.Net Security

Proseminar *Objektorientiertes Programmieren mit .NET und C#*

Sabahattin Giritli

Institut für Informatik
Software & Systems Engineering

Agenda

- Code Acces Security
  - Permissions
  - Security Transparency Model
  - Sandboxing
- Role-Based Security
- Sicherheitskonfigurationen

Code Access Security

- Ermöglicht dem Code zu verlangen, dass seine Aufrufer
  - bestimmte Berechtigungen verfügen
  - eine bestimmte digitale Signatur besitzen

- Einschränkung von Berechtigungen

→ Schützt Ressourcen und Operationen
→ Relevant für Host und Library-Entwicklung

Permissions

- FileIOPermission
- UIPermission
- ReflectionPermission
- SecurityPermission
- …

CodeAccessPermission Klasse

→ Oberklasse jeder Permission

→ Custom Permissions

PermissionSet Klasse

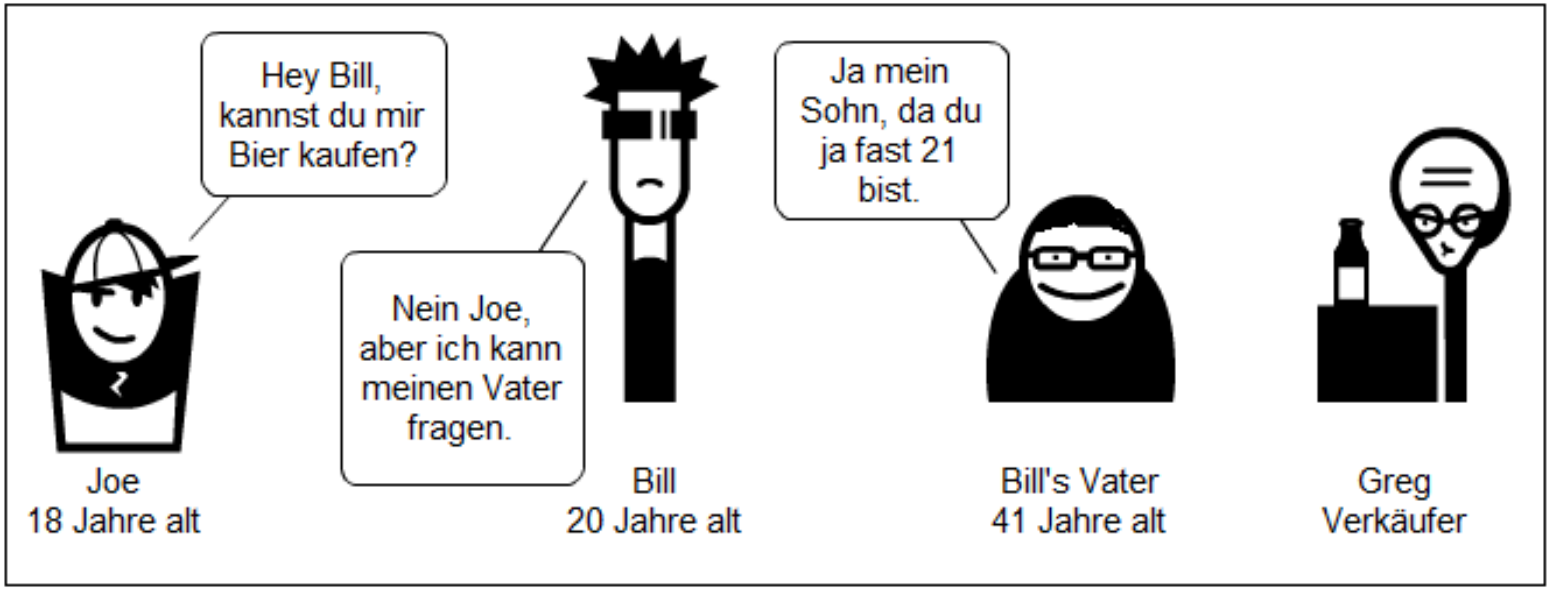
→ Assemblies besitzen jeweils eine Menge an Permissions

