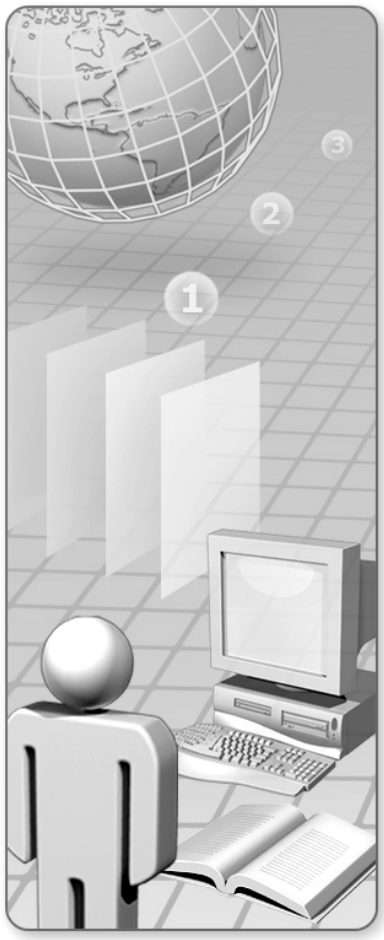
Microsoft Learning

Session: Writing Secure Code – Best Practices

Contents



Microsoft

What We Will Cover

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

In this session, we will focus on how to perform threat modeling during the design and coding stages of a software development project and the best practices for writing secure code. Specifically, we will discuss:

- The secure development process.
- Threat modeling.
- Risk mitigation.
- Security best practices.

Session Prerequisites

- Development experience with Microsoft Visual Basic® , Microsoft Visual C++® , or C#

Level 200

Secure Development Process

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

In this topic, we will review the essential components of a successful secure development process. Specifically, we will discuss:

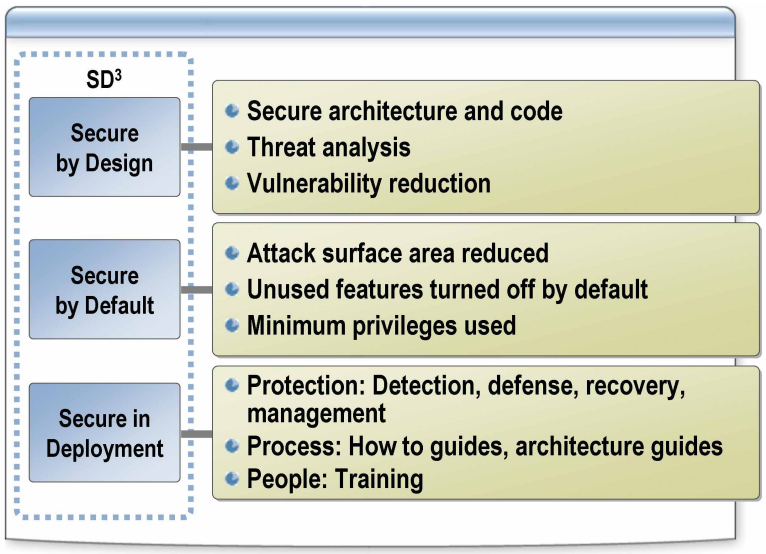
- Improving the application development process.
- The SD3 security framework.
- The secure product development timeline.
- Secure by design.

Improving the Application Development Process

- **Consider security**
  - At the start of the process
  - Throughout development
  - Through deployment
  - At all software review milestones
- **Do not stop looking for security bugs until the end of the development process**

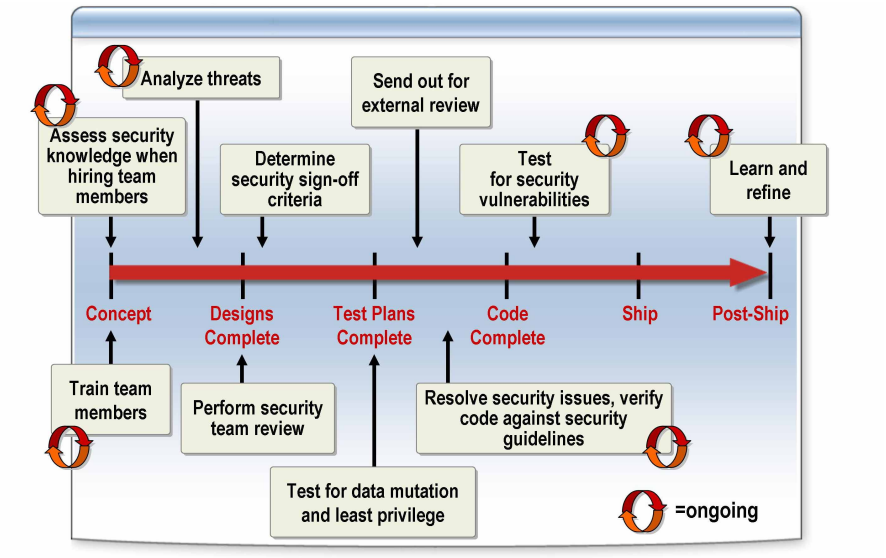
- Experiences at Microsoft and throughout the software industry have shown that successful delivery of trustworthy software products requires process improvements throughout the application development process.
- Often, security is considered as an afterthought, with security review of the product left until late in the development process. Although implementing security measures at this late stage does yield some benefits, it cannot substitute for considering security as an integral part of the process right from the beginning of the design phase, through development, to deployment.
- In short, you must build security into your application development lifecycle. For step-by-step guidance on how to achieve this, see Fast Track – How to Implement the Guidance at http://msdn.microsoft.com/library/en-us/dnnetsec/html/THCMFastTrack.asp

The SD3 Security Framework



- The first and most important step towards good security practice is to ensure that security is an integral part of the entire development process.
- The Secure Windows Initiative team at Microsoft has adopted a simple set of strategies called SD3. The SD3 framework has three core concepts: secure by design, secure by default, and secure in deployment. These concepts have shaped the development process to help deliver secure systems.
  - *Secure by design* means that you have taken appropriate steps to ensure that the overall design of the product is secure from the outset. Include threat modeling at the design phase and throughout the project to identify potential vulnerabilities. Use secure design, coding and testing guidelines.
  - *Secure by default* means that the product is shipped and that it is secure out of the box. If features are optional, turn them off by default. If a feature is not activated, then an attacker cannot use it to compromise your product. Ensure that only the least amount of privilege is required by user accounts to run your application. That way, a compromise has less serious consequences than if an attacker is able to run malicious code under an account with administrator privileges. Ensure that effective access controls are in place for resources.
  - *Secure in deployment* means that the system is maintainable after installation. If a product is hard to administer, it makes it more difficult to maintain protection against security threats. Ensure that users are educated to use the system in a secure manner. If a security vulnerability is discovered and a patch is necessary, ensure that the fix is fully tested internally and issued in a timely fashion.

Secure Product Development Timeline



**********ILLEGAL FOR NON-TRAINER USE**********

- Activities that are shown as ongoing start at the points as illustrated on the product development timeline. However, these activities continue on throughout the project lifecycle. For example, hiring of team members may not occur only at the start of the project, but happens at later stages as well.
- At the commencement of a project, during interviews with potential team members, ask security questions to assess the level of security knowledge. This will help you to select security conscious developers, and help you to assess the need for further security training.
- During the design phase and later in the project, threat modeling is a valuable tool for identifying security defects. Threat modeling is examined further in the next section of this presentation.
- During development, developers should be trained to recognize common security defects and how to code to avoid them. There should be special focus on security during code reviews.
- Test plans must include a significant element of security testing. Testers should devise ways of attacking the software, looking for security defects.
- After the product is shipped, there must be an appropriate plan of response for any security defects that are uncovered. At the least, this should involve education of developers so that the same or similar security issues can be uncovered. It may also involve the timely issuing of patches to existing installations.

