanding of the attacker's perspective (i.e., think like attackers) so that they can anticipate and thwart expected types of attacks.

- Attack patterns describe the techniques that would be employed by attackers to break software.

Example 1 for Generating an Attack Pattern

Source: https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/586-BSI.html

- **Pattern name and classification**: Denial of Service – Decompression Bomb
- **Attack Prerequisites**: The application must decompress compressed data. For the attack to be maximally effective, the decompression should happen automatically without any user intervention.
- **Description**: The attacker generates a small amount of compressed data that will decompress to an extremely large amount of data. The compressed data may only be a few kilobytes in size, whereas the decompressed data may be several hundred gigabytes in size. The target is running software that automatically attempts to decompress the data in memory to analyze it (such as with antivirus software) or to display it (such as with web browsers). When the target software attempts to decompress the malicious data in memory, it runs out of memory and causes the target software and/or target host to crash.
- **Method of Attack**: By maliciously crafting compressed data and sending it to the target over any protocol (e.g., e-mail, HTTP, FTP).

Example 1 for Generating an Attack Pattern

Source: https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/586-BSI.html

- **Attack Motivation-Consequences**: The attacker wants to deny the target access to certain resources.
- **Attacker Skill or Knowledge Required**: Creating the exploit requires a considerable amount of skill. However, once such a file is available, an unskilled attacker can find vulnerable software and attack it.
- **Resources Required**: No special or extensive resources are required for this attack.
- **Solutions and Mitigations:** Restrict the size of output files when decompressing to a reasonable value. Especially, handle decompression of files with a large compression ratio with care. Builders of decompressors could specify a maximum size for decompressed content and then cease decompression and throw an exception if this limit is ever reached.
- **Context Description:** Any application that performs decompression of compressed data in any format (e.g., image, archive, sound, gzip-ed HTML)

Example 2 for Generating an Attack Pattern

Source: https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/586-BSI.html

- **Pattern name and classification**: Shell Command Injection—Command Delimiters
- **Attack Prerequisites**: The application must pass user input directly into a shell command.
- **Description**: Using the semicolon or other off-nominal characters, multiple commands can be strung together. Unsuspecting target programs will execute all the commands. An example may be when authenticating a user using a web form, where the username is passed directly to the shell as in: exec("cat data_log_" + userInput + ".dat") The "+" sign denotes concatenation. The developer expects that the user will only provide a username. However, a malicious user could supply "username.dat; rm –rf / ;" as the input to execute the malicious commands on the machine running the target software. Similar techniques are also used for other attacks such as SQL injection. In the above case, the actual commands passed to the shell will be:
- cat data_log_username.dat; rm –rf /; .dat The first command may or may not succeed; the second command will delete everything on the file system to which the application has access, and success/failure of the first command is irrelevant.

Example 2 for Generating an Attack Pattern

Source: https://buildsecurityin.us-cert.gov/bsi/articles/knowledge/attack/586-BSI.html

- **Method of Attack**: By injecting other shell commands into other data that are passed directly into a shell command.
- **Attack Motivation-Consequences**: Execution of arbitrary code. The attacker wants to use the target software, which has more privilege than the attacker, to execute some commands that he/she does not have privileges to execute.
- **Attacker Skill or Knowledge Required**: Finding and exploiting this vulnerability does not require much skill. A novice with some knowledge of shell commands and delimiters can perform a very destructive attack. A skilled attacker, however, may be required to subvert simple countermeasures such as rudimentary input filtering.
- **Resources Required**: No special or extensive resources are required for this attack.
- **Solutions and Mitigations**: Define valid inputs to all fields and ensure that the user input is always valid. Also perform white-list and/or black-list filtering as a backup to filter out known command delimiters.
- **Context Description**: OS: UNIX.

