Segurança na Plataforma Microsoft .Net

João Garcia
IST - INESC ID
www.gsd.inesc-id.pt/~jog

Sumário

- Segurança de Acessos por Código (*Code Access*)
- Políticas de Segurança

- Tópicos não abordados:
  - Mecanismos de Recolha de Provas
  - Segurança baseada em Papéis (*Role-Based Security*)
  - Ferramentas de Segurança
  - Segurança em Aplicações ASP.Net

Segurança por Controlo de Acesso

- Motivação
- Segurança Baseada em Provas (*Evidence-Based Security*)
- Conceitos Básicos
- Análise da Pilha (*Stack Walk*)
- Quando é aplicada a segurança?
- Como funciona?
- Criação de Permissões de Acesso
- Requisitos de Segurança
- Forçar Verificações de Segurança

Motivação (1)

- O código pode ser descarregado da Internet ou circular anexo a mensagens de correio electrónico ou documentos
- Código malicioso (executado até por utilizadores confiáveis) pode estragar sistemas e dados resultando em danos elevados
- Mecanismos de segurança mais comuns:
  - Segurança aplicada ao nível do processos
  - Dar permissões a utilizadores com base em credenciais (habitualmente uma palavra-chave)
  - Restringir o acesso a recursos (directorias, ficheiros, configurações)

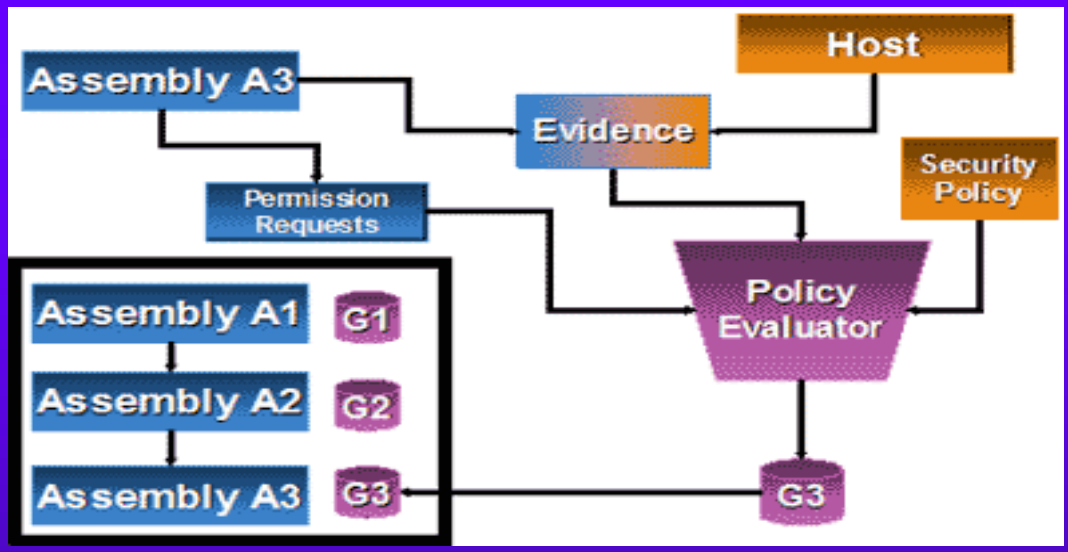
Motivação (2)

- Uma solução de segurança robusta adequada:
  - Tem de fornecer acesso a recursos de modo a que as aplicações possam operar
  - Requer um controlo de segurança fino de modo a que o código possa ser identificado, examinado e ser-lhe atribuído um grau de confiança.
- Pretende-se um mecanismo de segurança genérico:
  - Permita a execução de código vindo doutro computador
  - Mesmo que não exista uma relação de confiança
- A Segurança de Acessos por Código permite:
  - que o código possa ter vários graus de confiança consoante a origem do código e outros aspectos da sua identidade

Segurança Baseada em Provas

- Porquê ao nível da *assembly*?
  - A assembly é a unidade fundamental de instalação, controlo de versões, reutilização, activação e autorização
- De que URL foi obtida a *assembly*?
  - A política de segurança requer o endereço de origem da *assembly*
- Em que zona foi obtida a *assembly*?
  - Zonas são associações entre critérios de segurança e origens de código (e.g. Internet, intranet, máquina local)
- Qual é o *strong name* da *assembly*?
  - O *strong name* um identificador cifrado fornecido pelo autor da *assembly*
  - O *strong name* não autentica o autor
  - O *strong name* identifica univocamente a *assembly* e garante que esta não foi modificada

