

## Re: Cross-database execution permissions with certificates and sch

---

*Source:* <http://www.derkeiler.com/Newsgroups/microsoft.public.sqlserver.security/2008-02/msg00045.html>

---

- *From:* "Dan Guzman" <[guzmanda@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:guzmanda@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)>
  - *Date:* Fri, 8 Feb 2008 07:02:51 -0600
- 

Thanks for laying it out -- your code sample turned on the lights for me.

I'm glad I was able to help.

What I meant by schema scoped permissions is the use of another perimeter around the service broker activated user. I want to create the objects on which this user requires permissions in a schema that is owned by a user with no login, and not owned by the database owner. That way this service broker activated user cannot access objects in other schemas in this database.

I agree that this is a good use of schema as a security boundary. Coincidentally, I mentioned this in my blog post this week

<http://weblogs.sqlteam.com/dang/archive/2008/02/03/Keep-Schema-and-Ownership-Simple.aspx>.

--

Dan Guzman  
SQL Server MVP

"Craig Thomas" <[CraigThomas@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx](mailto:CraigThomas@xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx)> wrote in message [news:A2123E5C-6B84-490E-9465-5CE2E59809FD@xxxxxxxxxxxxxxxxxxxxx](mailto:news:A2123E5C-6B84-490E-9465-5CE2E59809FD@xxxxxxxxxxxxxxxxxxxxx)

Dan,

Thanks for laying it out -- your code sample turned on the lights for me.

The reason why things were working for me when I was using the [dbo] user in both databases is not (as I posited) some magical property of the [dbo]. It was simply because [dbo] user was present in both databases. The only magic here is that the [dbo] user is created when the database is created -- not by some explicit step I took.

The comment about schema scoped permissions not being applicable in my service broker activated stored procedure is probably a misconception layered on my misunderstanding about the properties of the OWNER of the stored

Re: Cross–database execution permissions with certificates and sch  
procedure.

What I meant by schema scoped permissions is the use of another perimeter around the service broker activated user. I want to create the objects on which this user requires permissions in a schema that is owned by a user with no login, and not owned by the database owner. That way this service broker activated user cannot access objects in other schemas in this database.

My motivation here is that the service broker activated procedure will have to execute dynamic sql, which introduces the possibility of a sql injection threat. Our application code will make every effort to assure that sql injection is defended against, however, the threat exists anywhere EXECUTE (@sqlString) (or sp\_executesql) occurs.

My understanding of least–permissions security leads me to the choice of sequestering the service broker activated user execution context to the narrow set of procedures and views that it needs to do the job it is intended to do, by confining permissions grants to single schema containing only a few objects. (I know I could grant it permissions on the objects explicitly, but leveraging the schema as a container for the grants makes managing them easier.)

Certificate based authentication will be used to allow the service broker activated stored procedure to access any resource it depends upon which is outside of its close–walled schema–scoped perimeter – views, stored procedures, and functions in other databases or even within the same database.

Thanks again, very much, for leading me through this.

--Craig.

"Dan Guzman" wrote:

> The evidence that Dan, Uri, and I find suggests that the owner must be >  
> the  
> database principal [dbo]. (If this understanding is wrong, I'd > appreciate  
> learning about it!)

Whatever principal you impersonate with EXECUTE AS must have a security context in both databases but it doesn't need to be dbo. As Uri and I mentioned earlier, the second script will work if you add the impersonated principal (or guest) to providerDB. No permissions need be granted since the user is only needed for context:

```
USE providerDB,  
CREATE USER dispatcher  
GO
```

--this now works

Re: Cross–database execution permissions with certificates and sch

```
USE [dispatcherDB]
GO
EXECUTE AS LOGIN = 'dispatcher'
GO
EXECUTE [dispatch]
GO
REVERT
GO
```

The reason EXECUTE AS OWNER works with dbo as the schema owner is that 1) the certificate is a trusted authenticator because you granted AUTHENTICATE and 2) the impersonated dbo principal exists in dispatchDB by virtue of the fact that both databases are owned by the same server principal (sa).

