

Re: List User's Permissions down to table.column action

Source: <http://www.derkeiler.com/Newsgroups/microsoft.public.sqlserver.security/2006-06/msg00116.html>

- *From:* "Neil Meyer" <nmeyer@xxxxxxxx>
 - *Date:* Thu, 15 Jun 2006 09:29:54 -0700
-

Works great! Thanks Greg. Well done.

Neil

"Greg Larsen" <GregLarsen@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:9F537B10-3A04-4C01-9006-4086C1F2F9FF@xxxxxxxxxxxxxxxxxxxx

You might try this SP. This SP generates the necessary T-SQL commands to add a new user based on an existing user. You can look at the generated T_SQL to review the permissions. I beleave it does column level permission. If nothing else you can use this code to build some thing that works for you.

I wrote an article about this code here that you can read if you have a subscription to SQL Server Magazine:

http://www.sqlmag.com/Article/ArticleID/41181/sql_server_41181.html

```
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

```
/****** Object: Stored Procedure
dbo.USB_GEN_USER_RIGHTS_BASED_ON_EXISTING_LOGIN Script Date: 10/21/2002
6:56:14 AM *****/
```

```
CREATE PROCEDURE USB_GEN_USER_RIGHTS_BASED_ON_EXISTING_LOGIN
```

Re: List User's Permissions down to table.column action

```
@OLDLOGIN VARCHAR(200), -- EXISTING LOGIN NAME
@NEWLOGIN VARCHAR(200), -- NEW LOGIN NAME
@NEWUSER CHAR(128) , -- NEW USER NAME FOR EACH DATABASE
@PASSWORD VARCHAR(200) =" -- PASSWORD FOR SQL SERVER
AUTHENTICATED USERS
AS
--
-- WRITTEN BY: GREG LARSEN FOR WASHINGTON STATE DEPARTMENT OF
HEALTH
-- DATE: DECEMBER 13, 2001
--
-- DESCRIPTION: THIS STORED PROCEDURE GENERATES COMMANDS
-- THAT WILL ADD A NEW USERS TO THE SERVER WITH THE SAME
-- RIGHTS AS AN EXISTING USER.
--
-- UPDATE HISTORY:
-- DATE: OCTOBER 21, 2002
-- FIXED PROBLEMS WITH STATEMENT LEVEL PERMISSIONS GRANTING.
--
-- BEGIN CALLOUT A
-- Declare variables and create temporary table to hold commands
-----
DECLARE @INDX INT
SET NOCOUNT ON
DECLARE @TEXT CHAR(100)
DECLARE @CNT INT
DECLARE @CMD NVARCHAR(200)
DECLARE @DB NVARCHAR(128)
DECLARE @OLDUSER VARCHAR(100)
--
-- CREATE TABLE TO HOLD GENERATED COMMANDS
--
CREATE TABLE #TMP_LOGIN_RIGHTS (
RIGHTS_TEXT CHAR(2000))
--
-- END CALLOUT A
-- BEGIN CALLOUT B
-- ADD USER TO SERVER
-----
-- TEST TO SEE IF #OLDLOGIN EXISTS ON SERVER
--
SELECT @CNT=COUNT(*) FROM [MASTER].[DBO].[SYSLOGINS] WHERE
LOGINNAME =
@OLDLOGIN
IF @CNT = 0
BEGIN
RAISERROR ('@OLDLOGIN IS NOT A VALID USER OF SQL SERVER',16,1)
RETURN
END
--
```

Re: List User's Permissions down to table.column action

Re: List User's Permissions down to table.column action