Attack Trees

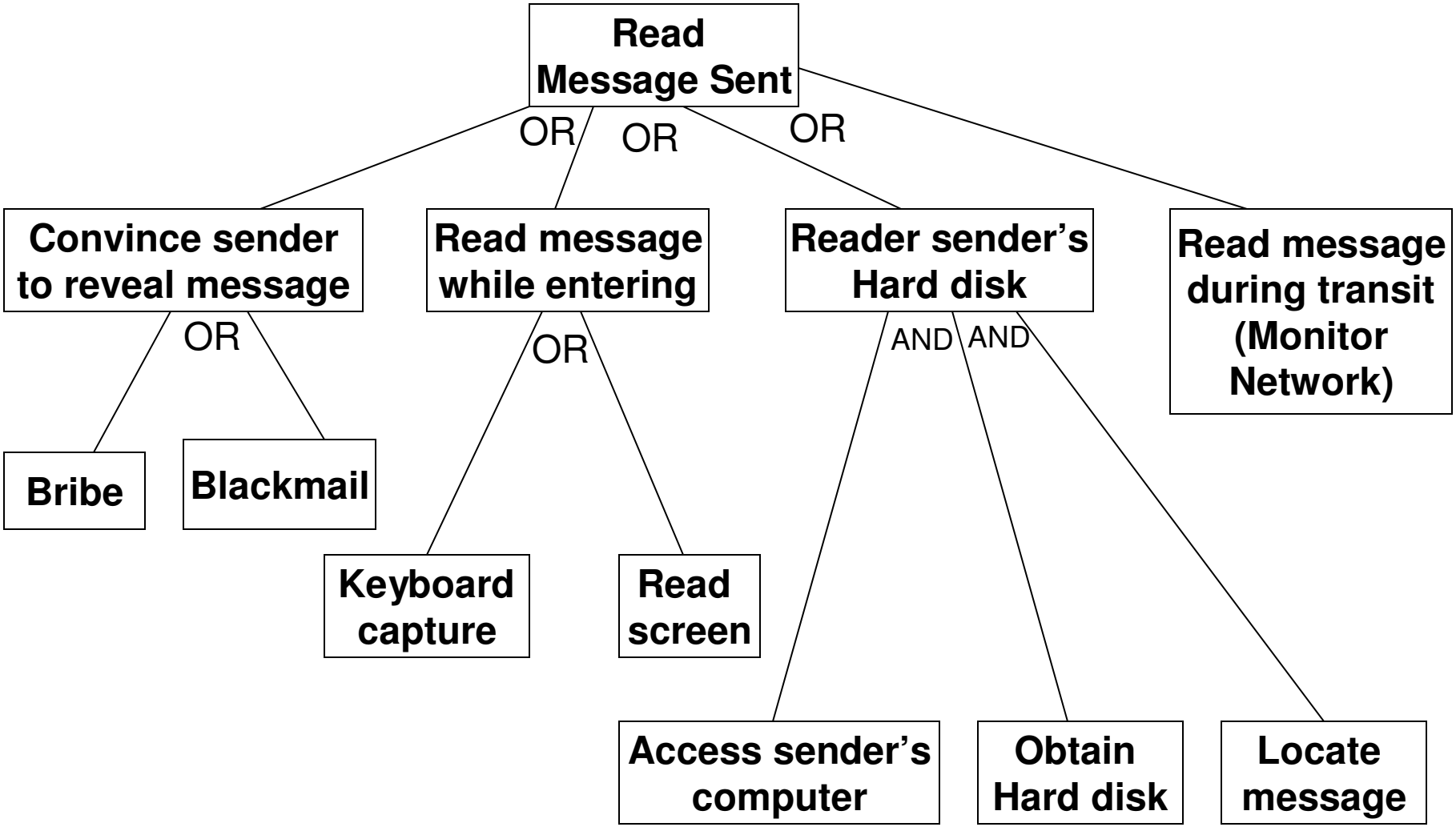
- Attack trees – represent attacks against a system in a tree structure, with the goal as the root node; an intermediate node becomes a sub goal, and children of that node are ways to achieve that sub goal.

- Each path (called the attack path) tracing from the root node to a leaf node represents a unique way to achieve the goal of the attacker.

- Attack trees can be used to answer the following questions:
  - What is the shortest path of the attack?
  - What is the easiest attack?
  - What is the cheapest attack?
  - Which attack causes the most damage?
  - Which attack path is the hardest to detect?

- A node in an attack tree can be either an “AND” node or an “OR” node.
  - OR nodes are alternatives;
  - AND nodes represent a combination of steps that must be successfully executed in order to achieve the goal.