Segurança de Acessos por Código



- Quando uma aplicação se executa:
  - é avaliada automaticamente e recebe um conjunto de permissões do CLR
  - consoante as permissões ou se executa correctamente ou gera uma excepção

Funções da Segurança de Acessos por Código (1)

- Definir permissões e conjuntos de permissões:
  - Que representam o direito de aceder a recursos
- Permitir aos administradores configurar as políticas de segurança:
  - associar conjuntos de permissões a grupos de código (*code groups*)
- Permitir ao código requerer permissões:
  - de que necessita para se executar
  - de que seria útil dispôr
  - que nunca deverá utilizar

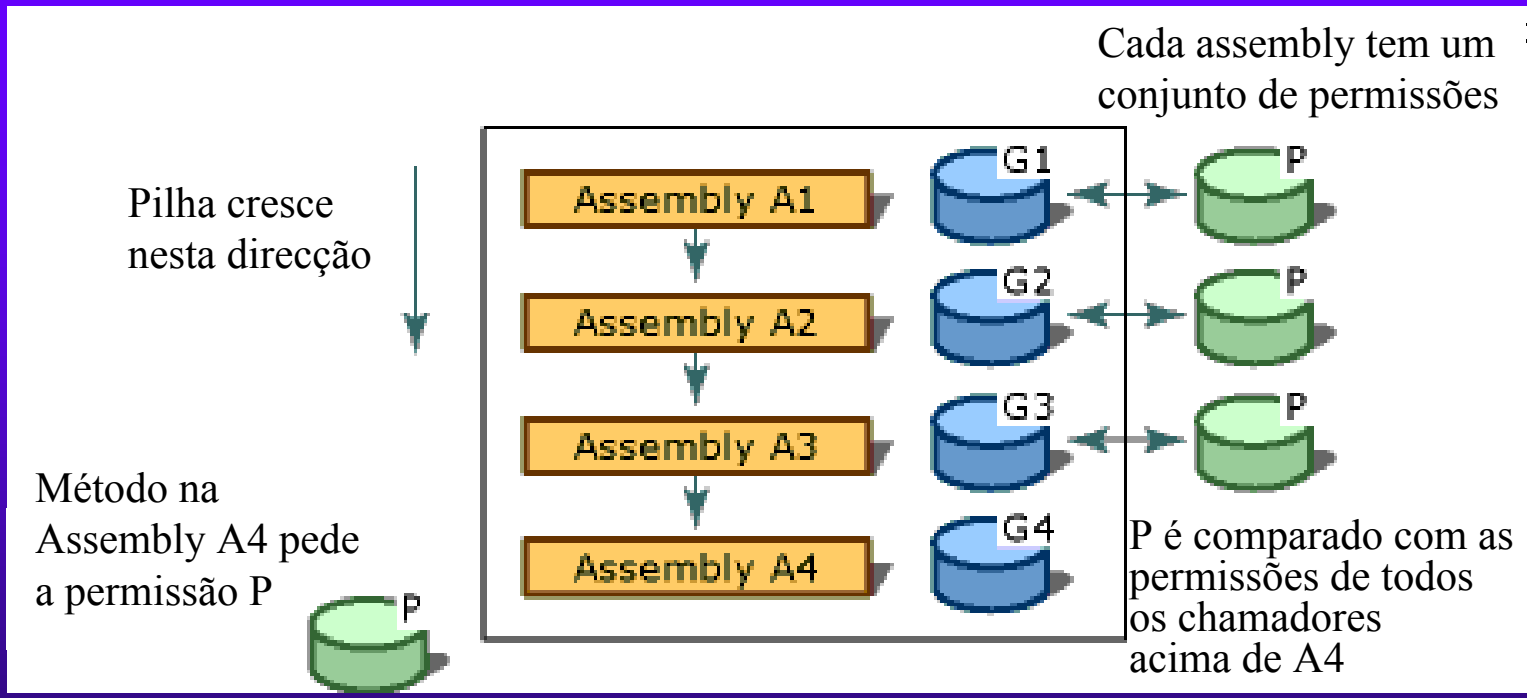
Funções da Segurança de Acessos por Código (2)

- Dar permissões a cada *assembly* que é carregada:
  - com base nas permissões pedidas e na política de segurança vigente
- Permitir ao código exigir que quem o chama:
  - tenha permissões específicas
  - tenha uma dada assinatura digital (e.g. de uma dada organização ou local)
- Impôr restricções ao código em execução:
  - comparando as permissões de cada chamador com as permissões exigidas