Secure By Design

- **Raise security awareness of design team**
  - Use ongoing training
  - Challenge attitudes - "What I don't know won't hurt me" does not apply!
- **Get security right during the design phase**
  - Define product security goals
  - Implement security as a key product feature
  - Use threat modeling during design phase

- Education is the key ingredient of an effective secure development process. You must ensure that all designers and programmers are informed about common security flaws in software so they can avoid repeating past mistakes. Perform regular training to raise security awareness and ensure that security remains a primary concern of all team members.
- Problems discovered late in the development cycle are much more expensive to fix than those discovered early on. This is particularly true for security issues. Be sure to consider the security of your application right at the beginning of the design phase.
- The security goals of the product should be considered with equal (or greater) importance to the other functional goals.
- If security is designed as a product feature, rather than considered as a "value-add" feature to be added on "if there is time," the resulting product will be much more secure than if it is added as an afterthought.
- Threat modeling during the design phase as well as the development phase has proven to be extremely effective in determining the highest level security risks that are posed to the product and how attacks can occur.

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.

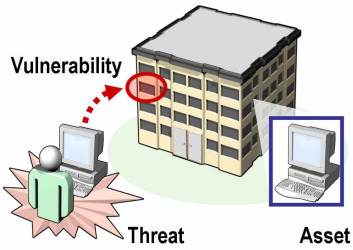
What Is Threat Modeling?

- **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.

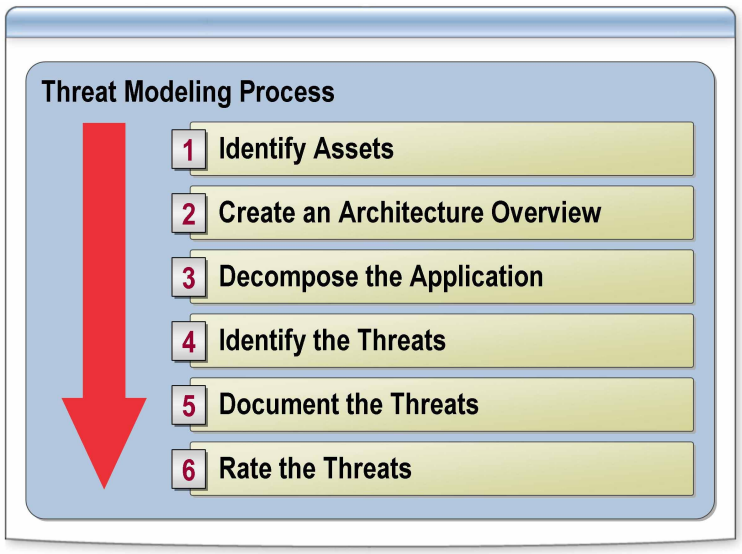
Benefits of Threat Modeling

- 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:

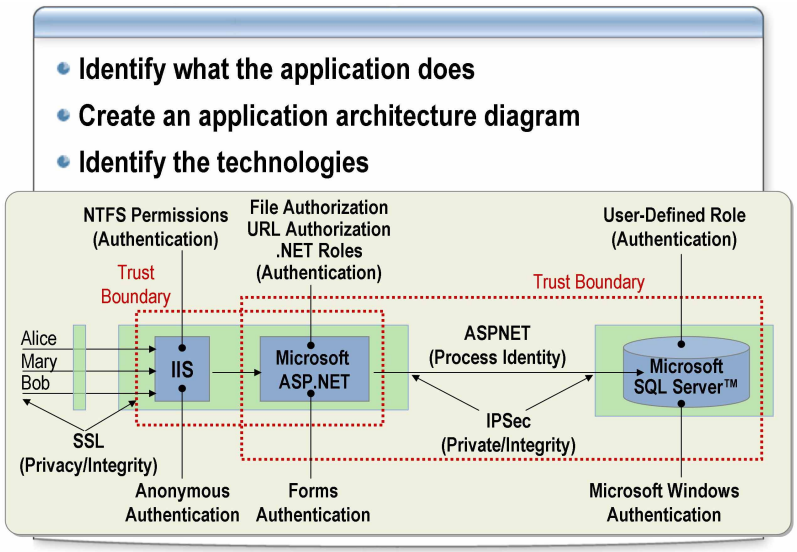
1. *Identify assets*. Identify the valuable assets that your systems must protect.
2. *Create an architecture overview*. Use simple diagrams and tables to document the architecture of your application, including subsystems, trust boundaries, and data flow.
3. *Decompose the application*. Decompose the 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.

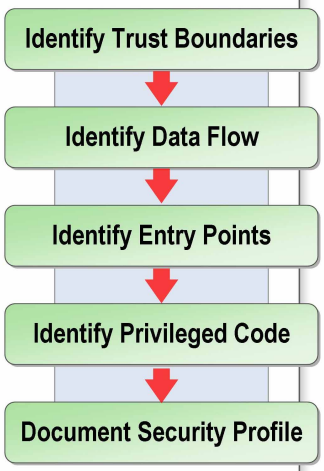
Threat Modeling Process – Step 2: Create An Architecture Overview



- 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

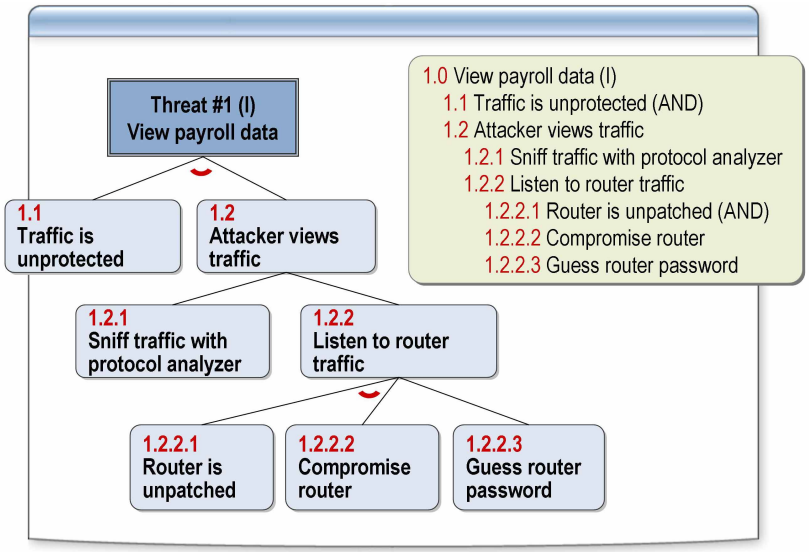
- In this stage of the threat modeling process, you identify threats to your system assets. The best way to do this is to assemble a team consisting of members of the development and test teams to conduct an informed brainstorming session in front of a whiteboard.
- There are three types of threats that 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