Attack Tree for Reading a Message

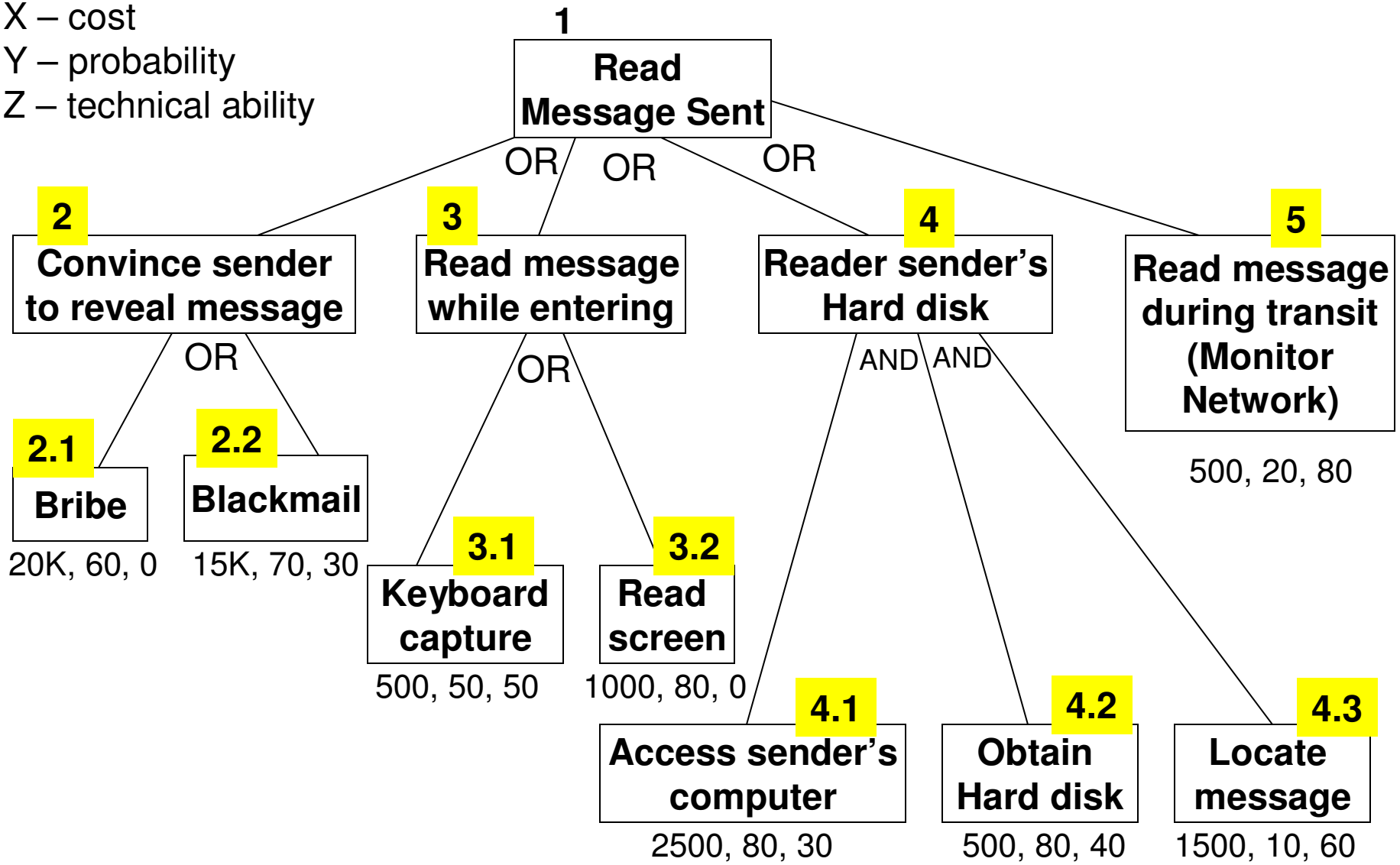


Attack Trees

- Once the basic attack tree is completed, values (called indicators) are assigned to the leaf nodes.
- Indicators are used to make calculations about the nodes and attack paths. One can calculate the security of the goal and rank each attack path according to the values of indicators.
- The indicators used for a particular node are:
  - Cost of attack: The cost (dollar value) to stage the attack on this node
  - Probability of apprehension: The probability of an attacker being caught performing the action in this node.
  - Technical ability: The skill level needed for this attack. It is usually on a scale of 1 to 100.
- For a node having two or more AND nodes as children:
  - Cost = Sum of the cost of the child nodes
  - Probability = Maximum of the probability of apprehension
  - Technical ability = Sum of the cost of the child nodes
- After evaluating the attack paths in the attack tree, the most vulnerable paths are selected depending on the attacker's characteristics and constraints. Proper security controls are added along these paths to decrease the chances of an attack.

