KAPITOLA 7

Zabezpečení SQL Serveru

Rob Walters

Témata kapitoly:

- Čtyři oblasti rámce zabezpečení
- Přehled zabezpečení SQL Serveru 2005
- Ověřování a autorizace
- Podpora šifrování v SQL Serveru 2005
- Ochrana SQL Serveru 2005
- Jak hackeři napadají SQL Server
- Shrnutí

S rostoucím znepokojením souvisejícím s ochranou osobních dat a rozšířením počítačových virů se pro Microsoft stal vývoj metodiky zajišťující bezpečnost počítačů nesmírně důležitým úkolem. V roce 2003 Microsoft představil iniciativu Trustworthy Computing, která popisovala kroky, jejichž provedení bylo nutné k tomu, aby lidé pociťovali jistotu při používání zařízení závisejících na počítačích a softwaru. (Tuto iniciativu naleznete na *http://www.microsoft.com/mscorp/twc/twc_whitepaper.mspx.*) Na základě této iniciativy staví Microsoft SQL Server 2005 takzvaný rámec zabezpečení (Security Framework), který vidíte na obrázku 7.1.

Čtyři oblasti rámce zabezpečení

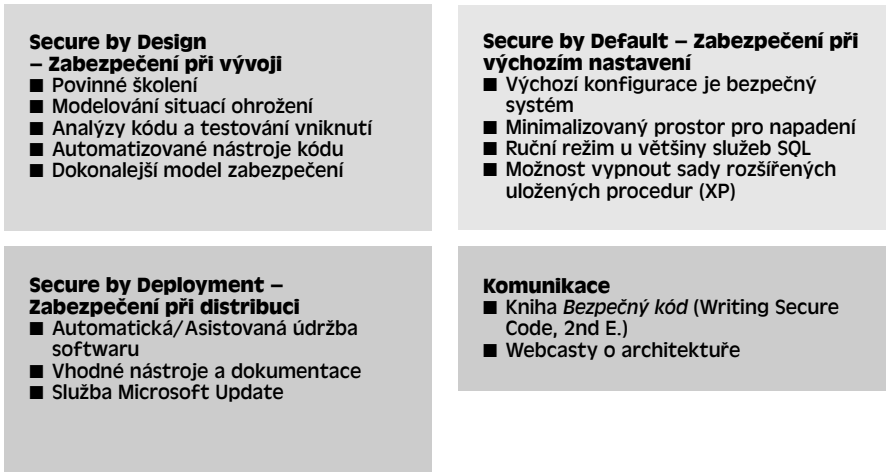
Zabezpečení SQL Serveru 2005 je uspořádáno do čtyř oblastí: zabezpečení při vývoji, zabezpečení při výchozím nastavení, zabezpečení při distribuci a komunikace.

Zabezpečení při vývoji

Na zabezpečení bylo dbáno při vývoji všech předchozích verzí SQL Serveru. U SQL Serveru 2005 skupina, která produkt vyvíjela, zajistila, aby všichni věděli, co je v souvislosti se zabezpečením v sázce. Celý vývojový tým prošel povinným školením týkajícím se zabezpečení a vytvářely se a analyzovaly modely ohrožení u všech komponent všech funkcí v rámci produktu. Microsoft bere zabezpečení velmi vážně a návrháři funkcí v rámci SQL Serveru považovali zabezpečení v konečném konceptu za nejvyšší prioritu.

Zabezpečení při výchozím nastavení

Zabezpečení při výchozím nastavení je jednou z nejmarkantnějších oblastí, se kterou se uživatelé SQL Serveru set-



Obrázek 7. 1: Rámec zabezpečení v SQL Serveru 2005

kají. Projeví se jednoduše při instalaci SQL Serveru s výchozími možnostmi. Uživatelé předchozích verzí SQL Serveru si všimnou, že služby jako například SQL Server Agent jsou implicitně zakázané. Navíc jsou zakázané rozšířené uložené procedury jako *xp_cmdshell* a dotazy používající funkci OPENROWSET. Toto výchozí zakázání funkcí má za cíl minimalizovat oblast náchylnou k napadení a účinky tohoto nastavení lze pozorovat v rámci celého produktu.

Zabezpečení při distribuci

Pravděpodobně nejnáročnější úkol týkající se SQL Serveru je efektivní distribuce do produkčního prostředí. S tolika různými konfiguracemi a funkcemi může být pro správce obtížné mít přehled o nejnovějších opravných souborech a doporučených postupech. SQL Server 2005 je nyní součástí služby Microsoft Update, čímž se zmírňují nepříjemnosti spojené s hledáním nejnovějšího opravného souboru, který by se měl aplikovat.

Komunikace

I před vlastním uveřejněním SQL Serveru 2005 bylo dostupné množství technických informací v různých podobách. Uživatelé beta verze se vzdělávali prostřednictvím dokumentů white paper, webcastů a aktivních diskusních skupin. Většina těchto webcastů a dokumentů white paper byla v současné době aktualizována a poskytuje bohatý vzdělávací obsah.

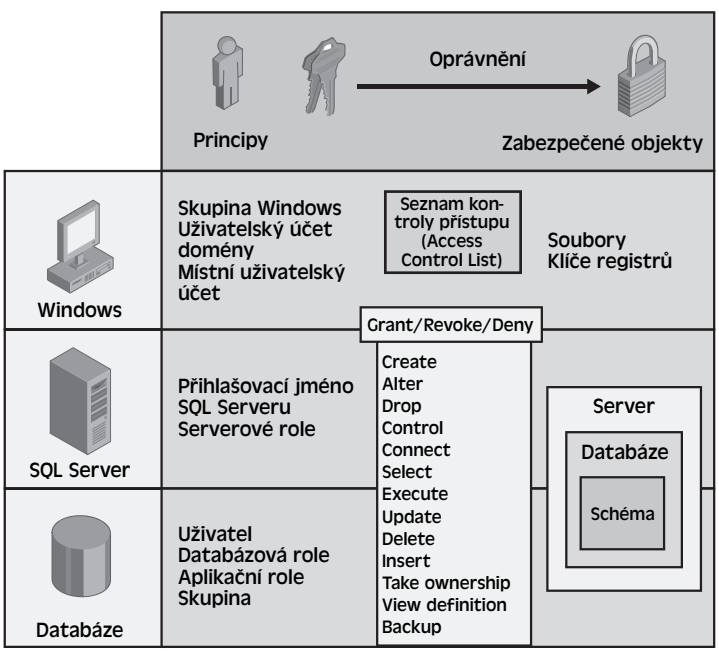
Všechny edice SQL Serveru 2005 obsahují funkce zabezpečení, které uživatelům pomáhají chránit jejich data. V této kapitole se budeme věnovat následujícím aspektům zabezpečení SQL Serveru:

- Přehled zabezpečení, včetně ověřování a autorizace
- Oddělení uživatele od schématu
- Šifrování dat v rámci databáze a při jejich přenosu
- Ochrana SQL Serveru 2005

Přehled zabezpečení SQL Serveru 2005

Pokud již dobře znáte koncept *uživatelů*, *rolí* a přihlašovacích jmen (logins) SQL Serveru, můžete pravděpodobně přejít k další části této kapitoly. Těm, kteří tyto koncepty neznají, zde poskytneme krátké vysvětlení. Koncepty uživatelů a rolí existují jak v prostředí Microsoft Windows, tak v SQL Serveru. V abstraktním smyslu se označují jako *principy (principals)*. Principy jsou entity, které mohou požadovat prostředky z aplikace nebo operačního systému. V případě Windows jsou principy entitami, jako například Domain Users a Local Users. V SQL Serveru jsou těmito entitami přihlašovací jména a serverové role. V rámci databáze jsou těmito entitami například uživatelé databáze, databázové role a aplikační role.

Co tedy lze s entitami provádět? Řekněme, že máte objekt, jako je soubor nebo databázová tabulka, ke které chcete umožnit přístup. Tyto objekty, neboli *zabezpečené objekty (securables)*, jsou prostředky, ke kterým autorizační systém řídí přístup. Některé zabezpečené objekty mohou být obsaženy uvnitř jiných objektů, čímž vytváří vnořené hierarchie zvané *obory (scope)*, které samy mohou být zabezpečené. K zabezpečeným oborům v databázovém stroji SQL Server Database Engine patří *server*, *databáze* a *schéma*. Každý objekt v SQL Serveru má definovanou množinu oprávnění, které mohou být přidělovány principu. Na obrázku 7.2 je graficky znázorněn model princip – zabezpečený objekt – oprávnění.



Obrázek 7.2: Model princip – zabezpečený objekt – oprávnění

Přihlašovací jména SQL Serveru

Po představení rozsáhlého schématu zabezpečení SQL Serveru a způsobu, jakým se vztahuje k zabezpečení Windows, je čas věnovat se oblastem specifickým pro SQL Server. Pro připojení k SQL Serveru potřebujete přihlašovací jméno. Toto jméno může být kombinací uživatelského jména (jako například *Login1*) a složitého hesla. Do SQL Serveru lze také jako přihlašovací jméno přidat existující účet systému Windows, a vyhnout se tak vytvoření samostatného uživatelského jména a hesla. SQL Server tedy podporuje dva typy přihlašovacích jmen: přihlašovací jména systému Windows a přihlašovací jména SQL Serveru.

Samotná přihlašovací jména nemají přístup k žádné konkrétní databázi v SQL Serveru; v podstatě umožňují pouze připojení k SQL Serveru. Přihlašovací jména jsou navíc entity, kterým lze udělovat oprávnění na úrovni serveru za účelem provádění určitých akcí. Tyto akce jsou spojeny se serverovými rolemi, jako jsou například sysadmin, diskadmin a dbcreator. V tabulce 7.1 je uveden seznam rolí serveru a jejich odpovídající funkce.

Tabulka 7.1: Pevně dané role SQL Serveru 2005

Role serveru	Popis
sysadmin	Členové mohou v serveru provádět jakoukoliv činnost. Implicitně jsou členem této fixní serverové role všichni členové skupiny BUILTIN\Administrators systému Windows, což je skupina lokálních správců. Účet služby SQL Serveru je také členem této role.
dbcreator	Členové mohou vytvořit (create), změnit (alter), vymazat (drop) a obnovit (restore) jakoukoliv databázi.
diskadmin	Tato role se používá pro správu souborů disku. Většina možností se týká přidávání a odebírání zálohovacích zařízení.
Processadmin	Členové mohou ukončit procesy spuštěné v instanci SQL Serveru. Tato role je užitečná pro případ, kdy někomu chcete umožnit, aby ukončil dlouho probíhající dotaz nebo osamocené připojení.
securityadmin	Členové mohou spravovat přihlašovací jména a jejich vlastnosti. Mohou spustit příkazy GRANT, REVOKE a DENY na úrovni serveru i databáze. Mohou také znovu nastavit hesla pro přihlašovací jména SQL Serveru. Tato role nemá právo přiřadit oprávnění k objektům uvnitř databáze. Pokud chcete, aby to členové securityadmin mohli provést, musíte jejich přihlašovací jména zařadit do fixní databázové role db_accessadmin v konkrétní databázi.
Bulkadmin	Členové mohou spustit příkaz BULK INSERT. I přes členství v této roli je stále vyžadováno, aby uživatel měl k aktualizovanému objektu přístup, pokud není zároveň členem role sysadmin.
serveradmin	Členové mohou měnit konfiguraci serveru a server vypnout.
setupadmin	Členové mohou přidat a odebrat propojené servery a také provádět některé systémové uložené procedury.
public	Členy role public jsou všechna platná přihlašovací jména SQL Serveru.

Uživatelé databáze

Přihlašovací jména lze vytvořit v SQL Server Management studiu nebo příkazem *CREATE LOGIN*. Poté, co vytvoříte přihlašovací jméno, je přirozeným dalším krokem povolení přístupu k určité databázi. Databáze samotné mají vlastní sady rolí definující specifický přístup a akce, které uživatelé těchto rolí mohou provádět. Před přidělením přístupu k databázi je nutné vytvořit uživatele databáze pro toto přihlašovací jméno. Uživatele databáze lze vytvořit pomocí SQL Server Management Studia nebo pomocí příkazu T-SQL *CREATE USER*.

