您的潜力，我们的动力

Microsoft
微软（中国）有限公司

开发高安全级别的企业应用系列课程（之四）
应用Microsoft .NET Framework编写安全的应用程序

钟卫

DPE

微软公司

msdn

MSDN Webcasts

Session Prerequisites

- Development experience with Microsoft Visual Basic®, Microsoft Visual C++, or C#
- Experience building Microsoft Windows or Web applications using the .NET Framework

Level 200

课程概述

- .NET Framework 安全特性
- 代码访问安全
- 角色访问安全
- 构建安全的ASP.NET Web 应用
- 构建安全的ASP.NET Web Services应用

.NET Framework 安全特性

- .NET Framework 安全特性
- 代码访问安全
- 角色访问安全
- 构建安全的ASP.NET Web 应用
- 构建安全的ASP.NET Web Services应用

.NET 托管代码的执行安全

- .NET Framework 安全特性
  帮助您开发更加安全的应用
  - 内置很多的安全组件
    - 类型察看
    - 异常管理
    - 安全引擎
  - 与Windows 的安全性相互补充
  - 具备很高的灵活性和扩展性

类型安全的系统

类型安全代码
- 防范缓冲区溢出
- 授权方式限制访问内存
- 允许相同进程运行多程序集

应用程序域提供：
- 性能增强
- 代码安全性的增强

代码访问安全

- .NET Framework 安全特性
- 代码访问安全
- 角色访问安全
- 构建安全的ASP.NET Web 应用
- 构建安全的ASP.NET Web Services应用

缓冲区溢出保护

- 应用类型检验防范内存改写
- 在.NET 中**System.String**的声明和调用没发生改变

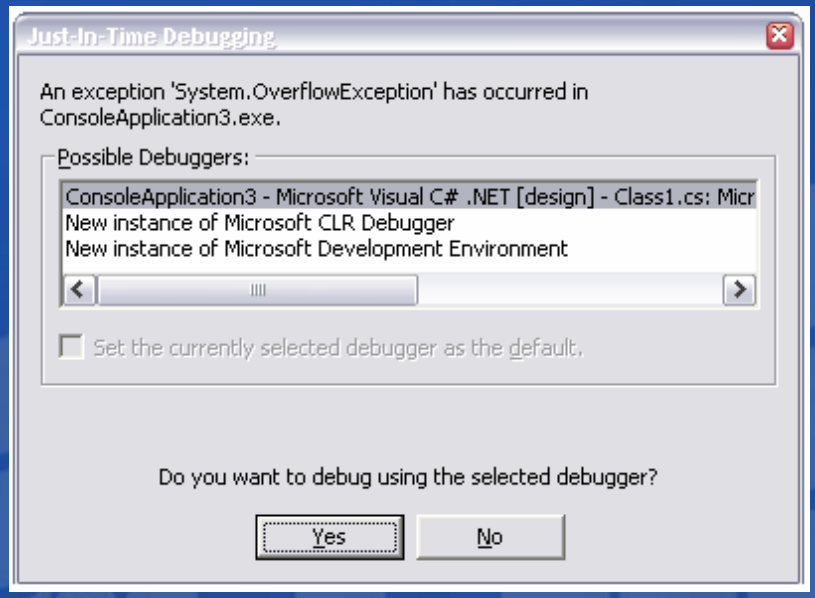
```
void CopyString (string src)
{
    stringDest = src;
}
```

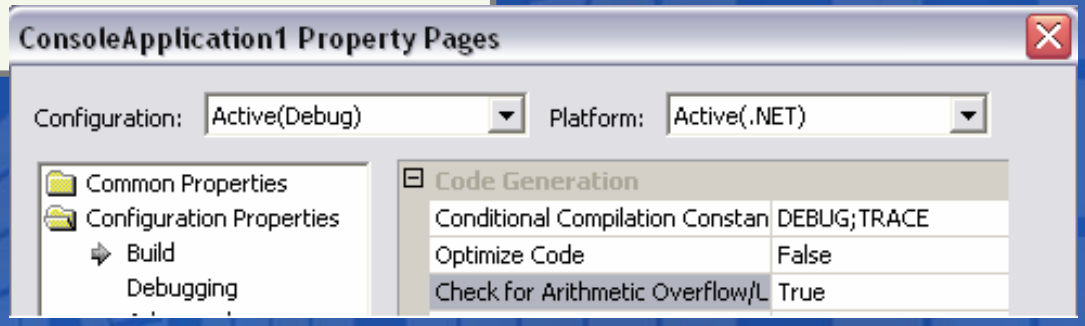
- .NET 中的**System.Text.StringBuilder**提供了缓冲区边界检查

