

# 在.NET上实现应用安全性

蔺华(Lin Hua)  
Developer Evangelist  
D&PE  
Microsoft China





# 我们今天主要讨论什么？

- **.NET Framework 安全特性**
- 代码访问安全性
- 基于角色的安全性
- 加密
- **Whidbey**的安全特性概览

# .NET 托管执行安全性

- **.NET Framework 安全性增强**

- 帮助您开发更为安全的应用
- 包括很多重要的安全特性，如：
  - （类型检查）**Type Checker**
  - （异常管理器）**Exception Manager**
  - （安全引擎）**Security Engine**
- 很好的增强的**Windows**平台的安全性

# Type-Safe 机制

- Type-safe 代码:
  - 防止缓冲区溢出(buffer overrun)
  - 限制访问未授权的内存地址空间
  - 允许多个程序集(assembly) 运行在同一进程
- 程序域(app domain)提供了:
  - 更好的性能
  - 更好的代码安全性

# 缓冲区溢出保护

- 类型检查可以防止任意的内存操作
- **.NET System.String** 对象是不变的

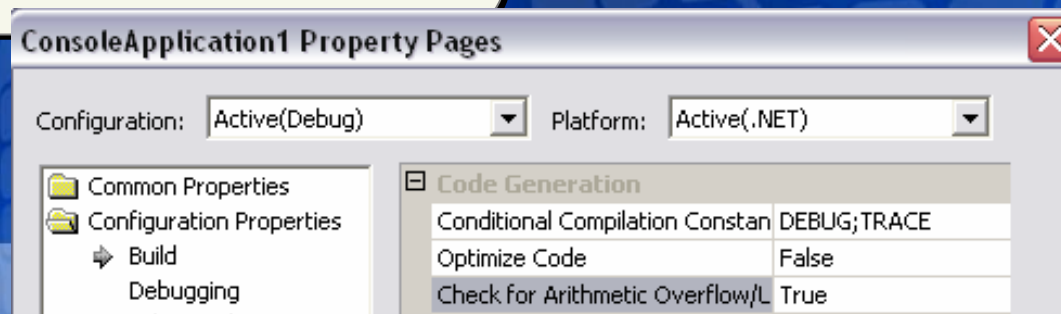
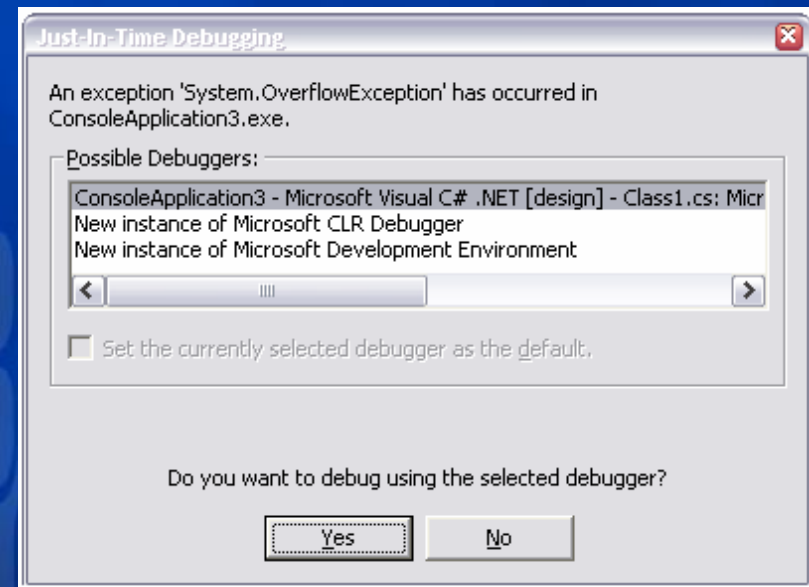
```
void CopyString (string src)
{
    stringDest = src;
}
```