Attack Tree with Indicator Values

X, Y, Z
X – cost
Y – probability
Z – technical ability

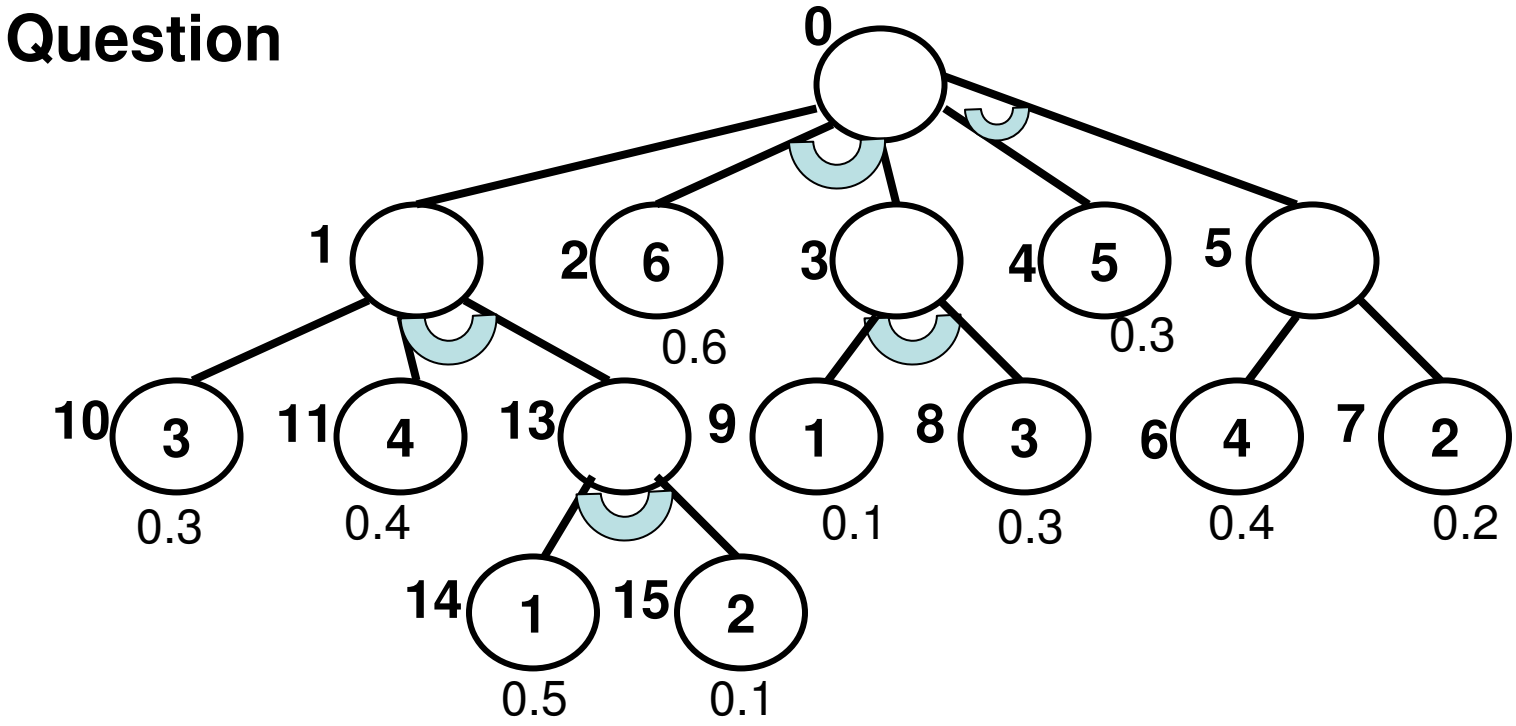


Attack Paths for the Tree to Read a Message

#	Path	Cost	Probability	Skill
1	1-2-2.1	$20,000	60%	0
2	1-2-2.2	$15,000	70%	30
3	1-3-3.1	$500	50%	50
4	1-3-3.2	$1000	80%	0
5	1-4-4.1, 4.2, 4.3	$2500 + $500 + $1500 = $4500	Max (80%, 80%, 10%) = 80%	30 + 40 + 60 = 130
6	1-5	$500	20%	80

