Segurança de dados

Aula 04 – Controle de acesso

Gustavo Bianchi Maia

gustavo.maia@faculdadeimpacta.com.br

Sumário

- Definições
  - Surface Area
  - Segurança ao longo do processo de desenvolvimento de software.
  - Autenticação vs autorização
  - Estabelecendo uma conexão com o SQL Server
- Controle de acesso no SQL Server:
  - Principals
  - Classes (grupos de permissões)
  - Securables
  - Permissions
  - Papéis
- Impersonate
- Exemplos e exercícios

Surface Area

Definimos como área de superfície tudo aquilo que fica “exposto” em um servidor, ou seja, seus serviços, as portas de comunicação etc.

Quanto mais portas e janelas uma casa tem, mais oportunidades um invasor terá em suas tentativas. Mesmo portas blindadas e super resistentes são consideradas como um ponto de vulnerabilidade.

Ou seja, uma das práticas comuns de segurança é justamente reduzir a quantidade de pontos de acesso ao servidor, ou seja, reduzindo assim sua área superficial.

De forma equivalente, quanto mais acesso uma pessoa tem, maior a área de superfície que ela expõe. Então, quanto mais acessos, maior o risco de exposição e vazamento.

Sabendo disso, quais boas práticas aprendemos ?

Segurança como processo

- Seguro na especificação (Secure by Design)
  - Treinamento dos usuários
  - Análise de riscos
  - Revisão de código e testes de invasão
  - Testes automatizados
  - Modelo avançado de segurança
- Seguro no desenvolvimento (Secure by Development)
  - Manutenções automáticas / assistidas
  - Seguindo as melhores práticas
  - Updates contínuos
- Seguro por padrão (Secure by Default)
  - A configuração padrão é a de um sistema seguro
  - Superfície de ataque minimizada
  - A maioria dos serviços não obrigatórios vem desligados
  - Habilidade de desligar procedures ou funções externas
- Seguro na comunicação (Secure by Communications)
  - Padrões de segurança de rede novos.
  - Separações por schemas e usuários
  - Criptografia de dados
  - Auditoria de eventos

Sabendo disso, quais boas práticas aprendemos ?

Autenticação vs autorização

Autenticação é o processo de validação de credenciais, ele visa comprovar que você é quem você diz que é.

Uma boa autenticação é baseado em :

- Algo que você é
- Algo que você sabe
- Algo que você tem

Normalmente é realizado na entrada ou nas primeiras camadas de um sistema / servidor.

__single sign on__* ou ***autenticação por modelo de confiança* um termo utilizado quando, ao se autenticar em um sistema, você está autenticado à outros sistemas que confiam uns nos outros.

Autorização é o processo de verificação de privilégios, ele visa determinar se você (já autenticado), tem direitos de realizar [ou não] uma certa ação.

Estabelecendo uma conexão

Protocolo de comunicação:

São as maneiras em que uma comunicação pode ser estabelecida com um servidor, como:

- Transport Control Protocol (TCP)
- Names Pipes
- Shared Memory
- Virtual Interface Architecture (VIA)

Endpoints: são formas de entrada em um servidor

Existe [pelo menos] um Endpoint para cada protocolo habilitado, mas outros podem ser criados para fins específicos (Database Mirroring cria um endpoint para a porte 5022)

Conexão: Após conhecer a porta, o cliente envia um request de conexão com o servidor, isto é chamado de pré-login ou handshake (apresentação).

Logo em seguida é feita a solicitação de login com o servidor (autenticação)

Uma vez autenticado, o servidor faz uma solicitação de acesso ao banco de dados padrão do usuário (autorização de uso daquele banco).

Finalmente o usuário é autenticado à um database (obrigatoriamente).

Controle de acesso no SQL Server

Todo permissionamento no SQL Server é sempre concedito nos seguintes moldes:

```
GRANT { ALL [ PRIVILEGES ] }
  | permission [ ( column [ ,...n ] ) ] [ ,...n ]
  [ ON [ class :: ] securable ] TO principal [ ,...n ]
  [ WITH GRANT OPTION ] [ AS principal ]
```

Então, tudo é uma questão de identificar:

O que ? → `{ALL [ PRIVILEGES ]} | permission [( column [ ,...n ] )]`

Onde ? → `ON [ class :: ] securable ]`

Para quem ? → `TO principal [ ,...n ]`

Direitos de passar adiante ? → `[ WITH GRANT OPTION ]`

Ex: de permissões de leitura (o que) na tabela (onde, classe: objeto) cliente (onde, nome do objeto) para o usuário [salas\fit123456] (quem) com permissões para que ele passe estes direitos para outra pessoa (Direitos de passar adiante).

```
GRANT SELECT ON cliente TO [salas\fit123456] WITH GRANT OPTION
```

Controle de acesso no SQL Server

Mas antes de sair testando no SQL, várias definições ainda precisam ser feitas:

- **Principals**: São entidades que podem solicitar recursos, às quais podemos conceder [ou remover] permissões, são exemplos :
  - Server Logins - responsáveis pela autenticação no servidor
    - SQL Server login
    - Windows Authentication login
    - Windows Authentication group
    - Logins baseados em certificados
  - Database User - contraparte do login dentro do banco de dados
  - Roles - papéis nomeados (representados por um conjunto de permissões necessárias para a execução de uma função)
    - Server Roles
    - Database Roles
    - Application Roles
  - Schemas - “sub-divisão” de um banco de dados

Controle de acesso no SQL Server

- **Securables**: é algo que você pode (ou deve) proteger, são divididos em classes, são sobre eles que as permissões são atribuídas :
  - Server
    - Login
    - Server Role
    - Database *entre outros
  - Database
    - Schema
    - User
    - Database Role *entre outros
  - Schema
    - Type
    - Object
      - Function
      - Procedure
      - Table
      - View *entre outros

Permissões - Class:Database

ALTER
ALTER ANY APPLICATION ROLE
ALTER ANY ASSEMBLY
ALTER ANY ASYMMETRIC KEY
ALTER ANY CERTIFICATE
ALTER ANY CONTRACT
ALTER ANY DATABASE AUDIT
ALTER ANY DATABASE DDL TRIGGER
ALTER ANY DATABASE EVENT NOTIFICATION
ALTER ANY DATASPACE
ALTER ANY FULLTEXT CATALOG
ALTER ANY MESSAGE TYPE
ALTER ANY REMOTE SERVICE BINDING
ALTER ANY ROLE
ALTER ANY ROUTE
ALTER ANY SCHEMA
ALTER ANY SERVICE
ALTER ANY SYMMETRIC KEY
ALTER ANY USER
AUTHENTICATE
BACKUP DATABASE
BACKUP LOG
CHECKPOINT
CONNECT
CONNECT REPLICATION
CONTROL
CREATE AGGREGATE
CREATE ASSEMBLY
CREATE ASYMMETRIC KEY
CREATE CERTIFICATE
CREATE CONTRACT
CREATE DATABASE
CREATE DATABASE DDL EVENT NOTIFICATION
CREATE DEFAULT
CREATE FULLTEXT CATALOG
CREATE FUNCTION
CREATE MESSAGE TYPE
CREATE PROCEDURE
CREATE QUEUE
CREATE REMOTE SERVICE BINDING
CREATE ROLE
CREATE ROUTE
CREATE RULE
CREATE SCHEMA
CREATE SERVICE
CREATE SYMMETRIC KEY
CREATE SYNONYM
CREATE TABLE
CREATE TYPE
CREATE VIEW
CREATE XML SCHEMA COLLECTION
DELETE
EXECUTE
INSERT
REFERENCES
SELECT
SHOWPLAN
SUBSCRIBE QUERY NOTIFICATIONS
TAKE OWNERSHIP
UPDATE
VIEW DATABASE STATE
VIEW DEFINITION

