

Re: server authentication & ASP authentication

Source: <http://www.derkeiler.com/Newsgroups/microsoft.public.sqlserver.security/2002-07/1142.html>

From: Aaron Margosis [MS] (aaronmaronline@microsoft.com)

Date: 07/07/02

From: "Aaron Margosis [MS]" <aaronmaronline@microsoft.com>

Date: Sun, 7 Jul 2002 00:47:31 -0400

Yes, 100% correct, with some clarifications:

* You are correct --- Integrated Windows authentication works only with Internet Explorer. No logon UI is presented if the user is already logged on to the client workstation with an authorized Windows account. (I don't know whether Integrated Windows works with IE on non-Windows platforms.)

* Your SQL administrative tasks are greatly reduced if you create SQL logins for Windows groups (domain or local) instead of for each Windows user account. Each member of an authorized group can authenticate. You can then add or remove members of the group in Windows; you don't need to make corresponding changes in SQL. (Note though, that users log on, not groups, so the SYSTEM_USER variable will always identify a specific user, not a group.)

* Regarding performance: there are two common ways that applications use SQL Server with Windows authentication. One is to flow the caller's Windows security context all the way to the database. The other is to call a middle-tier component (e.g., a COM+ application) that runs under a specific Windows identity. In the latter case, that component's identity is the one that is granted access to the database. That component must authorize the caller's request (e.g., via roles) before making the database request on the caller's behalf. While there are security benefits to the former approach, it also needs to create a separate database connection for each user. The latter approach scales better, because all callers can share the same pool of database connections. In your scenario, however, that would *probably* not be an issue.

* The connection string will not contain a username or password; instead it will contain something like "Integrated Security=SSPI" (the specifics depend on whether you're using ODBC, OLEDB/ADO, .NET, ...).

* Regarding server variables, here's what's in the docs:

AUTH_USER: The name of the user as it is derived from the authorization header sent by the client, before the user name is mapped to a Windows

account. This variable is no different from REMOTE_USER. If you have an authentication filter installed on your Web server that maps incoming users to accounts, use LOGON_USER to view the mapped user name.

LOGON_USER: The Windows account that the user is impersonating while connected to your Web server. Use REMOTE_USER or AUTH_USER to view the raw user name that is contained in the request header. The only time LOGON_USER holds a different value than these other variables is if you have an authentication filter installed.

REMOTE_USER: The name of the user as it is derived from the authorization header sent by the client, before the user name is mapped to a Windows account. If you have an authentication filter installed on your Web server that maps incoming users to accounts, use LOGON_USER to view the mapped user name.

You should use object permissions in SQL to grant access to specific stored procedures, etc., based on group membership. That is the most effective way to guarantee that unauthorized users do not gain access to those parts of the database they shouldn't access. To determine group membership without incurring "access denied" (e.g., to present the right UI for the user), you can use "IsCallerInRole" from COM+ apps, or ADSI calls from your ASP code. Example (testing membership in a local group "WorkerBees"):

```
dim oGroup
set oGroup = GetObject("WinNT://MyServer/WorkerBees,group")
' You will need to get the username into this format: "WinNT://" +
domain + "/" + username
if oGroup.IsMember(sUser) then
' your code here...
```

To secure the machine from interactive logon:

By default, Basic authentication creates a "local" logon on the web server, so users need the logon locally privilege. In order to secure the machine from users coming up to the console and interactively logging on, you may just need to physically secure the machine. There is also an IIS metabase setting that will make Basic auth logons "network" logons instead (I'd have to look up how to do this). With this setting, you could then deny local logon privileges to these users while still allowing access through the web site. (With this setting, the SQL Server would have to remain on the same machine with IIS -- the users' credentials would not be able to flow to a remote machine.)

You may also want to prevent users from connecting directly to SQL Server. One way, since IIS and SQL are on the same machine, would be to disable all the network libraries and just use shared memory as the connection mechanism. That will allow local connections only. (That also means only local administration.)

HTH

-- Aaron

"Jakub Jablonski" <jakubjab@data.pl> wrote in message
news:3D262E65.1050506@data.pl...

> Aaron,

>

> *This is all intranet, but some day it may be internet also. We use
> different versions of Windows, Unix, Linux and all sorts of browsers as
> clients.*

> *The SQL Server and web server are on the same machine (Windows 2000).*

>

> *Let me make sure I understood correctly your suggestion:*

>

> *1. Create Windows account for each employee and the local group for each
> group of employees on the machine where IIS and SQL Server run.*

>

> *2. Disable Anonymous and enable Basic authentication and SSL on the web
> server (Integrated Windows doesn't work with Netscape, am I correct?)*

>

> *3. Use windows instead of standard authentication on SQL Server. Do I
> also need to create a server login and a database user for each
> employee? Does it affect performance to have 120 logins in SQL Server?*

>

> *Then the sequence is (correct me if I'm wrong):*

> – *employee opens web browser and sends request to the IIS.*

> – *IIS responses with the request for basic authentication*

> – *employee enters his/her login and password*

> – *IIS accepts credentials and runs the ASP application under that user's
> context*

> – *ASP application connects to the SQL Server using connection string
> without password*

> – *SQL Server recognizes the user and grants him/her appropriate permission*

s.

>

> *I guess I can find the login name in ServerVariables. What is the
> difference between AUTH_USER, LOGON_USER and REMOTE_USER variables in my
> case? How to find the group of the user?*

>

> *How can I prevent users from logging in directly to the machine where
> they have accounts, and use only the web application?*

>

> *Thank you for the answer*

> *Regards,*

> *Jakub Jablonski*

>

>

> *Aaron Margosis [MS] wrote:*

> > *Is this all intranet? If not, is there a firewall between your web
server*

> > *and database? Are the web server and DB on the same machine?*

> >

> > *My primary inclination would be to use platform authentication across
the*

microsoft.public.sqlserver.security: Re: server authentication & ASP authentication

> > *board. That is, create Windows accounts for each of the 120 users. Use*
> > *Integrated Windows or Basic authentication on the web server (disallow*
> > *anonymous access), and Windows authentication to the database. This*
does a
> > *number of things for you that you would otherwise have to do in code:*
> > ** Secure management of credentials. Passwords are hashed, rather than*
> > *stored in clear text (the original password cannot be derived from the*
> > *hash). The hashes are accessible only to the OS. A user's password can*
be
> > *changed only by the user or an administrator. It is also easy to*
enforce
> > *password complexity, password expiration, and account lockout after a*
> > *specified number of unsuccessful attempts.*
> > ** With Integrated Windows (NTLM or Kerberos), passwords are not*
transmitted
> > *over the network in the clear. With Basic authentication, add SSL to*
> > *achieve the same.*
> > ** EVERY entry point to your application and database enforces*
authentication
> > *and authorization.*
> > ** Validation checks are performed correctly every time.*
> > ** Subsequent requests in the same user session are correctly associated*
with
> > *the initial authentication. You don't need to set/get encrypted cookies*
to
> > *determine who the user is.*
> >
> > *With SQL Server, it is easy to map Windows groups to roles. Groups can*
be
> > *defined locally -- you don't need them to be established at the domain*
> > *(although you could do that too).*
> >
> > *Embedding passwords in text files on the hard drive is not secure --*
> > *especially if the file lives at or under your web application's vroot.*
> > *There have been vulnerabilities in the past that allow an attacker's*
request
> > *for a specific file (e.g., a .asp page) to download the raw file rather*
than
> > *run it on the server. (In other words, a user could download the asp*
page
> > *that contains the username/password.)*
> >
> > *HTH*
> >
> > *-- Aaron*
>