算法错误检查

- 对算法的错误检查通过：
  - 关键字检查
  - 项目设定

```
byte  b=0;
while (true)
{
    Console.WriteLine (b);
    checked
    {
        b++;
    }
}
```

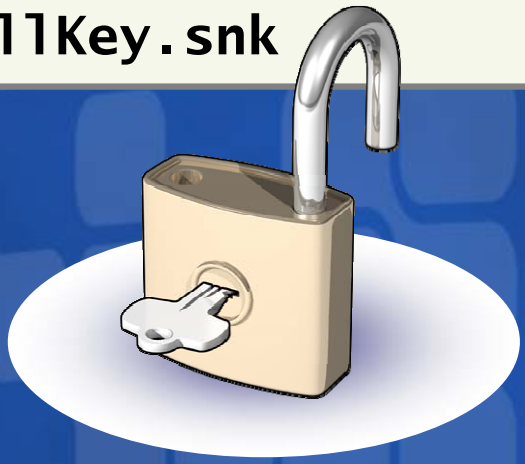




强命名保护

- 强命名
  - 具有唯一的标识符
  - 被用于对程序集的数字签名
- 经过强命名的程序集
  - 防止被篡改
  - 确认程序集原有者的身份
  - 允许组件并行调用

```
sn -k MyFullKey.snk
```



独立存储

- 提供了一个虚拟的文件系统
- 允许设置配额
- 文件的独立存储通过
  - 应用身份确认
  - 用户身份确认

```
IsolatedStorageFile isoStore =
   IsolatedStorageFile.GetUserStoreForAssembly();
```

