Viene qui utilizzata la funzione StrongNameIdentityPermissionAttribute con una richiesta di nome sicuro:

```
[StrongNameIdentityPermissionAttribute(SecurityAction.Demand,
PublicKey="…hex…", Name="App1", Version="0.0.0.0")]
public class Class1
{

}
```

7.6.10.5 Protezione e campi pubblici di sola lettura

Non utilizzare mai campi pubblici di sola lettura dalle librerie managed in quanto i campi pubblici di sola lettura possono essere modificati.
Alcune classi di framework .NET includono campi pubblici di sola lettura che contengono parametri di confine specifici per la piattaforma. Ad esempio, il campo InvalidPathChars è un array che descrive i caratteri che non sono consentiti in una stringa del percorso di file.
I valori dei campi pubblici di sola lettura come InvalidPathChars possono essere modificati dal codice o dal codice che condivide il dominio di applicazione. Se si utilizzano i campi pubblici in sola lettura, come InvalidPathCharsi, il codice dannoso può alterare le definizioni dei limiti e utilizzare il codice in modi inaspettati.
Nella versione 2.0 e versioni successive di .NET Framework, è necessario utilizzare metodi che restituiscono un nuovo array anziché utilizzare i campi di array pubblici. Ad esempio, invece di utilizzare il campo InvalidPathChars, è necessario utilizzare il metodo GetInvalidPathChars.
Si noti che i tipi di .NET Framework non utilizzano i campi pubblici per definire internamente i tipi di confini. Al contrario, il framework .NET utilizza campi privati separati. La modifica dei valori di questi campi pubblici non altera il comportamento dei tipi di .NET Framework.

7.6.10.6 Esclusione di classi e membri utilizzati da codice non attendibile

Utilizzare le dichiarazioni illustrate in questa sezione per impedire che classi e metodi specifici, nonché proprietà e eventi, siano utilizzati da un codice parzialmente attendibile. Applicare queste dichiarazioni a una classe, applica la protezione a tutti i suoi metodi, proprietà e eventi; tuttavia, si noti che l'accesso sul campo non è influenzato dalla sicurezza dichiarativa. Si noti inoltre che le richieste di collegamento aiutano a proteggere solo i chiamanti immediati e potrebbero essere ancora soggetti ad attacchi.
In associazione con il nome sicuro, un LinkDemand viene applicato a tutti i metodi, le proprietà e gli eventi accessibili a livello pubblico per limitarne l'uso a chiamanti affidabili. Per disattivare questa funzionalità, è necessario applicare l'attributo AllowPartiallyTrustedCallersAttribute. Pertanto, la selezione esplicita di classi per escludere i chiamanti non attendibili è necessaria solo per assemblies non assegnate o assemblies con questo attributo; è possibile utilizzare queste dichiarazioni per contrassegnare un sottoinsieme di tipi in esso che non sono destinati a chiamanti non attendibili.

Gli esempi seguenti mostrano come evitare che classi e membri siano utilizzati da codice non attendibile.
Esempi:
Per classi pubbliche non sealed:

```
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.InheritanceDemand,
Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name="FullTrust")]
public class CanDeriveFromMe
{
}
```

Per classi pubbliche sealed:

```
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name="FullTrust")]
public sealed class CannotDeriveFromMe
{
}
```

Per classi pubbliche abstract:

```
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.InheritanceDemand,
Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name="FullTrust")]
public abstract class CannotCreateInstanceOfMe_CanCastToMe{}
```

Per funzioni pubbliche virtual:

```
class Base1
{
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.InheritanceDemand,
Name="FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name="FullTrust")]
  public virtual void CanOverrideOrCallMe() {}
}
```

Per funzioni pubbliche abstract:

```
abstract class Base2{
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.InheritanceDemand, Name =
"FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name = "FullTrust")]
public abstract void MustOverrideMe();
}
```

Per funzioni di aggiornamento pubblico in cui la classe di base non richiede una completa fiducia:

```
class Derived : Base1
{
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.Demand, Name="FullTrust")]
  public override void CanOverrideOrCallMe()
  {
    base.CanOverrideOrCallMe();
  }
}
```

Per funzioni di aggiornamento pubblico in cui la classe di base richiede una completa fiducia:

```
class Derived : Base1
{
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name="FullTrust")]
  public override void CanOverrideOrCallMe()
  {
    base.CanOverrideOrCallMe();
  }
}
```

Per pubbliche interfacce:

```
public interface ICanCastToMe
{
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name = "FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.InheritanceDemand, Name =
"FullTrust")]
void CanImplementMe();
}
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.LinkDemand, Name = "FullTrust")]
[System.Security.Permissions.PermissionSetAttribute(
System.Security.Permissions.SecurityAction.InheritanceDemand, Name =
"FullTrust")]
class Implemented : ICanCastToMe
{
  public void CanImplementMe()
  {
  }
```