What to Audit at the SQL Server Instance Level

Written By: K. Brian Kelley -- 7/30/2013

Problem

I'm responsible for the security of my SQL Servers. I know I should be auditing them regularly, I just don't know where to start. What's your recommendation for auditing?

Solution

There's a multi-part answer to this, so let's focus on one of those parts in this article: the server level permissions with respect to Microsoft SQL Server. I don't mean the operating system, that's another part, but that which we call "server level" permissions inside SQL Server. Here are things we're interested in:

- What are the logins (server principals)?
- Of those logins, which SQL Server based logins are enforcing any sort of password policy?
- What are the members of the fixed server roles?
- What are the user-defined server roles (SQL Server 2012)?
- What are the server-level permissions assigned to the logins and server roles?

The Logins

Querying all the logins is easy. We have a catalog view, sys.server_principals, that will give us this information. Here are the basics I query for:

- Name
- Type of login
- Whether or not the login is disabled
- When the login was created
- When the login was last updated

This query returns all of that information:

```
SELECT name, type_desc, is_disabled,
       create_date, modify_date
FROM sys.server_principals;
```

You want to compare these results over time. When logins are Windows groups, you want to know who is a member of that group (including nesting) if the login has elevated permissions like:

- the ability to IMPERSONATE another login
- is a member of a fixed server role
- has the CONTROL SERVER permission
- has the ALTER ANY LOGIN permission

We'll talk more about permissions when we examine that section.

SQL Logins and Password Policy Enforcement

Some of the logins are bound to be SQL Server based ones. If that's the case, we want to know what logins have the password policy enforced. The password policy will be the same as for the operating system. We also want to know what logins may have the password policy enforced but have the password set to not expire. Here's a query to do that:

```
SELECT name, is_policy_checked, is_expiration_checked
FROM sys.sql_logins;
```

Whether or not it's okay to avoid password policy and/or password expiration is something that has to be evaluated on a case-by-case basis. SQL Server logins probably need to comply with the same sort of policy your organization has for "service accounts" within Active Directory.

Role Membership

Prior to SQL Server 2012, we only have to worry about built-in server roles. As a result, I'm tackling role membership next. It's important to know who is in what role. This query will report that information:

```
SELECT l.name AS 'login', r.name AS 'role'
FROM sys.server_principals AS l
  JOIN sys.server_role_members AS rm
    ON l.principal_id = rm.member_principal_id
  JOIN sys.server_principals AS r
    ON rm.role_principal_id = r.principal_id;
```

Obviously, any login that is a member of a server role needs to be verified. I will say this is especially true of members of *securityadmin*. In the past, using securityadmin was seen as an effective control where you wanted to give a DBA some rights within SQL Server but not rights to get to the data, for instance. However, the securityadmin role gives permissions to grant server-level permissions. Therefore, there's a known exploit where a member of securityadmin creates a new login, gives it impersonate rights to sa or gives it CONTROL SERVER and then the securityadmin logs in as that new login. Therefore, pay careful attention to all server role memberships.

User-Defined Roles

If you're dealing with SQL Server 2012, you have to handle the fact that user-defined roles are now available at the SQL Server level. If you want to see all the roles, execute this query. It will return the traditional built-in roles.

```
SELECT name
FROM sys.server_principals
WHERE type = 'R';
```

If you take a look at sys.server_principals, you'll likely note that all the traditional roles are numbered with a principal ID of 10 or less. Therefore, if we want to see roles created by a user, we just have to add in the appropriate clause:

```
SELECT name
FROM sys.server_principals
WHERE type = 'R'
  AND principal_id > 10;
```

Obviously, if it's a user-defined role, you want to investigate the permissions. If you're only looking at the traditional roles, you might miss where someone created a role and granted it CONTROL SERVER. That would be a huge gap in your security.

Server-Level Permissions

This brings us to the last set of permissions I recommend checking at the server level within SQL Server. Basically we're looking for anything related to:

- CONTROL SERVER
- IMPERSONATE
- ALTER *
- CREATE *
- VIEW SERVER STATE
- SHUTDOWN

The following query returns that information:

```
SELECT state_desc, permission_name, l.name
FROM sys.server_permissions AS perm
  JOIN sys.server_principals AS l
    ON perm.grantee_principal_id = l.principal_id
WHERE class_desc = 'SERVER';
```

Note that I'm intentionally focusing on "SERVER" permissions, meaning I'm filtering out endpoints. If you're using endpoints such that certain logins can only access SQL Server from a particular endpoint, then you probably want to issue the following query:

```
SELECT perm.state_desc, permission_name, e.name AS 'Endpoint', l.name
FROM sys.server_permissions AS perm
  JOIN sys.server_principals AS l
    ON perm.grantee_principal_id = l.principal_id
  JOIN sys.endpoints e
    ON perm.major_id = e.endpoint_id
WHERE class_desc = 'ENDPOINT';
```

A Word about Auditing:

Auditing only works if it's:

- Done regularly
- Checked regularly
- Compared against previous audit results

Therefore, try to schedule these queries and dump the output somewhere. PowerShell is a great way to handle this type of task. However, don't just let the audit results pile up. The results should be checked as soon after the queries are run as is possible. Also, the audit results should be checked against previous results. This is how we can spot changes in our SQL Server environments over time.

Next Steps

- Read up on what the server-level permissions mean.
- Learn why a member of securityadmin is now considered the equivalent of a sysadmin.
- Understand what each fixed server role can do.
- How to audit a SQL Server login's password age.