**[Audit Domain Group and User Permissions](http://www.sqlservercentral.com/blogs/sqlrnnr/2017/12/29/audit-domain-group-and-user-permissions/)**

[SQLRNNR](http://www.sqlservercentral.com/blogs/sqlrnnr/)

Posted on 29 December 2017

FROM: <http://www.sqlservercentral.com/blogs/sqlrnnr/2017/12/29/audit-domain-group-and-user-permissions/?utm_source=DBW&utm_medium=pubemail>

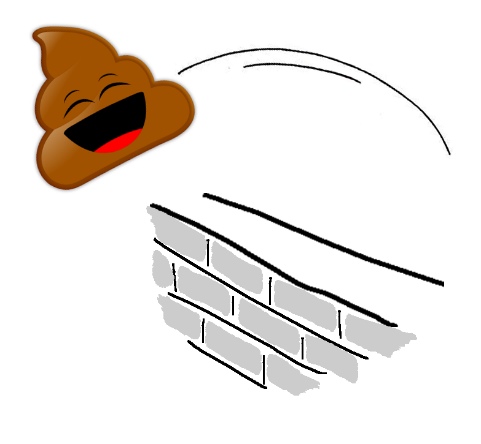
No matter how simple the task or how versed we are with doing a security audit, it seems like we can always stand to learn just a little bit more. No matter how many times we hand an audit report over to the auditor, there is always “just one” more report we have to provide.

I know it seems like I am almost always running some sort of audit report. Whether it is to audit which user may have [changed something](http://bit.ly/XEMissingDB) or what [access a user may have](http://bit.ly/anGB50) overall to the system, there is always a need for more audit.

One of the aspects of an audit that I have not yet written about is with regards to Domain Groups. It is a very common practice to grant access to SQL Server via Domain Group membership. We can easily see which groups have which access, but how often do we try to figure out who is in the group?

When you do try to figure out who is in a Domain Group, do you do like so many DBAs that I know and yell across a cubicle wall to the AD guys and ask them to look it up? Maybe you fire off an email to the AD Admins and ask them to do the grunt work for you. Nobody will shame you for doing either of those two methods. I mean, most people probably toss it over the fence in one way or another.

**Fence**



It is not a difficult stretch to imagine asking somebody who is an expert at a technology to see if they could get a quick answer to your question for you. That said, it is also really nice when you can service those types of questions by yourself and thus save everybody a little time and effort. Luckily, there is at least one easy way for the SQL Server professional to try and answer questions about group membership.

Microsoft has included a method for the DBA to try and capture information about AD related accounts. This method comes in the form of an extended procedure called xp\_logininfo. This procedure can tell me the members of domain groups, the access level the account has, and also validate if the account is still active in AD.

Suppose you needed to determine all of the accounts that may have access to SQL Server, via direct Login or via group membership. In addition, you must find out which of the accounts are no longer valid in AD. I have a script that will do just that by first perusing the groups and then cycling through each of the User accounts produced from the group perusal.

Let’s take a look at that script.

/\*

============================================================

Quick script to enumerate Active directory users who get permissions from An Active Directory Group

============================================================

\*/

--a table variable capturing any errors in the try...catch below

DECLARE @ErrorRecap TABLE

(

ID INT IDENTITY(1, 1) NOT NULL PRIMARY KEY

, AccountName VARCHAR(256)

, ErrorMessage VARCHAR(256)

);

DECLARE @groupname VARCHAR(256)

, @acctname VARCHAR(256);

IF OBJECT\_ID('tempdb.dbo.#tmp') IS NOT NULL

BEGIN

DROP TABLE #tmp;

END

IF OBJECT\_ID('tempdb.dbo.#tmpdeeper') IS NOT NULL

BEGIN

DROP TABLE #tmpdeeper;

END

CREATE TABLE [dbo].[#TMP]

