Threat Modeling

- Secure Development Process
- Threat Modeling
- Risk Mitigation
- Security Best Practices

In this topic, we will discuss threat modeling. Specifically, we will discuss:

- What threat modeling is.
- The benefits of threat modeling.
- The threat modeling process.
- Categorizing threats by using STRIDE.
- Identifying threats by using attack trees.
- Coding to a threat model.

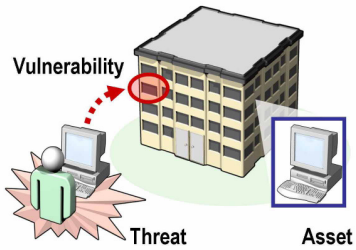
What Is Threat Modeling?

- **Threat modeling is a security-based analysis that:**
  - Helps a product team understand where the product is most vulnerable
  - Evaluates the threats to an application
  - Aims to reduce overall security risks
  - Finds assets
  - Uncovers vulnerabilities
  - Identifies threats
  - Should help form the basis of security design specifications

- Threat modeling is an analysis approach that helps designers to define the security aspect of the design specifications.
- The overriding driver of threat modeling is that you cannot build secure systems unless you understand your threats. The process aims to evaluate the threats to the application with the goal of reducing the consequences of an attack.
- Threat modeling is simple although it does require significant time investment and some practice to get it right. However, time spent during this phase is time well invested. Bugs found during this phase of development are much cheaper to fix than bugs discovered right before shipping!
- Threat modeling has a structured approach that is far more cost efficient and effective than applying security features in a haphazard manner. Without threat modeling, you may not know precisely what threats each feature is supposed to address. It leads naturally to a solid security design which is a part of the formal design specification of your application.
- Threat modeling allows you to systematically identify and rate the threats that are most likely to affect your system.
- An *asset* is also referred to in threat parlance as an attack target. It is anything that is worth protecting.
- A *vulnerability* is a weakness in a system, such as a coding bug or a design flaw.
- A *threat to a system* is a potential event that will have an unwelcome consequence if it becomes an attack.

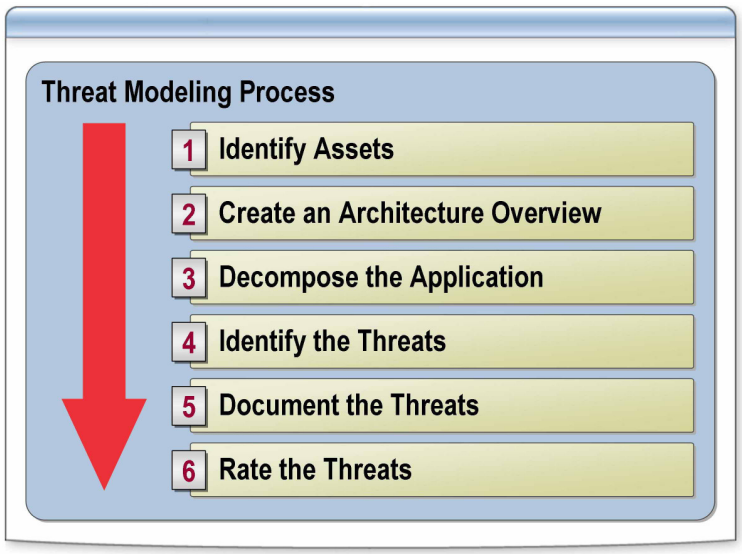
Benefits of Threat Modeling

- Helps you understand your application better
- Helps you find bugs
- Identifies complex design bugs
- Helps integrate new team members
- Drives well-designed security test plans



- Threat modeling helps you:
  - Understand your application better: Team members who spend time analyzing the application in a structured manner inevitably end up with a deeper understanding of how the application works.
  - Find bugs: The additional scrutiny of the threat modeling process leads to the discovery of other, non-security related bugs.
  - Identify complex design bugs: The analytical nature of the process can reveal complex multi-step security bugs where several small failures combine to cause a major failure. This kind of vulnerability may be missed by unit testing performed by the developer and the majority of test plans.
  - Integrate new members: Threat modeling is a useful tool for new team members to become familiar with the architecture of the product.
- Threat modeling should drive your security test plan design. Testers should test against the threat model, which will help them develop new test tools and procedures.