Análise da Pilha (1)

- Determina se um dado código pode aceder a um recurso ou realizar uma operação:
  - A análise da pilha pretende evitar que código com poucas permissões execute acções não permitidas através de chamadas a código com mais permissões.
  - A segurança do CLR analisa a pilha comparando as permissões de cada chamador com as permissões requeridas
  - Se algum chamador na pilha não tem a permissão pedida, é lançada uma excepção de segurança e o acesso é negado

Análise da Pilha (2)



Aplicação da Segurança (no carregamento)

- Por omissão, o CLR concede à *assembly*:
  - somente as permissões associados com o conjunto de permissões Internet
- Se a *assembly* tem uma assinatura digital:
  - recebe as permissões Internet
    e
  - eventualmente permissões adicionais associadas à sua assinatura.
- Informações disponíveis acerca da origem da *assembly* também influenciam o conjunto de permissões

Aplicação da Segurança (em execução)

- Sempre que o chamador acede a uma biblioteca que expõe recursos protegidos ou código *unmanaged*:
  - a biblioteca faz um pedido de segurança
- Este pedido de segurança causa:
  - a verificação das permissões necessárias junto dos chamadores

Como Funciona (1)

- Quando uma *assembly* é carregada:
  - o avaliador de política do CLR determina que permissões devem ser atribuídas reunindo as provas da *assembly* e avaliando-as no contexto da política de segurança.
  - o avaliador de política do CLR concede um conjunto de permissões baseado nas provas e em pedidos que a *assembly* faça
  - o autor de uma *assembly* pode saber que ela requer necessariamente certas permissões ou que nunca necessitará de outras
  - esse tipo de pedidos adicionais podem ser passados ao CLR através de pedidos para permissões específicas

Como Funciona (2)

- Consoante o tipo de pedido de permissões:
  - O avaliador de políticas pode limitar mais as concessões à *assembly* ou
  - recusar-se a carregar a *assembly* (se a política actual não preencher os requisitos mínimos da *assembly*)
- uma *assembly* não pode receber mais permissões do que as que receberia sem pedidos
- os pedidos só servem para limitar as concessões

Como Funciona (3)

- Quando se terminar a avaliação da política, tem-se um conjunto inicial de permissões.
- Os pedidos específicos das *assemblies* podem refinar estas concessões em 3 aspectos:
  - especificar um conjunto mínimo (**Required**) de permissões indispensáveis (se não estiverem presentes, a *assembly* não será carregada e é lançada uma excepção)
  - especificar um conjunto óptimo (**Optional**) de permissões (permissões úteis à *assembly* mas cuja inexistência não impede a execução)
  - recusar permissões que não sejam necessárias (**Refused**).
- Estes refinamentos são conseguidos com comandos declarativos aquando do carregamento

Permissões Pré-Existentes

Permission	Named permission set					
	Everything		**LocalIntranet**	**Internet**		**Nothing**
DnsPermission	All built-in unrestricted permissions, except the **SecurityPermission** for skipping verification.		Unrestricted	–		–
EnvironmentPermission			USERNAME, TEMP	–		–
FileDialogPermission			Unrestricted (read-only access)	Unrestricted (read-only access)		–
FileIOPermission			–	–		–
IsolatedStoragePermission			**AssemblyIsolationByUser** Unrestricted quota	**DomainIsolationByUser** 10240 bytes, 365 days		–
ReflectionPermisson			Unrestricted	–		–
RegistryPermisson			–	–		–
SecurityPermission			Execution, Assertion	Execution		–
SocketPermission			Connect/accept, site of origin	–		–
UIPermission			Unrestricted	**SafeSubWindows, OwnClipboard**		–
WebPermission			Connect, site of origin	–		–
	Code Group (Zone)					
	Local (My Computer)	**Intranet**	**Trusted sites**	**Internet**	**Restricted sites**	**All Code***

Criar Novas Permissões (1)

- No entanto, pode fazer sentido definir uma nova classe de permissões de acessos por código:
  - para componentes ou bibliotecas que acedem a recursos não contemplados nas classes pré-existentes
  - quando uma classes pré-existente não faz um controlo de acesso suficientemente fino
  - quando uma permissão pré-existente não protege um recurso adequadamente

Criar Novas Permissões (2)