Poté, co vytvoříte uživatele databáze, lze jej přiřadit do jedné z databázových rolí. V tabulce 7.2 je uveden seznam rolí, které mají všechny databáze.

Tabulka 7.2: Pevně dané role databáze v SQL Serveru 2005

Role databáze	Popis
db_accessadmin	Členové mohou přidávat a odebírat přístup pro přihlašovací jména libovolného typu (uživatelé systému Windows, skupiny Windows a přihlašovací jména SQL Serveru).
db_backupoperator	Členové mohou databázi zálohovat.
db_datareader	Členové mohou číst všechna data ze všech tabulek.
db_datawriter	Členové mohou přidávat, odstraňovat nebo měnit data ve všech tabulkách.
db_ddladmin	Členové mohou spustit jakýkoliv příkaz Data Definition Language (DDL) v databázi.
db_denydatareader	Členové nemohou číst žádná data v tabulkách v rámci databáze.
db_denydatawriter	Členové nemohou přidávat, upravovat nebo odstraňovat žádná data v tabulkách v rámci databáze.
db_owner	Členové mohou provádět všechny aktivity při konfiguraci a údržbě databáze, včetně jejího vymazání.
db_securityadmin	Členové mohou měnit členství v rolích a spravovat oprávnění.

Existuje zvláštní role, která je známá jako role public. Všichni vytvoření uživatelé databáze jsou členem role public. Tato role definuje výchozí oprávnění pro uživatele v určité databázi. Nemohou do ní být přiřazení uživatelé, skupiny nebo role, protože každý do této role implicitně patří. Tuto roli nelze vymazat. Kvůli ochraně před neoprávněným přístupem k datům byste proto měli minimalizovat oprávnění udělená roli public. Namísto toho udělte oprávnění jiným databázovým rolím a uživatelským účtům přiřazeným k přihlašovacím jménům.

Uživatelský účet Guest

V souvislosti s neoprávněním přístupem k datům je důležité zmínit zvláštní uživatelský účet dostupný v SQL Serveru, který se nazývá Guest. Tento účet je implicitně vytvořen v nových uživatelských databázích, ale existuje i v databázi master a v databázi tempdb. Účet Guest je ovšem implicitně zakázaný, což znamená, že v rámci databáze nemůže být použit pro přístup. Účet Guest umožňuje přihlášení a přístup k databázi bez uživatelského účtu. Přihlašovací jméno přijímá identitu uživatele Guest, když jsou splněny všechny následující podmínky:

- Přihlašovací jméno má přístup k instanci SQL Serveru, ale nemá přístup k databázi použitím vlastního uživatelského účtu nebo prostřednictvím členství ve skupině systému Windows.
- Databáze obsahuje uživatelský účet Guest.
- Účet Guest je v databázi povolen.

Oprávnění lze přidělit uživatelskému účtu Guest stejně jako jakémukoliv jinému uživatelskému účtu. Použití účtu Guest byste se ovšem měli vyhnout, protože oprávnění udělená tomuto účtu získají všechna přihlašovací jména, která nemají vlastní databázová oprávnění. Musíte-li účet Guest použít, udělte mu pouze minimální oprávnění.

Do této chvíle jsme se věnovali různým aspektům objektů princip na úrovni serveru a databáze. Úplné seznámení se s těmito tématy je mimo rámec této knihy; podrobné informace o správě uživatelů v SQL Serveru naleznete v SQL Server Books Online.

Ověření a autorizace

Dříve, než se podrobněji seznámíme s koncepty ověření a autorizace, je důležité zmínit novou funkci v SQL Serveru 2005 s názvem koncový bod. V předchozích verzích SQL Serveru se klienti mohli připojovat prostřednictvím protokolů TCP, Named Pipes, Shared Memory a VIA. Pokud byl jeden z těchto protokolů v serveru povolen a uživatel měl platné přihlašovací jméno, připojení bylo přijato. SQL Server 2005 zavádí oddělení tohoto chování pomocí koncových bodů.

Koncové body lze považovat za bod vstupu do SQL Serveru. Správci mohou vytvořit koncový bod nejen pro TCP, Named Pipes, Shared Memory a VIA, ale také pro HTTP. Poté, co je koncový bod vytvořen, lze omezit přístup tak, aby se uživatelé mohli připojit pouze prostřednictvím určitého typu koncového bodu. Například lze vytvořit přihlašovací jméno s názvem Login1 a povolit přístup ke koncovému bodu HTTP a zakázat přístup ke všem ostatním koncovým bodům (TCP, Named Pipes, Shared Memory a VIA). V tomto stavu má Login1 přístup k SQL Serveru pouze prostřednictvím koncového bodu HTTP; k SQL Serveru se nemůže připojit přímo pomocí protokolu TCP nebo jakéhokoliv jiného protokolu. Abychom viděli, jaký má takováto verifikace koncového bodu vliv na ověřování, uvedeme příklad procesu vytvoření klientského připojení.

Jak klienti vytváří připojení

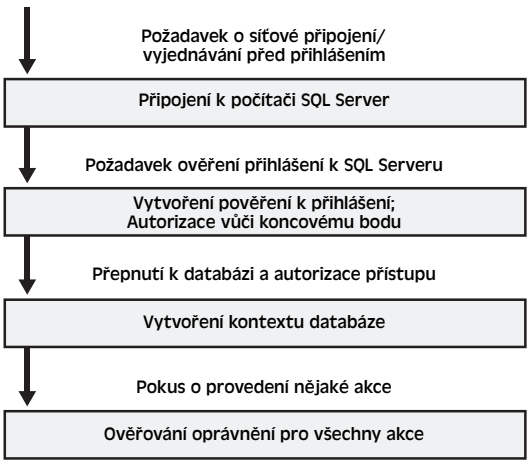
Pokud chce klient TCP vytvořit připojení k SQL Serveru, musí nejprve vědět, ke kterému portu se má připojit. V předchozích verzích SQL Serveru vždy na portu UDP 1434 čekalo vlákno procesu. Jeho účelem bylo vrátit informace o všech instancích SQL Serveru, které byly spuštěny, spolu s čísly portů těchto instancí. Vše, co musel klient učinit, bylo vytvořit připojení k tomuto portu a zjistit, ke kterému portu se musí připojit, aby se dostal na konkrétní instanci SQL Serveru. Tento proces všeobecně fungoval až do chvíle, kdy hackeři nalezli způsob, jak provést napadení Denial of Service (DoS) nepřetržitým odesíláním paketů tomuto portu s požadavkem o výčet portů. Vzhledem k tomu, že byl tento proces výčtu součástí služby SQL Server, vir „SQL Slammer“ způsobil uživatelům SQL Serveru značné potíže. V SQL Serveru 2005 byla tato funkce vyjmuta do oddělené služby

s názvem SQL Server Browser. Nyní lze tuto funkci vypnout a zapnout, aniž byste ovlivnili samotnou službu SQL Server.