- Internet Zone apps get 10MB by default

  Internet 域给应用程序默认分配10MB的空间

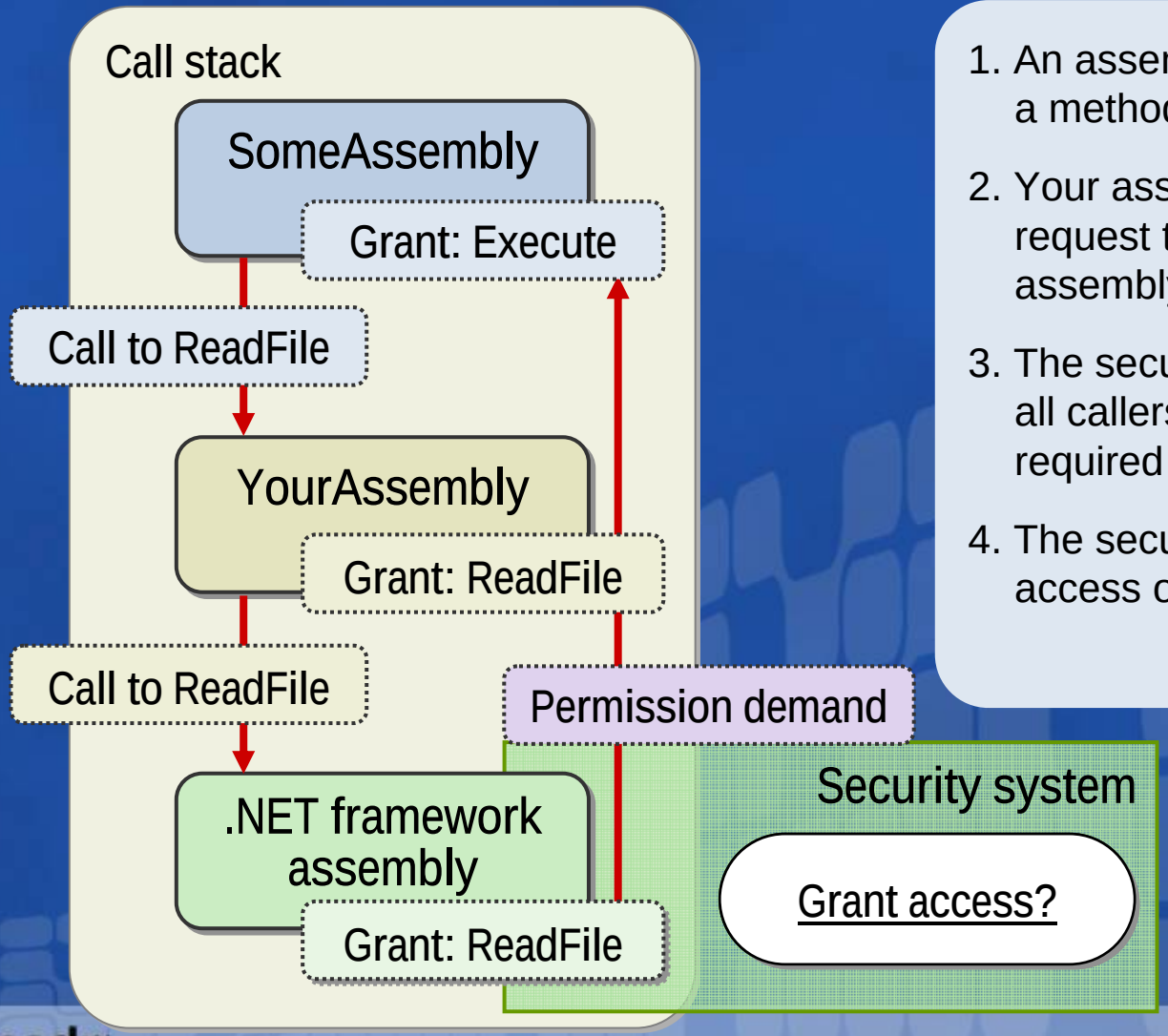
基于证据的安全机制

- 证据:
  - 当程序集被访问时评估证据
  - 被用于确定程序集所具有的权限
  - 包含了程序集的:
    - 强命名信息
    - URL（可信，非可信）
    - Zone（安全区域）
    - 可信的代码签名
    - 用户自定义信息

安全策略

Security entity	Description
Policy	• 由管理员进行设定 • 运行时强制检查 • 管理简单 • 包含**permissions** • 包含**code groups**
Code group	• 一类相似的程序集 • 基于证据 • 与一个**permission**集合相联系
Permission set	• 包含一组经过授权的**permission**

步入调用堆栈的安全机制

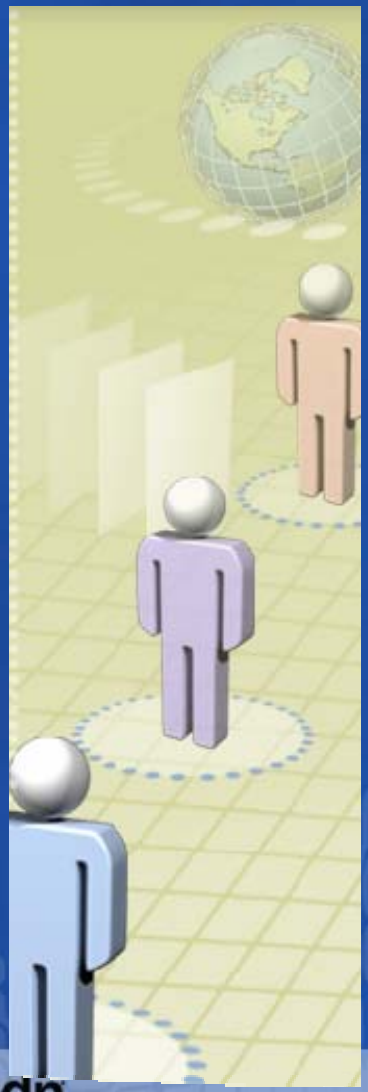


1. An assembly requests access to a method in your assembly
2. Your assembly passes the request to a .NET Framework assembly
3. The security system ensures that all callers in the stack have the required permissions
4. The security system grants access or throws an exception

访问许可请求

- 开发者通过许可请求获取必须的权限
- 属性引用实现
- 除非具备了运行程序的最小权限，否则不能运行

```
//I will only run if I can call unmanaged code
[assembly:SecurityPermission
   (SecurityAction.RequestMinimum,
   UnmanagedCode=true)]
```



Demonstration 1: Code Access Security

- Using the .NET Framework Configuration Tool
- Performing Security Checks
- Requesting Permissions

部分信任的程序集

- .NET Framework 1.1之前的web应用都是full trust运行
- .NET Framework 1.1提供下面几种信任级别：
  - Full
  - High
  - Medium
  - Low
  - Minimal

角色访问安全

- .NET Framework 安全特性
- 代码访问安全
- 角色访问安全
- 加密
- 构建安全的ASP.NET Web 应用
- 构建安全的ASP.NET Web Services应用

认证和授权

- 认证需要解决
  "Who are you?"
  "Am I sure you are who you say you are?"

- 授权需要解决
  "Are you allowed to … ?"