```
-- DETERMINE IF @NEWLOGIN IS ALREADY DEFINED TO SERVER
--
SELECT @CNT=COUNT(*) FROM [MASTER].[DBO].[SYSLOGINS] WHERE
LOGINNAME =
@NEWLOGIN
--
-- IF @NEWLOGIN EXIST ABORT
--
IF @CNT > 0
BEGIN
RAISERROR('@NEWLOGIN ALREADY EXISTS ON SQL SERVER', 16,1)
RETURN
END
--
-- IF @NEWLOGIN CONTAINS '\' THEN NT LOGIN
--
SELECT @INDX=CHARINDEX('\',@NEWLOGIN)
IF @INDX > 0
--
-- GENERATE COMMANDS TO ADD NT USER
--
INSERT INTO #TMP_LOGIN_RIGHTS SELECT 'EXECUTE
[MASTER].[DBO].[SP_GRANTLOGIN] [' + @NEWLOGIN + ']'
+ CHAR(13)+
'EXECUTE [MASTER].[DBO].[SP_DEFAULTDB] [' + @NEWLOGIN + '],[' +
RTRIM(DBNAME) + ']' AS RIGHTS_TEXT
FROM [MASTER].[DBO].[SYSLOGINS]
WHERE LOGINNAME = @OLDLOGIN
ELSE
BEGIN
IF @PASSWORD = "
BEGIN
RAISERROR('@PASSWORD MUST BE SPECIFIED FOR SQL SERVER
AUTHENTICATION',
16,1)
RETURN
END -- @PASSWORD = "
--
-- GENERATE COMMANDS TO ADD SQL SERVER AUTHENTICATION USER
--
INSERT INTO #TMP_LOGIN_RIGHTS SELECT 'EXECUTE
[MASTER].[DBO].[SP_ADDLOGIN] [' + @NEWLOGIN +
'],[' + @PASSWORD + ']' + CHAR(13)+
'EXECUTE [MASTER].[DBO].[SP_DEFAULTDB] [' + @NEWLOGIN + '],[' +
RTRIM(DBNAME) + ']' AS RIGHTS_TEXT
FROM [MASTER].[DBO].[SYSLOGINS]
WHERE LOGINNAME = @OLDLOGIN
END

-- END CALLOUT B
-- BEGIN CALLOUT C
```

Re: List User's Permissions down to table.column action

```
-- ADD USER TO DATABASES
```

```
-----  
SET NOCOUNT ON
```

```
SET @CMD= '[MASTER].[DBO].[SP_HELPUSER]'
```

```
--
```

```
-- GET THE NAME OF ALL DATABASES
```

```
--
```

```
DECLARE ALLDATABASES CURSOR FOR
```

```
SELECT NAME FROM [MASTER].[DBO].[SYSDATABASES]
```

```
OPEN ALLDATABASES
```

```
FETCH NEXT FROM ALLDATABASES INTO @DB
```

```
--
```

```
-- CREATE TABLE TO HOLD LIST OF USERS IN CURRENT DATABASE
```

```
--
```

```
CREATE TABLE #TMPUSERS (
```

```
  USERNAME VARCHAR(100),
```

```
  GROUPNAME VARCHAR(100),
```

```
  LOGINNAME VARCHAR(100),
```

```
  DEFDBNAME VARCHAR(100),
```

```
  USERID SMALLINT,
```

```
  SUSERID SMALLINT
```

```
)
```

```
WHILE (@@FETCH_STATUS = 0)
```

```
BEGIN
```

```
--
```

```
-- COMMAND TO RETURN ALL USERS IN DATABASE
```

```
--
```

```
SET @CMD = '[' + @DB + ']' + '[DBO].[SP_HELPUSER]'
```

```
--
```

```
-- GET ALL USERS IN DATABASE INTO TEMPORARY TABLE
```

```
--
```

```
INSERT INTO #TMPUSERS EXEC (@CMD)
```

```
--
```

```
-- DETERMINE WHETHER OLD USER IS IN DATABASE
```

```
--
```

```
SELECT @CNT = COUNT(*) FROM #TMPUSERS WHERE LOGINNAME =
```

```
@OLDLOGIN
```

```
--
```

```
-- IF OLD USER IS IN DATABASE THEN ADD NEW USER TO DATABASE
```

```
--
```

```
IF @CNT > 0
```

```
BEGIN
```

```
--
```

```
-- DETERMINE IF @NEWUSER ALREADY EXIST IN DATABASE
```

```
--
```

```
SELECT @CNT = COUNT(*) FROM #TMPUSERS WHERE USERNAME =
```

```
@NEWUSER
```

```
--
```

```
-- IF USER EXIST ABORT
```

```
--
```

Re: List User's Permissions down to table.column action

```
IF @CNT > 0
BEGIN
--
-- CLOSE AND DEALLOCATE CURSOR OF DATABASES SO NEXT TIME AROUND
NO
ERROR OCCURS
--
CLOSE ALLDATABASES
DEALLOCATE ALLDATABASES
--
-- SET TEXT OF ERROR MESSAGE
--
SET @TEXT = '@NEWUSER ALREADY EXIST IN DATABASE ' + @DB
-- RAISE ERROR AND RETURN
RAISERROR(@TEXT,16,1)
RETURN
END
--
-- GENERATE COMMAND TO ADD @NEWLOGIN TO CURRENT DATABASE
--
INSERT INTO #TMP_LOGIN_RIGHTS SELECT 'EXECUTE [' + @DB +
'].[DBO].[SP_GRANTDBACCESS] [' +
@NEWLOGIN +
'],[' + RTRIM(@NEWUSER) + ']' AS RIGHTS_TEXT
FROM (
SELECT DISTINCT USERNAME, LOGINNAME
FROM #TMPUSERS
WHERE LOGINNAME = @OLDLOGIN )A
END
--
-- TRUNCATE TABLE FOR NEXT DATABASE
--
TRUNCATE TABLE #TMPUSERS