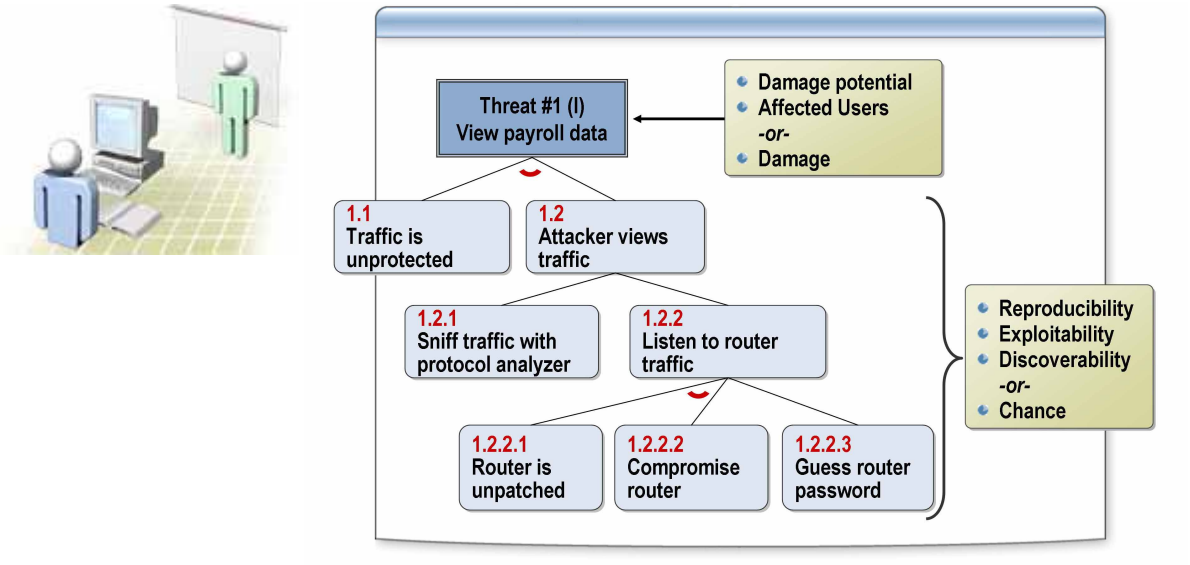
- **Use formula:**

  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.

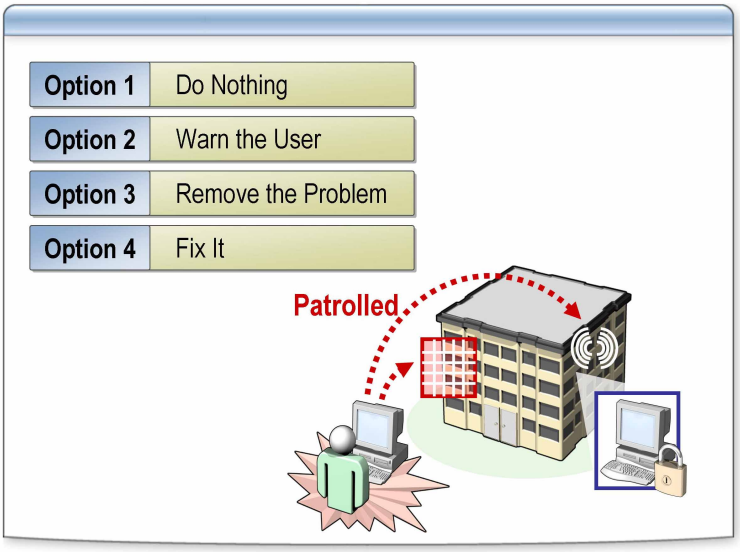
Risk Mitigation

- 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:

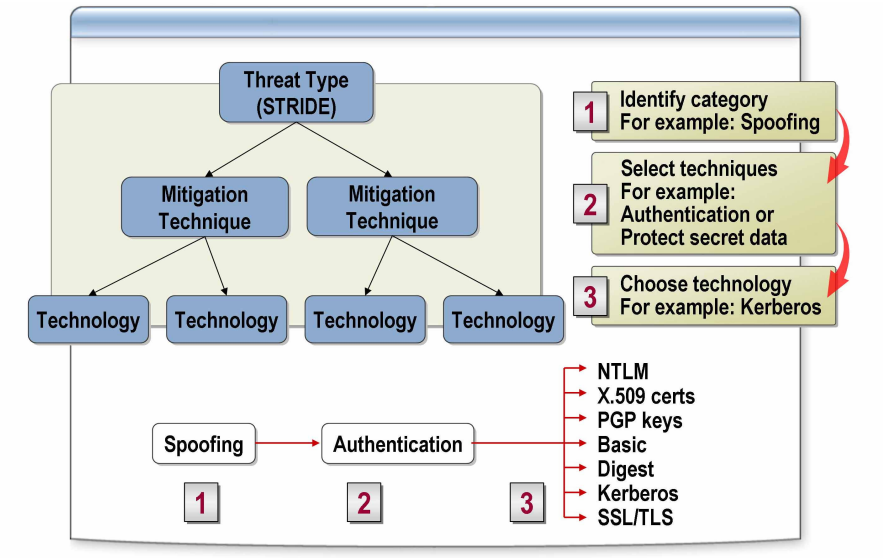
- Risk mitigation options.
- The risk mitigation process.
- Some sample mitigation techniques.

Risk Mitigation Options



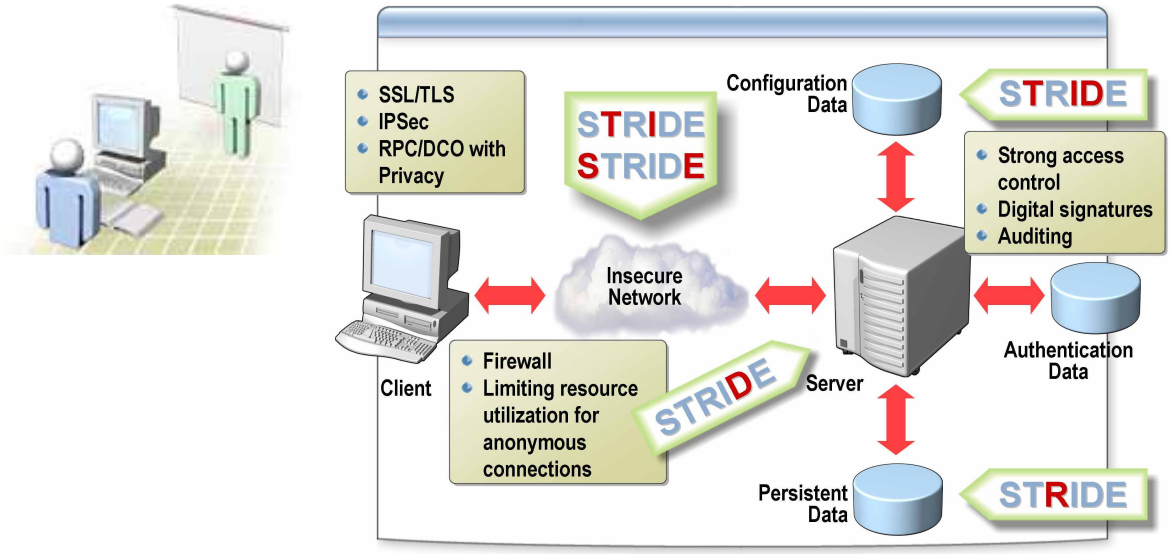
- Mitigation techniques protect assets by securing vulnerabilities in a system and preventing threats from being realized.
- When considering risks and how to mitigate them, you have four choices:
  - Option 1: Do nothing. You may decide to leave a low security risk without fixing it. This is rarely 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:

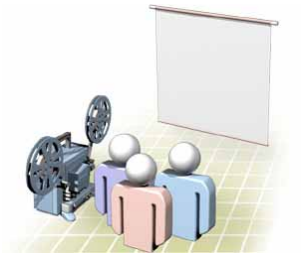
- Running with least privilege.
- Reducing the attack surface.
- Validating user input.
- The defense in depth model.
- Not relying on obscurity.
- Not storing secret information.
- Using DPAPI.
- Failing intelligently.
- Testing security.
- Learning from mistakes.