> The upshot here seems to be that I can't employ schema–scoped > permissions  
> on  
> the execution context of my service broker activated stored procedure.  
>  
> Is there any truth to this?

You can still use the certificate as the authenticator as long as the impersonated principal has a security context in the referenced database as well. I'm not exactly sure I understand what you mean by schema–scoped permissions, though. Can you elaborate?

--  
Hope this helps.

Dan Guzman  
SQL Server MVP

"Craig Thomas" <CraigThomas@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message  
[news:B3F2FDA3-9C46-4203-9FB5-B373204D83E6@xxxxxxxxxxxxxxxxxxxx](mailto:news:B3F2FDA3-9C46-4203-9FB5-B373204D83E6@xxxxxxxxxxxxxxxxxxxx)  
> Dan, Uri,  
>  
> Thanks much for looking into this, for the followup and the references.  
>  
> I confess, though, that I still don't see the principals at play here > (pun  
> intended).  
>  
> In the article Uri references  
> (<http://www.sommarskog.se/grantperm.html#ownershipchaining>) there is a  
> section that describes the perils of using "EXECUTE AS OWNER",  
> particularly  
> when the owner has plenty of privileges. Yet in every reference I read  
> about  
> Service Broker, EXECUTE AS OWNER is the way to achieve the >

Re: Cross–database execution permissions with certificates and sch

cross–database

> permissions needed for an "activated" stored procedure.

>

> The evidence that Dan, Uri, and I find suggests that the owner must be > the

> database principal [dbo]. (If this understanding is wrong, I'd > appreciate

> learning about it!)

>

> The upshot here seems to be that I can't employ schema–scoped > permissions

> on

> the execution context of my service broker activated stored procedure.

>

> Is there any truth to this?

>

> Any and all insights much appreciated.

>

> Thanks,

> --Craig.

>

>

> "Dan Guzman" wrote:

>

>> Thanks for the repro scripts.

>>

>>> Did I discover some magic property of the [dbo] user and [dbo] >>> > schema

>>> (not

>>> likely), or am I missing something conceptual about permissions >>> > grants

>>> through certificates?

>>>

>>> The dbo user is special for a number of reasons. However, I haven't

>>> worked

>>> with certificates much so I'm not exactly sure how this works with

>>> cross–database authentication.

>>>

>>> I'm short on time right now to identify the root cause but I'll share

>>> some

>>> of my observations when I played around with your scripts. The issue

>>> seems

>>> to be related to EXECUTE AS OWNER. The second script works when I

>>> change

>>> the "dsp" schema authorization to dbo but then breaks if the database

>>> owners

>>> are different. It also works if I add the dispatcher user to >>> providerDB

>>>

>>> -- >>> Hope this helps.