- **System.Text.StringBuilder** 检查缓冲区范围

# 算法错误捕获

- 通过下面两种方法来捕获算法错误:
  - **checked**关键字
  - **Project** 设置

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





# 类型安全

## demo

- Investigating .NET Data-Type Safety
- Using the *checked* keyword



# 强命名程序集

- 强命名是
  - 唯一标识符（含公钥）
  - 用来数字签名程序集

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

- 强命名程序集
  - 防止篡改
  - 确认程序集发布者的身份
  - 允许并行组件



# 隔离存储

- 提供了一个虚拟的文件系统
- 允许配额
- 基于下面两种方式实现了文件系统隔离
  - 应用程序身份
  - 用户身份

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



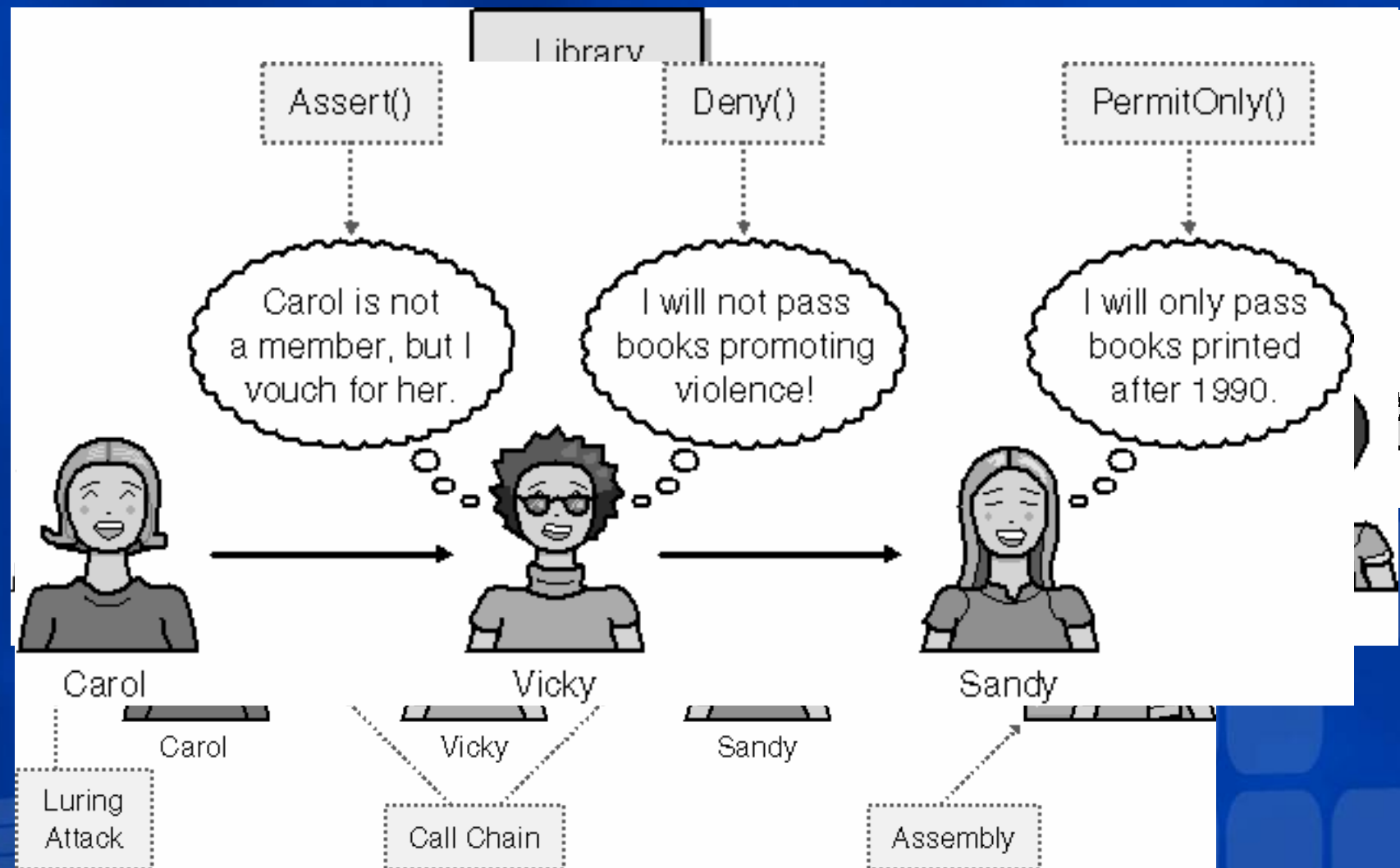
# 我们今天主要讨论什么？

- .NET Framework 安全特性
- 代码访问安全性(CAS)
- 基于角色的安全性
- 加密
- Whidbey的安全特性概览



# CAS 图示

Microsoft



# Evidence-Based 安全性

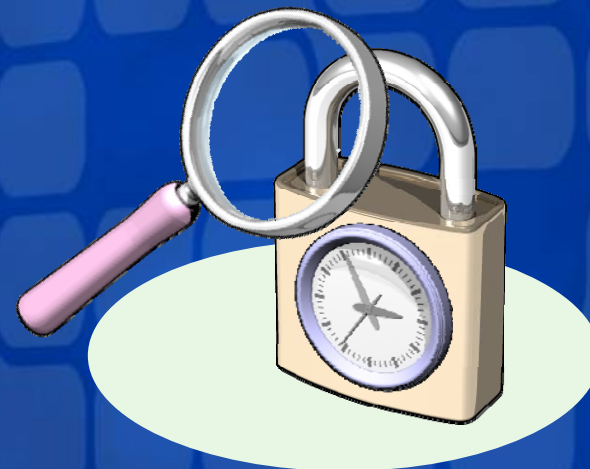
- Windows 安全机制

- [\(Video\)](#)

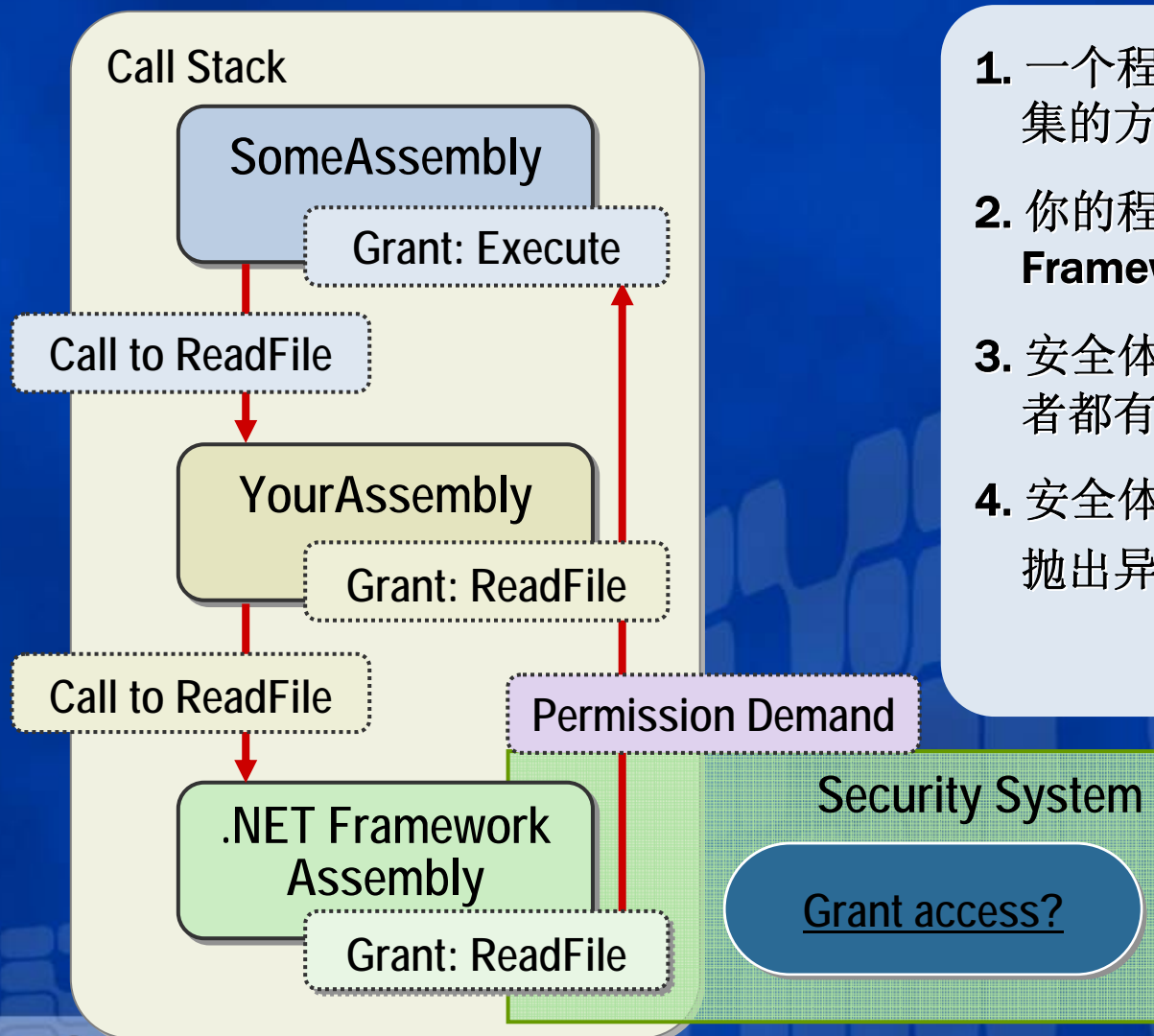
- Evidence(证据)