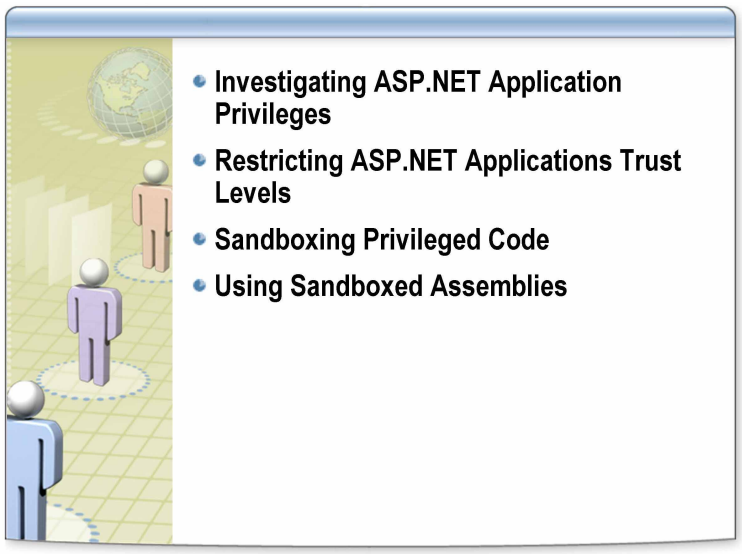
Run with Least Privilege

- **Well-known security doctrine:**
  - "Run with just enough privilege to get the job done, and no more!"
- **Elevated privilege can lead to disastrous consequences**
  - Malicious code executing in a highly privileged process runs with extra privileges too
  - Many viruses spread because the recipient has administrator privileges

- You must always run with just enough privilege to get the job done, and no more. For example, if you have a buffer overrun that runs under the security context of an administrator, a hacker can overwrite the return address and have malicious code execute with those same privileges.
- Many well-known viruses rely on infecting computers by using accounts that have elevated privileges.
- It is easier for developers to create applications that require users with administrator-level access, because there are no permission errors, but this should not be done.
- Many developers develop applications using Administrator privileges "just for now" because it makes development easier. Although the developer may intend to lower the required privileges later on, time pressures frequently mean that this does not happen. This results in applications going live with elevated privileges, because system admin staff discover that "If we do not run as administrator, the application does not work!"

Demonstration 1: ASP.NET Applications Security





In this demonstration, you will show that the Microsoft® ASP.NET worker process runs under a nonprivileged account by default. You will show how the secure developer grants access to an ASP.NET application to access the file system, write to a custom event log, and access the registry.

Investigating ASP.NET Application Privileges

Perform the steps on the LONDON VPC.

1. Log on to the **Administrator** account with a password of **P@ssw0rd**.
2. Open Internet Explorer.
3. Browse to **http://london/SecDev/lowPrivASPNET**.
4. Select the **Write File** option. Click the **Do It** button.
5. If you are prompted by Internet Explorer, click **Yes**.
6. Select the **Write to Event Log** option. Click the **Do It** button to show the security error.
7. Close Internet Explorer.
8. On the **Start** menu, click **Run**.
9. Type **regedt32** and then click **OK**.
10. Navigate to HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Eventlog.
11. Right-click **Eventlog**, and then click **Permissions**.
12. Click **Add**.
13. In the **Object names** box, type **NETWORK SERVICE**
14. Click **Check Names**, and then, click **OK**.

15. Select the **Allow** checkbox for **Full Control**, and then click **OK**.
16. Close the Registry Editor.
17. Open Internet Explorer.
18. Browse to **http://london/SecDev/lowPrivASPNET**.
19. Select the **Write to Event Log** option. Click the **Do It** button to show that it now works.
20. Close Internet Explorer.

Restricting ASP.NET Application Trust Levels

1. Using Notepad, open C:\Courses\DeveloperSecurity\Democode\Websites\LowPrivASPNET\Web.config.
2. Below the <system.web> tag, enter the following:
   <trust level="Medium" originUrl="" />
3. Save the file.
4. Open Internet Explorer.
5. Browse to http://london/SecDev/lowPrivASPNET.
6. Select the **Write File** option. Click the **Do It** button to show the security error.
7. Click **Back**, and then select the **Write to Event Log** option. Click the **Do It** button to show the security error.
8. Close Internet Explorer.

Sandboxing Privileged Code

1. Start Visual Studio .NET.
2. Open the following solution C:\Courses\DeveloperSecurity\Democode\Session03\lowPrivASPNET\lowPrivASPNET.sln.
3. Add a new **C# class library project** to the solution. Call the new project **EventLogger**.
4. Open a Visual Studio .NET 2003 Command Prompt.
5. cd **C:\Courses\DeveloperSecurity\Democode\Session03\lowPrivASPNET\EventLogger**
6. Type **sn.exe -k eventloggerkeypair.snk**.
7. In Visual Studio .NET, edit **AssemblyInfo.cs** for the **EventLogger** project.
8. Modify the AssemblyKeyFile line to:
   [assembly: AssemblyKeyFile(@"..\..\eventloggerkeypair.snk")]
9. Configure the Assembly Version:
   [assembly: AssemblyVersion("1.0.0.0")]
10. Below the AssemblyVersion line, add the AllowPartiallyTrustedCallersAttribute as follows:
    [assembly:AllowPartiallyTrustedCallersAttribute()]
11. Add **using System.Security;** to the top of the AssemblyInfo.cs module for the **EventLogger** project.

12. Right-click **default.aspx** and select **View Code**.
13. Locate the **Button1_Click** method.
14. Within the **Button1_Click** method, locate the **if** block that begin:

```
if (RadioButtonList1.SelectedValue == "Event")
```

15. Select all the code *inside* the braces for this if block ***except*** for the line that reads:

```
TextBox1.Text = "Event log entry written successfully";
```

16. From the **Edit** menu, click **Cut**.
17. Double-click **Class1.cs**.
18. Type **public static string writeEvent()** directly above the **public Class1()** constructor function and press ENTER.
19. Type **{** and press ENTER twice.
20. Type **}** and press ENTER.
21. Place the cursor on the line inside the curly braces **{}** you have just entered, and click **Edit - Paste**.

22. Modify the code you have just pasted to that it reads:

```
public static string writeEvent()
{
    try
    {
                string source = "Event Source";
                string log = "Application";
                string eventText = "Writing to the event log";
                EventLogEntryType eventType = EventLogEntryType.Information;

                //Assert permission
                EventLogPermission eventPerm;
                eventPerm = new
EventLogPermission(EventLogPermissionAccess.Instrument,
                "LONDON");
                eventPerm.Assert();

                //Check to see if the source exists.
                if(!EventLog.SourceExists(source))
                {
        //The keys do not exist, so register your application
        // as a source
                  EventLog.CreateEventSource(source, log);
                }

                //Write to the log.
                    EventLog.WriteEntry(source, eventText, eventType);
                    return "Event log entry written successfully";
                  }
                  finally
                  {
                    CodeAccessPermission.RevertAssert();
                  }
                }
```

23. Add **using System.Diagnostics;** and **using System.Security;** to the top of Class1.cs.

24. Select the **EventLogger** project in the **Solution Explorer** window, and click **Build – Build EventLogger**.

25. Install the assembly in the GAC by switching to the command window, and then typing **cd bin\debug**

26. Type **gacutil -i eventlogger.dll** and press ENTER.

Using Sandboxed Assemblies

1. In the command window, get the public key token of the signed assembly by typing **sn –Tp eventlogger.dll**
2. In Visual Studio .NET, edit **Web.config** in the **lowPrivASPNET** Web application. Change the <compilation> section to:.

```
    <compilation
         defaultLanguage="c#"
         debug="true">
         <assemblies>
                  <add assembly="eventlogger, Version=1.0.0.0,
Culture=neutral, PublicKeyToken=ea...d6"/>
               </assemblies>
    </compilation>
```

3. Replace **ea…d6** with the public key token you found in the previous step.
4. Add a Reference to the Web project. In the **Add Reference** window, click the **Browse** button and navigate to C:\Courses\DeveloperSecurity\Democode\Session03\lowPrivASPNET\EventLogger\bin\debug\ folder. Select **EventLogger.dll** and click **Open**.
5. Click **OK**.
6. Edit default.aspx.cs. In the **Button1_Click** method, modify the code to call the static method in EventLogger, as follows:

```
if (RadioButtonList1.SelectedValue == "Event")
{
               TextBox1.Text = EventLogger.Class1.writeEvent();
}
```

7. Build the solution.
8. If Internet Explorer is running, close Internet Explorer.
9. In the command window, type **iisreset** to restart IIS.
10. Open Internet Explorer.
11. Browse to http://london/SecDev/lowPrivASPNET
12. Select the **Write to Event Log** option. Click the **Do It** button to show that it works again.
13. Close all applications.