(

[ACCOUNTNAME] VARCHAR(256) NULL

, [TYPE] VARCHAR(8) NULL

, [PRIVILEGE] VARCHAR(8) NULL

, [MAPPEDLOGINNAME] VARCHAR(256) NULL

, [PERMISSIONPATH] VARCHAR(256) NULL

);

CREATE TABLE #tmpdeeper

(

[ACCOUNTNAME] VARCHAR(256) NULL

, [TYPE] VARCHAR(8) NULL

, [PRIVILEGE] VARCHAR(8) NULL

, [MAPPEDLOGINNAME] VARCHAR(256) NULL

, [PERMISSIONPATH] VARCHAR(256) NULL

);

DECLARE cgroup CURSOR LOCAL FORWARD\_ONLY STATIC READ\_ONLY FOR

SELECT name

FROM master.sys.server\_principals

WHERE type\_desc = 'WINDOWS\_GROUP';

OPEN cgroup;

FETCH NEXT FROM cgroup INTO @groupname;

WHILE @@FETCH\_STATUS <> -1

BEGIN

BEGIN TRY

INSERT INTO #TMP ( [ACCOUNTNAME]

, [TYPE]

, [PRIVILEGE]

, [MAPPEDLOGINNAME]

, [PERMISSIONPATH] )

EXEC master..xp\_logininfo @acctname = @groupname

, @option = 'members'; -- show group members

END TRY

BEGIN CATCH

--capture the error details

DECLARE @ErrorSeverity INT

, @ErrorNumber INT

, @ErrorMessage VARCHAR(4000)

, @ErrorState INT;

SET @ErrorSeverity = ERROR\_SEVERITY();

SET @ErrorNumber = ERROR\_NUMBER();

SET @ErrorMessage = ERROR\_MESSAGE();

SET @ErrorState = ERROR\_STATE();

--put all the errors in a table together

INSERT INTO @ErrorRecap ( AccountName, ErrorMessage )

SELECT @groupname, @ErrorMessage;

PRINT 'Msg ' + CONVERT(VARCHAR, @ErrorNumber) + ' Level '

+ CONVERT(VARCHAR, @ErrorSeverity) + ' State '

+ CONVERT(VARCHAR, @ErrorState);

PRINT @ErrorMessage;

END CATCH;

FETCH NEXT FROM cgroup INTO @groupname;

END;

CLOSE cgroup;

DEALLOCATE cgroup;

DECLARE cuser CURSOR LOCAL FORWARD\_ONLY STATIC READ\_ONLY FOR

SELECT DISTINCT ACCOUNTNAME

FROM #TMP

UNION

SELECT sp.name

FROM sys.server\_principals sp

WHERE type\_desc = 'WINDOWS\_LOGIN'; --'WINDOWS\_GROUP'

OPEN cuser;

FETCH NEXT FROM cuser INTO @acctname;

WHILE @@FETCH\_STATUS <> -1

BEGIN

BEGIN TRY

INSERT INTO #tmpdeeper ( [ACCOUNTNAME]

, [TYPE]

, [PRIVILEGE]

, [MAPPEDLOGINNAME]

, [PERMISSIONPATH] )

EXECUTE master..xp\_logininfo @acctname = @acctname,@option = 'all';

END TRY

BEGIN CATCH

SET @ErrorSeverity = ERROR\_SEVERITY();

SET @ErrorNumber = ERROR\_NUMBER();

SET @ErrorMessage = ERROR\_MESSAGE();

SET @ErrorState = ERROR\_STATE();

--put all the errors in a table together

INSERT INTO @ErrorRecap ( AccountName, ErrorMessage )

SELECT @acctname, @ErrorMessage;

--echo out the supressed error, the try catch allows us to continue processing, instead of stopping on the first error

PRINT 'Msg ' + CONVERT(VARCHAR, @ErrorNumber) + ' Level '

+ CONVERT(VARCHAR, @ErrorSeverity) + ' State '