Permissões - Class:Object

ALTER

CONTROL

DELETE

EXECUTE

INSERT

RECEIVE

REFERENCES

SELECT

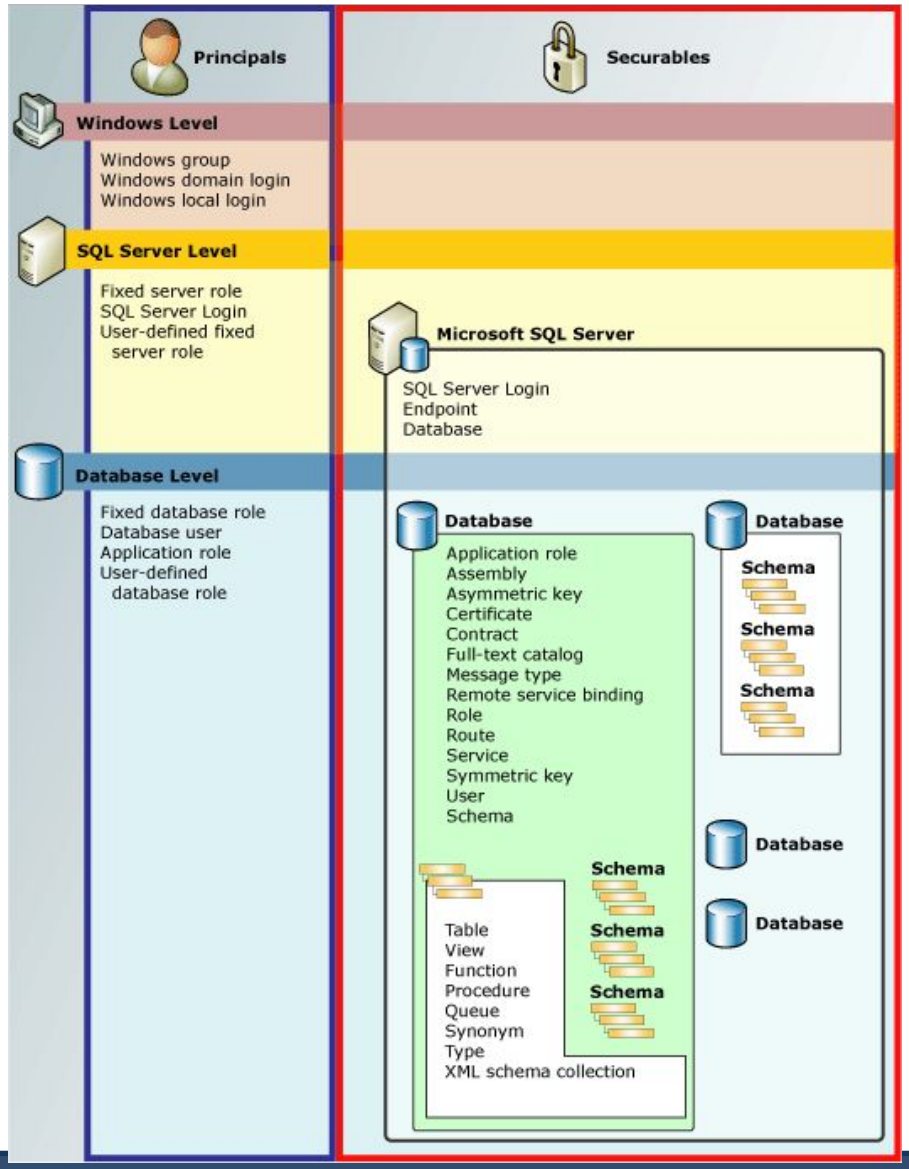
TAKE OWNERSHIP

UPDATE

VIEW CHANGE TRACKING

VIEW DEFINITION

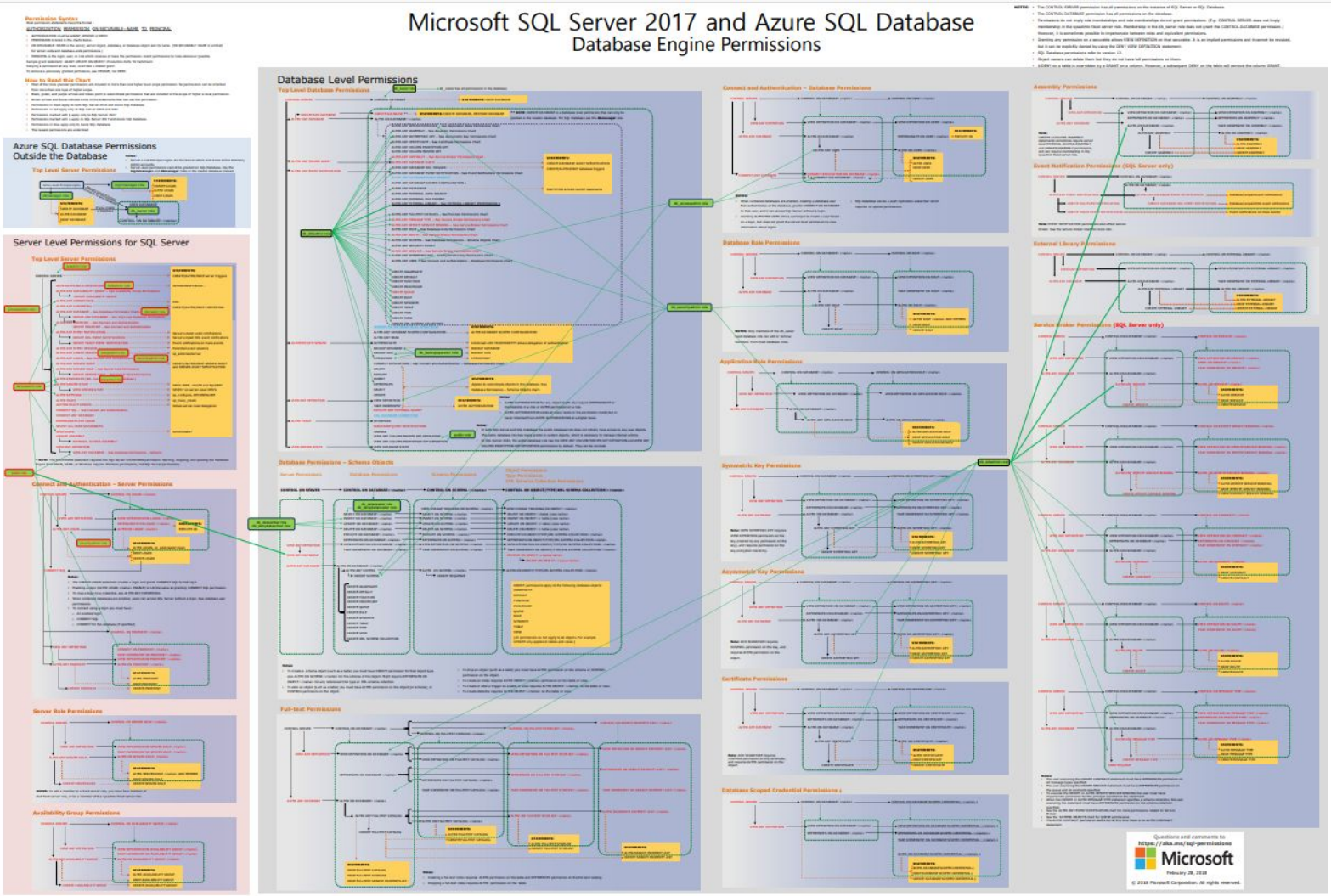
Controle de acesso no SQL Server



Fonte: [Permissions Hierarchy (Database Engine) - SQL Server | Microsoft Docs]

Detalhado: https://aka.ms/sql-permissions-poster

Controle de acesso no SQL Server



Fonte: https://aka.ms/sql-permissions-poster

Roles

Roles - papéis nomeados (representados por um conjunto de permissões necessárias para a execução de uma função)

- Server Roles (permissões para 'logins')
  - **sysadmin** (ou sa) - podem realizar quaisquer ações
  - **serveradmin** podem alterar quaisquer configurações do servidor e reinicia-lo.
  - **setupadmin** podem gerenciar linked servers (adicionar ou remover), replicação de dados, extended stored procedures e executar algumas procedures do sistema como sp_serveroption.
  - **securityadmin** podem criar e gerenciar logins, além de processos de auditoria e logs.
  - **processadmin** podem gerenciar processos em execução no SQL Server.
  - **dbcreator** podem criar, alterar e manipular o tamanho dos bancos.
  - **diskadmin** podem gerenciar os arquivos físicos no disco.

Roles

- Database Roles
  - **db_owner** podem executar todas as atividades de configuração e manutenção no banco de dados, bem como descartar o banco de dados.
  - **db_securityadmin** podem modificar a associação de funções e gerenciar permissões. A adição de entidades nesta função pode habilitar o escalonamento não intencional de privilégios.
  - **db_accessadmin** podem adicionar ou remover o acesso ao banco de dados para logons do Windows, grupos do Windows e logons do SQL Server.
  - **db_backupoperator** podem fazer backup do banco de dados.
  - **db_ddladmin** podem executar qualquer comando Data Definition Language (DDL) em um banco de dados.
  - **db_datawriter** podem adicionar, excluir ou alterar dados em todas as tabelas de usuário.
  - **db_datareader** podem ler todos os dados de todas as tabelas de usuário.
  - **db_denydatawriter** não podem adicionar, modificar ou excluir nenhum dado nas tabelas de usuário de um banco de dados.
  - **db_denydatareader** não podem ler nenhum dado nas tabelas de usuário de um banco de dados.