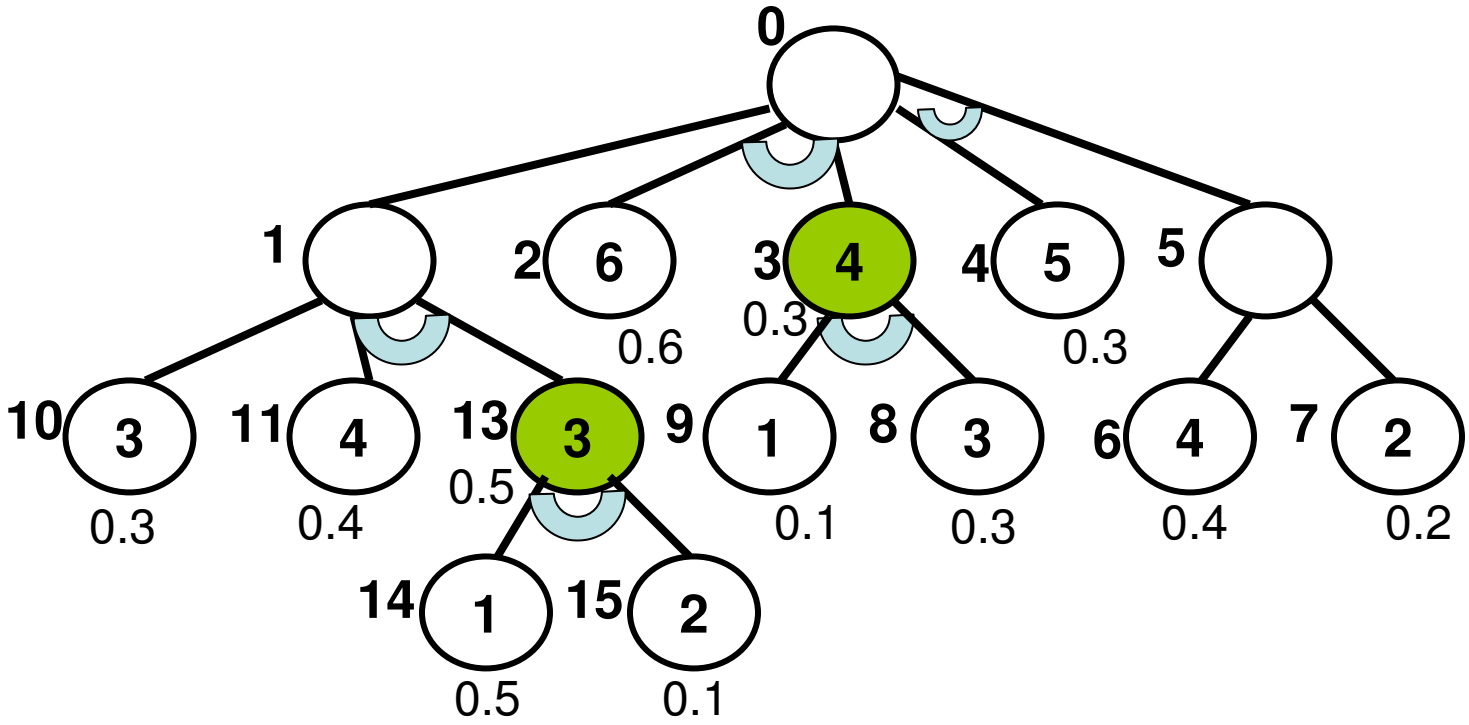
The selection of an attack path depends on the attacker's constraints and characteristics For example, if an attacker is rich, afraid of getting caught, and have low technical skills, then attack path 1 is most likely to be chosen compared to others. On the other hand, if an attacker is highly skilled; but is concerned about cost and also prefers an attack node that is difficult to catch; then attack path 6 is likely to be chosen.

Another Example on Attack Tree



The integers outside the circle indicate node ID
The integers inside the circle represent the cost
The decimal values outside the circle represent the probability of apprehension

Solution for Example on Attack Tree



The integers outside the circle indicate node ID
The integers inside the circle represent the cost
The decimal values outside the circle represent the probability of apprehension

Attack Paths for the Example of Attack Trees with Cost Values

#	Path	Cost	Probability of Apprehension
1	0 – 1 – 10	3	0.3 (30%)
2	0 – 1 – 11 - 13 – 14, 15	7	0.5 (50%)
3	0 – 2 – 3 – 8, 9	10	0.6 (60%)
4	0 – 4 – 5 – 6	9	0.4 (40%)
5	0 – 4 – 5 – 7	7	0.3 (30%)

If the **cost** denotes **Attack skills or the cost incurred in $ to launch the attack**, then path # 1 is the cheapest and it also has the least probability of apprehension. Hence, path # 1 is the most vulnerable to attack and hence more security controls should be added to increase the cost of node 10.

DREAD: Risk Quantification Model

- The DREAD model can be used to quantify, compare and prioritize the amount of risk presented by each evaluated threat so that the risks posed by different threats can be directly sorted.
- Risk_DREAD = (Damage + Reproducibility + Exploitability + Affected Users + Discoverability)/ 5
- Each of the above five categories can be assigned a score of 3-HIGH, 2-Medium, 1-Low and 0-None
- **D**amage potential:
  - How great is the damage if the vulnerability is exploited?
- **R**eproducibility:
  - How easy is it to reproduce the attack?
- **E**xploitability:
  - How easy is it to launch an attack?
- **A**ffected users:
  - As a rough percentage, how many users are affected?
- **D**iscoverability:
  - How easy is it to find the vulnerability?

DREAD: Threat Rating Criteria

- **Damage Potential**
  - High: The attacker can subvert the security system; get full trust authorization; run as administrator; upload content.
  - Medium: Leaking sensitive information
  - Low: Leaking trivial information
- **Reproducibility**
  - High: The attack can be reproduced every time and does not require a timing window
  - Medium: The attack can be reproduced, but only with a timing window and a particular race situation.
  - Low: The attack is very difficult to reproduce, even with knowledge of the security hole.
- **Exploitability**
  - High: A novice programmer could make the attack in a short time
  - Medium: A skilled programmer could make the attack, then repeat the steps.

DREAD: Threat Rating Criteria

- **Exploitability**
  - Low: The attack requires an extremely skilled person and in-depth knowledge every time to exploit.
- **Affected Users**
  - High: All users, default configuration, key customers
  - Medium: Some users, non-default configuration
  - Low: Very small % of users, obscure feature; effects anonymous users
- **Discoverability**
  - High: Published info explains the attack. The vulnerability is found in the most commonly used feature and is very noticeable.
  - Medium: The vulnerability is in a seldom-used part of the product, and only a few users should come across it. It would take some thinking to see malicious user.
  - Low: The bug is obscure, and it is unlikely that users will work out damage potential.