POZNÁMKA

Potíže s DoS lze zmírnit, pokud v bráně firewall zablokujete port 1434. Obáváte-li se však napadení v síti intranet, může být vhodné tuto službu neprovozovat a namísto toho explicitně dodávat čísla portů do propojovacího řetězce.

Poté, co byla učiněna žádost o síťové připojení a provedeno vyjednávání před přihlášením, musí SQL Server ověřit uživatele, aby nejprve zjistil, zda má povolen přístup. Na obrázku 7.3 je zobrazen proces ověřování.



Obrázek 7.3: Model ověřování SQL Serveru 2005

V této chvíli služba přijme pověření přihlášení poskytnuté uživatelem a pokusí se je ověřit. Je-li ověření úspěšné, přihlášení je autorizováno vůči koncovému bodu, který odpovídá připojení vytvořenému k SQL Serveru. V tomto příkladu je u přihlašovacího jména zkontrolováno, zda mu bylo uděleno oprávnění CONNECT ke koncovému bodu TCP. Je-li tomu tak, proces ověřování pokračuje; v opačném případě v tomto bodě připojení selže.

POZNÁMKA

Implicitně je novým přihlašovacím jménům uděleno oprávnění CONNECT pro koncový bod TCP.

Poté, co přihlášení projde kontrolou koncového bodu, přepne SQL Server do kontextu databáze (výchozí databáze definovaná ve vlastnostech přihlašovacího jména nebo stanovená v řetězci připojení). Dále se pokusí ověřit přihlašovací jméno jako uživatele této databáze. Je-li ověření úspěšné, připojení je také úspěšné; pokud nikoli, připojení selže. Když je vytvořen kontext databáze a přihlášení bylo ověřeno, uživatel může provádět svou práci na serveru.

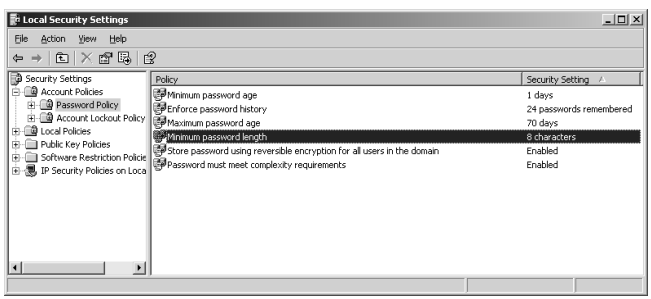
Zásady hesel

V prostředí Windows mohou správci nastavit vypršení platnosti přihlášení a vynutit zásady hesel (například aby hesla měla určitou délku a obsahovala kombinaci zvláštních znaků). V SQL Serveru tradičně tato globální nastavení zásad nikdy nebyla respektována přihlašovacími jmény SQL. V SQL Serveru 2005 se oba typy přihlašovacích jmen (jak ze systému Windows, tak standardní SQL) řídí podle nastavení zásad skupiny definované v doméně Windows.

POZNÁMKA

Vynucení zásady hesel je dostupné pouze v SQL Serveru nainstalovaném v systému Windows Server 2003 nebo novějším.

Pro lepší představu o zásadách hesel definujme minimální délku hesla v okně Místní nastavení zabezpečení (Local Security Settings), které se nachází ve složce Nástroje pro správu (Administrative Tools) v okně Ovládací panely (Control Panel). Tento nástroj je zobrazen na obrázku 7.4.



Obrázek 7.4: Uzel Zásady hesla (Password Policy) v okně Místní nastavení zabezpečení (Local Security Settings)

Pomocí tohoto nástroje lze změnit různé parametry. V tomto příkladu změníme minimální délku hesla na osm znaků. Po provedení této změny lze pokračovat s vytvářením našeho nového přihlašovacího jména.

Namísto příkazu sp_addlogin, který používaly předchozí verze SQL Serveru, má SQL Server 2005 pro vytvoření přihlašovacího jména nový příkaz DDL: *CREATE LOGIN*.

```
CREATE LOGIN foo
WITH PASSWORD='123' ,
CHECK_POLICY=ON ,
CHECK_EXPIRATION=ON
```

Provádění tohoto příkazu se nezdaří, protože naše heslo obsahuje pouze tři znaky a předtím jsme stanovili minimální délku osm znaků.

```
Msg 15116, Level 16, State 1, Line 1
Password validation failed. The password does not meet
policy requirements because it is too short.
```

Nový systémový pohled sys.sql_logins vrátí informace o přihlašovacích jménech, jako například zda je nastavena zásada a vypršení, ale pokud chcete další informace týkající

se přihlašovacích jmen, tedy například kolik nesprávných hesel bylo pro toto přihlašovací jméno odesláno, lze použít vestavěnou funkci *LOGINPROPERTY*, jak vidíte zde.

```
DECLARE @name varchar(20)

SET @name='foo'

SELECT LOGINPROPERTY( @name, 'PasswordLastSetTime') AS PasswordLastSetTime,
LOGINPROPERTY( @name, 'IsExpired' ) AS IsExpired,
LOGINPROPERTY( @name, 'IsLocked' ) AS IsLocked,
LOGINPROPERTY( @name, 'IsMustChange' ) AS IsMustChange,
LOGINPROPERTY( @name, 'LockoutTime' ) AS LockoutTime,
LOGINPROPERTY( @name, 'BadPasswordCount') AS BadPasswordCount,
LOGINPROPERTY( @name, 'BadPasswordTime') AS BadPasswordTime,
LOGINPROPERTY( @name, 'HistoryLength') AS HistoryLength,
LOGINPROPERTY( @name, 'PasswordHash' ) AS PasswordHash
```

Oddělení uživatele a schématu

Představte si zaměstnance v organizaci, který intenzivně využívá SQL Server. Vytvoří mnoho tabulek a uložených procedur a poté jej jednoho dne tato organizace náhle propustí. Jako správce databáze máte za úkol znovu přiřadit vlastnictví všech objektů, které zaměstnanec vytvořil a vlastnil. V předchozích verzích SQL Serveru to byl zdlouhavý úkol. V SQL Serveru 2005 je tento úkol zjednodušen díky existenci objektu schéma.