Roles

Pergunta - Qual a diferença entre:

Usar role db_datareader (direito de select em todas as tabelas do banco)
E dar grants individuais (select) em todas as tabelas do banco ?

Granularidade: db_datareader --> para tudo
grants individuais --> 1 tabela por vez

Manutenção: db_datareader --> +1 tabela --> automatico
grants individuais --> +1 tabela --> precisa de mais um grant

Maneira de revogar o acesso:

db_datareader + deny --> tudo menos 1 tabela
grants individuais --> só não liberar aquela tabela
ou dar revoke naquela permissão

Credenciais

Normalmente contas do SQL já tem por direito os direitos do usuário que executa o serviço do MSSQLSERVER, e os processos agendados no SQL Server Agent tem os direitos do usuário que executa o serviço MSSQLSERVERAGENT.

Porém, nunca é uma boa prática dar muitos privilégios à essas contas, que acabam muitas vezes sendo restritas ao próprio servidor (lembram-se da Surface Area ?)

Mas como dar direitos para contas acessarem o domínio, se nem o servidor pode ?

Credenciais (credentials) são um meio de prover à logins internos do SQL (SQL Server Logins) o direito de acesso externo ao servidor.

Por definição, um login do Windows (windows authentication login), já tem direitos fora do servidor SQL, porém, ainda sim, posso lhe associar uma credencial, dando-lhe um conjunto ‘extra’ de direitos.

Contas internas (SQL Server Login), para acessarem algo além do que a conta de serviço tem, precisam de credenciais novas.

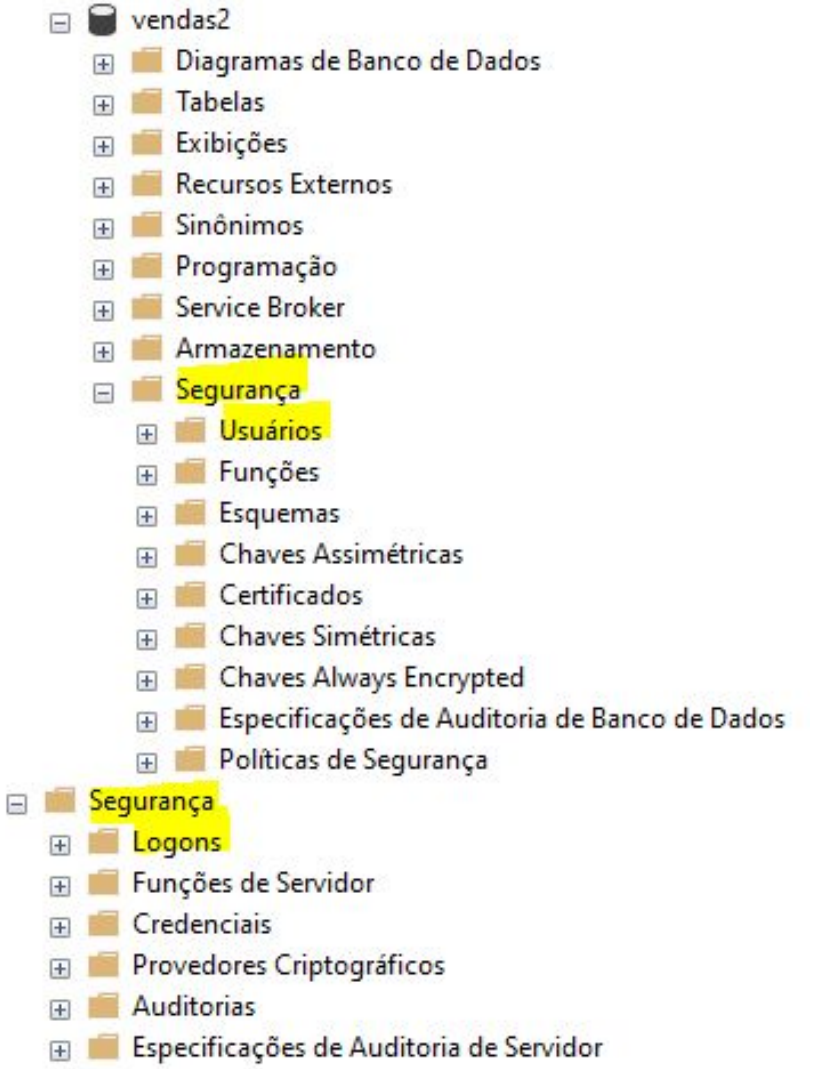
Interface

Existem dois menus de segurança:

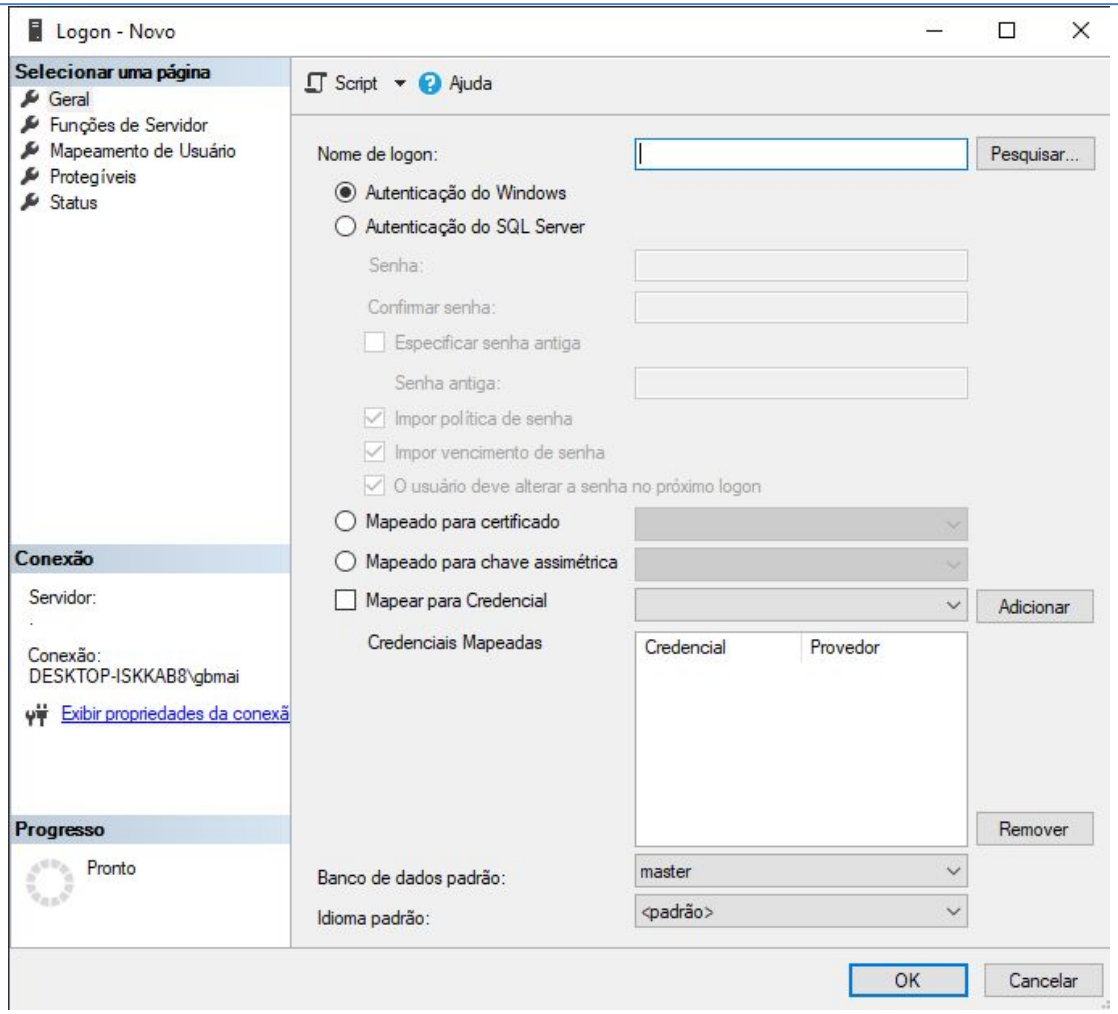
- Um para o servidor, onde é possível gerenciar logins, server roles e demais controle a nível de instância.
- Um para cada database criado, onde é possível gerenciar usuários, database roles, schemas e demais controles para aquele banco de dados.

Normalmente, um login está associado à um usuário para cada banco que aquele login tem acesso.

Isso é chamado de mapeamento.