The Threat Modeling Process



A threat modeling process requires several steps:

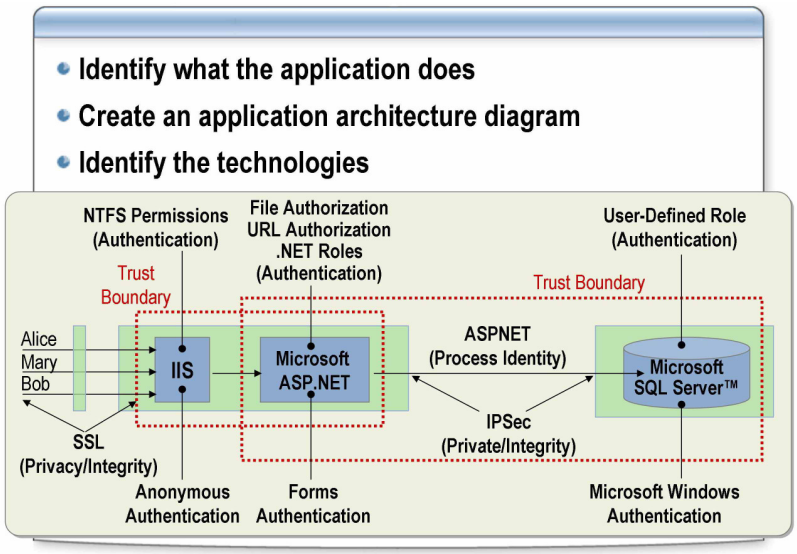
1. *Identify assets*. Identify the valuable assets that your systems must protect.
2. *Create an architecture overview*. Use simple diagrams and tables to document the architecture of your application, including subsystems, trust boundaries, and data flow.
3. *Decompose the application*. Decompose the architecture of your application, including the underlying network and host infrastructure design, to create a security profile for the application. This security profile helps to uncover vulnerabilities in the design, implementation, or configuration of your application.
4. *Identify the threats*. Keeping the goals of an attacker in mind, and with knowledge of the architecture and potential vulnerabilities of your application, identify the threats that could affect the application.
5. *Document the threats*. Document each threat using a common threat template that defines a core set of attributes to capture for each threat.
6. *Rate the threats*. Rate the threats to prioritize and address the most significant threats first. These threats present the biggest risk. The rating process weighs the probability of the threat against damage that could result should an attack occur. It might turn out that certain threats do not warrant any action when you compare the risk posed by the threat with the resulting mitigation costs.

Threat Modeling Process – Step 1: Identify Assets

- **Build a list of assets that require protection, including:**
  - Confidential data, such as customer databases
  - Web pages
  - System availability
  - Anything else that, if compromised, would prevent correct operation of your application

- The first stage of the threat modeling process is to identify the assets you need to protect.
- An *asset* is a system resource of value, such as the data in a database or on the file system.
- You must build a list of assets that require protection including:
  - Confidential data, such as customer databases.
  - Web pages.
  - System availability.
  - Anything else that, if compromised, would prevent correct operation of your application.

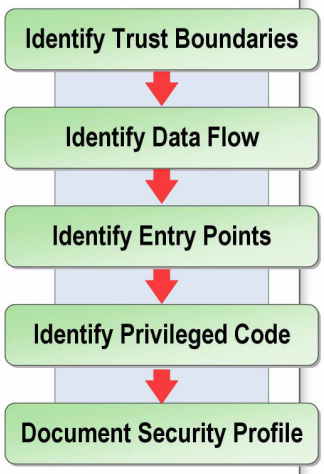
Threat Modeling Process – Step 2: Create An Architecture Overview



- The purpose of the architecture overview is to document the function of your application, its architecture and physical deployment configuration, and the technologies that form part of your solution.
- There are three main parts to the application overview:
  - *Identify what the application does*. Document use cases to help you and others understand how your application is supposed to be used. This also helps you determine how it can be misused. Use cases put application functionality in context.
  - *Create an architecture diagram*. This describes the composition and structure of your application and its subsystems as well as its physical deployment characteristics. Depending on the complexity of the application, this may need to split into several diagrams that focus on specific areas
  - *Identify the technologies*. This helps you focus on technology-specific threats later in the process. It also helps you determine the correct and most appropriate mitigation techniques.