SQL Server 2005 zavádí odlišný koncept schémat, než tomu bylo v předchozích verzích SQL Serveru. V SQL Serveru 2000 jsou uživatelé databází ekvivalentem schématu. Každý uživatel databáze je vlastníkem schématu, který má stejný název jako uživatel. Vlastník objektu je tedy identický s vlastníkem schématu, které objekt obsahuje. Toto chování mapování uživatele na schéma je příčina ztěžující nové přiřazení vlastnictví.

SQL Server 2005 uživatele a schémata odděluje, přičemž schémata jsou nezávislé objekty, které mohou obsahovat nula nebo více objektů. Uživatelé mohou vlastnit jedno nebo více schémat a lze jim přiřadit výchozí schéma. Není-li výchozí schéma určeno, je uživateli implicitně přiřazeno schéma databáze dbo. Toto výchozí schéma se používá pro rozlišování názvů objektů, na které se odkazuje bez použití jejich úplného názvu. Umístění, které se v SQL Serveru 2000 kontroluje nejdříve, je schéma vlastněné volajícím uživatelem a dále schéma, jež vlastní dbo. V SQL Serveru 2005 má každý uživatel výchozí schéma, takže když SQL Server analyzuje názvy objektů, kontroluje výchozí schéma.

Lze tedy přenést vlastnictví mnoha zabezpečených objektů a schéma může obsahovat objekty, jež vlastní jiný uživatel databáze než ten, který vlastní schéma. Toto je důležité poznamenat, protože pokud manipulujete se starými systémovými pohledy, jako například *sysobjects*, výsledky vrácené v dotazech se nezobrazí správně. Namísto toho musíte použít katalogové pohledy, jako je *sys.objects*.

Příkaz DDL *CREATE SCHEMA* se používá k vytvoření nového schématu. Lze jej také použít pro vytvoření tabulek a pohledů v rámci nového schématu a pro nastavení oprávnění *GRANT*, *DENY* nebo *REVOKE* u těchto objektů, jak je vidět v následujícím příkladu:

```
CREATE SCHEMA Sales
AUTHORIZATION Rob
CREATE TABLE Leads (id int, name varchar(max),phone nvarchar(20))
GRANT SELECT ON Sales.Leads TO Tammie
DENY SELECT ON Sales.Leads TO Vince;
GO
```

Tento příklad vytvoří schéma Sales a přiřadí jeho vlastnictví Robovi. V rámci schématu Sales se vytvoří tabulka Leads, ke které se udělí Tammie přístup pro čtení, zatímco Vince má přístup pro čtení zakázán.

Aby měla Tammie přístup k tabulce v SQL 2000, musí jako název schématu použít *Rob*, jak vidíte zde:

```
SELECT * FROM Production.SalesReps.Rob.Leads12
```

V SQL Serveru 2005 může namísto toho použít schéma *Sales*:

```
SELECT * FROM Production.SalesReps.Sales.Leads3
```

Pokud by Rob někdy zaměstnání opustil, musí člen role sysadmin pro přenos vlastnictví schématu Sales k Tammie učinit pouze následující:

```
ALTER AUTHORIZATION ON SCHEMA::Sales TO Tammie;
GO
```

Oddělení uživatelů databáze od schémat poskytuje vývojářům a správcům mnoho výhod, včetně následujících:

- Jedno schéma může prostřednictvím členství v rolích nebo skupin Windows vlastnit více uživatelů. Tím se daná funkčnost rozšiřuje, přičemž role a skupiny mohou vlastnit objekty.
- Odebírání uživatelů databáze je velmi zjednodušeno.
- Odebrání uživatele databáze nevyžaduje přejmenování objektů obsažených ve schématu daného uživatele. Proto nemusíte revidovat a testovat aplikace odkazující explicitně na zabezpečené objekty obsažené ve schématu poté, co odeberete uživatele, který objekty vytvořil.
- Jedno výchozí schéma může pro jednotný překlad názvu sdílet více uživatelů.
- Sdílená výchozí schémata umožňují vývojářům ukládat sdílené objekty ve schématu, které bylo vytvořeno specificky pro určitou aplikaci, namísto schématu dbo.
- Oprávnění ke schématům a zabezpečeným objektům obsaženým ve schématu lze spravovat na vyšší úrovni podrobnosti, než tomu bylo u předchozích verzí.
- Úplné názvy objektů mají čtyři části: server.databáze.schéma.objekt.