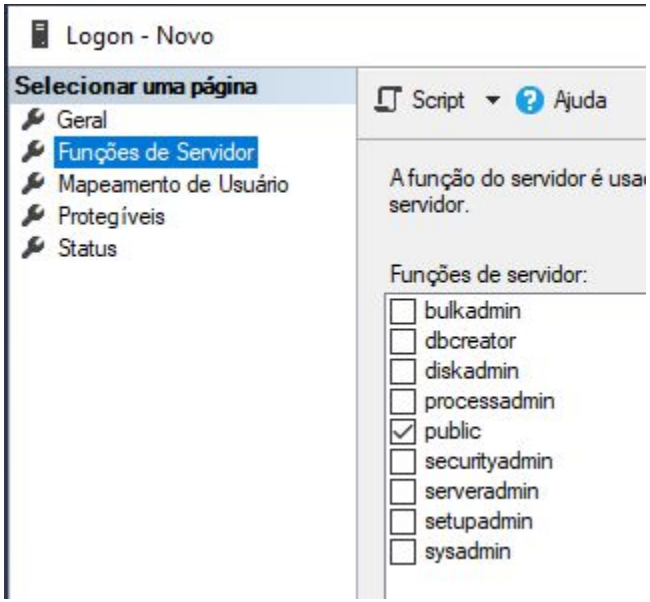
Interface

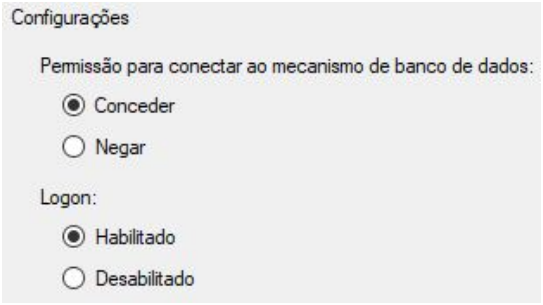


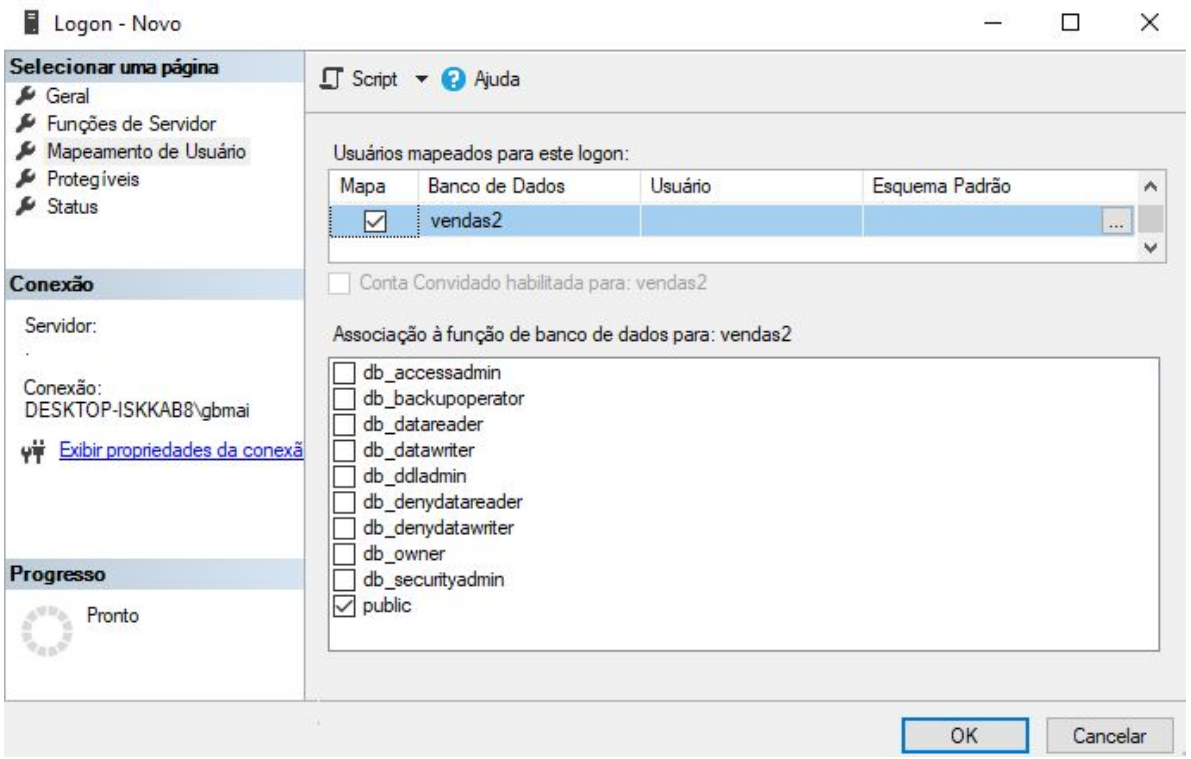
Quase todo o permissionamento básico é realizado no menu de logins.