Reduce the Attack Surface

- **Expose only limited, well documented interfaces from your application**
- **Use only the services that your application requires**
  - The Slammer and CodeRed viruses would not have happened if certain features were not on by default
  - ILoveYou (and other viruses) would not have happened if scripting was disabled
- **Turn everything else off**

- Hackers commonly exploit weaknesses in platform services, such as the Index Service in IIS, to compromise applications.
- Hackers can also attack all the interfaces that your applications expose.
- By turning all unused platform services off and limiting the number of interfaces exposed by your application, you can limit the attack vectors that hackers can exploit.
- Microsoft Windows Server™ 2003, for example, installs with the majority of services turned off. Administrators must explicitly enable services. This extends to features such as IIS and ASP.NET, which are disabled out of the box.

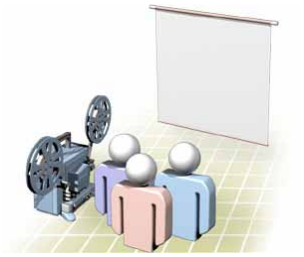
Do Not Trust User Input

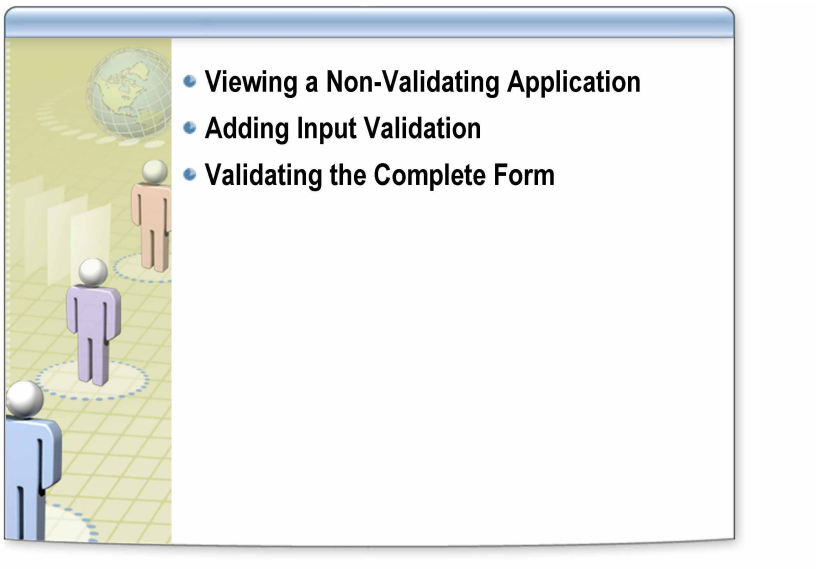
- **Validate all input**
  - Assume all input is harmful until proven otherwise
  - Look for valid data and reject everything else
- **Constrain, reject, and sanitize user input with**
  - Type checks
  - Length checks
  - Range checks
  - Format checks

```
Validator.ValidationExpression =
"\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*";
```


- Consider all user input as harmful until proven otherwise.
- Look for valid input and deny everything else. Do not do the opposite and look for invalid data, reject it, and then let everything else in.
  - Hackers can circumvent client-side validation. Consider performing validation often and close to the threats. For example, validate data on the server before committing it to a database.
  - Invalidate input that is too large.
  - Invalidate input that contains dangerous characters or keywords, such as scripting elements (<script>, <object>), reserved SQL characters and keywords (- -, INSERT, xp_cmdshell), or file traversal characters (..\).
- Use the mantra: Constrain, Reject, and Sanitize.
  - One of the most powerful ways to constrain data input is through validation controls and regular expressions.
  - The example regular expression (\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*) checks that the input string is of the correct form for an Internet e-mail address.

Demonstration 2: Windows Forms Validation





In this demonstration, you will show how to add input validation code to a Windows Form application. ASP.NET developers have the benefit of having a large number of controls to help with input validation. In Windows Forms, the developer must write more code to achieve the same result. However, it is still easy to implement.

Viewing a Non-Validating Application

Perform the steps on the LONDON VPC.

1. If you are not already logged on, log on to London by using the **OrdinaryUser** account with a password of **Passw0rd**.
2. Using Windows Explorer, navigate to the C:\Courses\DeveloperSecurity\DemoCode\Session03\WindowsFormsValidation\bin folder and double-click **WindowsFormsValidation.exe**.

   If file extensions are not shown, this will simply be **WindowsFormsValidation**
3. Do not enter any name input and select a date on a Sunday (weekends should not be allowed) to show that no validation is being performed.
4. No errors occur. Close the application

Adding Input Validation

1. In Visual Studio .NET, open C:\Courses\DeveloperSecurity\DemoCode\Session03\ WindowsFormsValidation\ WindowsFormsValidation.sln.
2. Open Form1.vb in design view.
3. From the **Toolbox**, drag an **ErrorProvider** control onto the form. Visual Studio .NET places it in the Component Box under the form.
4. Right-click the form, and then click **View Code**.
5. Select **nameTextBox** from the left-hand dropdown box and Validating from the right-hand dropdown box.
6. Inside the **nameTextBox_Validating** method, type the following:

```
    If (nameTextBox.Text = "") Then
        ErrorProvider1.SetError(nameTextBox, _
            "Please enter your Name")
    Else
        ErrorProvider1.SetError(nameTextBox, "")
    End If
```

7. Select **eventDateTimePicker** from the left-hand dropdown box and **Validating** from the right-hand dropdown box.
8. Inside the **eventDateTimePicker_Validating** method, type the following:

```
        If ((eventDateTimePicker.Value.DayOfWeek = _
              DayOfWeek.Sunday) _
          Or (eventDateTimePicker.Value.DayOfWeek = _
              DayOfWeek.Saturday)) Then
            ErrorProvider1.SetError(eventDateTimePicker, _
                "Please select a weekday")
        Else
            ErrorProvider1.SetError(eventDateTimePicker, "")
        End If
```