Kontext provedení

Spravování oprávnění a jejich udělení uživatelům, kteří nejsou členy role sysadmin, byl vždy nesnadný úkol zejména v případě uživatelů vlastnících uložené procedury, které používají tabulky a další objekty, jež uživatelé nevlastní. V typickém případě si představme tři uživatele: Uživatel1, Uživatel2 a Uživatel 3. Uživatel3 vlastní tabulku Customer-Information. Uživatel2 vytvořil uloženou proceduru, která z této tabulky získává pouze jméno zákazníka. Uživatel1 potřebuje provést uloženou proceduru Uživatele2. V před-

chozích verzích SQL Serveru potřebuje Uživatel1 k uložené proceduře Uživatele2 nejen oprávnění *EXECUTE*, ale potřebuje také přístup k související tabulce, kterou vlastní Uživatel3. Tento požadavek znásobte počtem zaměstnanců v organizaci a zjistíte rozsah potíží spojených se správou oprávnění. V SQL Serveru 2005 může Uživatel2 změnit kontext provedení uložené procedury na některý z následujících:

- ***EXECUTE AS CALLER*** Provede se v kontextu volajícího. Toto je stejné výchozí chování jako v předchozích verzích SQL Serveru.
- ***EXECUTE AS SELF*** Provede se v kontextu uživatele, který jako poslední uloženou proceduru upravil.
- ***EXECUTE AS* <uživatel databáze>** Provede se v kontextu specifikovaného uživatele lokální databáze. Aby toto fungovalo, musí mít uživatel, který uloženou proceduru vytváří nebo upravuje, pro tohoto uživatele přidělené oprávnění *IMPERSONATE*.
- ***EXECUTE AS OWNER*** Provede se v kontextu uživatele, který uloženou proceduru vlastní. Pokud se vlastník změní poté, co je objekt vytvořen, kontext provedení se automaticky mapuje na nového vlastníka.

Vraťme se nyní k původnímu problému v našem příkladu (výpis 7.1).

Výpis 7.1: Příklad kontextu provedení

```
--Vytvořit přihlašovací jména
create login user1 with password='A*ahfn2@^(K'
go
create login user2 with password='*HABa7s7aas'
go
create login user3 with password='zxd837&^gqF'
go
create database ExampleDB
go
use ExampleDB
go

--User3 bude vlastnit tabulku
create user User3 with default_schema=User3
go
create schema User3 authorization User3
go
--User2 bude mít právo SELECT a vytvoří proceduru pro přístup k datům
create user User2 with default_schema=User2
go
create schema User2 authorization User2
go

--User1 bude mít právo spustit proceduru
create user User1 with default_schema=User1
go
create schema User1 authorization User1
go

grant create table to User3
```

```
go
grant create table to User2
go
execute as login='User3'
go
create table User3.CustomerInformation
(CustomerName nvarchar(50))
go
insert into CustomerInformation values ('Bryan''s Bowling Alley')
insert into CustomerInformation values ('Tammie''s Tavern')
insert into CustomerInformation values ('Frank''s Fresh Produce')
go
grant select on CustomerInformation to User2
go
revert
go
execute as login='User2'
--vytvořit uloženou proceduru, která vrací záznamy z naší tabulky
create proc viewCustomerNames
AS
BEGIN
select * from User3.CustomerInformation
END
go
grant execute on ViewCustomerName to User1
go
revert
go
```

Když jsou nyní vytvořeni všichni uživatelé a objekty, přihlásíme se jako User1 a pokusíme se o přímý přístup k tabulce a uložené proceduře.

```
execute as login='User1'
--User1 nemůže k tabulce přistupovat přímo
select * from User3.CustomerInformation
--User1 může spustit uloženou proceduru, ale nemá právo na zdrojovou tabulku
exec User2.ViewCustomerNames
go
revert
go
```

V tomto bodě bychom v předchozích verzích SQL Serveru museli poskytnout uživateli User1 určitý typ přístupu k tabulce uživatele User3. V SQL Serveru 2005 může User2 změnit kontext provedení uložené procedury. V našem příkladě User2 změní kontext provedení na *EXECUTE AS OWNER*. To způsobí, že se uložená procedura provede v kontextu uživatele User2, protože User2 je vlastníkem dané uložené procedury.

```
execute as login='User2'
go
ALTER PROCEDURE ViewCustomerNames
WITH EXECUTE AS OWNER
AS
BEGIN
select * from User3.CustomerInformation
```

```
END
go
revert
go
```

Když nyní uživatel User1 provede uloženou proceduru uživatele User2, uložená procedura se provede v kontextu uživatele User2, a bude tak mít přístup k tabulce uživatele User3 pouze v rámci kontextu uložené procedury. Tímto se uživateli User1 umožní provést uloženou proceduru uživatele User2, aniž by měl explicitní přístup ke zdrojové tabulce.

```
execute as login='User1'
--User1 stále nemá právo přistupovat k tabulce přímo
select * from User3.CustomerInformation
--User1 může vykonat proceduru, jež používá tabulku CustomerInformation
exec User2.ViewCustomerNames
go
revert
go
```

V předchozím příkladu je použita další forma změny kontextu provedení. V SQL Serveru 2005 je možné změnit kontext provedení aktuálního připojení, aniž by bylo nutné připojení uzavřít a znovu otevřít. Aby k tomu mohlo dojít, musí být uživatel buď členem role sysadmin, nebo mít pro použité přihlašovací jméno oprávnění *IMPERSONATE*. Eventuálně lze podobný příkaz (EXECUTE AS USER='user') použít ke změně kontextu pro uživatele databáze. V tomto případě přepnutí kontextu je obor zosobnění omezen na aktuální databázi. Při přepnutí kontextu na uživatele databáze se oprávnění daného uživatele na úrovni serveru nepřejímají. Uživatel navíc musí být členem role sysadmin nebo mít oprávnění *IMPERSONATE* vůči uživateli databáze. Provádění přepínání kontextu je funkční a účinný nástroj snižující počet spravovaných oprávnění. Pro vývojáře a členy role sysadmin představuje toto přepínání kontextu snadný způsob, jak testovat skripty a provádět ladění, aniž by bylo nutné se odhlásit a znovu připojit k SQL Serveru.

Podpora šifrování v SQL Serveru 2005

Data uložená v databázích jsou zajímavá nejen pro nás uživatele, ale i pro mnohé jiné lidi, které ani nemusíme znát. Se vzrůstajícím významem relačních databází u organizací a spotřebitelů jsou dodavatelé databází pod vzrůstajícím tlakem souvisejícím s požadavkem zajištění více bezpečnostních prvků.

Kromě uzamčení přístupu k databázím, jako je SQL Server, mohou správci a vývojáři zajistit další úroveň zabezpečení proti neoprávněnému přístupu k datům, kterou je šifrování. Na vyšší úrovni jsou důležité informace, jako je například vaše rodné číslo, šifrováním přeloženy do binární podoby, které rozumí pouze pověřené zúčastněné strany. Data lze šifrovat pro jejich přenos, například když odesíláte své heslo webovému serveru, nebo mohou být uložená v šifrovaném formátu (například v systému souborů nebo v databázi).