- Implementar uma classe de permissões envolve os seguintes passos:
  - escrever algum código:
    - conceber uma classe Permission
    - implementar IPermission e IUnrestrictedPermission
    - implementar ISerializable, por questões de desempenho ou para suportar tipos de dados particulares
    - suportar conversões para XML
    - adicionar suporte para segurança declarativa o que requer a implementação de uma classe Attribute
  - pedir a permissão criada nos pontos do código relevantes
  - actualizar a política de segurança com a nova classe
  - adicionar a assembly da permissão ao repositório de código confiado

Fazer Pedidos de Segurança (1)

- Garantir que só os chamadores a que foi concedida uma permissão específica podem chamar o código:
  - Pedir declarativa ou imperativamente que quem chama o código tenha uma dada permissão.
- Um pedido faz com que o CLR faça uma verificação de segurança do código chamador:
  - O CLR analisa a pilha, examina as permissões de cada chamador e determina se a permissão pedida foi concedida a todos eles.
  - Se um chamador não tem a permissão pretendida, a verificação falha e é gerada uma excepção **SecurityException.**

Fazer Pedidos de Segurança (2)

- Pode-se causar uma verificação de segurança:
  - sempre que um método é chamado
  - antes da execução de um bloco de código
- Se se pretender uma verificação antes da execução de qualquer método de uma classe :
  - coloca-se um pedido antes da classe

Segurança Declarativa

- Pedidos declarativos:
  - colocar informação no código usando atributos
  - pode-se usar segurança declarativa para colocar um pedido ao nível da classe ou do método
  - os pedidos ao nível do método sobrepõem-se aos feitos ao nível da classe

```
[CustomPermissionAttribute(SecurityAction.Demand, Unrestricted = true)]
 public static string ReadData() {
      //Read from a custom resource.
}
```

Segurança Imperativa

- Pedidos imperativos:
  - Colocados ao nível do método criando uma instância de um objecto permissão e chamando o método **Demand.**
  - A sintaxe imperativa não pode ser usada ao nível da classe.
  - A verificação de segurança é feita aquando da chamada ao método **Demand.**

```
public static void ReadData() {
    CustomPermission MyPermission =
        new CustomPermission(PermissionState.Unrestricted);
    MyPermission.Demand();
    //Read from a custom resource
}
```

Orientações Para Fazer Pedidos de Segurança

- Para garantir que o chamador origina de um sítio particular ou foi assinado por um dado fornecedor:
  - Pode-se exigir que os chamadores tenham uma dada identidade.
- Para garantir que um objecto só pode ser criado por chamadores que tenham uma dada permissão:
  - Coloca-se o pedido no constructor dessa classe
- Para impôr restricções de segurança ao nível da *assembly* podem ser declarados atributos da *assembly*:

```
[assembly:...]
```

Forçar Verificações de Segurança

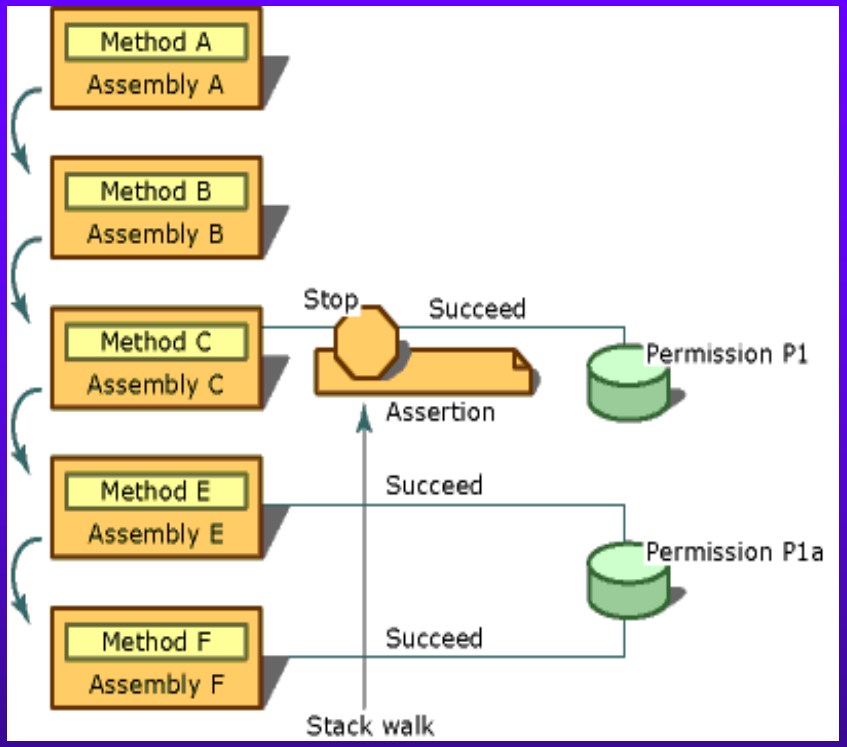
- Aplicadas ao nível da classe e do método para se sobreporem a decisões do CLR
- Invocadas quando um chamador usa o código
- Usadas para parar a análise da pilha e limitar o acesso de chamadores que já tenham certas permissões
- Podem ser perigosas, devendo ser usadas cuidadosamente
- Fazem-se chamando num objecto permissão ou conjunto de permissões:
  - Assert
  - Deny
  - PermitOnly

