rusanu.com

- About
- Links
- Articles
- Blog

Call a procedure in another database from an activated procedure

March 7th, 2006

In my previous post http://rusanu.com/2006/03/01/signing-an-activated-procedure/ I've showed how to use code signing to enable a server level privilege (view server state) when running an activated stored procedure. I'll show now how to solve a very similar issue: call a stored procedure from another database. Why this is a problem is explained in this post: http://rusanu.com/2006/01/12/why-does-feature-not-work-under-activation/

So let's say the 'SessionsService' from my previous post needs some new functionality: it has to audit the requests it's receive. An audit infrastructure already exists in the [AuditInfrastructure] database, all is needed is to call the stored procedure [audit_record_request] in that database and the request will be audited. First create this 'audit infrastructure', which for our example will be very simple:

```
create database [AuditInfrastructure]

go


use [AuditInfrastructure]

go


-- An audit table, simply stores the time of the request

--

create table audit_table (request_time datetime);

go


-- this is the audit procedure

--

create procedure audit_record_request

as

begin

      set nocount on

      insert into audit_table (request_time) values (getdate());
```

```
end
```

So now all we have to do is to change the SessionService procedure to call this audit procedure:

```
USE [DemoActivationSigning];
GO

ALTER PROCEDURE [SessionsServiceProcedure]
AS
BEGIN
      SET NOCOUNT ON;
      DECLARE @dh UNIQUEIDENTIFIER;
      DECLARE @mt SYSNAME;

      BEGIN TRANSACTION;
      WAITFOR (RECEIVE TOP (1) @dh = conversation_handle,
            @mt = message_type_name
            FROM [SessionsQueue]), TIMEOUT 1000;
      WHILE (@dh IS NOT NULL)
      BEGIN
            - If the message is a request message,
            - send back a response with the list of sessions
            -
            IF (@mt = N‘RequestSessions’)
            BEGIN
                  - Get the list of current sessions
                  - and send it back as a response
                  -
                  DECLARE @response XML;
                  SELECT @response = (
                        SELECT * FROM sys.dm_exec_sessions
                              FOR XML PATH (’session’), TYPE);
                  SEND ON CONVERSATION @dh
```

```
                        MESSAGE TYPE [Sessions]
                        (@response);
                  --  This is the extra call to the audit procedure
                  EXECUTE [AuditInfrastructure]..[audit_record_request];
            END
            - End the conversation and commit
            END CONVERSATION @dh;
            COMMIT;

            - Try to loop once more if there are message
            -
            SELECT @dh = NULL;
            BEGIN TRANSACTION;
            WAITFOR (RECEIVE TOP (1) @dh = conversation_handle,
                  @mt = message_type_name
                  FROM [SessionsQueue]), TIMEOUT 1000;
      END
      COMMIT;
END
GO
```

So now we send a request to the 'SessionsService'

```
DECLARE @dh UNIQUEIDENTIFIER;
BEGIN DIALOG CONVERSATION @dh
      FROM SERVICE [Requests]
      TO SERVICE 'SessionsService'
      ON CONTRACT [SessionsContract]
      WITH ENCRYPTION = OFF;
SEND ON CONVERSATION @dh
      MESSAGE TYPE [RequestSessions];
```

We expect a response back, but nothing happens. The 'SessionsService' has gone silent. If we look into the Event Viewer, will find some troublesome entries:

```
Event Type: Information
Event Source: MSSQLSERVER
Event Category: (2)
Event ID: 9724
```

```
Date:  3/7/2006
Time:  10:36:25 AM
User:  REDMOND\remusr
Computer: REMUSR10
Description:
The activated proc [dbo].[SessionsServiceProcedure] running on queue
DemoActivationSigning.dbo.SessionsQueue output the following:  'The server
principal "REDMOND\remusr" is not able to access the database
"AuditInfrastructure" under the current security context.'
For more information, see Help and Support Center at
http://go.microsoft.com/fwlink/events.asp.
Data:
0000: fc 25 00 00 0a 00 00 00   ü%……
0008: 09 00 00 00 52 00 45 00   ….R.E.
0010: 4d 00 55 00 53 00 52 00   M.U.S.R.
0018: 31 00 30 00 00 00 14 00   1.0…..
0020: 00 00 41 00 75 00 64 00   ..A.u.d.
0028: 69 00 74 00 49 00 6e 00   i.t.I.n.
0030: 66 00 72 00 61 00 73 00   f.r.a.s.
0038: 74 00 72 00 75 00 63 00   t.r.u.c.
0040: 74 00 75 00 72 00 65 00   t.u.r.e.
0048: 00 00         ..
```

Note: In case you wonder 'so what happened to my request?.The activated stored procedure has thrown an error and it rolled back. Because Service Broker is a fully transactional, the dequeued request was rolled back and is again available for dequeue. The activation will kick in again, causing the same error and again the request to be rolled back. Then again activation will kick in and so on and so forth. Eventually (after 5 consecutive rollbacks) the Service Broker Poisson Message support will detect this situation and will disable the queue.