Identities and Principals

- identity 包含了一个用户的信息，比如用户的登陆名称
- principal 包含了一种角色的信息，比如角色包含的用户和计算机
- .NET Framework 提供了：
  - WindowsIdentity and WindowsPrincipal objects
  - GenericIdentity and GenericPrincipal objects

创建Windows Identities and Principals

- 使用WindowsIdentity和WindowsPrincipal 对象
  - Single validation

```
WindowsIdentity myIdent = WindowsIdentity.GetCurrent();
WindowsPrincipal myPrin = new WindowsPrincipal(myIdent);
```

  - Repeated validation

```
AppDomain.CurrentDomain.SetPrincipalPolicy(PrincipalPolicy.WindowsPrincipal);
WindowsPrincipal myPrin = (Thread.CurrentPrincipal as WindowsPrincipal);
```

创建 Generic Identities and Principals

- 创建一个 **GenericIdentity** 和 **GenericPrincipal**

```
GenericIdentity myIdent = new GenericIdentity("User1");
string[] roles = {"Manager", "Teller"};
GenericPrincipal myPrin =
   new GenericPrincipal(myIdent, roles);
```

- 将创建的**GenericPrincipal**加载到当前的策略上

```
System.Threading.Thread.CurrentPrincipal = myPrin;
```

执行安全检查

- 在代码中使用Identity and Principal members
  - 比如，使用Identity.Name属性校验用户的登陆名称

```
if (String.Compare(myPrin.Identity.Name, "DOMAIN\\Fred",
   true)==0)
{
   // Perform some action
}
```

  - 比如，使用类Principal的IsInRole方法检验角色成员

```
if (myPrin.IsInRole("BUILTIN\\Administrators"))
{
   // Perform some action
}
```

命令与声明式的安全检查

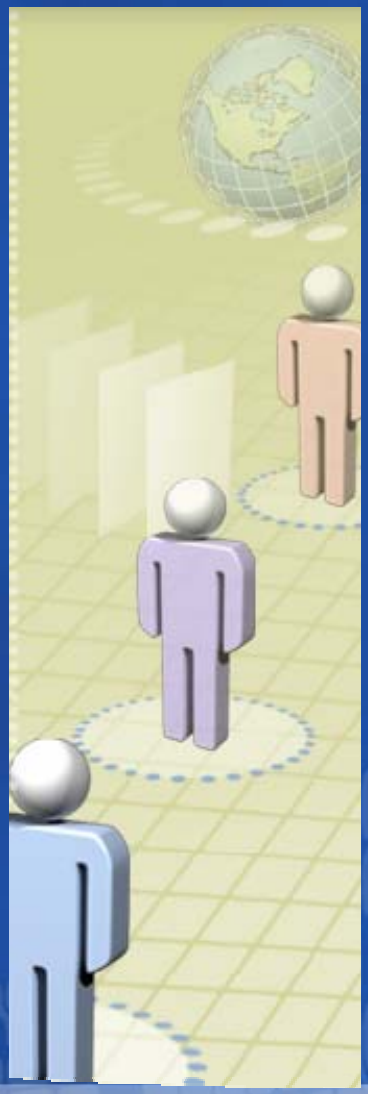
- Use permissions to perform role-based security checks
  - 命令式

```
PrincipalPermission prinPerm = new   PrincipalPermission("Teller", “Manager”, true);
try
{
   prinPerm.Demand();  //Does the above match the active principal?
}
```

  - 声明式 (通过属性调用实现)

```
[PrincipalPermission(SecurityAction.Demand, Role="Teller", Authenticated=true)]
```

Demonstration 2: Role-Based Security



- Using Windows Role-Based Security
- Using Generic Role-Based Security

构建安全的ASP.NET Web 应用

- .NET Framework 安全特性
- 代码访问安全
- 角色访问安全
- 加密
- 构建安全的ASP.NET Web 应用
- 构建安全的ASP.NET Web Services应用

ASP.NET认证方式

认证方式	优点	缺点
Windows	• 使用现有**windows**的架构 • 控制对于敏感数据的访问权	• 不支持所有的客户端类型 • 每个用户都必须有一个**windows**账户
Forms	• 支持任意客户端	• 需要**cookies**支持
Microsoft Passport	• 提供对于多个**Internet Web** 站点的 一次性登陆验证 • 允许开发人员定制注册叶面	• 需要**cookies**的支持 • 涉及费用问题