9. Build and run the application. Show that the user must enter a name and that the date must be a weekday.

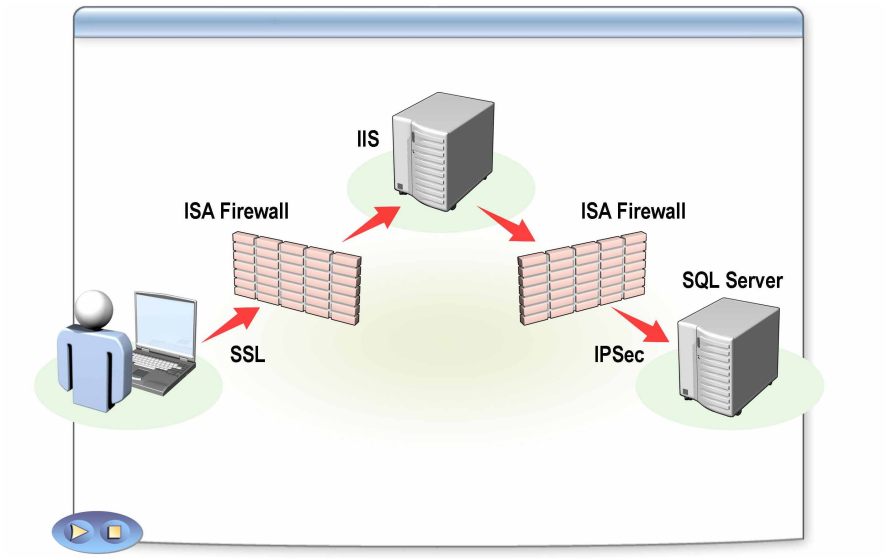
Validating the Complete Form

1. Go to Design view of the form, and then double-click the button.
2. Inside the **Button1_Click** method, type the following:

```
If ErrorProvider1.GetError(nameTextBox) <> "" _
  OR ErrorProvider1.GetError(eventDateTimePicker) <> "" Then
    MessageBox.Show("Please enter valid data")
End If
```

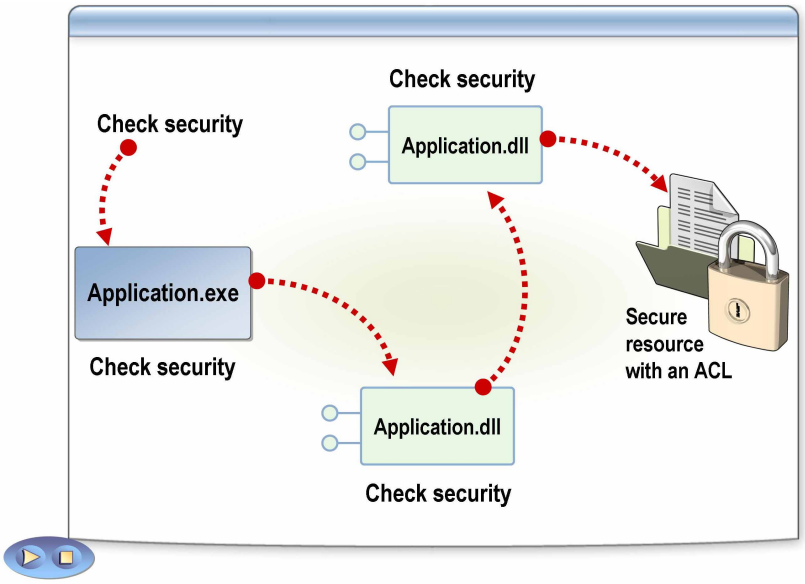
3. Build and run the application. Show that the warning message displays until the user enters a name and a date during the week.
4. Close all applications.

Defense in Depth (1 of 3) – Use Multiple Gatekeepers



- People make mistakes, which means that one layer of defense is not good enough. By building multiple layers of defense, you can increase the security of a system.
- Software architects should design systems with multiple defense barriers. For example, if you are exposing sensitive data in a server running SQL Server on a Web page, you can place a firewall in front of the Web server to prevent unauthorized access to it. In addition, you can require clients to connect by using Secure Sockets Layer (SSL). You can strengthen the security perimeter by placing an additional firewall in front of the server running SQL Server and requiring all the computers on the internal network to use Internet Protocol security (IPSec).
- Threat modeling defines the location and importance of defensive strategies.

Defense in Depth (2 of 3) – Apply Appropriate Measures for Each Layer



- In addition to implementing multiple layers of security in systems, you can also add layers of security in applications. For example, you can have an application .exe file authenticate and authorize users. Then, each supporting application .dll file can also authenticate and authorize users.
- If only the application .exe file authenticates and authorizes users, then a hacker might circumvent it and call the application .dll file independently.
- As a last line of defense, be sure to use access control lists (ACLs) to secure files and folders.
- Increased security often comes with a cost in performance. Determine the depth of defense for software components based on threat modeling and prioritization that is performed during the envisioning phase.

Defense in Depth (3 of 3) – Use Strong ACLs on Resources

- Design ACLs into the application from the beginning
- Apply ACLs to files, folders, Web pages, registry settings, database files, printers, and objects in Active Directory
- Create your own ACLs during application installation
- Include DENY ACEs
- Do not use NULL DACLs

- During the design phase of a software project, document all of the ACL requirements of the application and develop them into the application.
- You can use ACLs to restrict access to a variety of items including files, folders, Web pages, registry settings, database files, printers, and objects in Active Directory® directory service.
- To protect sensitive files and folders for your application, programmatically create ACLs during installation.
- Be explicit, and use the DENY ACE often.
- A NULL discretionary access control list (DACL) is a way of granting all access for an object to all users, including hackers. For example, a hacker can change a NULL DACL to Everyone Deny Access. Other dangerous access control entries (ACEs) include WRITE, WRITE_DAC, WRITE_OWNER, and DELETE.

Do Not Rely on Security by Obscurity

- Do not hide security keys in files
- Do not rely on undocumented registry keys
- Always assume an attacker knows everything you know

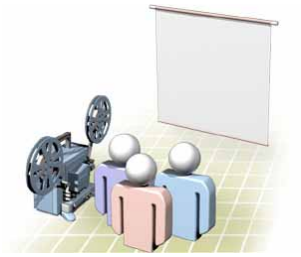
- One issue with encryption is the need to store encryption keys securely.
- You may be tempted to hide keys inside text files, html files, or data files and use to extract the required data when required. However, it is very easy for an attacker to find hidden information. Concealing secrets can contribute to overall security, but it must never be your primary security mechanism.
- You should always assume that an attacker knows everything you do, and has access to all source code and diagrams.

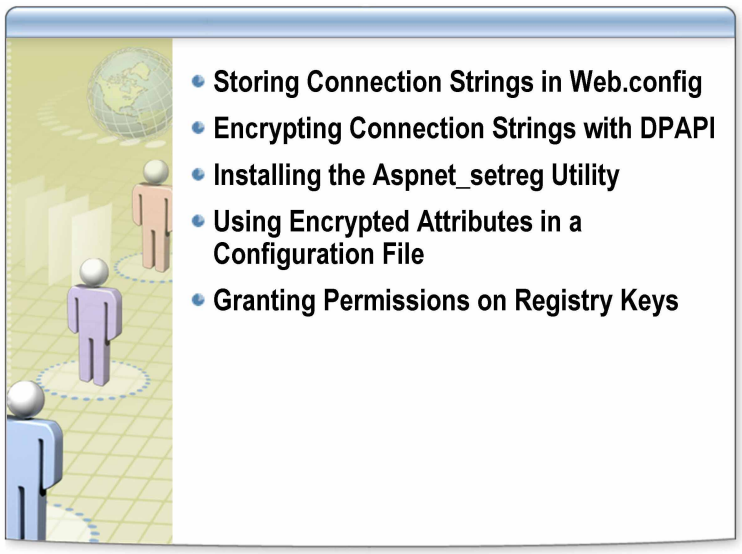
Use Data Protection API (DPAPI) to Protect Secrets

- **Two DPAPI functions:**
  - CryptProtectData
  - CryptUnprotectData
- **Two stores for data encrypted with DPAPI:**
  - User store
  - Machine store

- DPAPI is an encryption application programming interface (API) that ships with Microsoft Windows Server 2003 and Microsoft Windows XP. DPAPI is a powerful encryption class built into the operating system that uses a user's password to derive user-specific or computer-specific encryptions.
- The Data Protection API can be used to protect security-sensitive data, such as database connection strings and service account credentials. It is password-based, and can store data in either a user or machine store.
- There are two ways to use it: the user store approach, where you protect data such that only the data owner can access it, or the machine store approach, where any user on the machine can access it. You select the latter case by setting the CRYPTPROTECT_LOCAL_MACINE flag in the dwFlags field when calling the API.
- The user store approach affords an additional layer of security because it limits who can access the secret. Only the user who encrypts the data can decrypt the data. However, use of the user profile requires additional development effort when DPAPI is used from an ASP.NET Web application. You need to take explicit steps to load and unload a user profile. ASP.NET does not automatically load a user profile.
- The machine store approach is easier to develop because it does not require user profile management. However, unless an additional entropy parameter is used, it is less secure because any user on the computer can decrypt data. (Entropy is a random value designed to make deciphering the secret more difficult.) The problem with using an additional entropy parameter is that this must be securely stored by the application, which presents another key management issue. With this approach, you should also protect the encrypted data produced by DPAPI with an ACL when you store it in persistent storage.

