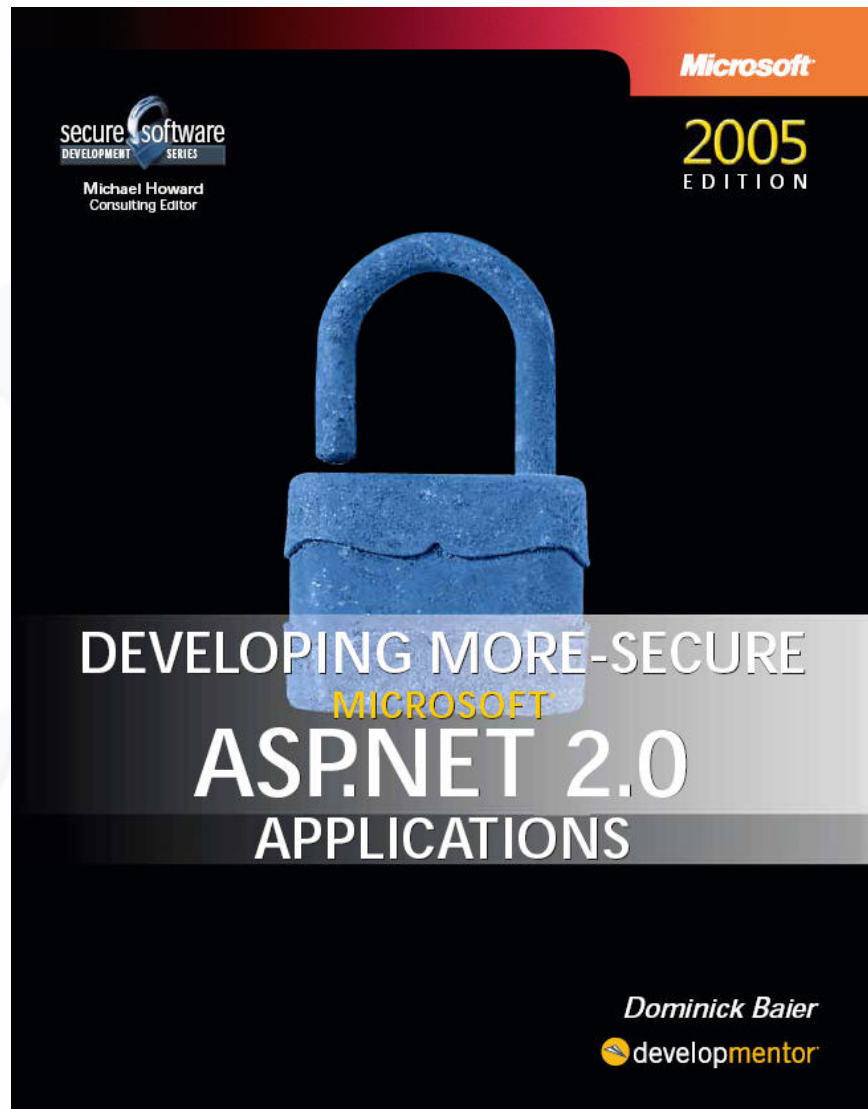


Smart Client Deployment und Security

Dominick Baier
<http://www.leastprivilege.com>



<shamelessPlug />



Deployment

Sicherheit  Produktivität

Windows Installer

- „High Impact“ Software
 - Systemweite Änderungen notwendig
 - Registry
 - COM Komponenten
 - auf System Dateien zugreifen
 - Admin Rechte benötigt
- Div. Verteilungs-Mechanismen vorhanden
 - Group Policies, Systems Management Server...