- 当程序集装载时评估
  - 用来确定程序集的权限
  - 它包括程序集的下列信息:

Evidence	Description
Hash	Hash of the assembly
Publisher	AuthentiCode® signer
StrongName	Public key+name+version
Site	Web site of code origin
Url	URL of code origin
Zone	Internet Explorer zone of code origin



# 安全检查堆栈审核



1. 一个程序集请求访问你的程序集的方法
2. 你的程序集就把请求传给 **.NET Framework** 程序集
3. 安全体系确保堆栈中每个调用者都有需要的权限
4. 安全体系将授予访问权限或者抛出异常





# 安全检查的类型

- 强制安全检查(imperative)
  - 创建 **Permission** 对象
  - 调用 **Permission** 方法
- 声明式安全检查(declarative)
  - 使用 **Permission** 属性
  - 应用到方法和类
- 覆盖安全检查(overriding)
  - 使用 **Assert** 方法
  - 防止堆栈审核

# 代码访问安全性

## demo

- .Net Framework 配置工具
- 执行安全检查
- 请求权限

# 我们今天主要讨论什么？

- **.NET Framework 安全特性**
- 代码访问安全性
- 基于角色的安全性
- 加密
- Whidbey的安全特性概览



# 认证和授权

- 认证(Authentication)是问:  
“你是谁?”  
“我能确认你是你说的那个人吗?”
- 授权(Authorization)是问:  
“你被允许做 ... ?”



# 身份和主体特征

- 身份包含关于用户的信息，如登陆名等
- 主体特征包含用户和机器和角色信息
- **.NET Framework 提供了：**
  - **WindowsIdentity 和 WindowsPrincipal 对象**
  - **GenericIdentity 和 GenericPrincipal 对象**

# 创建 Windows 身份和主体特征

- 使用 **WindowsIdentity** 和 **WindowsPrincipal**:

- Single validation

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

- Repeated validation

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



# 创建 Generic 身份和主体特征

- 创建 GenericIdentity 和 GenericPrincipal

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

- 把 GenericPrincipal Attach 到当前线程

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

# 执行安全检查

- 在代码中用**Identity**和**Principal**成员
  - 使用**Identity**对象的**Name**属性来检查用户登录名

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

- 使用**Principal**的**IsInRole**方法来判定角色成员关系

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

# 强制和声明式安全检查

- 用权限来做基于角色的安全检查

- 强制检查

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

- 声明式检查

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

# 基于角色的安全性

## demo

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



# 我们今天主要讨论什么？

- .NET Framework 安全特性
- 代码访问安全性
- 基于角色的安全性
- 加密
- Whidbey的安全特性概览

# 加密技术

密码学术语	描述
对称加密	Encrypting and decrypting data with a secret key
非对称加密	Encrypting and decrypting data with a public/private key pair
哈希	Mapping a long string of data to a short, fixed-size string of data
数字签名	Hashing data and encrypting the hash value with a private key

**.NET Framework** 为这些操作都  
提供了很好的支持



# 使用对称加密

- 选择算法
  - TripleDESCryptoServiceProvider
  - RijndaelManaged
- 产生一个密钥
- 用同样的密钥加、解密数据：
  - FileStream
  - MemoryStream
  - NetworkStream

# 使用非对称加密

- 选择算法
  - RSACryptoServiceProvider
  - DSACryptoServiceProvider
- 产生一个公钥、密钥对
- 加密、解密数据

