

DON'T LET YOUR PERMISSIONS BE HIJACKED!

Erland Sommarskog
Data Platform MVP

ERLAND SOMMARSKOG

Independent consultant based in Stockholm

SQL Server MVP since 2001

esquel@sommarskog.se

<http://www.sommarskog.se>

Slides and scripts are available on
<http://www.sommarskog.se/present>



PARTNERS & SPONSORS



ORGANIZERS



WYDZIAŁ ELEKTRONIKI,
TELEKOMUNIKACJI
I INFORMATYKI

GOLD SPONSORS



PATRONS



PATRONAT HONOROWY

Prezydent
Miasta Gdańska



WHAT THIS IS ABOUT

Permission Hijacking:

A person with less permissions than you have makes you unknowingly run code that (ab)uses your permissions to perform actions that are advantageous to the attacker.

PREAMBLE

ISN'T THIS TOO FANTASTIC?

- When it comes to security, nothing is “too unlikely”.
- If an attack can be performed, it will be attempted sooner or later, some place.
 - You don't want that place be your workplace.
- It all depends on the stakes.
 - Perform advantageous updates.
 - Steal data.
 - Install ransomware.
 - Just cause a mess (e.g., a disgruntled employee).

PREAMBLE

NOT ALL SITES ARE EQUAL

- In some places: only sysadmin and plain users and nothing in between.
 - Not very susceptible to the attacks I will discuss. (But see below.)
- Elsewhere: sysadmin, db_owner (application admin), db_ddladmin or similar (devs).
 - This is where you need to watch out!
- Don't overlook application logins.
 - They could have elevated permissions and SQL-injection holes.
- What about developers whose code gets deployed on the server?

THE EVIL DDL TRIGGER

[sysadmin.sql](#)

- When running index and statistics maintenance, an application admin could have installed a DDL trigger that performs malicious actions abusing sysadmin permissions.
 - Adding a user to sysadmin is just one example.
 - It could be data theft, data manipulation etc.
- Line of defence: PoLP, [Principle of Least Privilege](#).
- In this case: `EXECUTE AS USER = 'dbo'`.
 - This way you sandbox yourself into the current database and renounce all your permissions on server level.

EXECUTE AS COMMANDS

- **EXECUTE AS USER:** run commands as a database user.
 - Sandboxed into the current database.
 - No server-level access or access to other databases.
 - Can still use temp tables.
- **EXECUTE AS LOGIN:** run commands as a server login.
 - Sandboxed inside the instance.
 - Can access any database and server-level objects.
 - No access to file system or linked servers set up with self-mapping.
- To impersonate a user/login you need **IMPERSONATE** permission on that user/login.

ABOUT REVERT

- Impersonation is in force until the REVERT command is executed *in the same scope*.
 - REVERT in an inner scope has no effect towards an EXECUTE AS in outer scope.
- Extraneous REVERT are ignored without error.
- Tip: Always put REVERT in its own batch, so that is executed also if there is an error.

A SMALL HICCUP

[sysadmin.sql](#)

- `EXECUTE AS USER = 'dbo'` may produce error 15517:
Cannot execute as the database principal because the principal "dbo" does not exist, this type of principal cannot be impersonated, or you do not have permission.
- Happens easily with databases restored from other servers.
 - Due to SID mismatch between sys.databases and the database itself.
- Routine: set database owner when you restore a database.
- My opinion: a database should be owned by an SQL login that exists solely to own that database.

RUNNING DATABASE MAINTENANCE

1. Ola Hallengren's Maintenance Solution.

- IndexOptimize accepts the parameter `@ExecuteAsUser`, which you set to `'dbo'` (or another user of your choice).