Assert (1)

- Método que pode ser chamado nas classes de permissões ou de conjuntos de permissões (**PermissionSet**)
- Permite ao código que chama **Assert** e a código chamado realizar acções, mas não altera as permissões dos chamadores
- Uma asserção de segurança altera a processo de análise da pilha:
  - diz à segurança do CLR para não verificar a permissão acima de quem chamou **Assert**
- Pode criar faltas na segurança por interromper o processo de verificação do CLR.

Assert (2)

- P1A: permissão para ler *.txt no disco C
- P1: permissão para ler todo o disco C
- Ambas P1A and P1 são **FileIOPermissions**, e P1A é um subconjunto de P1
- As a*ssemblies* E e F possuem P1A
- A *assembly* C faz **Assert** de P1
- As *assemblies* A e B não possuem P1 nem P1A



Assert: sintaxe declarativa

```
[C#]
namespace LogUtil {
   using System; using System.IO; using System.Security.Permissions;

   public class Log {
        public Log() { }

        [FileIOPermission(SecurityAction.Assert, All = @"C:\Log.txt")]
        public void MakeLog() {
           StreamWriter TextStream = new StreamWriter(@"C:\Log.txt");
           TextStream.WriteLine("This Log was created on {0}", DateTime.Now);
           TextStream.Close();
        }
   }
}
```

Assert: sintaxe imperativa

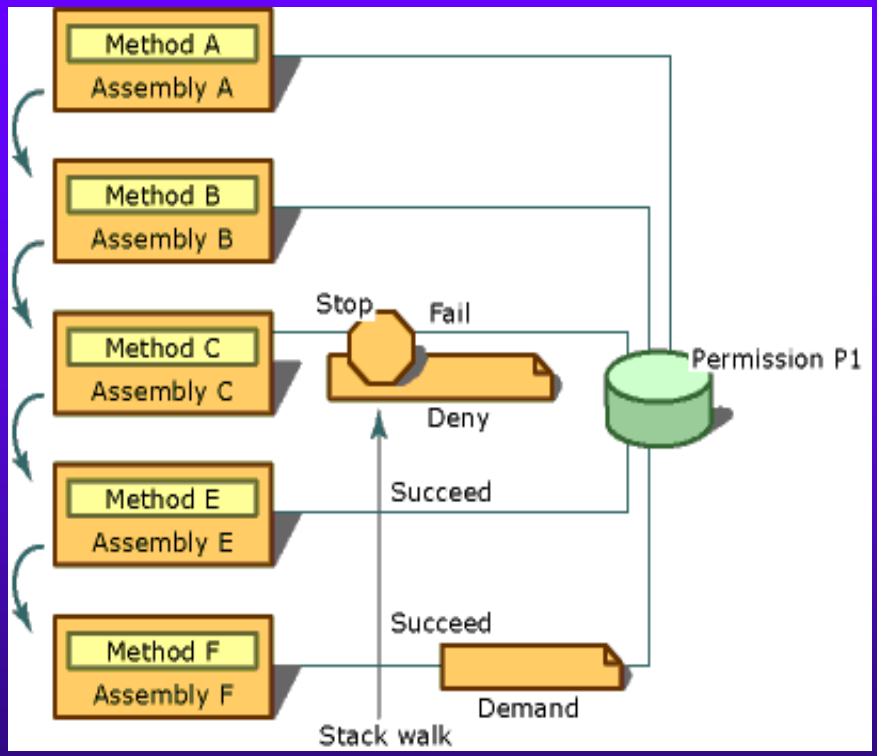
```
[C#]
namespace LogUtil {
   using System; using System.IO; using System.Security.Permissions;

  public class Log {
        public Log() { }

        public void MakeLog() {
           FileIOPermission FilePermission = new
           FileIOPermission(FileIOPermissionAccess.AllAccess,@"C:\Log.txt");
           FilePermission.Assert();
           StreamWriter TextStream = new StreamWriter(@"C:\Log.txt");
           TextStream.WriteLine("This Log was created on {0}", DateTime.Now);
           TextStream.Close();
        }
  }
}
```