--
-- GET NEXT DATABASE
--
FETCH NEXT FROM ALLDATABASES INTO @DB
END -- WHILE (@@FETCH_STATUS = 0)
--
-- CLOSE CURSOR OF DATABASES
--
CLOSE ALLDATABASES
-- END CALLOUT C
-- BEGIN CALLOUT D
-- GRANT USER TO ROLES WITHIN DATABASES

-----
OPEN ALLDATABASES
FETCH NEXT FROM ALLDATABASES INTO @DB
WHILE (@@FETCH_STATUS = 0)
BEGIN
```

Re: List User's Permissions down to table.column action

```
--
-- SET COMMAND TO FIND USER PERMISSIONS HAS IN CURRENT DATABASE
--
SET @CMD = '[' + @DB + '].[DBO].[SP_HELPUSER]'
--
-- EMPTY TEMPORARY TABLE #TMPUSERS
--
TRUNCATE TABLE #TMPUSERS
--
-- GET USER PERMISSIONS FOR ALL USERS IN CURRENT DATABASE
--
INSERT INTO #TMPUSERS EXEC (@CMD)
--
-- DETERMINE WHETHER THE OLD USER IS IN A ROLE
--
SELECT @CNT = COUNT(*) FROM #TMPUSERS WHERE LOGINNAME =
@OLDLOGIN AND
GROUPNAME <> 'PUBLIC'
--
-- IF OLD USER IS IN A ROLE THEN ADD NEW USER TO THAT ROLE
--
IF @CNT > 0
--
-- GENERATE COMMANDS TO ADD @NEWUSER TO APPROPRIATE ROLES IN
CURRENT
DATABASE
--
INSERT INTO #TMP_LOGIN_RIGHTS SELECT DISTINCT 'EXECUTE [' + @DB +
'].[DBO].[SP_ADDROLEMEMBER] [' + RTRIM(A.GROUPNAME) +
'],[' + RTRIM(@NEWUSER) + ']' AS RIGHTS_TEXT
FROM #TMPUSERS A
WHERE A.LOGINNAME = @OLDLOGIN AND A.GROUPNAME <>
'PUBLIC'
--
-- GET NEXT DATABASE
--
FETCH NEXT FROM ALLDATABASES INTO @DB
END -- WHILE (@@FETCH_STATUS = 0)
CLOSE ALLDATABASES
DROP TABLE #TMPUSERS
-- END CALLOUT D
-- BEGIN CALLOUT E
-- GRANT USER ACCESS TO SERVER ROLES
-----
-- CREATE TABLE TO HOLD SERVER ROLES
--
CREATE TABLE #TMPSRVROLES (
SERVERROLE VARCHAR(100),
MEMBERNAME VARCHAR(100),
MEMBERSID VARBINARY (85))
--
```

Re: List User's Permissions down to table.column action

```
-- COMMAND TO GET SERVER ROLES
--
SET @CMD = '[MASTER].[DBO].[SP_HELPsrvROLEMEMBER]'
--
-- GET SERVER ROLES INTO TEMPORARY TABLE
--
INSERT INTO #TMPSRVROLES EXEC (@CMD)
--
-- DETERMINE WHETHER THE OLD USER IS IN A SERVER ROLE
--
SELECT @CNT = COUNT(*) FROM #TMPSRVROLES WHERE MEMBERNAME =
@OLDLOGIN
--
-- IF OLD USER IS IN A ROLE THEN ADD NEW USER TO THAT SERVER ROLE
--
IF @CNT > 0
--
-- GENERATE COMMANDS TO ADD @NEWLOGIN INTO THE APPROPRIATE
SERVER ROLES
--
INSERT INTO #TMP_LOGIN_RIGHTS SELECT 'EXECUTE
[MASTER].[DBO].[SP_ADDSRVROLEMEMBER]' + '[' +
RTRIM(@NEWLOGIN) + ']' +
',' + RTRIM(A.SERVERROLE) + ']' AS RIGHTS_TEXT
FROM #TMPSRVROLES A
WHERE A.MEMBERNAME = @OLDLOGIN
--
-- DROP SERVER ROLE TABLE
--
DROP TABLE #TMPSRVROLES
-- END CALLOUT E
-- BEGIN CALLOUT F
-- GRANT USER PERMISSIONS TO OBJECTS AND STATEMENTS
-----
-- CREATE TEMPORARY TABLE TO HOLD INFORMATION ABOUT OBJECTS
PERMISSIONS
--
CREATE TABLE #TMPPROTECT (
OWNER VARCHAR(100),