+ CONVERT(VARCHAR, @ErrorState);

PRINT @ErrorMessage;

END CATCH;

FETCH NEXT FROM cuser INTO @acctname;

END;

CLOSE cuser;

DEALLOCATE cuser;

--display both results and errors

SELECT \*

FROM #TMP;

SELECT \*

FROM #tmpdeeper;

SELECT \*

FROM @ErrorRecap;

You will see here that I have a couple of cursors built into this script. This is helping me cycle through first the groups and then the users. As I cycle through each user, I am able to determine the validity of the account. Not only will I know if an AD account is dead and needs to be removed from SQL Server, but I will also know if a Domain Group no longer exists.

This little script has proven to be a major time saver with some clients. Not only has it saved me some time, but it has also helped to figure out how a developer was able to perform sysadmin functions on a prod box even though his account did not have sysadmin access granted explicitly.

http://jasonbrimhall.info/?feed-stats-post-id=4001

**Related Posts:**

* [Security as a Fleeting Thought](http://jasonbrimhall.info/2015/02/10/security-as-a-fleeting-thought/) February 10, 2015
* [Login from an Untrusted Domain – Back to Basics](http://jasonbrimhall.info/2016/11/08/login-from-an-untrusted-domain-back-to-basics/) November 8, 2016
* [Failed to Create the Audit File](http://jasonbrimhall.info/2017/12/31/failed-to-create-the-audit-file/) December 31, 2017
* [Auditing Needs Reporting](http://jasonbrimhall.info/2015/10/13/auditing-meets-xevents/) October 13, 2015
* [All about the Change](http://jasonbrimhall.info/2016/01/12/all-about-the-change/) January 12, 2016

[http://feeds.feedburner.com/~ff/SqlRnnrSsc?d=yIl2AUoC8zA](http://feeds.feedburner.com/~ff/SqlRnnrSsc?a=IPdgTqyRWGo:vU0zU7C3C0A:yIl2AUoC8zA)[http://feeds.feedburner.com/~ff/SqlRnnrSsc?i=IPdgTqyRWGo:vU0zU7C3C0A:F7zBnMyn0Lo](http://feeds.feedburner.com/~ff/SqlRnnrSsc?a=IPdgTqyRWGo:vU0zU7C3C0A:F7zBnMyn0Lo)[http://feeds.feedburner.com/~ff/SqlRnnrSsc?i=IPdgTqyRWGo:vU0zU7C3C0A:V_sGLiPBpWU](http://feeds.feedburner.com/~ff/SqlRnnrSsc?a=IPdgTqyRWGo:vU0zU7C3C0A:V_sGLiPBpWU)[http://feeds.feedburner.com/~ff/SqlRnnrSsc?d=I9og5sOYxJI](http://feeds.feedburner.com/~ff/SqlRnnrSsc?a=IPdgTqyRWGo:vU0zU7C3C0A:I9og5sOYxJI)[http://feeds.feedburner.com/~ff/SqlRnnrSsc?i=IPdgTqyRWGo:vU0zU7C3C0A:3QFJfmc7Om4](http://feeds.feedburner.com/~ff/SqlRnnrSsc?a=IPdgTqyRWGo:vU0zU7C3C0A:3QFJfmc7Om4)

[](http://www.sqlservercentral.com/blogs/sqlrnnr/)

[**SQL RNNR**](http://www.sqlservercentral.com/blogs/sqlrnnr/)

**Comments**

[Leave a comment on the original post [jasonbrimhall.info, opens in a new window]](http://jasonbrimhall.info/2017/12/29/audit-domain-group-and-user-permissions/#comments)

http://www.sqlservercentral.com/Resources/Images/loading.gifLoading comments...

**Posted by jeff\_yao on 29 December 2017**

Have you considered a Domain group within another group?

**Posted by Jason Brimhall on 30 December 2017**

i have and it would be a similar routine. The question with nested groups is what point do we stop recursing through them.