CodeAccessPermission Klasse

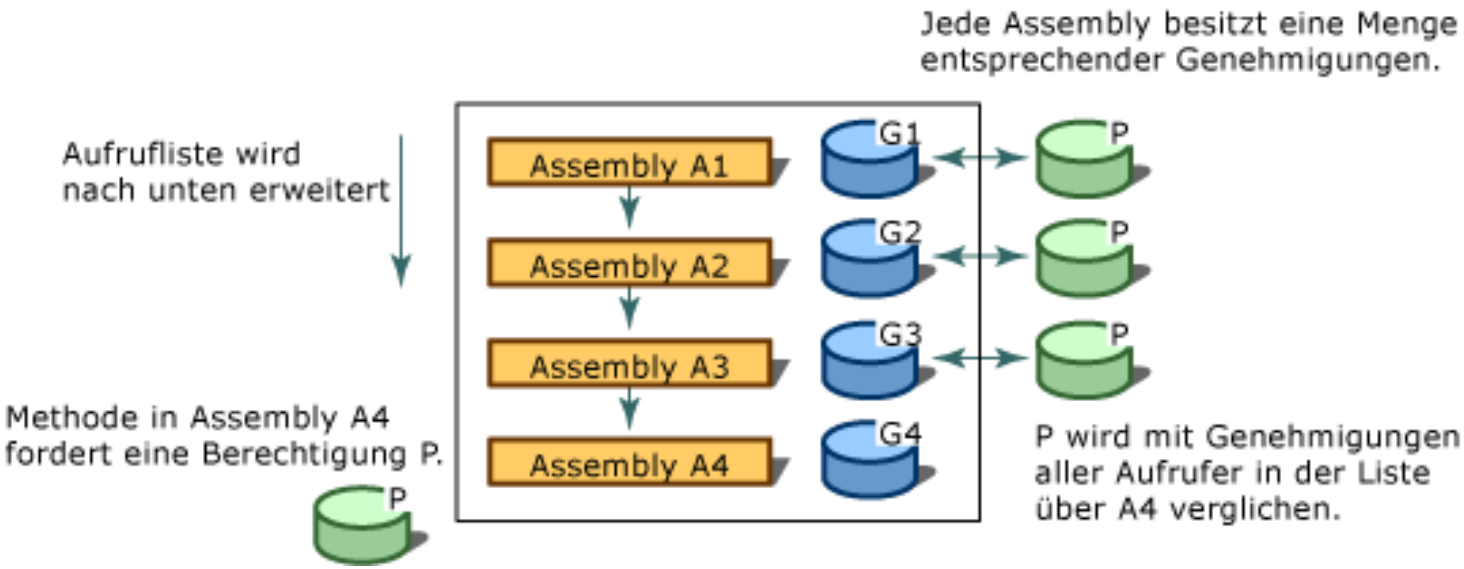
Methoden:

Demand	Fordert, dass Caller bestimmte Permission besitzen
Assert	Deklariert, dass Caller bestimmte Permission besitzen
PermitOnly	Zugriff ausschließlich auf die angegeben Ressourcen
IsSubsetOf	Ist die Permission Untermenge der anderen Permission?
Intersect	Was ist der Schnitt zwischen den Permissions?
Union	Was ist die Vereinigung?
…	

Was stimmt mit diesem Szenario nicht?