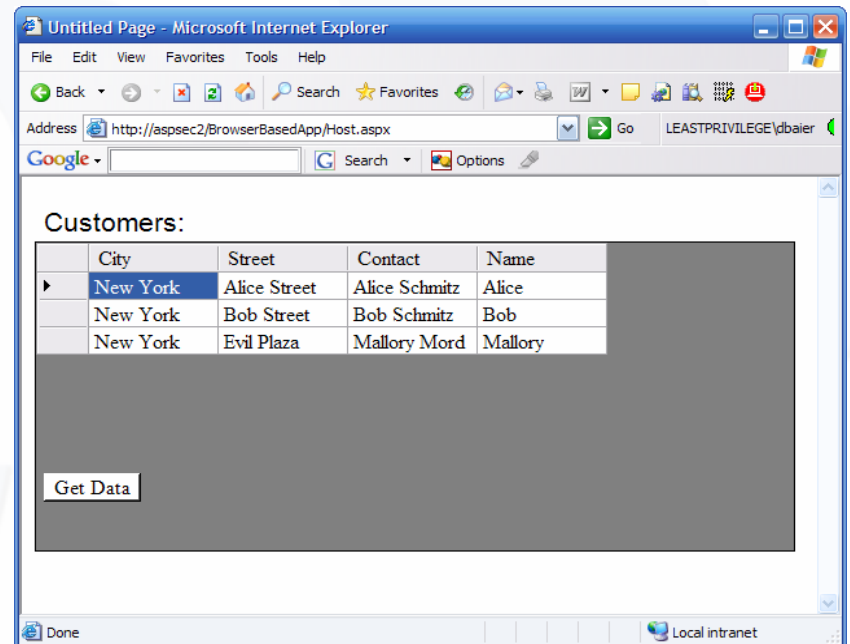
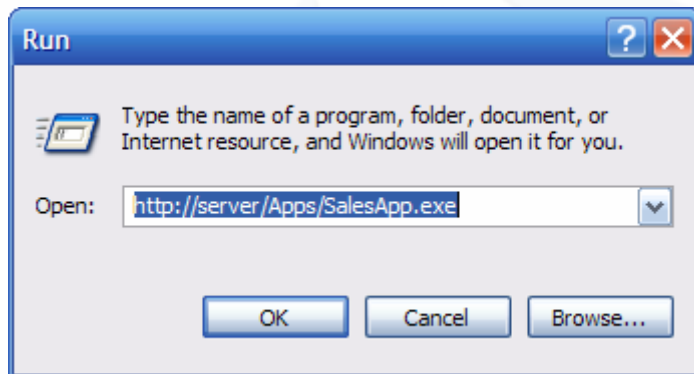
Smart Clients

- „Low Impact“ Software
 - xcopy Style
 - per-Benutzer Installation / private Daten
 - keine Admin Rechte notwendig
- Start von Share/URL
 - keine explizite Installation
- Aber: Ist das nicht auch gefährlich???

No-Touch Deployment

- Browser based apps
- HREF-EXE

```
<object id="CustomerControl "  
classid="http://Control.s.dll#.Customers"
```



Code Access Security

- Möglichkeit APIs zu deaktivieren bzw. einzuschränken
 - per default keine Einschränkungen (Full Trust)
 - CLR benutzt Zonen-Konzept um Code einzuschränken
 - Restriktiv aber Sicher
 - Anpassbar

Tips für Partially Trusted Code

- Nur managed Code verwenden
 - kein MC++, /unsafe, COM, Win32
- kein System.Runtime.Serialization
 - dafür System.Xml.Serialization
- kein Remoting, Enterprise Services, WCF
 - dafür System.Web.Services (und nur zum Herkunfts-Server)
- Benutzen von Common Dialogs
 - OpenFileDialog, SaveFileDialog, PrintDialog
- Reflection nur gegen Public Member
- Isolated Storage um Daten auf Client zu speichern

Policy Änderungen

- Nötig wenn default Policy zu restriktiv
- Muss auf **jedem** Client durchgeführt werden
 - manuell (mscorcfg.msc)
 - caspol.exe
 - programmatisch (-> .MSI Paket)
 - benötigt Admin Rechte
- Policy Design nicht trivial

Beispiel

Finden der Maschinen-Policy

```
PolicyLevel machineLevel = null;
IEnumerator policyLevelEnumerator = SecurityManager.PolicyHierarchy();
while (policyLevelEnumerator.MoveNext())
{
    PolicyLevel lvl = (PolicyLevel)policyLevelEnumerator.Current;
    if (PolicyType.Machine == lvl.PolicyType) {
        machineLevel = lvl;
        break;
    }
}
```

Beispiel

Hinzufügen eines Permission Sets

```
NamedPermissionSet nps = new NamedPermissionSet(
    "AcmeExpense Permissions", PermissionState.None);

nps.AddPermission(new FileIOPermission(PermissionState.Unrestricted));
if (null != machineLevel.GetNamedPermissionSet(nps.Name)) {
    machineLevel.ChangeNamedPermissionSet(nps.Name, nps);
}
else {
    machineLevel.AddNamedPermissionSet(nps);
}
```