Threat Modeling Process – Step 3: Decompose the Application

- Break down the application
- Create a security profile based on traditional areas of vulnerability
- Examine interactions between different subsystems
- Use DFD or UML diagrams



- Identify the trust boundaries that surround each of the assets in your application. For each subsystem, consider whether the upstream data flows, user input or calling code is trusted, and if not, consider how the data flows and input can be authenticated and authorized. Also consider server trust relationships.
- Analyze the data flow between different subsystems, paying particular attention to data flows crossing trust boundaries.
- Carefully consider all entry points to your application – such as Web pages or socket servers – as these are potential routes for attack. Also consider entry points on internal subsystems or components. Although these entry points may only exist to allow internal communication between components in your application, they could also serve as points of attack, if the attacker manages to bypass the normal "front door" security controls.
- Consider code that accesses secure resources such as DNS servers, directory services, environment variables, event logs, file systems, message queues, performance counters, printers, the registry, sockets, and Web services. Any code that demands special privileges or which works with the security system (for example .NET code access security) demands special vigilance.
- Create a security profile for the application by documenting the design and implementation approaches for input validation, authentication, authorization, configuration management, and the remaining areas where applications are most susceptible to vulnerabilities.
- Data flow diagrams (DFDs) are a useful tool for decomposing an application into its key components. They provide the basis of a good technique for illustrating the flow of data between processes. An alternative is UML activity diagrams, which are similar in purpose, although focus more on the transfer of control between processes. The guiding principle for DFDs is that an application or a system can be decomposed into subsystems, and subsystems can be decomposed into still lower-level subsystems. Ignore the inner workings of the application, but focus on the scope of the application, not functional details. Decompose the application down only two, three or four levels deep – just deep enough to understand the threats to the application. Do not fail in the threat modeling process by analyzing too deeply and losing sight of the original objective.

Threat Modeling Process – Step 4: Identify the Threats

- **Assemble team**
- **Identify threats**
  - Network threats
  - Host threats
  - Application threats

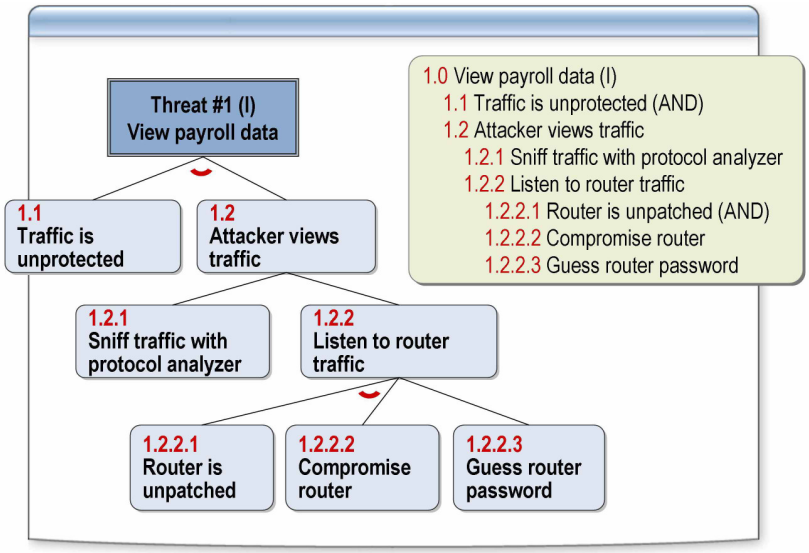
- In this stage of the threat modeling process, you identify threats to your system assets. The best way to do this is to assemble a team consisting of members of the development and test teams to conduct an informed brainstorming session in front of a whiteboard.
- There are three types of threats that you need to identify. You need to:
  - *Identify network threats*: Analyze the network topologies and look for vulnerabilities. Ensure your network is not vulnerable to incorrect device and server configuration.
  - *Identify host threats*: Look for vulnerabilities through poor configuration of hosts security. Consider patch levels and updates, services, protocols, accounts, files and directories, shares, ports, and auditing and logging.
  - *Identify application threats*: Scrutinize the security profile you created in the previous stage of the threat modeling process. Focus on application threats, technology-specific threats, and code threats. Look for things such as poor input validation, passing authentication credentials over unencrypted network connections, or using weak password or account policies.