Data v SQL Serveru 2005 jsou šifrována pomocí konceptu *šifrovacích klíčů*. Tyto klíče mohou být buď symetrické, nebo asymetrické a k oběma se vztahují výhody a nevýhody. V šifrování pomocí symetrického klíče mají odesílatel i příjemce dat stejný klíč. Odesílatel data šifruje pomocí klíče a šifrovacího algoritmu. Příjemce dat poté tato data dešifruje pomocí stejného šifrovacího algoritmu a klíče. Hlavní výhodou tohoto postupu je

lepší výkon šifrování a dešifrování v porovnání s použitím asymetrických klíčů. Potíže se šifrováním symetrickými klíči hrají roli tehdy, když zvážíme, co se stane, když náš symetrický klíč nějakým způsobem získá někdo jiný. Vzhledem k tomu, že používáme pouze jeden klíč, může data dešifrovat kdokoliv, kdo daný klíč získá.

Šifrování asymetrickými klíči využívá dvou klíčů, veřejného klíče a privátního klíče. Odesílatel šifruje data pomocí veřejného klíče příjemce, který může získat kdokoliv. Zabezpečení se projeví v okamžiku, kdy příjemce data obdrží; příjemce totiž data dešifruje pomocí svého privátního klíče. Veřejný klíč v tomto případě data dešifrovat nemůže. Privátní klíč je tedy kladem při použití asymetrického šifrování.

Dalším konceptem, který je součástí tématu šifrování, jsou certifikáty. Certifikáty jsou v podstatě asymetrické klíče obsahující zvláštní metadata. Tato metadata obsahují informace, jako je lhůta vypršení platnosti a certifikační úřad, který daný certifikát vydal. Certifikáty může vytvořit kdokoliv a za určitých okolností je vhodné se ujistit, že odesílatel nebo příjemce dat je skutečně tím, za koho se vydává. Pro tyto případy slouží právě certifikační úřady. Tyto organizace slouží jako prostředník mezi odesílatelem a příjemcem. Poté, co zaplatíte manipulační poplatek a úřad provede kontrolu totožnosti, vám úřad může poskytnout certifikát, který podepsal. Když nyní podepsaný certifikát použijete k odeslání dat, příjemce může certifikát ověřit u certifikačního úřadu, a protože jak vy, tak odesílatel certifikačnímu úřadu důvěřujete, je prakticky jisté, že zpráva byla skutečně podepsána odesílatelem.

Existuje další typ certifikátu, zvaný *certifikát podepsaný sám sebou*, který může vytvořit kdokoliv. V některých případech je přípustné použít certifikát podepsaný sám sebou. SQL Server 2005 při prvním spuštění automaticky vytvoří certifikát podepsaný sám sebou. Tento certifikát se používá k šifrování spojení během ověřování SQL Serveru.

Šifrování přenášených dat

Pokud klient používá rozhraní API SQL Server Native Access Client, všechny požadavky o připojení odeslané SQL Serveru 2005 jsou šifrovány. To je obrovský posun, protože při použití předchozích verzí se v případě, že ověření identity chce uživatel provést pomocí ověřování SQL Serveru, uživatelské jméno a heslo v podstatě odešlou v čitelné textové podobě. SQL Server 2005 může informace v paketech přihlášení šifrovat pomocí certifikátu podepsaného sebou samým, který vytvořil při prvním spuštění služby.

V připojení nejsou šifrovány pouze pakety přihlášení. Celé připojení může být volitelně šifrováno po dobu trvání tohoto připojení. Požadavek šifrovaného kanálu může být vynucen serverem (aby byla všechna připojení implicitně šifrována) nebo může šifrování požadovat klient vytvářející připojení. Doporučuje se, aby správci používali reálný certifikát namísto certifikátu, který je podepsaný sám sebou, z důvodu potenciálních napadení v průběhu zprostředkování.

Abyste v serveru vynutili šifrování, spusťte nástroj SQL Server Configuration Manager (který naleznete ve složce Programy (Programs) / SQL Server 2005 / Configuration Tools). Tento nástroj se používá ke správě protokolů a služeb SQL Serveru. Z místní nabídky Protocols for MSSQLSERVER vyberte Properties. Otevře se dialog Protocol Properties (obrázek 7.5)

Obrázek 7.5: Dialog Protocol Properties

Dialog má dvě karty. Karta Flags umožňuje vynutit v serveru šifrování. Na kartě Certificate lze vybrat certifikát, který je již nainstalovaný v místním systému, aby jej SQL Server použil k šifrování dat. Pokud certifikát nevyberete a zvolíte vynucení šifrování, SQL Server použije k šifrování dat svůj certifikát, který je podepsaný sám sebou. Nezapomeňte, že certifikát SQL Serveru, který je podepsaný sám sebou, není v klientských připojeních považován za důvěryhodný. Aby mohli klienti používat certifikát SQL Serveru, který je podepsaný sám sebou, musí tuto možnost nastavit v dialogu Properties v uzlu SQL Native Client Configuration ve stejném nástroji Configuration Manager. Tento dialog je zobrazen na obrázku 7.6.

Obrázek 7.6: Dialog Protocol Properties pro SQL Native Client Configuration

Tento dialog nabízí dvě možnosti. Možnost Force Protocol Encryption požaduje na klientech, aby vždy vytvářeli šifrovaná připojení. Druhou možnost je třeba zvolit v případě, že chce klient vytvářet šifrovaná připojení k SQL Serveru a využít přitom certifikát SQL Serveru, který je podepsaný sám sebou.

Šifrování statických dat

Rodná čísla, čísla kreditních karet, platové informace – rozsah citlivých informací se stále rozšiřuje. Přístup k těmto informacím uloženým v databázi byl tradičně zabezpečen pomocí oprávnění. Toto zabezpečení se stále používá, ovšem nyní je v SQL Serveru 2005 nativně dostupná další vrstva ochrany. Citlivá data lze šifrovat pomocí symetrických klíčů, asymetrických klíčů nebo certifikátů.