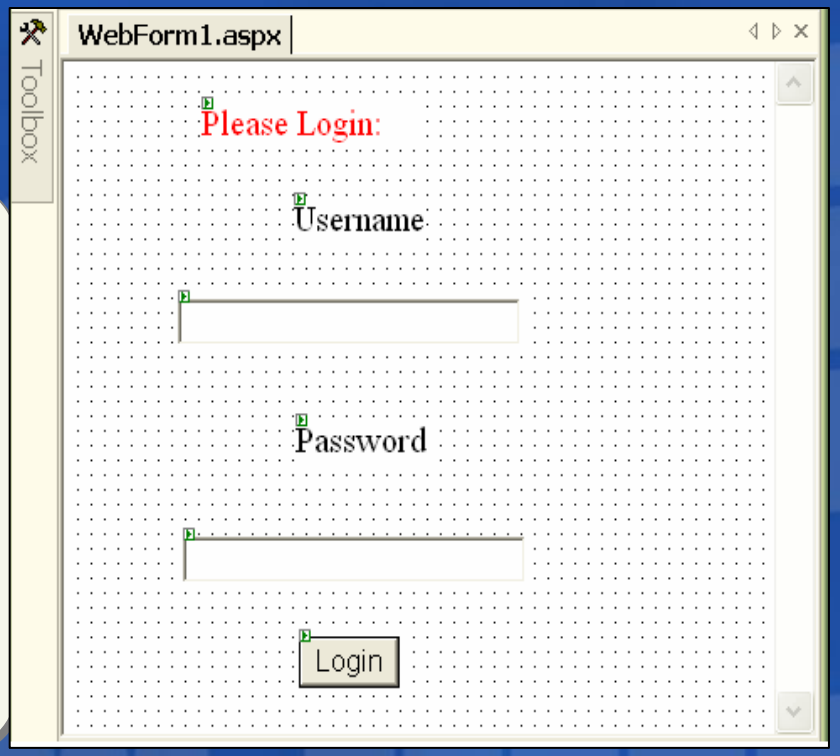
基于Forms认证方式的设置

- 设置IIS Anonymous authentication
- 通过设置Web.config 建立Form认证方式
- 建立授权
- 创建一个登陆叶面

```
<system.web>
    <authentication mode="Forms">
        <forms

loginUrl="WebForm1.aspx"/>
    </authentication>

    <authorization>
       <deny users="?"/>
    </authorization>
</system.web>
```



窗体认证方式的增强

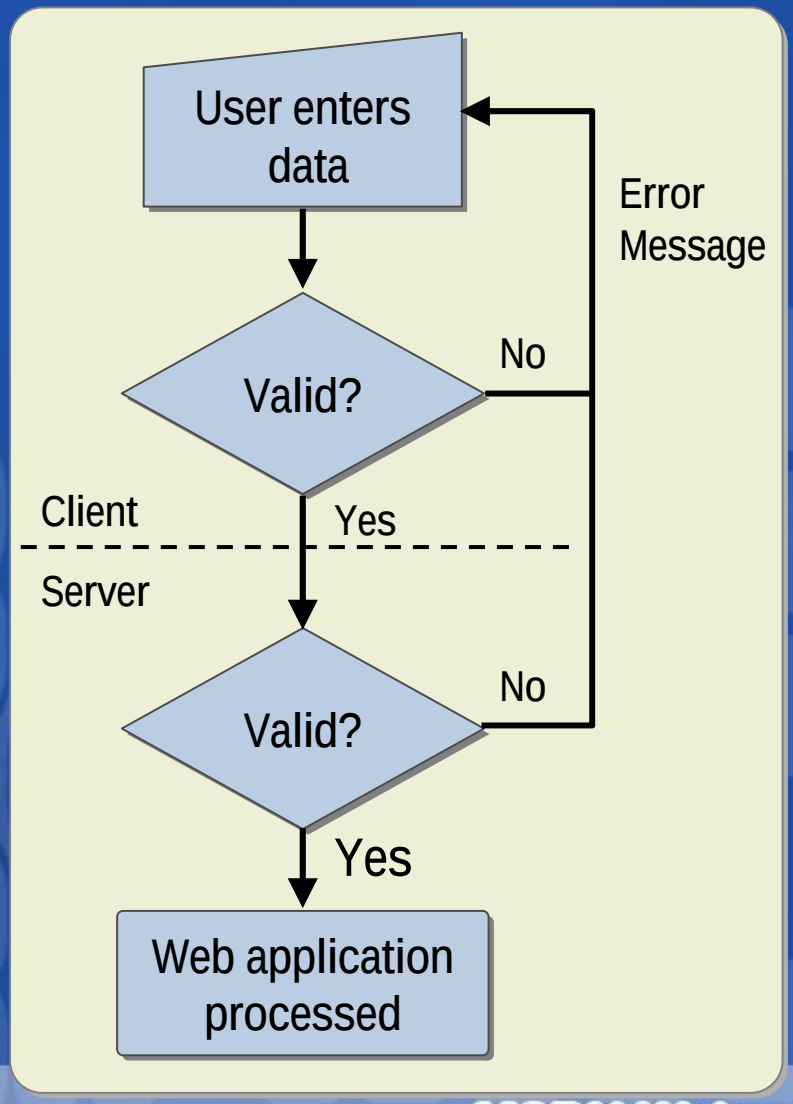
- 开发人员可以通过设置确保cookies安全

```
<authentication mode="Forms">
    <forms loginUrl="login.aspx"
        protection="All"
        requireSSL="true"
        timeout="10"
        name="AppNameCookie"
        path="/FormsAuth"
        slidingExpiration="true"
    </forms>
</authentication>
```