Threat Modeling Process – Identify the Threats by Using STRIDE

Types of threats	Examples
Spoofing	• Forging e-mail messages • Replaying authentication packets
Tampering	• Altering data during transmission • Changing data in files
Repudiation	• Deleting a critical file and deny it • Purchasing a product and deny it
Information disclosure	• Exposing information in error messages • Exposing code on Web sites
Denial of service	• Flooding a network with SYN packets • Flooding a network with forged ICMP packets
Elevation of privilege	• Exploiting buffer overruns to gain system privileges • Obtaining administrator privileges illegitimately

- Categorizing the threats that hackers pose has many benefits. For example, by categorizing the threats, you can identify and develop security policies that apply to a range of threats in a category and not just to individual threats.
- The STRIDE categories are:
  - *Spoofing*: Illegally accessing and then using another user's authentication information, such as user name and password. For example, a hacker can bypass authorization by using SQL injection and assuming the "sa" role.
  - *Tampering with Data*: Maliciously modifying data. An example would be making unauthorized changes to persistent data such as data in a database or alternating data as it flows between two computers over an open network, such as the Internet.
  - *Repudiation*: Threats associated with users denying (repudiating) an action when other parties do not have any way of proving otherwise.
  - *Information disclosure*: Exposing information to unauthorized individuals. An example would be reading data in transit between two companies. Another example is using cross-site scripting to steal somebody's cookies.
  - *Denial of service*: Denying service to valid users. An example would be making a Web server temporary unavailable or unusable. Another example is exploiting a buffer overrun that causes a server to restart.
  - *Elevation of privilege*: An unprivileged user gaining privileged access and thereby having sufficient access to compromise or destroy the entire system. An example would be exploiting a buffer overrun to gain a command shell and adding the hacker to the Administrators group.

Threat Modeling Process – Identify the Threats by Using Attack Trees



- Building threat trees is an effective method for determining computer system security issues. The idea behind threat trees is that an application is composed of threat targets and that each target could have vulnerabilities that when successfully attacked could compromise the system.
- The threat tree describes the decision-making process an attacker would go through to compromise the software component.
- The decomposition process has helped you to identify all the system components. You then identify potential threats to each component. Threat trees then help you decide how that threat could manifest itself.
- The way to write a threat tree is to identify goals and sub-goals of an attack, as well as what must be done so that the attack succeeds.
- The threat tree on this slide shows the ways in which an attacker could view another user's confidential payroll data.
- Although threat trees communicate data well, they can become cumbersome when building large threat models. An alternative way of representing the threat tree is using an outline, as illustrated on the slide.

Threat Modeling Process – Step 5: Document the Threats

- Document threats by using a template:

Threat Description	Injection of SQL Commands
Threat target	Data Access Component
Risk	
Attack techniques	Attacker appends SQL commands to user name, which is used to form a SQL query
Countermeasures	Use a regular expression to validate the user name, and use a stored procedure with parameters to access the database

- Leave Risk blank (for now)

- Document all the threats you identify.
- You must include the threat target and threat description. The Risk is left blank for now, and is completed in the final stage of the process.
- You may want to include the attack techniques, which can also highlight the vulnerabilities exploited, and the countermeasures that are required to address the threat.

