Impersonisation: Wen, wann und wie authentifizieren

Denn Sie wissen nicht, was sie tun

Mit Impersonation lässt sich Code im Kontext eines definierten Benutzers ausführen. Wer damit arbeitet, sollte genau wissen, was er tut, um nicht das Gegenteil dessen zu erreichen, was er eigentlich bezweckte.

Für die Webentwicklung mit ASP.NET und dem Internet Information Server ist Impersonation ein besonders nützliches Feature. Allerdings passiert es schnell, dass genau das Gegenteil vom eigentlichen Ziel der Impersonation erreicht wird.

In einer Multiuserumgebung sind Mechanismen erforderlich, die kontrollieren, unter welcher Identität der Code ausgeführt wird. Der Code einer Webanwendung, die nicht nur im Intranet steht, lässt sich von jedem, zu jeder Zeit und von jedem Ort aus ausführen. Da sollten alle Alarmglocken läuten. Wird jedoch ein Benutzerkonto zwischengeschaltet, gibt es nur eine Identität – die Prozessidentität –, der Berechtigungen zugewiesen werden können. Die Idee dahinter: Sollte, aus welchem Grund auch immer, jemand eigenen Code einschleusen, kann er dann am wenigsten Schaden anrichten, wenn der Code mit minimalen Berechtigungen läuft. Die Angriffsfläche wird damit schlicht und einfach minimiert.

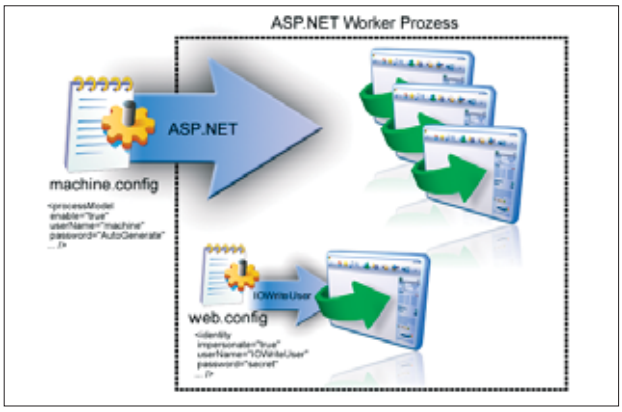

Abbildung 1 Impersonation per Konfiguration.

Auf einen Blick

Autor

Daniel Fisher ist Software Engineer bei der newtelligence AG. Er ist Microsoft Certified Professional für Webanwendungen mit C#, Experte und Editor für CodeZone.de und Managet die INETA Usergroup VfL-NiederRhein. Sie erreichen ihn über sein Blog www.lennybacon.com oder per Mail unter DanielF@newtelligence.com.

dotnetpro.code
A0510Impersonation

Sprachen C#

Technik ASP.NET

Voraussetzungen .NET 1.1

Lösungsansätze

Berechtigungen für die Prozessidentität setzen: Berechtigungen, Access Control Lists (ACLs), für den ASPNET/Network-Service-Benutzer für die benötigten Ressourcen setzen.

Privilegien für die Prozessidentität vergeben: Kommen ACLs nicht in Frage, beispielsweise weil sie zu aufwändig sind, etwa bei Lesezugriff auf alle Dateien, oder wenn ACLs das Problem nicht lösen, kann die Prozessidentität mit den erforderlichen Privilegien ausgestattet werden.

Prozessidentität ändern: Für alle Webanwendungen lässt sich in der machine.config ein eigener Benutzer konfigurieren.

Webanwendungs-Prozessidentität ändern: In der web.config wird für diesen Zweck ein eigener Benutzer konfiguriert, der Webanwendungen ausführen darf.

Web-Form-Prozessidentität ändern: Mithilfe einer <location>-Sektion wird in der web.config ein Benutzer konfiguriert, der eine einzelne Web Form ausführt.

Aufrufidentität ändern: Welche Codezeilen oder Aufrufe unter einer anderen Identität ausgeführt werden sollen, wird programmatisch definiert.