Demonstration 3: DPAPI





**********ILLEGAL FOR NON-TRAINER USE**********

This demonstration shows how to use DPAPI to encrypt data in Web.config, such as a connection string. It also shows how to use the Aspnet_setreg utility to encrypt user credentials in Web.config.

IMPORTANT: This demo uses user accounts that are set up in Demo 4: Authorization Techniques using the Trusted Subsystem Model in the session *Essentials of Application Security*. If you have not run that demonstration, you must perform the following tasks to prepare for this demonstration:

- Create a local account on London called **OrderReaders** with password **Passw0rd**. Ensure that the **User must change password at next logon** checkbox is **not** selected. Ensure that the **Password never expires** checkbox **is** selected.
- Grant the new account **write** and **modify** access to the C:\Windows\Microsoft.NET\Framework\v1.1.4322\Temporary ASP.NET Files folder. If you are prompted with a message about system folder security, click **Yes**.
- Create a local account on Denver called **OrderReaders** with password **Passw0rd**. Ensure that the **User must change password at next logon** checkbox is **not** selected. Ensure that the **Password never expires** checkbox **is** selected.
- In SQL Server Enterprise Manager on Denver:
  - Create a login for **OrderReaders**.
  - Set the default database for **OrderReaders** to **Northwind**.
  - Permit the **OrderReaders** access to the **Northwind** database.
  - Add **OrderReaders** to the **db_datareader** role for the **Northwind** database.

Storing Connection Strings in Web.config

Perform the steps on the LONDON VPC.

1. If you are not already logged on, log on to London by using the **OrdinaryUser** account with a password of **Passw0rd**.
2. Open Visual Studio .NET 2003.
3. Open the solution C:\Courses\DeveloperSecurity\Democode\Session03\dpapi.sln
4. Right-click **default.aspx** and click **Set at Start Page**.
5. Run the solution without debugging by pressing CTRL + F5.
6. If prompted about Enhanced Security, select the **In the future, do not show this message** checkbox and click **OK**.
7. Close Internet Explorer.
8. Show the code in default.aspx.cs.
9. Expand the **Webform Designer Generated Code** region, and show where the connection string is hard-coded into the application.
10. Show **default.aspx** in design view.
11. Right-click **sqlConnection1** and then click **Properties**.
12. Expand **Dynamic Properties**. Click **Connection String**, and then click the ellipsis button. Select the **Map property to a key in configuration file** checkbox, and then click **OK**.
13. Show Web.config and Default.aspx.cs to show that Visual Studio .NET has moved the connection string definition to the configuration file.

Encrypting Connection Strings with DPAPI

1. In Visual Studio .NET, view the code for class1.cs in the EncryptString project.
2. Point out the call to the **dp.Encrypt** method.
3. Optionally, briefly show the code to DataProtection.cs in the dpapiLibrary project showing the calls to the data protection API.
4. In Solution Explorer, right-click the **EncryptString** project, point to **Debug**, and click **Start new instance**.
5. When the **Input:** prompt appears, switch to viewing **Web.Config**.
6. Copy the text of the connection string from where it is defined at the bottom of Web.config.
7. Switch to the Command window, and then right-click the title bar, point to **Edit**, and click **Paste**.
8. Press ENTER. The encrypted text is written to a file.
9. Press ENTER.
10. In Visual Studio .NET, on the **File** menu, point to **Open** and click File.

11. Open the file C:\Courses\DeveloperSecurity\Democode\Session03\dpapi\EncryptString\Encrypted.txt.
12. Copy the contents of the file.
13. Switch to Web.config. Remove the existing connection string text and paste in the encrypted version.
14. Save Web.config.
15. In default.aspx.cs, expand the WebDesigner Generated Code region if it isn't already.
16. Locate the line beginning **this.sqlConnection1.ConnectionString = (…**
17. Change to:

```
                //
                // sqlConnection1
                //
                DataProtection dp = new
    DataProtection(DataProtection.Store.USE_MACHINE_STORE);
                string appSettingValue =
                        ((string)(configurationAppSettings.GetValue
            ("sqlConnection1.ConnectionString", typeof(string))));
                byte[] dataToDecrypt =
Convert.FromBase64String(appSettingValue);
                string connStr =
    Encoding.ASCII.GetString(dp.Decrypt(dataToDecrypt,null));
                this.sqlConnection1.ConnectionString = connStr;
```

18. Press CTRL + F5 to run the application.
19. Close Internet Explorer.

Installing the Aspnet_setreg Utility

1. View Web.Config again. Find the <Identity …> element.
2. Notice that this application uses impersonation, and the username and password are in the file in clear text.
3. Open Windows Explorer.
4. Navigate to C:\Courses\DeveloperSecurity\Democode\Session03\Aspnet_setreg.
5. Double-click **aspnet_setreg.exe**.
6. Enter **C:\Tools** as the unzip folder.
7. Open a Command window.
8. Type **cd C:\Tools** and press ENTER.
9. Enter **aspnet_setreg /?** to show all the options available with this utility.

Using Encrypted Attributes in a Configuration File

1. From the **Start** menu, point to **All Programs | Accessories**. Right-click **Command Prompt** and click **Run as**.
2. Select **The following user** and provide a **User name** of **Administrator** and a password of **P@ssw0rd**.
3. Type **cd C:\Tools** and press ENTER.
4. At the command prompt, type **aspnet_setreg.exe -k:SOFTWARE\MY_SECURE_APP\ identity -u:"OrderReaders" -p:"Passw0rd"**
5. This command also generates output that specifies how to change your Web.config or Machine.config file so that ASP.NET will use these keys to read that information from the registry.
6. Edit Web.Config in Visual Studio .NET.
7. Change the <identity> section to:

```
<identity impersonate="true"
userName="registry:HKLM\SOFTWARE\MY_SECURE_APP\identity\ASPNET_SETREG,userN
ame"
password="registry:HKLM\SOFTWARE\MY_SECURE_APP\identity\ASPNET_SETREG,passw
ord" />
```

Granting Permissions on Registry Keys

1. Switch back to the command window.
2. Type **regedt32** then press ENTER. Registry Editor starts.
3. Right-click the **HKEY_LOCAL_MACHINE\SOFTWARE\MY_SECURE_APP** subkey, and click **Permissions**.
4. Click **Add**. In the dialog box that opens, type **NETWORK SERVICE** and then click **Check Names**.
5. Click **OK**.
6. Select **Allow** checkbox for the **Read** permissions, and then click **Advanced**.
7. Select the permission entry for the **Network Service** account and then select the **Replace permission entries on all child objects with entries shown here that apply to child objects** checkbox.
8. Click **OK**.
9. In the **Security** message box, click **Yes**.
10. Click **OK** to close the **Permissions** dialog box.
11. Close Registry Editor.
12. In Visual Studio .NET, press CTRL + F5 to demonstrate that the application works.
13. Close all applications.