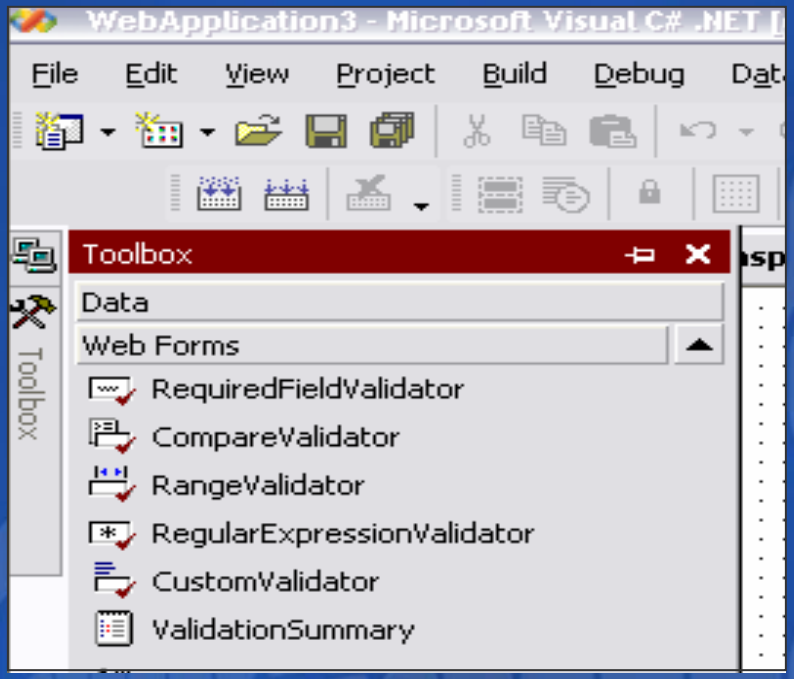
- 开发人员可以创建 application-specific keys来加解密 cookies

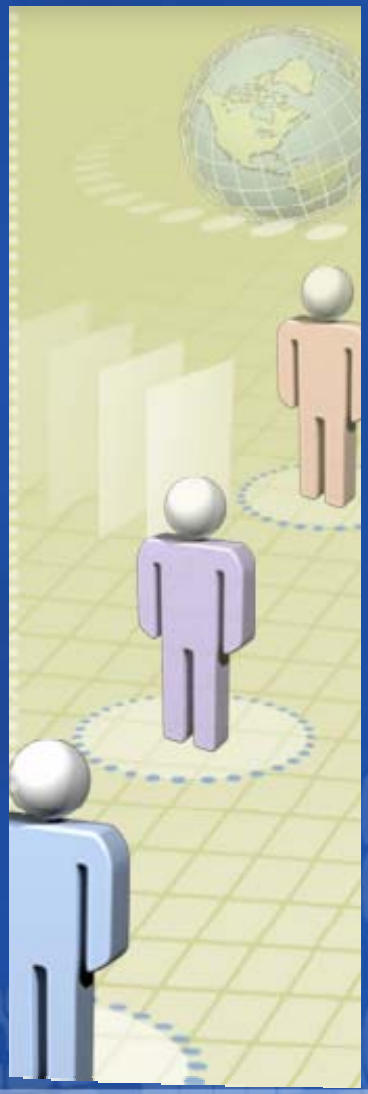
校验控件

- 客户端校验
  - 提供即时反馈
  - 减少postback
- 服务器校验
  - 重复了客户端的校验
  - 可以结合存储的数据进行验证



校验控件的类型


WebApplication3 - Microsoft Visual C# .NET
File
Edit
View
Project
Build
Debug
Toolbox
Data
Web Forms
RequiredFieldValidator
CompareValidator
RangeValidator
RegularExpressionValidator
CustomValidator
ValidationSummary
Toolbox