>>>

>>> Dan Guzman

>>> SQL Server MVP

Re: Cross-database execution permissions with certificates and sch

```
>>
>> "Craig Thomas" <CraigThomas@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote
in >> message
>>
news:62334118-872D-49FA-9BB8-4EFFF114F9CF@xxxxxxxxxxxxxxxxxxxx
>> >I have run into an obstacle while implementing cross-database >>
>execution
>> > permissions using certificates and schemas.
>> >
>> > The scenario I'm working with is pretty simple, and I've got a >> >
>formula
>> > that
>> > works and a formula that doesn't (complete with scripts below). Can
>> > someone
>> > take a look and help me out?
>> >
>> > Database 1 (named [dispatcherDB]) has a stored procedure called
>> > [dispatch].
>> > This stored procedure executes a procedure in database 2 (named
>> > [providerDB]).
>> >
>> > I follow these steps to create everything and grant cross-database
>> > permissions:
>> > 1. Executing as login='sa':
>> > 1.1 create a database [dispatcherDB], and a login [dispatcher],
>> > 1.2 create a second database [providerDB], and a login [provider],
>> > 1.3 in the [dispatcherDB] create a user [dispatcher],
>> > 1.4 make the dispatcher user the owner (EXEC sp_addrolemember
>> > N'db_owner',
>> > N'dispatcher')
>> > 2. Executing as user='dispatcher':
>> > 2.1 create a stored procedure [dispatch],
>> > 2.2 create a certificate,
>> > 2.3 sign the stored procedure (ADD SIGNATURE TO
OBJECT::[dispatch] >> > ...)
>> > 2.4 Discard the private key from the certificate,
>> > 2.5 Back up the certificate to a file.
>> > 3. Executing as login='sa':
>> > 3.1 in the [providerDB] create a user [provider] with a default >> >
>schema
>> > [pvd]
>> > 3.2 make the provider user the owner (EXEC sp_addrolemember
>> > N'db_owner',
>> > N'provider')
>> > 4. Executing as user='provider':
>> > 4.1 create a schema [pvd]
>> > 4.2 create a certificate from the file.
>> > 4.3 Create a user from the certificate,
>> > 4.4 GRANT AUTHENTICATE TO the certificate user,
>> > 4.5 GRANT EXECUTE ON [providerDB].[pvd].[testProc] to the >> >
certificate
```

## Re: Cross-database execution permissions with certificates and sch

```
>>> user.
>>> 5. Executing as user='dispatcher'
>>> 5.1 execute [dispatch]
>>>
>>> Everything works in this case.
>>>
>>> But here's a description of the subtly different case (which is the
>>> condition my application demands), which does not work:
>>> 1. Executing as login='sa':
>>> 1.1 create a database [dispatcherDB], and a login [dispatcher],
>>> 1.2 create a second database [providerDB], and a login [provider],
>>> 1.3 in the [dispatcherDB] create a user [dispatcher] *** with a >>>
default
>>> schema [dsp] ***,
>>> 1.4 make the dispatcher user the owner (EXEC sp_addrolemember
>>> N'db_owner',
>>> N'dispatcher')
>>> 2. Executing as user='dispatcher':
>>> 2.0 *** create a schema [dsp] ***,
>>> 2.1 create a stored procedure [dispatch] (this time it is
>>> [dsp].[dispatch],
>>> not [dbo].[dispatch]),
>>> 2.2 create a certificate,
>>> 2.3 sign the stored procedure (ADD SIGNATURE TO
OBJECT::[dispatch] >>> ...)
>>> 2.4 Discard the private key from the certificate,
>>> 2.5 Back up the certificate to a file.
>>> 3. Executing as login='sa':
>>> 3.1 in the [providerDB] create a user [provider] with a default >>>
schema
>>> [pvd]
>>> 3.2 make the provider user the owner (EXEC sp_addrolemember
>>> N'db_owner',
>>> N'provider')
>>> 4. Executing as user='provider':
>>> 4.1 create a schema [pvd]
>>> 4.2 create a certificate from the file.
>>> 4.3 Create a user from the certificate,
>>> 4.4 GRANT AUTHENTICATE TO the certificate user,
>>> 4.5 GRANT EXECUTE ON [providerDB].[pvd].[testProc] to the >>>
certificate
>>> user.
>>> 5. Executing as user='dispatcher'
>>> 5.1 execute [dispatch]
>>>
>>> This time there's the error message:
>>> Msg 916, Level 14, State 1, Procedure testProc, Line 10
>>> The server principal "dispatcher" is not able to access the database
>>> "providerDB" under the current security context.
>>>
>>> Okay, there is a difference in the security context of the caller ---
```