OBJECT VARCHAR(100),
GRANTEE VARCHAR(100),
GRANTOR VARCHAR(100),
PROTECTTYPE CHAR(10),
ACTION VARCHAR(20),
COLUMNX VARCHAR(100))

OPEN ALLDATABASES
SET @CMD = 'SELECT @OLDUSER=NAME FROM [' + @DB +
'].[DBO].[SYSUSERS] WHERE SID = (SELECT SID FROM
[MASTER].[DBO].[SYSLOGINS] WHERE LOGINNAME = ' +
```

Re: List User's Permissions down to table.column action

```
CHAR(39) + @OLDLOGIN + CHAR(39) + ')'
FETCH NEXT FROM ALLDATABASES INTO @DB
WHILE (@@FETCH_STATUS = 0)
BEGIN
--
-- INITIALIZE @OLDUSER VARIABLE
--
SET @OLDUSER = ""
--
--GENERATE COMMAND TO SET @OLDUSER TO THE USER NAME IN
DATABASE
-- IF @OLDLOGIN HAS ACCESS TO CURRENT DATABASE
--
SET @CMD = 'SELECT @OLDUSER=NAME FROM [' + @DB +
'].[DBO].[SYSUSERS] WHERE SID = (SELECT SID FROM
[MASTER].[DBO].[SYSLOGINS] WHERE LOGINNAME = ' +
CHAR(39) + @OLDLOGIN + CHAR(39) + ')'
--
-- EXECUTE COMMAND TO SET @OLDUSER TO THE USER NAME IN DATABASE
-- IF @OLDLOGIN HAS ACCESS TO CURRENT DATABASE
--
EXEC [MASTER].[DBO].[SP_EXECUTESQL] @CMD,N'@OLDUSER CHAR(200)
OUTPUT',@OLDUSER OUT
--
-- IF @OLDUSER IS NOT BLANK THEN @OLDLOGIN HAS ACCESS TO CURRENT
DATABASE
--
IF @OLDUSER <> ""
BEGIN
--
-- GENERATE COMMAND TO GET OBJECT PERMISSIONS FOR CURRENT
DATABASE
--
SET @CMD = '[' + @DB + '].[DBO].[SP_HELPROTECT]'
--
-- GET OBJECT PERMISSIONS INTO TEMPORARY TABLE
--
INSERT INTO #TMPPROTECT EXEC (@CMD)
--
-- DETERMINE IF THERE ARE ANY OBJECT PERMISSIONS FOR @OLDUSER
--
SELECT @CNT = COUNT(*) FROM #TMPPROTECT WHERE GRANTEE =
@OLDUSER
IF @CNT > 0
--
-- SWITCH TO THE APPROPRIATE DATABASE
--
INSERT INTO #TMP_LOGIN_RIGHTS SELECT 'USE [' + @DB + ']'
--
-- GENERATE COMMANDS TO GRANT OBJECTS PERMISSIONS FOR
REFERENCES, SELECT,
```

Re: List User's Permissions down to table.column action

UPDATE TO @NEWUSER

--

```
INSERT INTO #TMP_LOGIN_RIGHTS
SELECT CASE WHEN RTRIM(PROTECTTYPE) = 'GRANT_WGO'
THEN
'GRANT ' +
ACTION +' ON [' +
@DB + '].[ ' + OWNER + '].[ ' + OBJECT +
'] TO [' + RTRIM(@NEWUSER) + ']' + ' WITH GRANT
OPTION'
ELSE
'GRANT ' +
ACTION +' ON [' +
@DB + '].[ ' + OWNER + '].[ ' + OBJECT +
'] TO [' + RTRIM(@NEWUSER) + ']'
END AS RIGHTS_TEXT
FROM #TMPPROTECT WHERE GRANTEE = @OLDUSER AND OBJECT <> '!'
AND COLUMNX = '(ALL+NEW)'
```

--

-- GRANT COLUMN PERMISSION ON OBJECTS

--