Offenes Scheunentor

Die ersten drei Lösungsmöglichkeiten, Berechtigungen oder Privilegien für die Prozessidentität zu vergeben sowie das Ändern der Prozessidentität widerstreben der Absicht, den Code von Webapplikationen in einer gesicherten Umgebung, einer Sandbox, ablaufen zu lassen. Diese Optionen würden ein Scheunentor aufmachen und eher das Gegenteil vom Ziel der Impersonation bewirken.

```
...
<processModel enable="true"
 userName="machine" password="AutoGenerate"
... />
...
```

Auch mit dem Ändern der Webanwendungs-Prozessidentität kann man noch mit Kanonen auf Spatzen schießen. Es

Abbildung 2 Sicherheitsrisiken und impersonierter Code.

lohnt sich zu klären, ob diese außerordentliche Berechtigung wirklich in jeder einzelnen Web Form benötigt wird.

```
<identity impersonate="true" userName=
   "dbreadwrite" password="password" />
```

Die Impersonation seitenweise, also pro Web Form, durchzuführen, verringert das Risiko beträchtlich. Angenehmer Nebeneffekt ist dabei, dass jede einzelne Seite, die Impersonation verwendet, an einer zentralen Stelle, der web.config, aufgelistet wird.

```
...
<location path="upload.aspx">
 <system.web>
  <identity impersonate="true"
   userName="dbreadwrite" password="password" />
 </system.web>
</location>
...
```

Aufrufidentität ändern

Der Lösungsweg, den Kontext des ausführenden Benutzers nur für die Aufrufe zu ändern, die eine höhere Berechtigung erfordern, ist zwar nicht so übersichtlich, wie das Impersonieren pro Web Form, die Anwendung wird jedoch noch ein Stück sicherer.

Ein weiterer Vorteil gegenüber der deklarativen Lösung ist die Möglichkeit, den Benutzernamen und das Passwort in der web.config verschlüsselt abzulegen.

Ohne Impersonation arbeiten

Der Lesezugriff auf eine Datei, die auf einer Netzwerkresource liegt, lässt sich auch ohne Impersonation oder das Setzen von Berechtigungen und Privilegien realisieren. Ein weiterer Internet Information Server, auf den die Webanwendung per HTTP zugreift, kann in einem solchen Fall den anonymen Lesezugriff auf Dateien zur Verfügung stellen.

Fazit

Man sollte so früh als möglich klären, ob und wo Impersonation erforderlich ist, und nur darauf zurückzugreifen, wenn es keine Alternative zur Impersonation gibt. Auch wenn sie genutzt wird, versteht sich von selbst, dass die Tür zur Webanwendung nur einen so kleinen Spalt geöffnet werden sollte, dass gerade die gewünschte Operation auszuführen ist – sonst hat man schnell ungeladene Gäste, die man selbst hereingelassen hat.

|||||||

[1] msdn – Configuring the ASP.NET process Identity, msdn.microsoft.com/library/en-us/cpguide/html/cpconconfiguringaspnetprocessidentity.asp
[2] msdn – ACL's required to execute ASP.NET, msdn.microsoft.com/library/en-us/cpguide/html/cpconaspnetrequiredaccesscontrollistsacls.asp
[3] msdn – How To: Create a DPAPI Library, msdn.microsoft.com/library/en-us/dnnetsec/html/SecNetHT07.asp
[4] lennybacon.com – Re: Impersonation, www.lennybacon.com/PermaLink,guid,42d8a312-db3f-48d3-80f7-e6513ccdb2bb.aspx

Listing I

Programmatisches Festlegen der Identität, unter der eine Codezeile oder ein Aufruf ausgeführt wird.

```
...
public class WebForm1 : System.Web.UI.Page
{
const int LOGON32_LOGON_INTERACTIVE = 2;
const ...

[DllImport("advapi32.dll", SetLastError=true)]
public static extern int LogonUser(
string lpszUsername,
string lpszDomain,
string lpszPassword,
int dwLogonType,
int dwLogonProvider,
out IntPtr phToken );

[DllImport("advapi32.dll",  SetLastError=true)]
public static extern int ImpersonateLoggedOnUser(
IntPtr hToken
);

[DllImport("advapi32.dll", SetLastError=true)]
static extern int RevertToSelf();

[DllImport("kernel32.dll", SetLastError=true)]
static extern int CloseHandle(IntPtr hObject);

private void Page_Load(object sender, System.EventArgs e)
{
IntPtr lnToken;
int TResult = LogonUser("IOWriteUser",".","P@sswOrt",
LOGON32_LOGON_NETWORK,LOGON32_PROVIDER_DEFAULT,
out lnToken);

if ( TResult > 0 )
{
ImpersonateLoggedOnUser(lnToken);

/*Code, der unter anderem Benutzer ausgeführt werden soll*/

RevertToSelf();
CloseHandle(lnToken); }}}
...
```