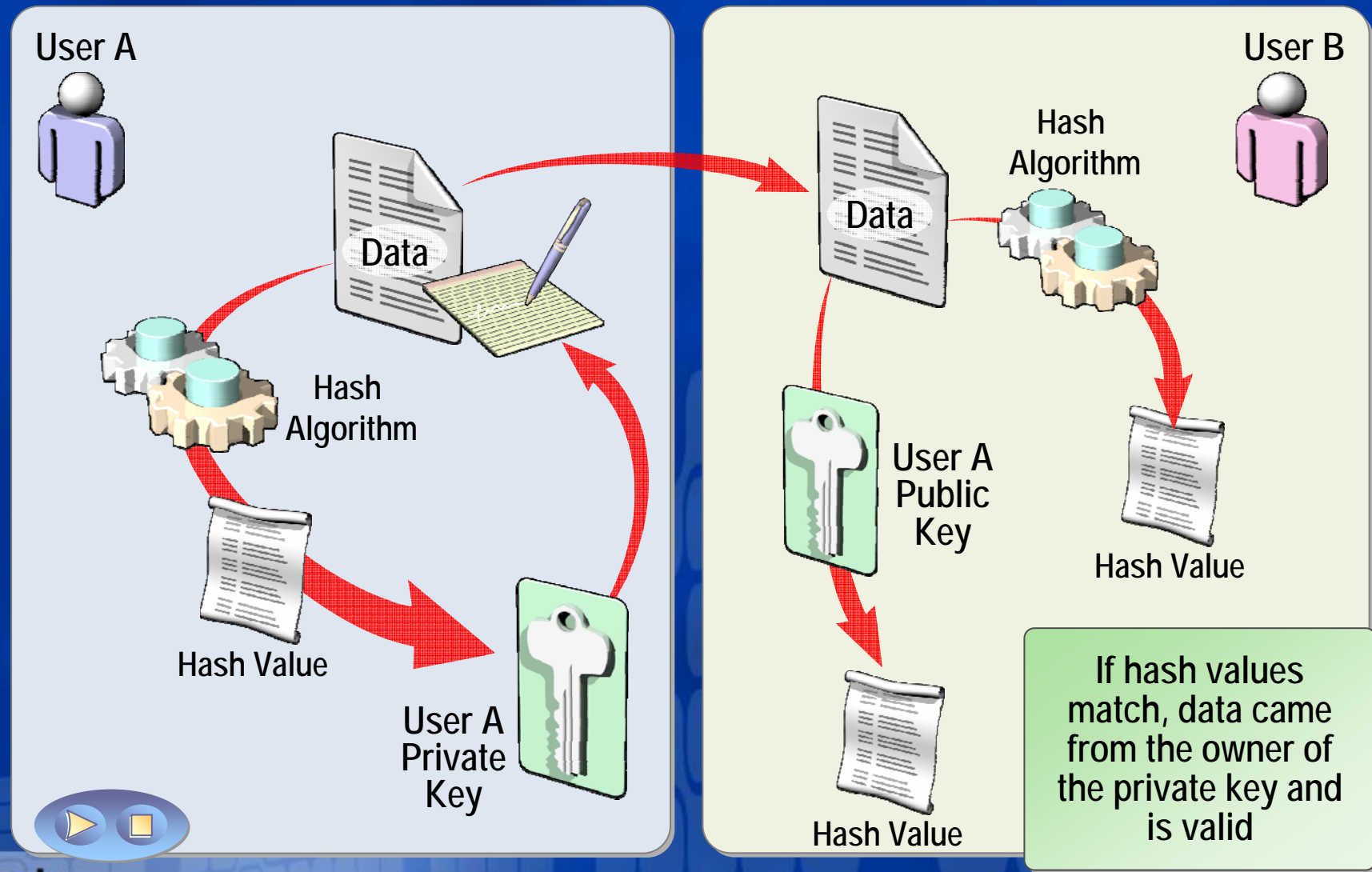


# 签名数据和验证签名

动作	步骤
签名数据	<ul style="list-style-type: none"><li>Hash the data</li><li>Encrypt the hash value with a private key</li></ul>
验证签名	<ul style="list-style-type: none"><li>Decrypt the signature by using sender's public key</li><li>Hash the data</li><li>Compare the decrypted signature to the hash value</li></ul>