Threat Modeling Process – Step 6: Rate the Threats

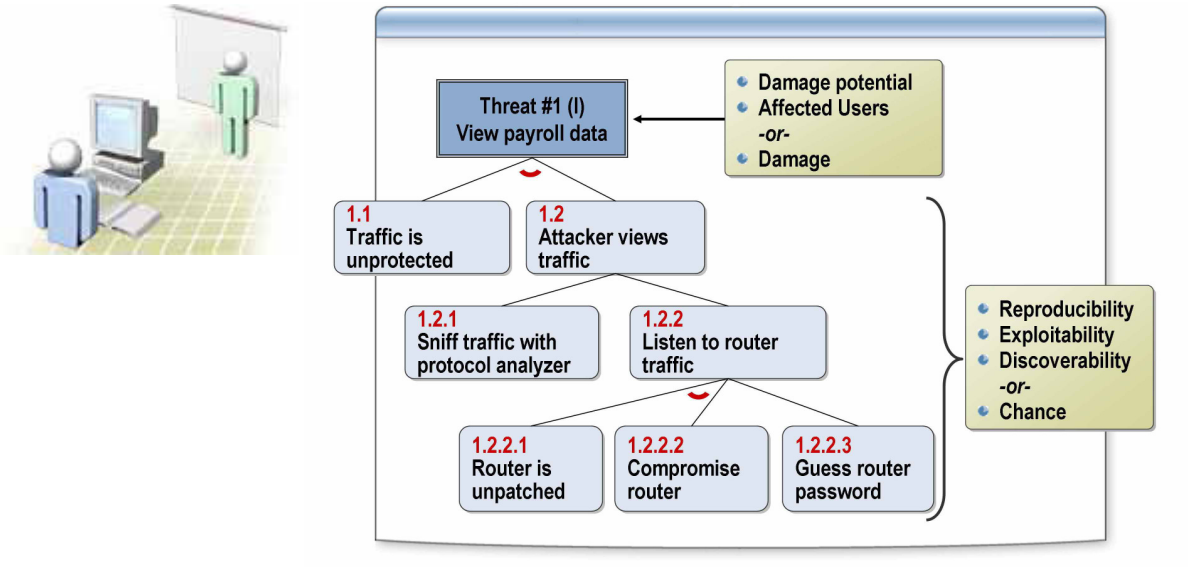
- **Use formula:**

  Risk = Probability * Damage Potential

- **Use DREAD to rate threats**
  - Damage potential
  - Reproducibility
  - Exploitability
  - Affected users
  - Discoverability

- At this step, you can fill in the risk data that you left blank in step 5.
- After you document your threats, you need a way to determine which threats are the most dangerous. To do this, you need a way to rank them.
- You can calculate risk using a 1-10 scale for probability (where 1 is not probable, 10 is very probable), and 1-10 for damage potential (1 – minimal damage, 10 – catastrophe). This results in a risk in the range of 1 – 100. You can divide the scale into three bands to generate a High, Medium and Low risk rating.
- This way of assessing risk is too simplistic for some, so at Microsoft, the DREAD model is used to refine this calculation to add a dimension that assesses what the impact of a security threat really means.
  - *Damage potential*: How much damage can the threat inflict on the system?
  - *Reproducibility*: How difficult is it to reproduce the threat?
  - *Exploitability*: How easy is it for hackers to succeed in exploiting a vulnerability?
  - *Affected users*: How many users will be affected?
  - *Discoverability*: How difficult is it to discover the threat?
- Applications and companies use different criteria for different elements of the DREAD model. The formula used to calculate the rank is not as important because the values derived are relative to one another. You just need to be consistent.

Threat Modeling Process – Example: Rate the Threats



- DREAD provides a further refinement of the simple Risk = Damage * Chance formula.
  - D (Damage Potential) and A (Affected users) are analogous to Damage.
  - R (Reproducibility), E (Exploitability) and D (Discoverability) are analogous to Chance.
- You can use this strategy to prioritize risks. After you calculate the priority for each identified security risk, you can prioritize the risks and determine a management strategy based on the priority value.
- Within each DREAD category, use a score of 3 for high risk, 2 for medium risk, and 1 for low risk.
- Assess each threat in your threat tree and assign the risk rating for each of the DREAD categories and add up the result.
- For example, the SQL Injection threat identified earlier could have a score of D – 3, R – 3, E – 3 , A – 3, D – 2. This gives a total score of 14.
- Across the total possible range of 5 – 15, 12 – 15 may be considered high risk, 8 – 11 as medium risk, and 5 – 7 as low risk.

Coding to a Threat Model

- **Use threat modeling to help**
  - Determine the most “dangerous” portions of your application
  - Prioritize security push efforts
  - Prioritize ongoing code reviews
  - Determine the threat mitigation techniques to employ
  - Determine data flow

- In summary, threat modeling is a valuable analysis tool which improves the design process. It allows you to identify the most vulnerable parts of your application so you can focus your efforts on those places. It also helps you to identify appropriate techniques for mitigating threats.
- The benefit of threat modeling does not end there. Use threat modeling through the coding process to revisit and retest security decisions made in the design phase and to direct focus in security code reviews.
- The results of the threat modeling process provide the information necessary to select appropriate security technologies and techniques to mitigate the threats that have been identified.
- Data flow diagrams are an effective tool for identifying data flows and are particularly useful for identifying all inputs to your application. Any inputs are a particular source of security vulnerabilities.