Demonstration 4: ASP.NET Web Application Security

- Configuring Forms Authentication
- Using Validation Controls

构建安全的ASP.NET Web Services应用

- .NET Framework 安全特性
- 代码访问安全
- 角色访问安全
- 加密
- 构建安全的ASP.NET Web 应用
- 构建安全的ASP.NET Web Services应用

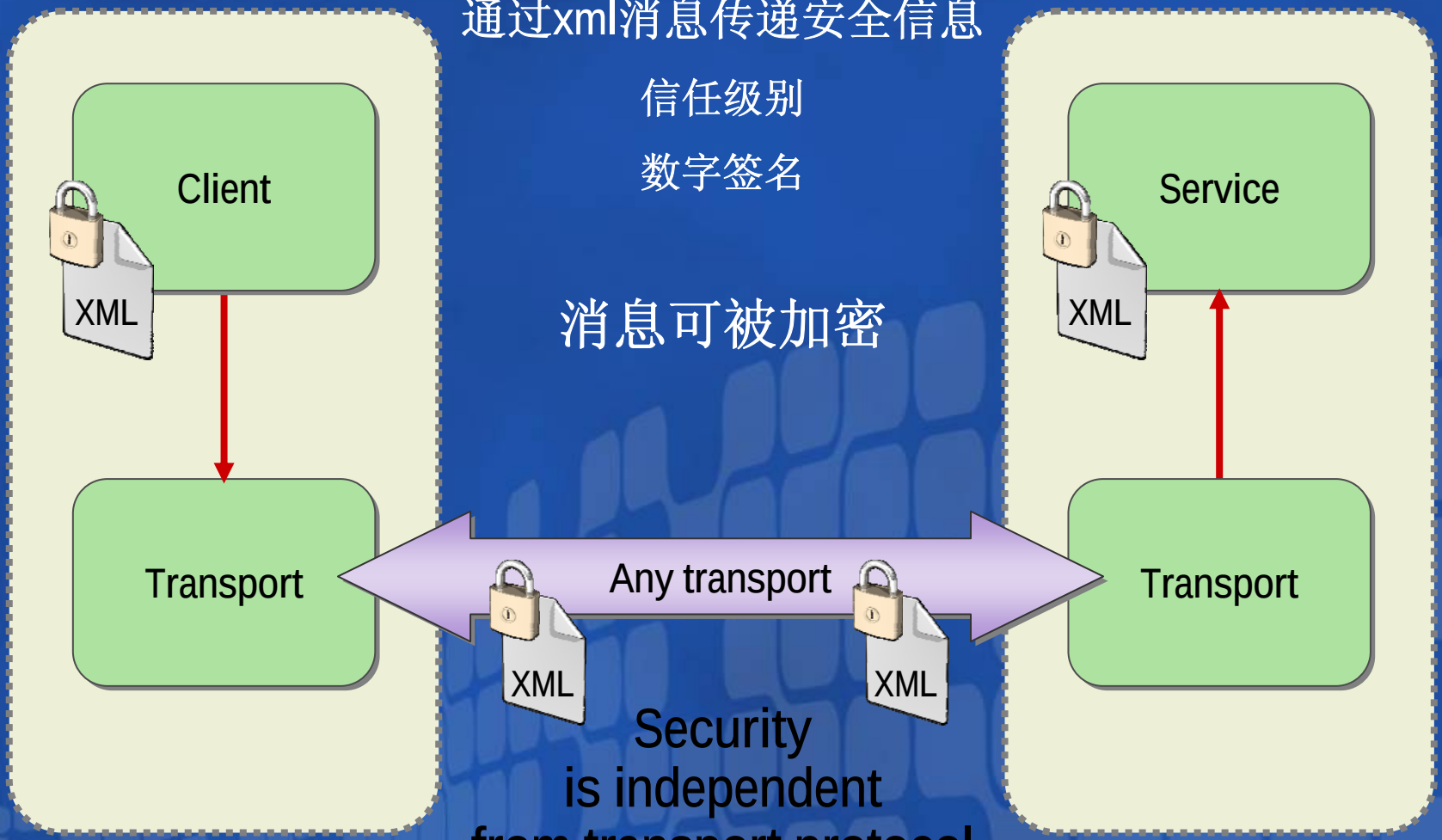
消息级别的安全

通过xml消息传递安全信息

信任级别

数字签名

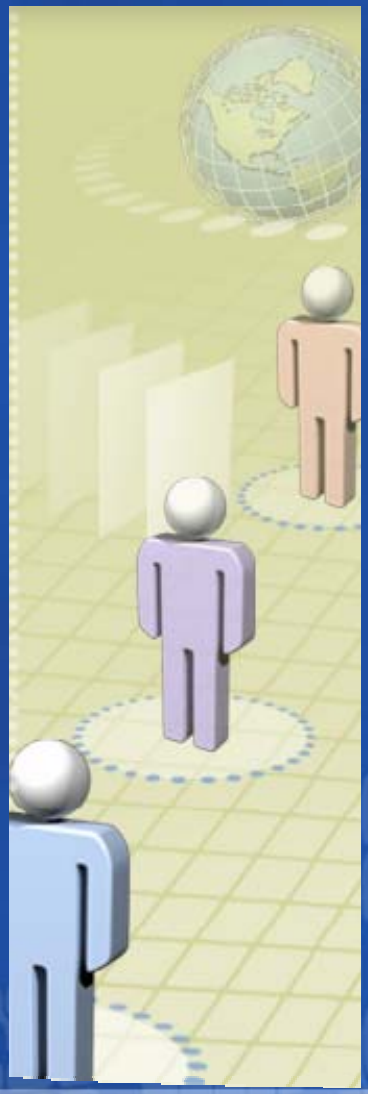
消息可被加密



Security
is independent
from transport protocol

Web Service Enhancements (WSE)

- 包含以下内容：
  - SOAP headers包含认证信息
  - 支持消息加密
  - 支持消息数字签名
- 支持消息路由
- 支持附件传递
- 通过程序集 Microsoft.Web.Services.dll 运行



Demonstration 5: Web Services Enhancements