Beispiel

Code Gruppe hinzufügen

secutil.exe ermittelt den public key...

```
CodeGroup myCodeGroup = new UnionCodeGroup(  
    new StrongNameMembershipCondition(  
        new StrongNamePublicKeyBlob(pubkey),  
        "AcmeExpense",  
        null),  
    new PolicyStatement(nps));  
myCodeGroup.Name = "AcmeExpense Application";  
myCodeGroup.Description = "Grants the AcmeExpense app access to ...";  
  
machineLevel.RootCodeGroup.AddChild(myCodeGroup);  
SecurityManager.SavePolicyLevel(machineLevel);
```

Speichern der Policy

ClickOnce

- CAS Modell hat sich als zu restriktiv erwiesen
 - Hohe Komplexität
 - Kunden waren unzufrieden
- ClickOnce soll diese Probleme lösen
 - Downloader „hebelt“ CAS aus
 - Neues Sicherheits-Modell (deaktiviert by default)
 - Lokale Installation per Benutzer
 - Automatische Updates

Manifeste

- Deployment Manifest (.application)
 - Dokumentiert Installations-Anforderungen
 - Installations-Quellen
 - Update-Anforderungen
 - Off-Line Verfügbarkeit
 - Digitale Signatur
- Application Manifest (.manifest)
 - Dokumentiert Laufzeit-Anforderungen
 - benötigte / optionale Dateien
 - benötigte CAS Berechtigungen
 - Digitale Signatur

Beispiel

SmartClient.application

```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly manifestVersion="1.0" >
  <assemblyIdentity ... />
  <description ... />
  <deployment ... />
  <dependency>
    <dependentAssembly
      codebase="SmartClient_1_0_0_0\SmartClient.exe.manifest">
        <hash>...</hash>
      </dependentAssembly>
    </dependency>
  <Signature>...</Signature>
</asmv1:assembly>
```