# 数字签名图示

Microsoft®



# .NET Framework 加密

## demo

- Performing Symmetric Encryption
- Signing Data

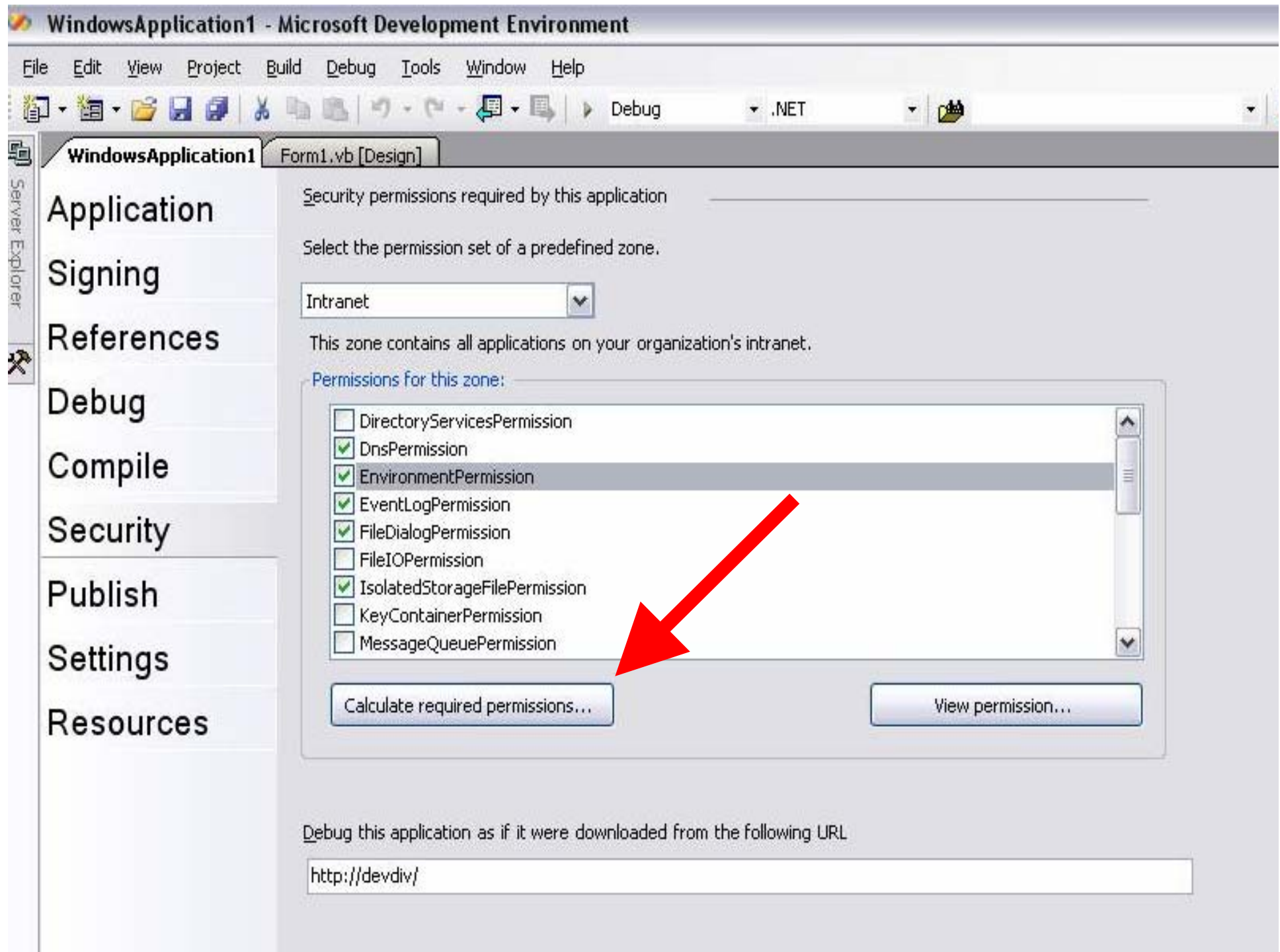
# 我们今天主要讨论什么？

- .NET Framework 安全特性
- 代码访问安全性
- 基于角色的安全性
- 加密
- Whidbey的安全特性概览



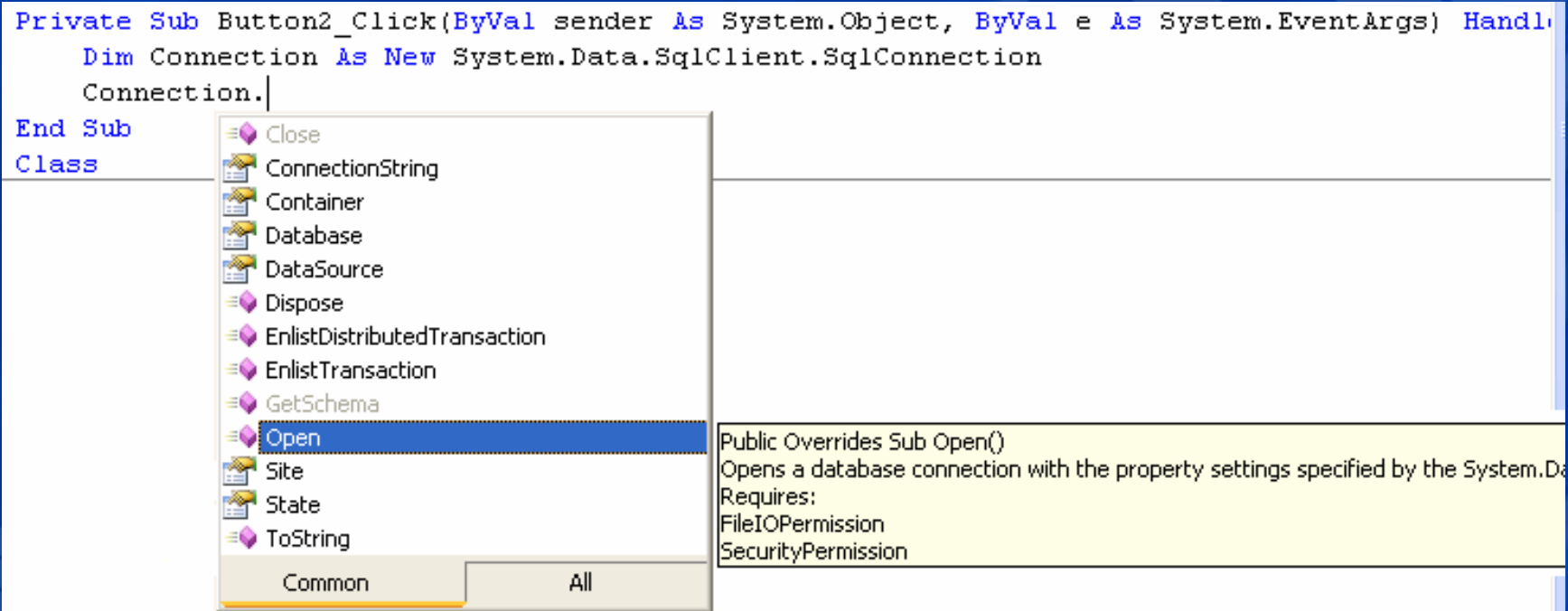
# VS2005的安全增强

- Static Analysis Tools
  - PREfast 针对C/C++应用
  - FxCop 针对托管代码
- /GS and AppVerifier
- SafeCRT Libraries
- Least Privilege and CAS
- IntelliSense in Zone



# IntelliSense in Zone

- Set a zone
- Grays out items in the intellisense list that would cause an app to violate the security settings
- Allows developers to catch security issues before they actually write the code



# 总结

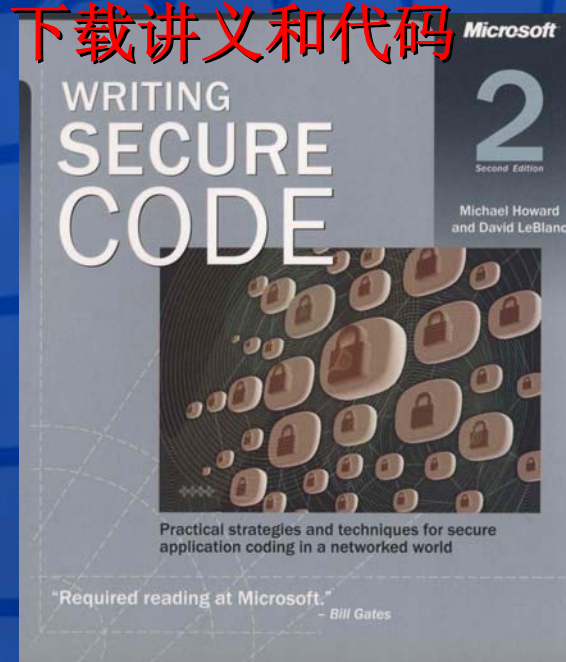
- .NET Framework 安全特性
- 代码访问安全性
- 基于角色的安全性
- 加密
- Whidbey的安全特性概览



# 相关资源

- 告诉我您的建议和要求: [hualin@microsoft.com](mailto:hualin@microsoft.com)
- <http://www.microsoft.com/MSPress/books/5957.asp>
- <http://msdn.microsoft.com/library/en-us/secmod/html/secmod71.asp?frame=true>
- <http://msdn.microsoft.com/library/en-us/secmod/html/secmod00.asp?frame=true>
- <http://www.microsoft.com/china/msdn>

下载讲义和代码



**Microsoft®**

**Microsoft®**

***Your potential. Our passion.™***


© 2004 Microsoft Corporation. All rights reserved.

This presentation is for informational purposes only. Microsoft makes no warranties, express or implied, in this summary.




# 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。

 **问题和解答 (无问题)** ▲ ×

在此会议中尚未解答任何问题。

要向演示者提问，请在此处键入问 

提问(A)

删除(D)

问题管理器(Q)