Deny (1)

- Evita o acesso ao recurso especificado na permissão
- Se o código chama **Deny** e um chamador posteriormente pede a permissão negada, a verificação de segurança falhará (mesmo que todos os chamadores a possuam)

Deny (2)

- P1: permissão para ler todo o disco C
- Assemblies A, B, C, D, e E possuem P1
- Método C cria uma instância da classe P1 e chama **Deny** sobre P1
- Método F pede a permissão P1



Deny: sintaxe declarativa

```
[C#]
using System; using System.Security.Permissions;

[RegistryPermissionAttribute(SecurityAction.Deny, Write ="HKEY_XXX")]

public class MyClass {
    public MyClass() { }

    public void ReadRegistry() {
        //Access the registry
    }
}
```

Deny: sintaxe imperativa

```
[C#]
using System; using System.Security.Permissions;

public class MyClass {
    public MyClass() { }
    public void ReadRegistry() {
        RegistryPermission MyPermission =
            new RegistryPermission(RegistryPermissionAccess.Write, " HKEY_XXX ");
        MyPermission.Deny();
        //Access the registry
    }
}
```

PermitOnly

- Basicamente, é o mesmo que **Deny**, mas:
  - é uma maneira diferente de especificar as condições em que a verificação de segurança deve falhar
  - só os recursos especificados podem ser acedidos
  - pedir **PermitOnly** sobre a permissão X é o mesmo que pedir **Deny** a todas as permissões menos à permissão X.
- Garante que só os recursos especificados por este objecto-permissão podem ser acedidos através do código a que se refere o pedido mesmo que código mais acima na pilha tenha mais permissões.
- Usa-se **PermitOnly** em vez de **Deny** quando:
  - é mais conveniente descrever o que é permitido em vez do que não é permitido.

PermitOnly: sintaxe declarativa

```
[C#]
using System; using System.Security.Permissions;

public class MyClass {
    public MyClass() { }

    [UIPermissionAttribute(SecurityAction.PermitOnly, Unrestricted=true)]
    public void ReadRegistry() {
        //Access a UI resource
    }
}
```

PermitOnly: sintaxe imperativa

```
[C#]
using System; using System.Security.Permissions;

public class MyClass {
        public MyClass() { }

        public void ReadRegistry() {
           UIPermission MyPermission = new UIPermission(PermissionState.Unrestricted);
           MyPermission.PermitOnly();
           //Access a UI resource
        }
}
```

Eficiência

- Asserts e outras sobreposições são mais eficientes se feitos declarativamente
- Os pedidos (**Demand**) são mais eficientes se feitos imperativamente

Políticas de Segurança

- Modelo de Políticas de Segurança
- Política de Segurança por Omissão
- Concessão de Permissões

- Tópicos não abordados:
  - Administração de Políticas de Segurança

Modelo de Políticas de Segurança

- Engloba os seguintes elementos:
  - Níveis de política de segurança:
    empresa (*enterprise*), máquina,utilizador (e domínio de aplicações).
  - uma hierarquia de grupos de código dentro de cada um destes níveis.
  - Conjuntos de permissões com nome associados a cada grupo de código.
  - Provas que fornecem informação acerca de características do código.
  - Domínios de aplicação associados a nós que fornecem provas acerca do código ao CLR.

Níveis de Política de Segurança .Net

- Cada nível tem a sua hierarquia de grupos de código o que permite definir e configurar a política de segurança.
- As políticas para utilizadores e domínios de aplicações são menos restrictivas que aquelas para a máquina e a empresa.

Policy type	Specified by	Applies to
Enterprise policy	Administrator	All managed code in an enterprise setting.
Machine policy	Administrator	All managed code on the computer.
User policy	Administrator or user	Code in all the processes associated with the current operating system user when the common language runtime starts.
Application domain policy	Application domain host code	All managed code in the host's application domain.

Grupos de Código

- Um agrupamento lógico de código que tem uma dada condição de identidade.
- Qualquer código que verifique a condição de pertença é incluído no grupo.
- Os administradores configuram a política de segurança gerindo os grupos de código.
- Estabelecem a correspondência entre provas e um conjunto de permissões.