SmartClient.exe.manifest

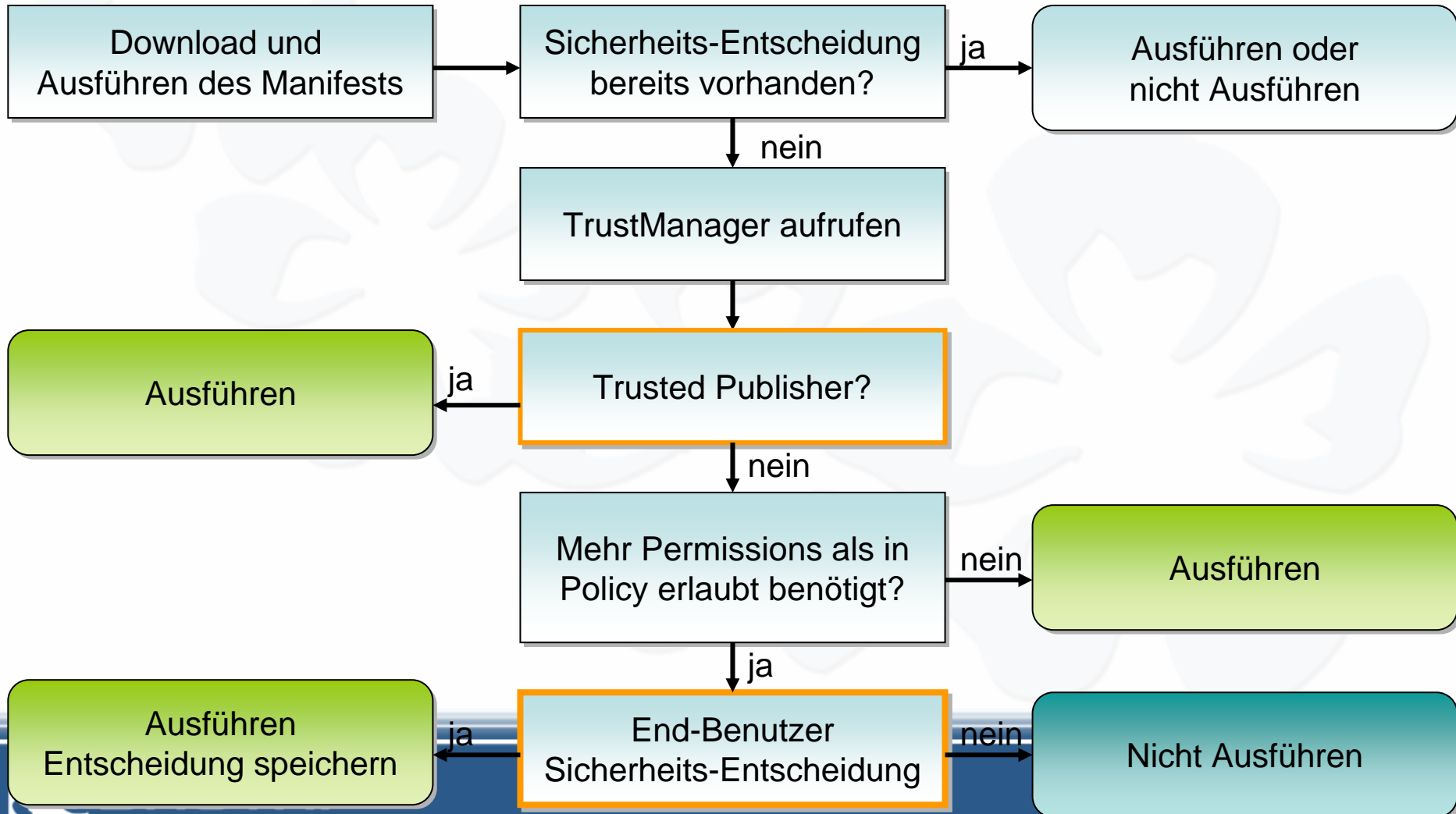
```
<?xml version="1.0" encoding="utf-8"?>
<asmv1:assembly manifestVersion="1.0">
  <asmv1:assemblyIdentity name="SmartClient.exe" version="1.0.0.0" />
  <entryPoint>...</entryPoint>
  <trustInfo>
    <security>
      <applicationRequestMinimum>
        <PermissionSet>...</PermissionSet>
      </applicationRequestMinimum>
    </security>
  </trustInfo>
  <dependency>...</dependency>
  <Signature>...</Signature>
</asmv1:assembly>
```

← Verweise auf Dateien

Sicherheits-Modell

- Ist Opt-In by Default
 - End-Benutzer führt Sicherheits-Entscheidung durch
- Extra Schritte für Lock-Down benötigt
 - Einführen von Code Signaturen
 - Konfiguration des Trust Managers
 - Automatisierbar in Active Directory Umgebungen

Sicherheits-Entscheidungen



Code Signaturen

- X509 Zertifikat für „Code Signing“ benötigt
 - VeriSign etc.
 - Interne PKI
- Clients müssen Zertifikat und CA trauen
 - Zertifikat im „Trusted Publishers“ Ordner
- Zertifikat kann über AD verteilt werden

TrustManager Konfiguration

- Via Registry Key
 - **HKLM\SOFTWARE\Microsoft\ .NETFramework\Security\TrustManager\PromptingLevel**
- Pro Zone folgende Einstellungen
 - **Enabled** - End-Benutzer entscheidet
 - **AuthenticodeRequired** - Signatur benötigt für Entscheidung
 - **Disabled** - End Benutzer kann nicht entscheiden / nur trusted Anwendungen dürfen Berechtigungen erhöhen

Default Settings

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\Security]

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\Security\TrustManager]

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\Security\TrustManager\PromptingLevel]

"My Computer"="Enabled"

"LocalIntranet"="Enabled"

"Internet"="Enabled"

"TrustedSites"="Enabled"

"UntrustedSites"="Disabled"



Zusammenfassung

- Spannung zwischen **Sicherheit** und **Einfachheit**
- **CAS** ist sicher aber restriktiv
 - CAS Policy Änderungen oft notwendig
 - mehr Funktionalität mit WPF
- **ClickOnce** vereinfacht Deployment
 - „entspannteres“ Sicherheits-Modell
 - End-Anwender trifft Sicherheits-Entscheidung
 - Lock Down notwendig in Firmen-Umgebungen