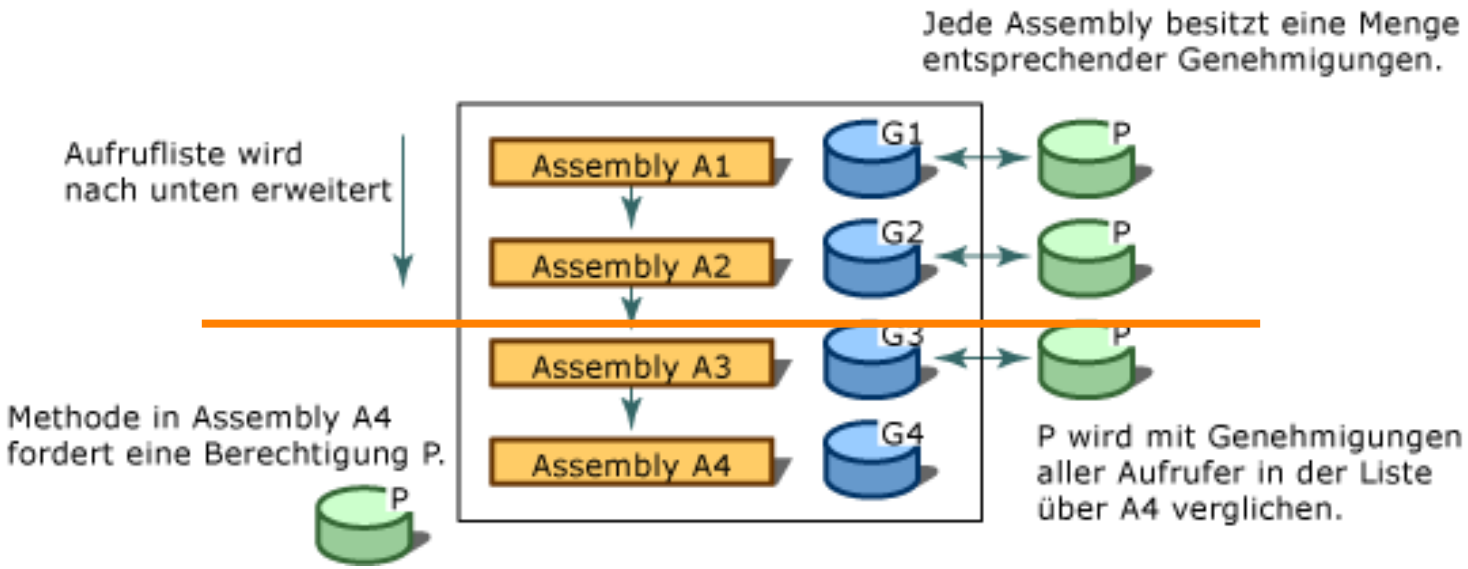


Permission Stackwalk



assert?

Permission Stackwalk



Vater übernimmt volle Verantwortung

Deklarative Syntax

- Mithilfe von Attributen auf Assembly-, Klassen- oder Memberebene
- Sicherheitsinformationen in Metadaten
- Requests, Demands, Overrides

```
[FileIOPermissionAttribute(SecurityAction.Demand,ViewAndModify =
    "C:\\example\\sample.txt")]
public class MyClass
{
    public MyClass()
    {
        //Dieser Konsturktor ist geschützt
    }

    public void MyMethod()
    {
        //diese Methode ist geschützt
    }
}
```

Imperative Syntax

- Erstellen von Permissions innerhalb von Funktionen
- Aufruf der Permission-Methoden
- Keine Requests, nur Demands und Overrides

```
FileIOPermission f = new FileIOPermission(PermissionState.Unrestricted);
try
{
    f.Demand();
}
catch (SecurityException s)
{
    Console.WriteLine(s.Message);
}
```

Security Transparency Model

Unterteilung von Code in kritische und unkritische Bereiche

Transparent
- Kein Assert und Berechtigungserweiterung
- Darf keinen unsicheren oder nicht überprüfbaren Code enthalten
- Kein Aufrufen von systemeigenem Code
- Kein direktes Aufrufen von Critical Code

Critical
- Full Trust Code, der alles darf
- Zugriff auf weiteren Critical Code

Security Transparency Model

Unterteilung von Code in kritische und unkritische Bereich

Transparent
- Kein Assert und Berechtigungserweiterung
- Darf keinen unsicheren oder nicht überprüfbaren Code enthalten
- Kein Aufrufen von systemeigenem Code
- Kein direktes Aufrufen von Critical Code

Critical
- Full Trust Code, der alles darf
- Zugriff auf weiteren Critical Code

Security Transparency Model

Unterteilung von Code in kritische und unkritische Bereich

Transparent
- Kein Assert und Berechtigungserweiterung
- Darf keinen unsicheren oder nicht überprüfbaren Code enthalten
- Kein Aufrufen von systemeigenem Code
- Kein direktes Aufrufen von Critical Code

Safe Critical
- Full Trust Code, gleichmächtig wie Critical Code
- Ermöglicht Zugriff auf zugrundeliegendenCritical Code

→ **Korrektheits- und Sicherheitsvalidation hier**

Critical
- Full Trust Code, der alles darf
- Zugriff auf weiteren Critical Code

PermissionSets von Assemlies

- Lokale (und Intranet) Applikationen laufen unter Full-Trust
- Bei hosted Assemblies entscheidet der Host
  (z.B. ASP .NET, IE, Silverlight …) über die zugeteilten Permissions
- Einschränkungen der Permissions über Sandboxing

Nutzung von Transparency-Attributen

	Assembly Level	Type Level	Member Level
SecurityTransparent	Alle Typen und Member sind Transparent	N/A	N/A
SecuritySafeCritical	N/A	Type und alle Member sind Safe Critical	Member ist Safe Critical
SecurityCritical	Alle Typen und Member sind Critical	Type und alle Member sind Critical	Member ist Critical
AllowPartiallyTrustedCallers	Alle Typen und Member sind Transparent solange nicht anders deklariert	N/A	N/A

Sandboxing

- Zum Ausführen teilweise vertrauenswürdiger Assemblies
- Isolierung vom restlichen Code
- Permissions der Sandbox werden vom Host/Entwickler selbst spezifiziert

Sandboxing

```
static void Main(string[] args)
{
    Evidence e = new Evidence();
    e.AddHostEvidence(new Zone(SecurityZone.Internet));
    PermissionSet pset = SecurityManager.GetStandardSandbox(e);

    AppDomainSetup ads = new AppDomainSetup();
    ads.ApplicationBase = "C:\\Sandbox";
    File.Copy("HelloWorld.exe", "C:\\Sandbox\\HelloWorld.exe", true);

    AppDomain sandbox = AppDomain.CreateDomain("Sandboxed Domain", e, ads, pset,
         null);
    sandbox.ExecuteAssemblyByName("HelloWorld");
}
```

Role-Based Security

- **Authentification**
  Wer ist der User?