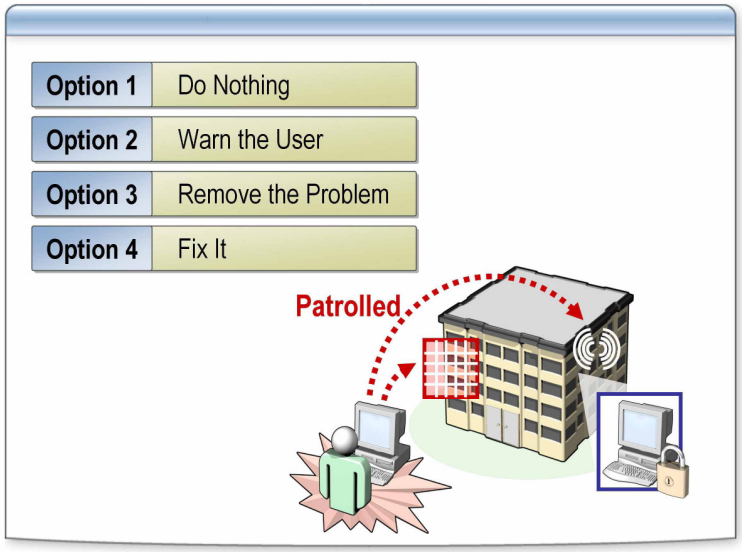
Risk Mitigation

- Secure Development Process
- Threat Modeling
- Risk Mitigation
- Security Best Practices

In this topic, we will examine how to mitigate the risks that the threat modeling process has uncovered. Specifically, we will discuss:

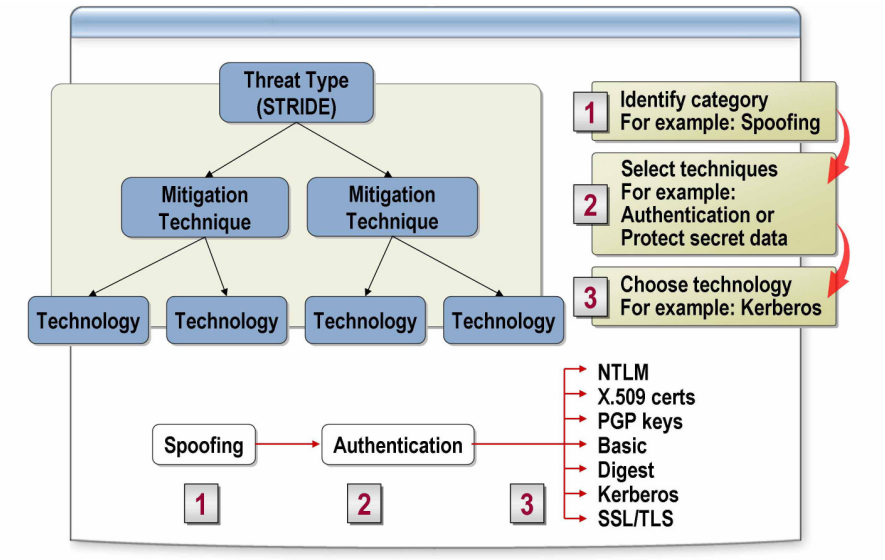
- Risk mitigation options.
- The risk mitigation process.
- Some sample mitigation techniques.