Botão direito sobre o menu logins > new login.

Interface







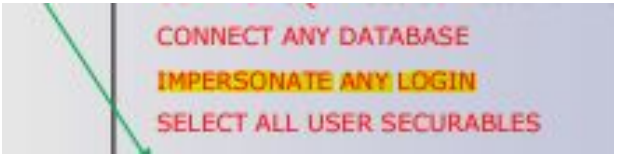
Quase todo o permissionamento básico é realizado no menu de logins.

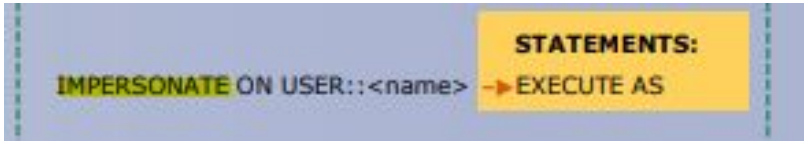
Botão direito sobre o menu logins > new login.

Permissões especiais

Permissão de visualizar DMV's administrativas

```
GRANT VIEW server state TO dbajr
```





Impersonate é a capacidade de se passar por outro login / usuário utilizando as novas permissões para a realização das atividades.

Muitas vezes utilizado por um sysadmin para testar as permissões de outros usuários.

```
EXECUTE AS login = '<nome do login>'
    <comandos>
REVERT
```

O comando para remover permissões é o **revoke**, de forma análoga existe o comando **deny**, que adiciona uma restrição, ambos funcionam nos mesmos moldes do GRANT.

Exercícios

Escolha um de seus bancos de dados. (não vale: master, model, msdb, tempdb)

Logado como um administrador. Via interface, crie um login chamado teste sem nenhuma server role além de public (a básica, já pré-definida).

Mapei-o ao banco escolhido, deixando-o [novamente] só com a database role de public (a básica, já pré-definida).

No console de comando (janela padrão para digitar código SQL), entre no banco escolhido (use <database>) e digite:

```
GRANT VIEW SERVER STATE TO [teste]
```

Por que deu errado ?

Como você faria para corrigir este erro ? ...então corrija-o...

Exercícios

Para testar se a permissão de view server state foi devidamente aplicada:

```
EXECUTE AS login = 'teste'
SELECT SYSTEM_USER
     --DMV para consulta de 'working threads'
     SELECT scheduler_id,
            current_tasks_count,
            runnable_tasks_count
     FROM   sys.dm_os_schedulers
     WHERE  scheduler_id < 255
REVERT
SELECT SYSTEM_USER
```

Este comando utitiliza-se do recurso de impersonate para, ainda logado como sysadmin, testar se o usuário teste já tem as permissões necessárias.

Exercícios

Agora, ainda logado como sysadmin, crie a seguinte procedure no banco de sua escolha (não vale: master, model, msdb, tempdb)

```
CREATE PROCEDURE Sp_aproceduremaissensacional
AS BEGIN
      SELECT 'Hoje é: ' + CONVERT(VARCHAR, Getdate(), 103)
  END
go
```

Agora tente executá-la com o usuário teste

(logando-se com ele, ou usando a função impersonate)

```
EXECUTE AS login = 'teste'
SELECT SYSTEM_USER
     EXEC Sp_aproceduremaissensacional
REVERT
SELECT SYSTEM_USER
```

Por que deu errado ?

Como você faria para corrigir este erro ? ...então corrija-o...

Exercícios

Agora, logado com o usuário administrador, execute a seguinte procedure:

```
CREATE PROCEDURE Sp_umaprocedurequalquer WITH EXECUTE AS owner
AS  BEGIN
      SELECT SYSTEM_USER
END
```

De as devidas permissões de execução ao usuário teste e execute o seguinte código:

```
EXECUTE AS login = 'teste'
     EXEC Sp_umaprocedurequalquer
REVERT
```

Agora altere a procedure sp_umaprocedurequalquer e remova o with execute as owner e veja qual o resultado.

Tente explicar a mudança de comportamento.

Obrigado

Segurança de dados

Aula 04 – Controle de acesso

- **Gustavo Bianchi Maia**

gustavo.maia@faculdadeimpacta.com.br