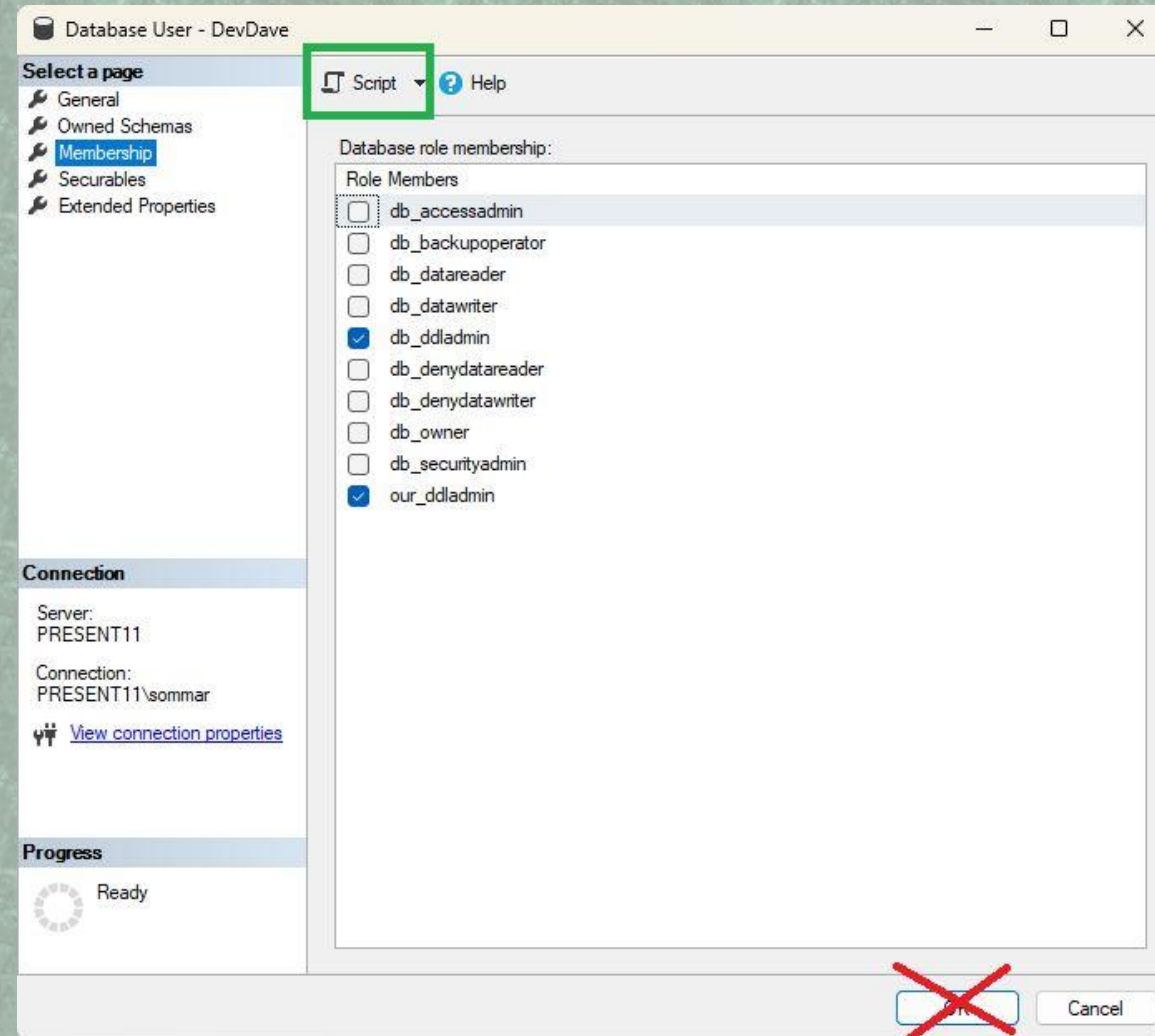
2. The maintenance plans in SSMS – no provision for EXECUTE AS USER.

- Only do backup and DBCC. Tell users to run index and stats maintenance themselves.
- Or for index and stats maintenance, have one task per database. Each task has its own login that is `db_owner` solely in that database. (Could be the database owner.)

DDL IN GENERAL

- For the server DBA any DDL on database level is dangerous, when there is a local db_owner or similar: creating indexes, users, granting permissions etc.
- Must bracket all DDL with `EXECUTE AS USER = 'dbo'` in that case.
- Corollary: cannot use UI in SSMS directly. Must use SQL commands.
- Or use the script button, see next slide.

SCRIPT BUTTON IN SSMS UI



DEVS AND DDL TRIGGERS

[sysadmin.sql](#)

- A user in db_ddladmin could create a DDL trigger to hijack the permissions of a db_owner user.
- This role should not have rights to create DDL triggers!
- Take out all users from db_ddladmin.
- Create a new role, e.g. our_ddladmin (or ask for an AD group) that you add as a member to db_ddladmin.
- Add devs to this role.
- Deny our_ddladmin ALTER ANY DATABASE DDL TRIGGER.
- Why not DENY db_ddladmin directly? SQL Server doesn't let us.

DML TRIGGERS AND STORED PROCEDURES

- When you insert/update/delete data in tables, your permissions can be hijacked by a trigger.
- When running stored procedures, a developer may have hidden malicious code somewhere.
- If you only need to protect sysadmin (no one between db_owner and plain users) same as before: impersonate dbo.
- To get data from outside the database, read into temp tables before starting impersonation.

HOW TO PROTECT DB_OWNER?