STRIDE Analysis Example

- **Application Scenario**
- *Student-Faculty Appointment System for Course Registration*: Through this application, students can login and request for appointments from their faculty/academic advisors to register for their classes. Care will be taken to ensure that there is no denial of service attacks wherein a student makes multiple fake appointments for a particular faculty during a particular time period. The financial aid opportunities available to the student for the particular semester are also displayed when the student makes the appointment for registration.
- Threats

1. Making multiple fake appointments - ***D****enial of service*
   - Student making multiple fake appointments within a time period so that genuine students are denied appointments
2. SQL injection attacks to login to the database – ***T****ampering with data*
   - A student may try to delete the records in the database by passing mischievous inputs for the username and password

DREAD Analysis for Fake Appointments Threat

RISK_DREAD Score
= (2+3+3+3+2)/5
= 2.6

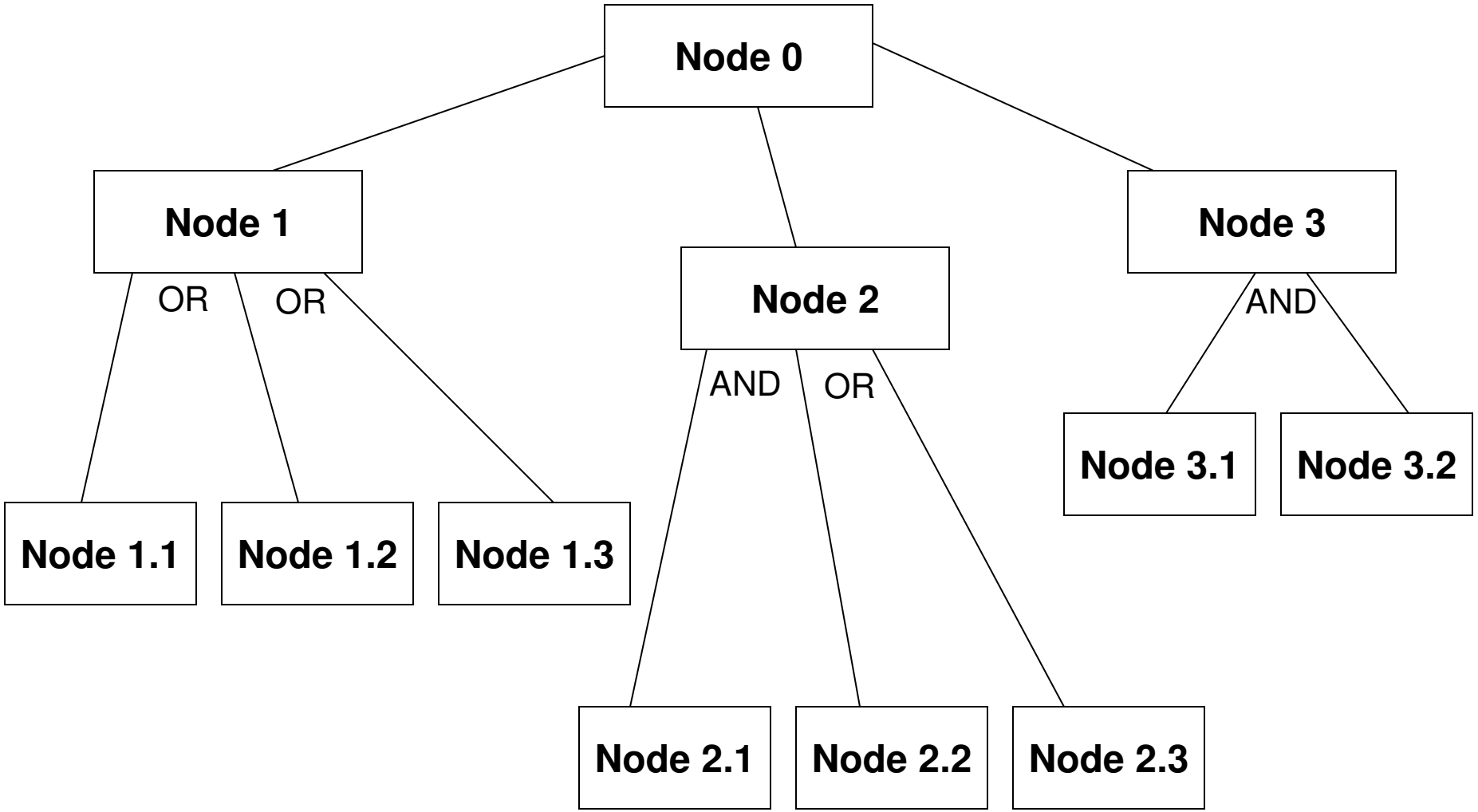
- **D**amage potential: How great is the damage if the vulnerability is exploited?
  - **MEDIUM (2)** – While the whole registration server system will not be affected, the students who want to make appointments to an advisor will be affected
- **R**eproducibility: How easy is it to reproduce the attack?
  - **HIGH (3)** – A student can make any number of appointments with any number of advisors if there are no proper controls set up.
- **E**xploitability: How easy is it to launch an attack?
  - **HIGH (3)** – It does not require sufficient expertise to make fake appointments
- **A**ffected users: As a rough percentage, how many users are affected?
  - **MEDIUM (2)** – While all students may not be affected, the students who require appointments to meet with an advisor will be affected.
- **D**iscoverability: How easy is it to find the vulnerability?
  - **MEDIUM (2)** – The vulnerability is not so common like a SQL injection attack on the login screen or username/password exposed in plaintext during transmission. The vulnerability of not controlling the number of appointments a student makes is very unique for such registration applications. Hence, it requires some thinking to exploit this vulnerability, if present.