- Implementing Security for a Web Service
- A look at WSE messages

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/

For More Information

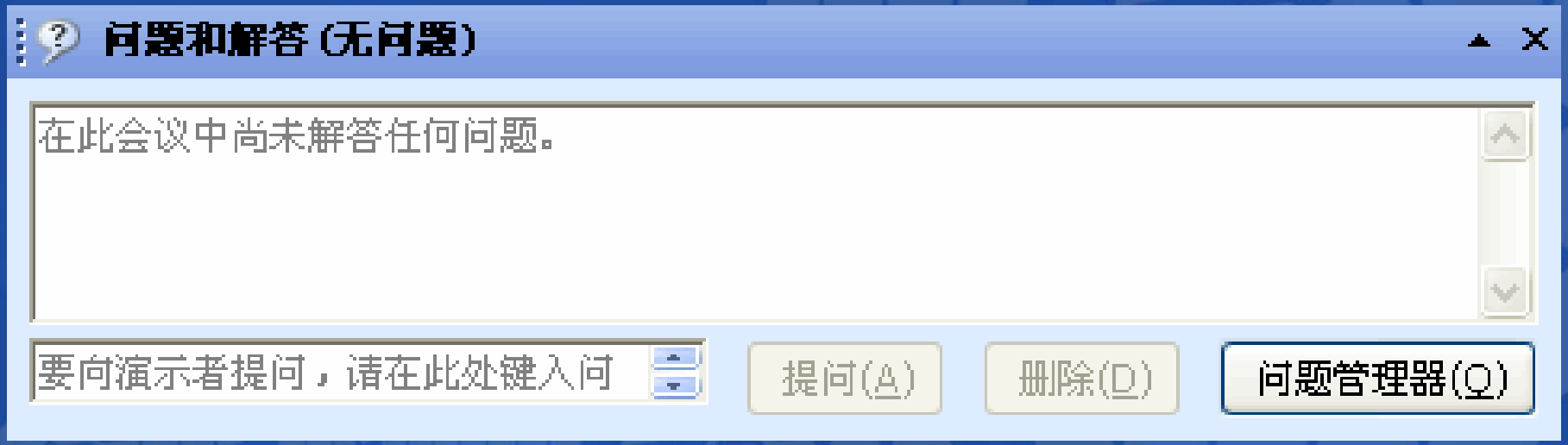
- Microsoft Security Site (all audiences)
  http://www.microsoft.com/security
- MSDN Security Site (developers)
  http://msdn.microsoft.com/security
- TechNet Security Site (IT professionals)
  http://www.microsoft.com/technet/security

您的潜力，我们的动力

Microsoft
微软(中国)有限公司

Q&A

如需提出问题，请单击“提问”按钮并在随后显示的浮动面板中输入问题内容。一旦完成问题输入后，请单击“提问”按钮。




您的潜力，我们的动力
Microsoft
微软（中国）有限公司
Microsoft
msdn
MSDN Webcasts