Membership condition	Condition based on
Application directory	The application's installation directory.
Cryptographic hash	An MD5 or SHA1 cryptographic hash.
Custom	An application- or system-defined condition.
File	The rights to access a file.
Net	The network where the code originates.
Software publisher	The public key of a valid Authenticode signat
Strong name	A cryptographically strong signature.
URL	The URL where the code originates, including example, http://site/app/*.
Web site	The Web site where the code originates; for www.microsoft.com or *.microsoft.com.
Zone	The zone where the code originates.

Conjuntos de Permissões com Nome

- Conjunto de Permissões com Nome:
  - Conjunto de permissões que os administradores podem associar a um grupo de código.
  - Consiste em, pelo menos, uma permissão, um nome do conjunto e a sua descrição.
  - Os administradores podem usá-los para definirem ou modificarem a política de segurança de um grupo de código.

Permission set	Description
Nothing	No permissions (code cannot run).
Execution	Permission to run (execute), but no permissions to use protected resources.
Internet	The default policy permission set suitable for content from unknown origin.
LocalIntranet	The default policy permission set within an enterprise.
Everything	All standard (built-in) permissions, except permission to skip verification.
FullTrust	Full access to all resources protected by permissions that can be unrestricted.

Provas (*evidence*)

- Informação que o CLR usa para tomar decisões acerca da política de segurança
- Indicação ao CLR que o código tem uma dada característica
- Exemplos comuns são assinaturas digitais e locais de origem, mas...
- pode ser qualquer informação que seja relevante para a aplicação

- O CLR usa provas fornecidas por uma máquina confiada (ou pelo carregador) para determinar a que grupo de código pertence o código e, consequentemente, que permissões recebe.

Permissões de Identidade

- As permissões de identidade são características que identificam uma *assembly*:

Class name	Identity represented
PublisherIdentityPermission	The software publisher's digital signature.
SiteIdentityPermission	The Web site where the code originated.
StrongNameIdentityPermission	The strong name of the assembly.
URLIdentityPermission	The URL where the code originated (including the protocol prefix—http, https, ftp, and so on).
ZoneIdentityPermission	The zone where the code originated. For more information, see System.Security.SecurityZone.

Política de Segurança por Omissão

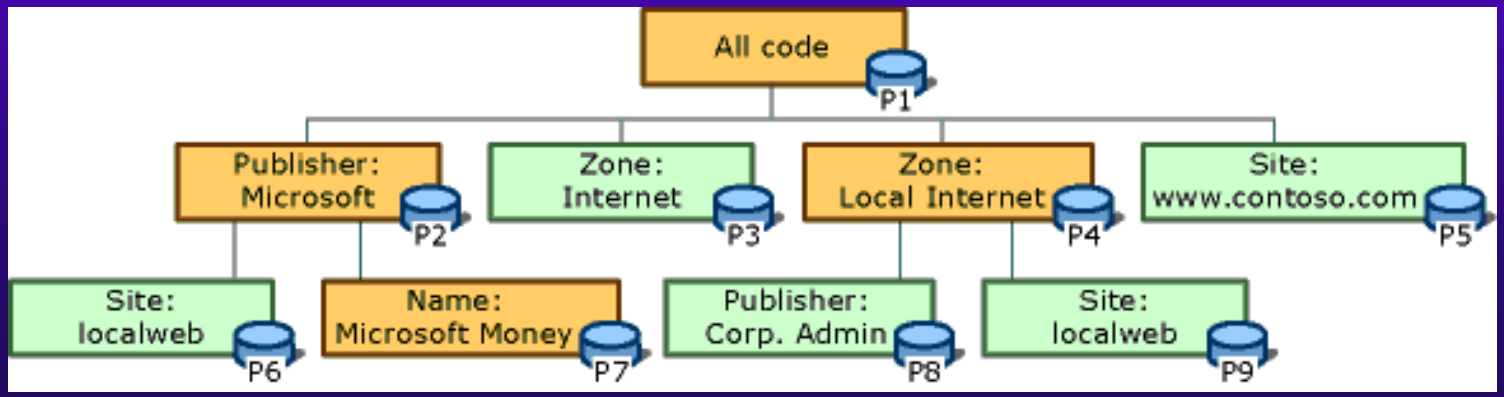
Permission	Named permission set					
	Everything		**LocalIntranet**	**Internet**		**Nothing**
DnsPermission	All built-in unrestricted permissions, except the **SecurityPermission** for skipping verification.		Unrestricted	–		–
EnvironmentPermission			USERNAME, TEMP	–		–
FileDialogPermission			Unrestricted (read-only access)	Unrestricted (read-only access)		–
FileIOPermission			–	–		–
IsolatedStoragePermission			**AssemblyIsolationByUser** Unrestricted quota	**DomainIsolationByUser** 10240 bytes, 365 days		–
ReflectionPermisson			Unrestricted	–		–
RegistryPermisson			–	–		–
SecurityPermission			Execution, Assertion	Execution		–
SocketPermission			Connect/accept, site of origin	–		–
UIPermission			Unrestricted	**SafeSubWindows, OwnClipboard**		–
WebPermission			Connect, site of origin	–		–
	Code Group (Zone)					
	Local (My Computer)	**Intranet**	**Trusted sites**	**Internet**	**Restricted sites**	**All Code***