STRIDE Analysis Example (contd..)

- Threats

3. Student registers for a course/appointment and later denies that he did not do it – ***Repudiation***
4. Login using the username and password (either reset or original) corresponding to a different user – ***Spoofing the identity***
   - **Node 1:** Through a social engineering attack –
     - ***Node 1.1***: Nexus with the system/db admin or
     - ***Node 1.2***: Send email from a fake address that appears to belong to that user
     - ***Node 1.3***: Stand next to the targeted user and read the username/password to the registration server when the user logs in
   - **Node 2:** Privacy Intrusion
     - ***Node 2.1, 2.2:*** By knowing a list of secret questions and the answers to these questions (learn the personal information of the targeted user) – lets say the system requires users to answer two questions correctly
     - ***Node 2.3:*** Learn the email address/password of the targeted user and click the Forget Password button in the Registration Page – Have the system send the reset password to the compromised email account.
   - **Node 3:** Hack the channel
     - ***Node 3.1*****:** Capture the username/password of the targeted user
     - ***Node* 3.2:** Do a cryptanalysis on it, if it is encrypted.

Attack Tree for Spoofing the Identity Threat



Attack Paths for Spoofing the Identity

#	Path	Cost	Probability of Apprehension
1	Node 0 – 1 – 1.1		
2	Node 0 – 1 – 1.2		
3	Node 0 – 1 – 1.3		
4	Node 0 – 2 – 2.1, 2.2		
5	Node 0 – 2 – 2.3		
6	Node 0 – 3 – 3.1, 3.2		

Annualized Loss Expectancy Model

- Single loss expectancy (SLE) = Asset value * Exposure factor
- Annualized loss expectancy (ALE) = SLE * ARO

- SLE – the dollar amount that is assigned to a single event that represents the company's potential loss amount if a specific threat were to take place.
- Exposure factor – represents the percentage of loss a realized threat could have on a certain asset.
  - For example, if a data warehouse has the asset value of $150,000, it might be estimated that if a fire were to occur, 25% of the warehouse would be damaged (and not more, because of a sprinkler system and other fire controls, proximity of a firehouse, and so on), in which case the SLE would be $37,500.
- Annualized rate of occurrence (ARO) – the value that represents the estimated frequency of a specific threat taking place within a one-year timeframe. The range can be from 0.0 (never) to 1.0 (once a year) to greater than one (several times a year).
  - An ARO value of 0.001 indicates that the probability of a threat realizing is once in 1,000 years.
  - Continuing with the above fire example, if the ARO of a fire taking place is 0.1 (i.e., once in ten years), then the ALE value is $37,500 *0.1 = $3,750.

Cost/Benefit Analysis

- A security countermeasure, sometimes called a safeguard, must make good business sense, meaning it is cost-effective (its benefits outweighs its cost).
- Value of safeguard to the company = (ALE before implementing the safeguard) – (ALE after implementing the safeguard) – (Annual cost of installing and maintaining the safeguard)
- The safeguard is implemented if its value is above zero. The safeguard that yields the largest value to the company is chosen and a risk mitigation strategy is implemented.
- If there exists no safeguard that has a positive value, then the company has to either:
  - (1) Accept the risk, if the activity leading to the risk cannot be terminated without any major loss to the company and/or the ALE is not too high

    OR
  - (2) Avoid the risk, if the ALE is too high and/or the activity leading to the risk can be terminated without any major loss to the company.
- Residual risk is the risk left for the company to deal with even after implementing a countermeasure.
  - There cannot be any system or environment that is 100% secure.
- Total risk is the risk a company faces if it chooses not to implement any type of safeguard.