Re: Cross-database execution permissions with certificates and sch

```
>>> the
>>> callee's context is the same in both scenarios.
>>>
>>> Here's a look at the rows in sys.user_token in the scenario that >>>
works:
>>> <user_token principal_id="1" name="dbo" type="SQL USER" >>>
usage="GRANT
>>> OR
>>> DENY" />
>>> <user_token principal_id="0" name="public" type="ROLE"
usage="GRANT >>> OR
>>> DENY" />
>>> <user_token principal_id="16384" name="db_owner" type="ROLE"
>>> usage="GRANT
>>> OR DENY" />
>>>
>>> Here's what is in sys.user_token in the scenario that doesn't work:
>>> <user_token principal_id="5" name="dispatcher" type="SQL USER"
>>> usage="GRANT OR DENY" />
>>> <user_token principal_id="0" name="public" type="ROLE"
usage="GRANT >>> OR
>>> DENY" />
>>> <user_token principal_id="16384" name="db_owner" type="ROLE"
>>> usage="GRANT
>>> OR DENY" />
>>>
>>> Did I discover some magic property of the [dbo] user and [dbo] >>>
schema
>>> (not
>>> likely), or am I missing something conceptual about permissions >>>
grants
>>> through certificates?
>>>
>>> Thanks,
>>> --Craig.
>>>
>>> Here's the script that works: (the script that doesn't follows)
>>> -----
>>> SELECT 0, SUSER_NAME(), USER_NAME()
>>> GO
>>> REVERT
>>> GO
>>> SELECT 1, SUSER_NAME(), USER_NAME()
>>> GO
>>> USE [master]
>>> GO
>>> IF EXISTS (SELECT * FROM [sys].[databases] WHERE [name] =
>>> 'dispatcherDB')
>>> DROP DATABASE [dispatcherDB]
>>> GO
>>> IF EXISTS (SELECT * FROM [sys].[databases] WHERE [name] = >>>
```

Re: Cross-database execution permissions with certificates and sch

```
> 'providerDB')
>>> DROP
>>> DATABASE [providerDB]
>>> GO
>>> IF EXISTS (SELECT * FROM [sys].[server_principals] WHERE
[name] =
>>> 'dispatcher') DROP LOGIN [dispatcher]
>>> GO
>>> IF EXISTS (SELECT * FROM [sys].[server_principals] WHERE
[name] =
>>> 'provider') DROP LOGIN [provider]
>>> GO
>>> EXECUTE xp_cmdshell 'del c:\temp\dispatchSproc.cert',
NO_OUTPUT;
>>> GO
>>> CREATE DATABASE [dispatcherDB]
>>> GO
>>> CREATE DATABASE [providerDB]
>>> GO
>>> CREATE LOGIN [dispatcher]
>>> WITH PASSWORD = '53cr37',
>>> DEFAULT_DATABASE = [dispatcherDB],
>>> CHECK_EXPIRATION=OFF,
>>> CHECK_POLICY=OFF;
>>> GO
>>> CREATE LOGIN [provider]
>>> WITH PASSWORD = '53cr37',
>>> DEFAULT_DATABASE=[providerDB],
>>> CHECK_EXPIRATION=OFF,
>>> CHECK_POLICY=OFF;
>>> GO
>>> -- Create objects in dispatcher database
>>> USE [dispatcherDB]
>>> GO
>>> CREATE USER [dispatcher]
>>> -- WITH DEFAULT_SCHEMA = [dsp];
>>> GO
>>> EXEC sp_addrolemember N'db_owner', N'dispatcher'
>>> GO
>>> -- Continue to create objects as the dispatcher user, in the default
>>> schema
>>> for this user, dsp:
>>> EXECUTE AS USER = 'dispatcher'
>>> GO
>>> -- CREATE SCHEMA [dsp] AUTHORIZATION [dispatcher];
>>> GO
>>> CREATE PROCEDURE [dispatch]
>>> WITH EXECUTE AS OWNER
>>> AS
>>> BEGIN
>>> DECLARE @lclRequest xml;
```

Re: Cross-database execution permissions with certificates and sch

```
>>> DECLARE @lclResponse xml;
>>>
>>> SET @lclRequest = (
>>> SELECT
>>> [principal_id],
>>> [name],
>>> [type],
>>> [usage]
>>> FROM
```