The problem is, as expected, the database impersonation context, as explained here. Same as in the case of server level privileges, the easiest fix is to mark the database trustworthy:

```
ALTER DATABASE [DemoActivationSigning] SET TRUSTWORTHY ON
```

If we cannot afford this due to security risks (marking the database trustworthy elevates the database dbo to a de-facto sysadmin), we must reside to code signing. The steps are these:

- alter the procedure to have an EXECUTE AS clause (otherwise the code signing infrastructure does not work)
- create a certificate with a private key in the [DemoActivationSigning] database
- sign the procedure
- drop the private key of the certificate
- copy the certificate into the [AuditInfrastructure] database (backup the certificate to a file and the create from that file)
- derive a user from the certificate in the [AuditInfrastructure]database
- grant the desired privileges to this user

Here is the code for these steps:

```
USE [DemoActivationSigning];
GO

— Create aprocedure that implements
```

```
— the [SessionsService] service
—
ALTER PROCEDURE [SessionsServiceProcedure]
      WITH EXECUTE AS OWNER
AS
BEGIN
      SET NOCOUNT ON;
      DECLARE @dh UNIQUEIDENTIFIER;
      DECLARE @mt SYSNAME;

      BEGIN TRANSACTION;
      WAITFOR (RECEIVE TOP (1) @dh = conversation_handle,
            @mt = message_type_name
            FROM [SessionsQueue]), TIMEOUT 1000;
      WHILE (@dh IS NOT NULL)
      BEGIN
            — If the message is a request message,
            — send back a response with the list of sessions
            —
            IF (@mt = N'RequestSessions')
            BEGIN
                  — Get the list of current sessions
                  — and send it back as a response
                  —
                  DECLARE @response XML;
                  SELECT @response = (
                        SELECT * FROM sys.dm_exec_sessions
                              FOR XML PATH ('session'), TYPE);
                  SEND ON CONVERSATION @dh
                        MESSAGE TYPE [Sessions]
                        (@response);
                  EXECUTE [AuditInfrastructure]..[audit_record_request];
            END
            — End the conversation and commit
            END CONVERSATION @dh;
            COMMIT;

            — Try to loop once more if there are message
            —
            SELECT @dh = NULL;
            BEGIN TRANSACTION;
            WAITFOR (RECEIVE TOP (1) @dh = conversation_handle,
                  @mt = message_type_name
                  FROM [SessionsQueue]), TIMEOUT 1000;
      END
      COMMIT;
END
GO

— Create a certificate with a private key
— to sign the procedure with. The password
— used is not important, we'll drop the
— private key
—
CREATE CERTIFICATE [SessionsServiceProcedureAudit]
      ENCRYPTION BY PASSWORD = 'Password#1234'
      WITH SUBJECT = 'SessionsServiceProcedure Signing  for audit certificate';
GO

— Sign the procedure with the certificate's private key
—
```

```
ADD SIGNATURE TO OBJECT::[SessionsServiceProcedure]
      BY CERTIFICATE [SessionsServiceProcedureAudit]
            WITH PASSWORD = 'Password#1234';
GO

— Drop the private key. This way it cannot be
— used again to sign other procedures.
—
ALTER CERTIFICATE [SessionsServiceProcedureAudit]
      REMOVE PRIVATE KEY;
GO

— Copy the certificate in [master]
— We must backup to a file and create
— the certificate in [master] from this file
—
BACKUP CERTIFICATE [SessionsServiceProcedureAudit]
      TO FILE = 'C:\SessionsServiceProcedureAudit.CER';
GO

USE [AuditInfrastructure]
GO

CREATE CERTIFICATE [SessionsServiceProcedureAudit]
      FROM FILE = 'C:\SessionsServiceProcedureAudit.CER';
GO

CREATE USER [SessionsServiceProcedureAudit] FROM CERTIFICATE
[SessionsServiceProcedureAudit];
G0

— 'AUTHENTICATE' permission is required for all other permissions to take effect
—
GRANT AUTHENTICATE TO [SessionsServiceProcedureAudit];
GRANT EXECUTE ON [audit_record_request] TO [SessionsServiceProcedureAudit];
GO

— Enable back the disabled 'SessionsService' queue
—
ALTER QUEUE [SessionsQueue] WITH STATUS = ON;
GO

— Check that the response is now sent back
WAITFOR (RECEIVE CAST(message_body AS XML) FROM [RequestsQueue]);
```

Notice that I did not sent another request. The existing request was still there, waiting for the procedure to be fixed so it can be processed correctly.

Posted in Samples, Tutorials

6 responses to "Call a procedure in another database from an activated procedure"

1. *alzdba* says:
   October 12, 2009 at 5:57 am

   Nice operating principal demo !

   Keep in mind this must be documented in the DRP info whenever used !

2.  *Sam* says:
February 3, 2010 at 9:03 am

Remus,
Thanks for the code – what's going on with your comment char's? They come across as ascii 150 instead of 45, which is recognized as a dash.

3.  *Sam* says:
February 3, 2010 at 9:08 am

"– Copy the certificate in [master]

– We must backup to a file and create

– the certificate in [master] from this file

BACKUP CERTIFICATE [SessionsServiceProcedureAudit]

TO FILE = 'C:\SessionsServiceProcedureAudit.CER';

GO

USE [AuditInfrastructure]

GO"

Should this USE statement be going to master? It is mentioned but never switched to.

4.  *Remus* says:
February 3, 2010 at 9:16 am

These old posts were formatted originally in MS Office's Word and then pasted into WordPress. I am slowly going through them and reformat into a sane HTML, including the code (dash to –, phrase quote ` to true string delimiter ' etc)

5.  *Remus* says:
February 3, 2010 at 9:18 am

The commend about 'copy to [master]' is wrong. The first article at http://rusanu.com/2006/03/01/signing-an-activated-procedure/ needed to copy the certificate to [master] and when I changed the code for this article I forgot to change the comment. Will modify, thanks.

6.  *Sam* says:
February 3, 2010 at 1:40 pm

Remus,
You helped me with this question on SO – http://stackoverflow.com/questions/1870240/security-for-requesting-temporary-logins . I posted another comment there and was wondering if you could take a look.

Interested in SQL Server monitoring and configuration management?

Sign up at DBHistory.com

- ## Recent Posts

Interested in SQL Server monitoring and configuration management?

Sign up at DBHistory.com

© RUSANU CONSULTING LLC 2007-2016. All Rights Reserved CC-BY

Entries (RSS)