```
INSERT INTO #TMP_LOGIN_RIGHTS
SELECT CASE WHEN RTRIM(PROTECTTYPE) = 'GRANT_WGO'
THEN
'GRANT ' +
ACTION +' ON [' +
@DB + '].[ ' + OWNER + '].[ ' + OBJECT +
']([' + COLUMNX + '])' +
' TO [' + RTRIM(@NEWUSER) + ']' + ' WITH GRANT
OPTION'
ELSE
'GRANT ' +
ACTION +' ON [' +
@DB + '].[ ' + OWNER + '].[ ' + OBJECT +
']([' + COLUMNX + '])' +
' TO [' + RTRIM(@NEWUSER) + ']'
END AS RIGHTS_TEXT
FROM #TMPPROTECT WHERE GRANTEE = @OLDUSER
AND OBJECT <> '!' AND COLUMNX <> '(ALL+NEW)'
AND COLUMNX <> '!'
```

--

-- GRANT INSERT, DELETE, AND EXECUTE PERMISSION ON OBJECTS

--

```
INSERT INTO #TMP_LOGIN_RIGHTS
SELECT CASE WHEN RTRIM(PROTECTTYPE) = 'GRANT_WGO'
THEN
'GRANT ' +
ACTION +' ON [' +
@DB + '].[ ' + OWNER + '].[ ' + OBJECT +
'] TO [' + RTRIM(@NEWUSER) + ']' + ' WITH GRANT
```

Re: List User's Permissions down to table.column action

```
OPTION'  
ELSE  
'GRANT ' +  
ACTION +' ON [' +  
@DB + '].[' + OWNER + '].[' + OBJECT +  
' ] TO [' + RTRIM(@NEWUSER) + ']  
END AS RIGHTS_TEXT  
  
FROM #TMPPROTECT WHERE GRANTEE = @OLDUSER AND OBJECT <> '' AND  
ACTION IN ('INSERT', 'DELETE', 'EXECUTE') AND COLUMNX = ''  
--  
-- GRANT STATEMENT PERMISSIONS  
--  
--  
INSERT INTO #TMP_LOGIN_RIGHTS  
SELECT 'GRANT ' +  
ACTION +  
' TO [' + RTRIM(@NEWUSER) + ']  
AS RIGHTS_TEXT  
FROM #TMPPROTECT WHERE GRANTEE = @OLDUSER AND OBJECT = ''  
  
--  
-- REMOVE RECORDS FOR TEMPORARY TABLE IN PREPARATION FOR THE  
NEXT DATABASE  
TO BE PROCESSES  
--  
TRUNCATE TABLE #TMPPROTECT  
END  
--  
-- GET NEXT DATABASE TO PROCESS  
--  
FETCH NEXT FROM ALLDATABASES INTO @DB  
END -- WHILE (@@FETCH_STATUS = 0)  
--  
-- CLOSE AND DEALLOCATE DATABASE LIST CURSOR  
--  
CLOSE ALLDATABASES  
DEALLOCATE ALLDATABASES  
--  
-- DROP TEMPORARY TABLE THAT HELD OBJECT PERMISSIONS  
--  
DROP TABLE #TMPPROTECT  
-- END CALLOUT F  
-- BEGIN CALLOUT G  
-- PROCESS ALL GENERATED COMMANDS ONE AT A TIME  
-----  
--  
-- GET ALL THE GENERATED COMMANDS  
--  
DECLARE COMMANDS CURSOR FOR
```

Re: List User's Permissions down to table.column action

```
SELECT * FROM #TMP_LOGIN_RIGHTS

OPEN COMMANDS
FETCH NEXT FROM COMMANDS INTO @CMD
WHILE (@@FETCH_STATUS = 0)
BEGIN
--
-- PRINT COMMAND TO BE PROCESSED
--
PRINT @CMD
--
-- UNCOMMENT IF YOU WANT THE STORED PROCEDURE TO EXECUTE THE
COMMANDS TO
ADD
THE PERMISSIONS
--
--EXEC (@CMD)
--
-- GET NEXT COMMAND
--
FETCH NEXT FROM COMMANDS INTO @CMD
END -- WHILE (@@FETCH_STATUS = 0)
--
-- CLOSE AND DEALLOCATE COMMAND CURSOR
--
CLOSE COMMANDS
DEALLOCATE COMMANDS
--
-- DROP TABLE THAT HELD THE GENERATED RIGHTS THAT WERE GRANTED
TO
@NEWLOGIN
--
DROP TABLE #TMP_LOGIN_RIGHTS

-- END CALLOUT G

GO
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS ON
GO
```

"Neil Meyer" wrote:

Anybody have a script to list a user's effective permissions down to the table.column [action] level?

With records produced by this script, I would be able to say for example:

Re: List User's Permissions down to table.column action

Re: List User's Permissions down to table.column action

userID has select and update permissions on dbo.cdata.cfid through public role

I saw a similar, recent post on a similar topic on this newsgroup, but the suggestions did not work for me.

Thanks anybody.