Risk Mitigation Options



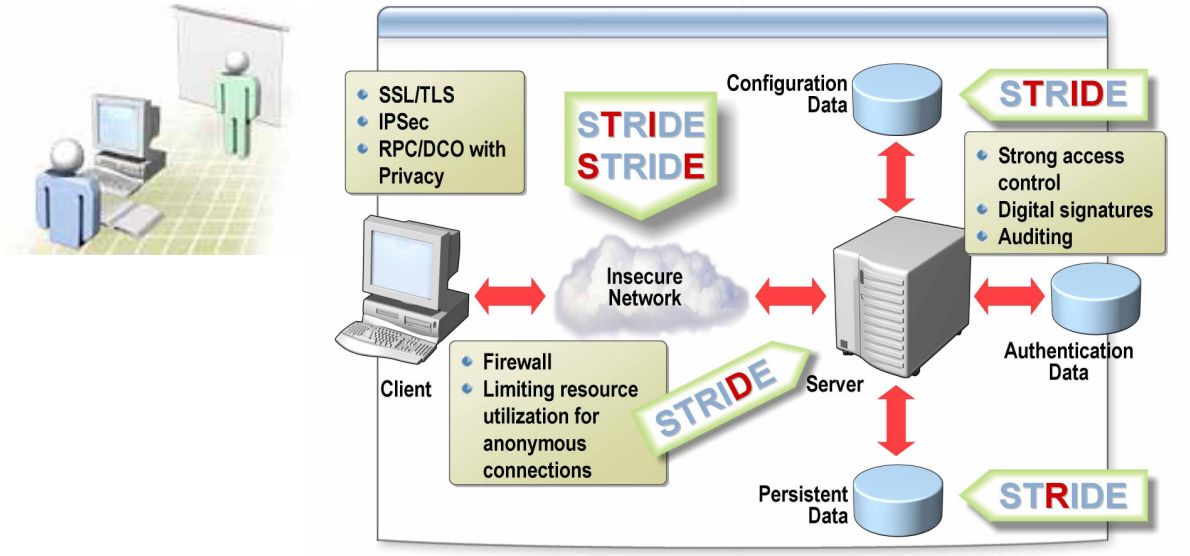
- Mitigation techniques protect assets by securing vulnerabilities in a system and preventing threats from being realized.
- When considering risks and how to mitigate them, you have four choices:
  - Option 1: Do nothing. You may decide to leave a low security risk without fixing it. This is rarely a good choice as the fault will lie latent in the application and it is likely that it will be discovered and you will be forced to fix it. This is bad for your users and bad for your business reputation.
  - Option 2: Warn the user. Inform the user of the problem and allow them to decide whether to use the feature. However, you cannot expect users to have sufficient detailed knowledge of the system or the nature of the limitation to make a decision.
  - Option 3: Remove the problem. If there is no time to fix the problem, you should consider pulling the feature from the product before shipping. It is better to fix it for the next release than allow it to go out and compromise your user's computers.
  - Option 4: Fix it. Select the technologies required to fix the problem.

Risk Mitigation Process



- Determining how to allay the risks identified by the threat modeling process is a three-step process.
  1. First, identify the category of threat, using STRIDE.
  2. Next, determine which techniques can help.
  3. Then, decide on appropriate technologies.
- For example, if the threat modeling process has identified a threat of spoofing identity, the technologies that can mitigate this include authentication or protecting secret data such as security credentials. If authentication is selected as the appropriate mitigating technique, the technologies you can use for authentication include NTLM, X.509 certificates, PGP (Pretty Good Privacy) Keys, Digest authentication, Kerberos, and SSL/TLS.

Sample Mitigation Techniques



- Some of the mitigating techniques for each of the STRIDE threat categories are shown on the slide. In general, the mitigation techniques for each of these threats include:
  - Spoofing:
    - Use appropriate authentication.
    - Protect secret data.
    - Do not store secrets.
  - Tampering with data:
    - Use appropriate authorization.
    - Use hashes or message authentication codes (MAC).
    - Use digital signatures.
    - Use tamper-resistant protocols such as SSL/TLS.
  - Repudiation:
    - Use digital signatures.
    - Use timestamps.
    - Use audit trails.

- Information disclosure:
  - Use authorization.
  - Use privacy-enhanced protocols.
  - Use encryption.
  - Protect secrets.
  - Do not store secrets.
- Denial of service:
  - Use appropriate authentication.
  - Use appropriate authorization.
  - Use filtering.
  - Use throttling.

Security Best Practices

- Secure Development Process
- Threat Modeling
- Risk Mitigation
- Security Best Practices

Fortunately, there are effective techniques developers can use to prevent attacks. Some of these can mitigate a wide range of attacks and should be used when developing any application on any platform. Some attacks can be mitigated by specific methods. In this topic, we will discuss a common set of techniques that developers can apply to mitigate security threats. Specifically, we will discuss:

- Running with least privilege.
- Reducing the attack surface.
- Validating user input.
- The defense in depth model.
- Not relying on obscurity.
- Not storing secret information.
- Using DPAPI.
- Failing intelligently.
- Testing security.
- Learning from mistakes.