Concessão de Permissões

- Cálculo do conjunto de permissões possíveis:
  - No carregamento, o CLR determina o conjunto de permissões que cada nível da política permite.
  - Em seguida, calcula a intersecção de todos os níveis relevantes para obter um conjunto único para o domínio de aplicação ou *assembly*.
- Para determinar as permissões concedidas:
  - O CLR compara o conjunto de permissões possíveis com as permissões pedidas pela *assembly*.

Calculo do Conjunto de Permissões Possíveis (e.g.)

- Hierarquia de grupos de código em que Microsoft Money é membro de quatro grupos:
  - all code, Microsoft (o Fornecedor), local Internet (a zona), Microsoft Money (o nome)
- O conjunto de permissões concedidas para um dado nível da política (máquina, utilizador ou domínio de aplicação) é:
  - a soma de todas os conjuntos de permissões com nome associados a esses grupos de código



Conclusão

- A segurança é reconhecidamente um aspecto importante numa plataforma de execução
- O .Net fornece um conjunto de conceitos e ferramentas potentes para tratar o problema da segurança
- Haveria muito mais a falar nesta apresentação ☹
- Para mais informação:
  - Documentação do .Net Framework SDK
  - MSDN
  - Livros (e.g. Introducing Microsoft .Net)

Nós e Domínio de Aplicações

- Application domain hosts:
  - cada aplicação corre num domínio de aplicações (nalgum nó)
  - o nó cria o domínio de aplicações e carrega *assemblies* nele
  - o nó tem acesso à informação acerca do código (provas), tais como a zona de onde o código provém ou a sua assinatura
  - um nó confiável é aquele que tem permissões para fornecer este tipo de informações ao CLR

Application domain host	Description
Browser host (Microsoft Internet Explorer)	Runs code within the context of a Web site.
Custom-designed hosts	Creates domains and/or loads assemblies into domains, including dynamic assemblies.
Server host (ASP.NET)	Runs code that handles requests submitted to a server.
Shell host	Launches applications (.exe files) from the shell.

Segurança ao nível da assembly

```
[assembly:PermissionSetAttribute(SecurityAction.RequestMinimum, Name =
    "FullTrust")]
```

Como Funciona (3)

- Uma política de segurança consiste em:
  - Conjuntos de permissões associados a grupos a que o código é atribuído com base nas provas recolhidas
- Os grupos de código podem descrever as permissões disponíveis a *assemblies* obtida de uma dada zona, ou de um dado fornecedor, etc...
- O CLR tem por omissão vários grupos de código e suas permissões, mas...
- Os administradores podem customizar este aspecto da segurança do CLR para o adaptar às suas necessidades
- Qualquer informação pode ser usada como provas desde que a política de segurança a use para identificar um grupo de código

Como Funciona (4)

- O procedimento para conceder prioridades envolve:
  - avaliar as provas para determinar que grupos de código são aplicáveis a três níveis diferentes:
    - a empresa (*enterprise*),a máquina e o utilizador
  - a política avalia esses três níveis por essa ordem, daí resultando a intersecção dos três.
  - Os administradores podem marcar um nível como **final** daí resultando a interrupção da avaliação
  - Isto permite, por exemplo, finalizar a política ao nível da máquina, não permitindo a aplicação de políticas de utilizador

Concessão de Permissões (2)

- Uma *assembly* pode conter pedidos declarativos de segurança que especificam as permissões de que o código precisa ou que pretende.
- Conceitos suportados:

Permission set	Description
Required	Specifies the minimum set of permissions the code must have to run.
Optional	Identifies permissions the code wants to have, in addition to the minimum set.
Refused	Specifies permissions that should never be granted to the code.