- **Autorisation**
  Was kann der User machen?

Role-Based Security Klassen

- **Identity** Klasse kapselt Infromationen über User
- **Principal** Klasse repräsentiert Rolle und Identity eines Users

→ Windows, Generic oder Custom Identity und Principal

Überprüfung durch Permissions

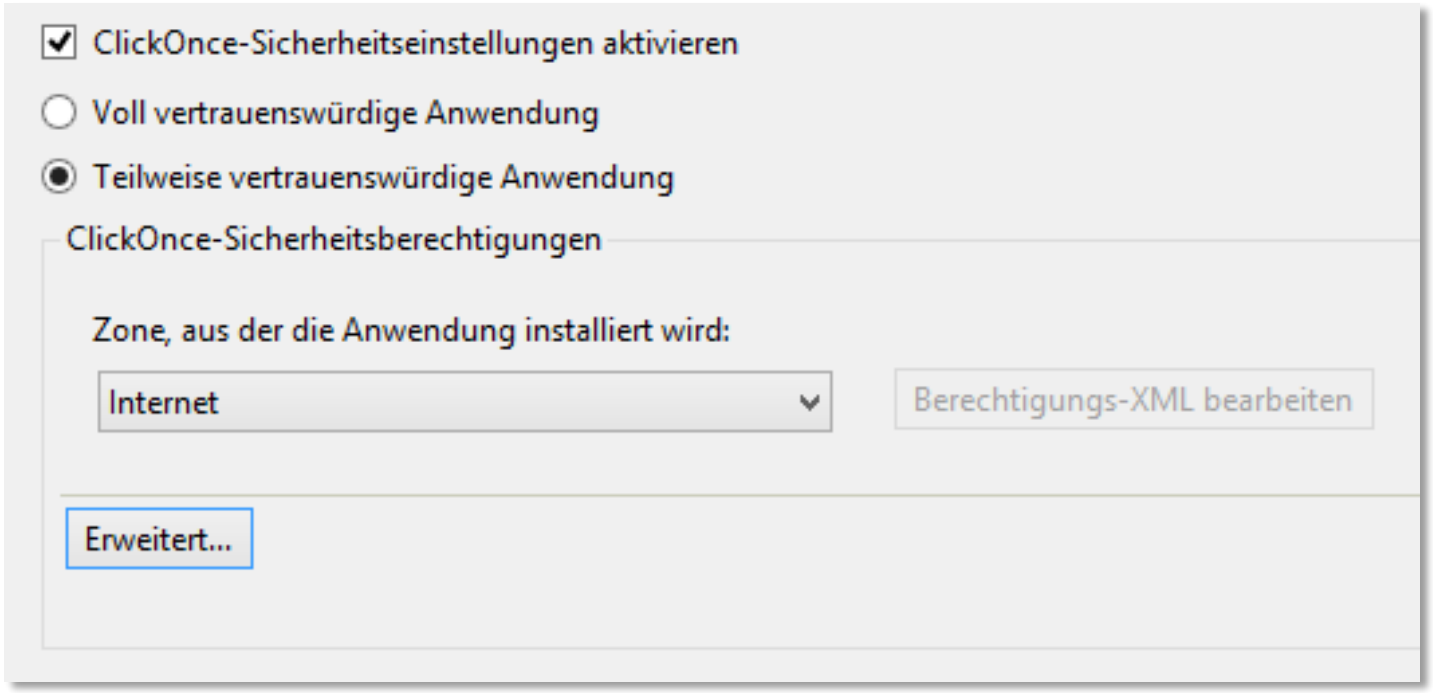
- → PrincipalPermission
- → Deklarative, Imperative Syntax

```
static void checkPricipalByPermission()
{
    AppDomain.CurrentDomain.SetPrincipalPolicy(PrincipalPolicy.WindowsPrincipal);

    PrincipalPermission p = new PrincipalPermission(null,"Administrator");

    p.Demand();

    Console.WriteLine("Hi Admin!");

}
```

Durch direkten Zugriff auf das Pricipal-Objekt

```
static void checkPricipalDirectly()
{

    WindowsIdentity wi = WindowsIdentity.GetCurrent();

    WindowsPrincipal wp = new WindowsPrincipal(wi);

    if(wp.IsInRole(WindowsBuiltInRole.Administrator));
        Console.WriteLine("Hi Admin!");

}
```

Sicherheitskonfigurationen



[Visual Studio 2012]

Vielen Dank für die Aufmerksamkeit!

Fragen?