Fail Intelligently (1 of 2)

```
DWORD dwRet = IsAccessAllowed(…);
if (dwRet == ERROR_ACCESS_DENIED) {
   // Security check failed.
      // Inform user that access is denied
} else {
   // Security check OK.
      // Perform task…
}
```

What if IsAccessAllowed() returns ERROR_NOT_ENOUGH_MEMORY?

- If your code does fail, make sure it fails securely

- When programmatically authorizing users based on their credentials or validating input, always follow a specific pattern. First, attempt to authorize or validate the data. Accept only valid input, and reject everything else. Do not attempt to invalidate the user or data and then accept everything else.

Fail Intelligently (2 of 2)

- **Do not:**
  - Reveal information in error messages

```
<customErrors mode="On"/>
```

  - Consume resources for lengthy periods of time after a failure
- **Do:**
  - Use exception handling blocks to avoid propagating errors back to the caller
  - Write suspicious failures to an event log

- Do not reveal error information to end users. Ensure production code does not contain error messages that developers use during production. If it is necessary to capture error information, write it to an event log.
- Do not consume resources for lengthy periods of time after a failure. It may result in a DoS attack.
- Your error handling should be structured in such a way that details of the error are not propagated back to the user.
- If your program has vulnerabilities in specific areas, write code to update an error log with any activities that appear suspicious.

Test Security

- **Involve test teams in projects at the beginning**
- **Use threat modeling to develop security testing strategy**
- **Think Evil. Be Evil. Test Evil.**
  - Automate attacks with scripts and low-level programming languages
  - Submit a variety of invalid data
  - Delete or deny access to files or registry entries
  - Test with an account that is not an administrator account
- **Know your enemy and know yourself**
  - What techniques and technologies will hackers use?
  - What techniques and technologies can testers use?

- Involve test teams at the very beginning of the design phase for each project. They can provide valuable feedback.
- Most testing is designed to prove product features work correctly. Security testing should be designed to make features fail. Examples of testing security include attempting to spoof user identities, submitting bad input, or creating DoS attacks.
- To defend against hackers, you must ensure that your test team has the same level of knowledge and access to the same technologies as hackers.
- Decomposing an application involves logically separating components, interfaces, and data to identify important assets and likely vulnerabilities. Use threat modeling to define security testing strategies.
- Attack every conceivable and inconceivable vulnerability. Explicitly test security, not just features.
- Use all of the means at your disposal to attack the application on all fronts.
- Use low-level programming languages such as C++ to find hidden vulnerabilities in the application. You can also use common scripts to automate attacks and create bad data.
- Submit data that is too long or too short, contains empty or NULL values, and contains completely random or only partially incorrect data.
- Try to delete critical files or place DENY DACLs on critical resources.
- Test with an account that is not an administrator account to accurately capture error information.
- If you have access to source code, review it.

Learn from Mistakes

- **If you find a security problem, learn from the mistake**
  - How did the security error occur?
  - Has the same error been made elsewhere in the code?
  - How could it have been prevented?
  - What should be changed to avoid a repetition of this kind of error?
  - Do you need to update educational material or analysis tools?

- "There is only one thing more painful than learning from experience and that is not learning from experience." – Archibald McLeish (1892 – 1982), American poet.
- Approach every bug as a learning opportunity. If you fail to learn from a mistake, it is highly probable that you will make the same mistake in the future.

Session Summary

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

Next Steps

1. **Stay informed about security**
   - Sign up for security bulletins:
     http://www.microsoft.com/security/security_bulletins/alerts2.asp
   - Get the latest Microsoft security guidance:
     http://www.microsoft.com/security/guidance/
2. **Get additional security training**
   - Find online and in-person training seminars:
     http://www.microsoft.com/seminar/events/security.mspx
   - Find a local CTEC for hands-on training:
     http://www.microsoft.com/learning/

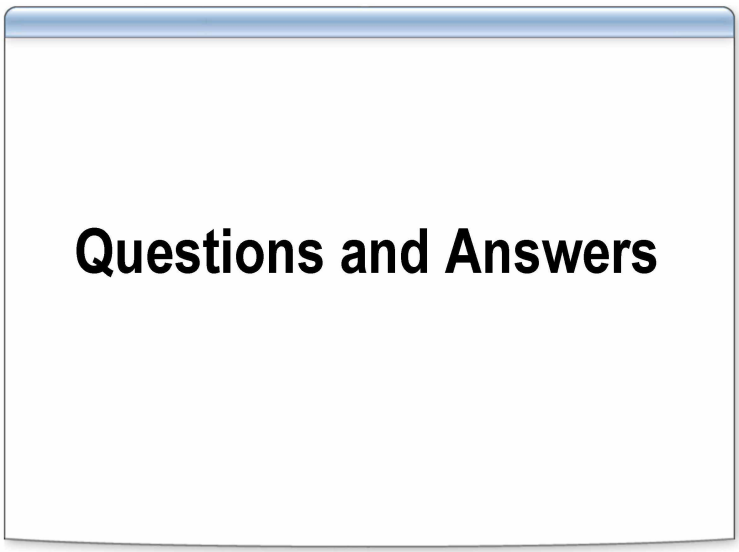
- Next steps include going to the Microsoft Web site to:
  - Get the latest security information.
  - Get additional security training.

For More Information

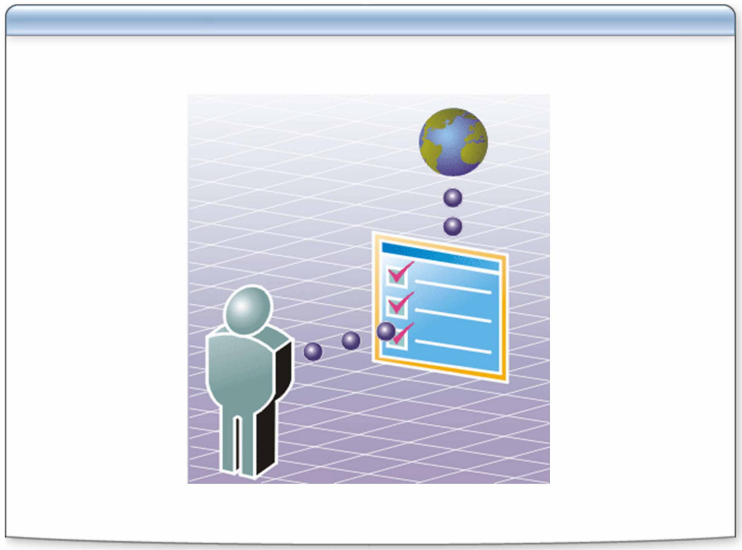
- **Microsoft Security Site (all audiences)**
  http://www.microsoft.com/security
- **MSDN Security Site (developers)**
  http://msdn.microsoft.com/security
- **TechNet Security Site (IT professionals)**
  http://www.microsoft.com/technet/security

- More technical information for IT professional and developers is available on the following Web sites:
  - Microsoft Security Site (all audiences)
    - http://www.microsoft.com/security
  - MSDN Security Site (developers)
    - http://msdn.microsoft.com/security
  - TechNet Security Site (IT professionals)
    - http://www.microsoft.com/technet/security

Questions and Answers



Clinic Evaluation



**********ILLEGAL FOR NON-TRAINER USE**********

Your evaluation of this clinic will help Microsoft understand the quality of your learning experience.

At a convenient time before the end of the clinic, please complete a clinic evaluation, which is available at http://www.CourseSurvey.com.

Microsoft will keep your evaluation strictly confidential and will use your responses to improve your future learning experience.