Countermeasures to Mitigate STRIDE Threats

- **Spoofing user identity**
  - Use strong authentication.
  - Do not store secrets (for example, passwords) in plaintext.
  - Do not pass credentials in plaintext over the wire.
  - Protect authentication cookies with Secure Sockets Layer (SSL).
- **Tampering with data**
  - Use data hashing and signing.
  - Use digital signatures.
  - Use strong authorization.
  - Use tamper-resistant protocols across communication links.
  - Secure communication links with protocols that provide message integrity.
- **Repudiation**
  - Create secure audit trails.
  - Use digital signatures.

Countermeasures to Mitigate STRIDE Threats

- **Information disclosure**
  - Use strong authorization.
  - Use strong encryption.
  - Secure communication links with protocols that provide message confidentiality.
  - Do not store secrets (for example, passwords) in plaintext.
- **Denial of service**
  - Use resource and bandwidth throttling techniques
  - Validate and filter input
- **Elevation of privilege**
  - Follow the principle of least privilege and use least privileged service accounts to run processes and access resources

5. Risk Likelihood Determination

- The likelihood of a risk is a qualitative estimate of how successful an attack (equivalent to the probability of a successful attack) and it will be based on analysis and past experience.
  - Based on the presence of threats and the vulnerabilities that might exist to be exploited.
  - Likelihood can be useful when prioritizing risks and evaluating the effectiveness of potential mitigations.
- Since, the likelihood of a risk is determined based on past experience, the likelihood cannot be accounted for new types of attacks or vulnerabilities that have not yet been discovered.
- The following factors must be considered in the likelihood estimation:
  - The motivation for the attack and the attacker's capability
  - The vulnerability's directness and impact
  - The effectiveness of current controls
- The motivation for the threat and the attacker's capability vary widely.
- Vulnerabilities vary in their directness and the severity of their impacts.
- The effectiveness of security controls can be stronger or weaker depending on the level of difficulty they pose for an intentional attacker or how they can deter an unlikely accidental failure.

Risk Likelihood Determination

The likelihood is a subjective combination of the three qualities: Motivation, Directness of vulnerability and Strength of the Security controls

A quality is said to be weak if it increases the risk and

A quality is said to be strong if it decreases the risk

High	All the three qualities are weak: a threat is highly motivated and sufficiently capable, a vulnerability exists that is severe and direct, and controls to prevent the vulnerability from being exploited are ineffective.
Medium	One of the three qualities is strong, but the others are not. The threat is perhaps not very motivated or not sufficiently capable, the controls in place may be reasonably strong, or the vulnerability might be indirect or not very severe.
Low	Two or more of the three qualities are strong. The threat might lack motivation or capability. Strong controls might be in place to prevent, or at least significantly impede, the vulnerability from being exploited. The vulnerability might be very indirect or very low impact.

6. Risk Impact Determination

- The impact of a risk (if it was to materialize) is determined by the following three aspects:
  - Identification of the threatened assets – The assets threatened by a risk and the nature of what will happen to them must be identified.
    - Common impacts are loss of data, corruption of data, unauthorized or unaudited modification of data, unavailability of data, corruption of audit trails and insertion of invalid data.
  - Impact Locality: All impacts will have a locality in space, time, policy and law.
    - Example: If an encryption key is stored unencrypted, it matters whether that key is in the dynamically allocated RAM of an application on a trusted server, or on the hard disk of a server on the Internet, or in the memory of a client application.

Risk Impact Determination

- Identification of business impact – The primary concern is monetary. In addition, there could be tangible impacts like exposing the business to liability to lawsuits or a race condition in database update operations leading to duplication or loss of data.

  - The company would weigh the return on investment for mitigation vs. the loss to the business because of a software failure. For example, if the worst possible consequence of a software failure is the loss of $10,000 to the business, but it will take $20,000 in labor hours and testing to fix the software, the company may not go for the risk mitigation as it does not make sense financially.
  - The monetary worth of the risks will play a key role in prioritizing them.

7. Risk Exposure Statement

- The risk exposure statement combines the likelihood of the risk occurring with the impact of the risk and provides the overall summary of risk exposure for the organization for each risk.

		Impact of the Risk		
		Low	Medium	High
Likelihood of the Risk	Low	Low	Low	Medium
	Medium	Low	Medium	High
	High	Medium	High	High

Risk Exposure Key

High (H)	**Indicates a strong need for corrective measures. An existing system may continue to operate, but a corrective action plan must be put in place as soon as possible**
Medium (M)	**Indicates that corrective actions are needed and a plan must be developed to incorporate these actions within a reasonable period of time (possibly in a future release)**
Low (L)	**Indicates that the system's decision authorities must determine whether corrective actions are still required or decide to accept the risk.**