- If there are users who can change stored procedures, triggers etc, they can hijack db_owner.
- Protection: PoLP, Principle of Least Privilege.
- Create a sandbox user with limited permissions and impersonate that user. For instance:

```
CREATE USER SandboxUser WITHOUT LOGIN
GRANT EXECUTE TO SandboxUser
GRANT SELECT, INSERT, UPDATE, DELETE TO SandboxUser
EXECUTE AS USER = 'SandboxUser'
```

BUT SELECTS SHOULD BE SAFE?

[sysadmin.sql](#)

- A table could have a computed column that calls a UDF.
- UDF could call xp_cmdshell to connect back as sysadmin.
- Best practice: Keep xp_cmdshell disabled, enable and re-disable when you need it.
- But there are other extended stored procedures...
- There is also OPENQUERY & OPENROWSET (if enabled).
- So plain SELECTs also need EXECUTE AS USER.

AGENT JOBS

- DevDave approaches you and says: We need to run purges at night, can you create an Agent job to run this procedure?
- If you schedule it to run as sysadmin, Debbie and Dave can do whatever they like in that procedure.
- First line of defence: Hey, why don't you schedule it from Task Scheduler on the application server?
 - After all, Agent is for management tasks, not application jobs, isn't it?
- But you may not win that battle...

AGENT ALTERNATIVES

How we can run Agent jobs with a user with limited permissions:

- Set user on job step (Advanced page).
 - Uses EXECUTE AS USER, so sandboxed into the database.
- Set job owner.
 - Uses EXECUTE AS LOGIN. Cross-database works, linked servers may not.
- Use an Agent proxy user.
 - Most general solution, but requires a new AD account.
- User should always be fictive user, not a person that can leave the company.

ATTACKS THROUGH DEPLOYMENT

- You get a deployment package from the application team.
- It could include all sorts of nasty things:
 - Adding extra logins/users with elevated permissions.
 - Manipulate data (in any database).
- Or what about this:
 - Set up a temporary linked server to system B with sensitive data and where you also are sysadmin.
 - They set up another linked server to a test system C they have access to and dump the sensitive data there.
- How do you defend yourself against hijacking?

DEFENCE AGAINST DEPLOYMENT ATTACKS

- For a plain script, one means of protection:
`EXECUTE AS USER = 'dbo' WITH NO REVERT`
- `WITH NO REVERT` causes any `REVERT` in the script to fail.
- Rather than `dbo`, use a sandbox user in our `_ddladmin`, or whatever permission you are prepared to grant the app team.
- Use a sandbox login, if actions are expected outside the database.
- First run in QA, this will reveal unexpected but legit actions.
 - But not necessarily malicious actions; they may check the permissions.

DEPLOYMENT ATTACKS, CONT'D

- Deployment script may have legit actions that requires high permissions.
- Or deployment is through DACPAC or MSI install, with no options to control login.
- Auditing may be your only choice here.
- There are several options, but SQL Server Audit is the only that audits that it has been turned off.
- DDL Triggers? They can be turned off with `DISABLE TRIGGER` – which does not fire DDL triggers...

SUMMARY:

WHAT THIS HAS BEEN ABOUT

- Malicious users with some elevated permission can inject code that you unknowingly execute with your permissions, performing actions you don't agree to.
- Recall: Code can be DDL triggers, DML triggers, stored procedures and more.
- “Elevated permission” here includes access to the deployment pipeline!

SUMMARY: MEANS OF PROTECTION

- Principle of Least Privilege, PoLP.
- Impersonate dbo to get rid of your server-level permissions.
- To protect db_owner from being hijacked, use sandbox users.
- Never put people directly in db_ddladmin, but create a role our_ddladmin that you deny the rights to DDL triggers.
- Application jobs must never run as sysadmin, but always with a user with limited permissions.
- The application team may be in practice be equal to db_owner — even if they don't have access to the server.

FINAL WORDS

- Some of the attacks I have suggested may seem very contrived.
- Remember: if the stakes are high enough, a possible attack will be attempted, sooner or later at some place.
- Don't overlook the benevolent attacker.
 - That is, someone who performs an attack not out of malicious intent, but to circumvent corporate red tape to get their job done.
- You could have made those users sysadmin, but you decided to lock down their permissions. Certainly not a bad idea in itself.
- But if you don't watch out, they may find a way to “work around” that restriction.

THAT'S ALL FOLKS!

Erland Sommarskog, esquel@sommarskog.se.

Slides and scripts at <https://www.sommarskog.se/present>.

Credits to **Baptiste Devigne** for pointing out the SELECT threat.

Don't Let Your Permissions be Hijacked:

<https://www.sommarskog.se/perm-hijack.html>.

Clean-up script:

[sysadmin.sql](#)

SQLDay Lite 2026 Gdańsk

Session feedback



Slides and scripts



Article: Don't Let Your
Permissions Be Hijacked!