Dříve než přistoupíme k šifrování dat ve sloupci, je důležité definovat stavební kameny šifrování podporovaného v SQL Serveru. Při prvním spuštění SQL Serveru se vytvoří zvláštní symetrický klíč pro server, zvaný Service Master Key (SMK). Tento klíč se používá k šifrování všech klíčů databáze (DMK, Database Master Key) a všech tajných údajů na úrovni serveru, jako jsou tajné údaje o pověření nebo přihlašovací hesla k propojenému serveru. Jedná se o 128bitový klíč 3DES. Algoritmus 3DES se používá díky jeho dostupnosti na všech platformách Windows podporovaných SQL Serverem 2005.

POZNÁMKA

Dostupnost šifrovacího algoritmu závisí na poskytovateli kryptografických služeb operačního systému, na němž běží SQL Server 2005. Například Windows XP SP2 podporuje DES, 3DES, RC2, RC4 a RSA, zatímco Windows Server 2003 podporuje kromě těchto algoritmů také AES128, AES192 a AES256.

Klíč SMK je šifrován pomocí zabezpečovacího API Windows, DPAPI a pověření účtu služby SQL Server. Vzhledem k tomu, že se klíčem SMK šifrují všechny klíče DMK a další tajné údaje na úrovni serveru, je tento klíč velmi důležitý a měl by být pravidelně zálohován. Klíč SMK zálohujete a obnovíte pomocí příkazu *BACKUP SERVICE MASTER KEY* a *RESTORE SERVICE MASTER KEY*. V případě ohrožení klíče SMK či pokud chcete změnit klíč SMK z důvodu zvýšení zabezpečení, lze provést jeho vygenerování příkazem DDL *ALTER SERVICE MASTER KEY REGENERATE*.

Pokud jde o šifrování dat, SQL Server klíčem SMK dešifruje klíč DMK, aby klíč DMK mohl pro klienta dešifrovat požadovaná data. Pro každou databázi existuje pouze jeden klíč DMK a žádný z těchto klíčů se nevytváří implicitně, protože klíč DMK se používá pouze k šifrování dat.

K vytvoření klíče DMK slouží nový příkaz DDL, který vidíte zde:

```
USE <<databáze>>
CREATE MASTER KEY
ENCRYPTION BY PASSWORD='<<sem vložte své oblíbené heslo>>'
```

Vytvořené klíče DMK jsou šifrovány klíčem SMK (takže SQL Server může pro klienta data dešifrovat) a heslem. Šifrování SMK lze odstranit, ale heslo klíče DMK by muselo být uživatelem specifikováno při každém přístupu k tomuto klíči.

Poté, co jsme představili klíčové komponenty šifrování, uvedeme příklad. Tento příklad používá tabulku s názvem SalaryInfo, která obsahuje jméno, oddělení a plat zaměstnanců v naší organizaci. Uživatel HR_User potřebuje v této tabulce data prohlížet a vkládat.

Při vytváření tohoto příkladu je nutné provádět mnoho různých kroků, a je tedy vhodné spouštět jej příkaz po příkazu. (Viz výpis 7.2.) V příkladu je uvedeno mnoho komentářů, které přesně vysvětlují, co každý příkaz T-SQL provede a proč.

Výpis 7.2: Encryption_HR.sql

```
CREATE LOGIN HR_Login WITH PASSWORD='NějakéSložitéHeslo'
GO
CREATE DATABASE ExampleDB
GO
use ExampleDB
GO
CREATE USER HR_User FOR LOGIN HR_Login
GO

--Vytvořit klíč DMK pro databázi ExampleDB
CREATE MASTER KEY ENCRYPTION BY PASSWORD='JinéSložitéHeslo'
GO

--Vytvořit tabulku, která obsahuje citlivé informace
--Všimněte si, že používáme varbinary pro informaci o platu,
--protože zašifrovaná data jsou binární
CREATE TABLE SalaryInfo
(employee nvarchar(50),
department nvarchar(50),
salary varbinary(60))
GO

--Povolit k této tabulce přístup uživateli HR_User pro přidávání dat
GRANT SELECT,INSERT TO HR_User
GO

--Vytvořit symetrický klíč
--Zašifrovat klíč heslem
--Umožnit přístup ke klíči uživateli HR_User
CREATE SYMMETRIC KEY HR_User_Key
AUTHORIZATION HR_User
WITH ALGORITHM=TRIPLE_DES
ENCRYPTION BY PASSWORD='DalšíSložitéHeslo'
GO

--Použít přihlašovací jméno HR_User a zašifrovat nějaká data
EXECUTE AS LOGIN='HR_Login'
GO

--Nejdříve musíme otevřít klíč, který bude použit pro šifrování dat
--Všimněte si, že k tomu používáme heslo klíče
OPEN SYMMETRIC KEY HR_User_Key DECRYPTION BY PASSWORD='DalšíSložitéHeslo'
GO
```

```
--Tento systémový pohled zobrazuje otevřené klíče, které je možné použít pro
šifrování dat
select * from sys.openkeys

--Vložit citlivá data do tabulky
--funkce encryptByKey potřebuje jedinečný identifikátor (GUID) klíče a data
pro zašifrování
--jelikož zapamatovat si GUID není jednoduché, funkce Key_GUID
--tuto informaci pro nás získá
INSERT INTO SalaryInfo VALUES
('Bryan','Sales',encryptByKey(Key_GUID('HR_User_Key'),'125000'))
INSERT INTO SalaryInfo VALUES
('Tammie','Sales',encryptByKey(Key_GUID('HR_User_Key'),'122000'))
INSERT INTO SalaryInfo VALUES
('Frank','Development',encryptByKey(Key_GUID('HR_User_Key'),'97500'))
INSERT INTO SalaryInfo VALUES
('Fran','Marketing',encryptByKey(Key_GUID('HR_User_Key'),'99500'))

--Po dokončení práce s klíči je velmi vhodné je uzavřít
CLOSE ALL SYMMETRIC KEYS
GO

--Zobrazit surová data v databázi, všimněte si binární informace
select * from SalaryInfo

--Nyní proveďme dešifrování obsahu tabulky
--Použijeme funkci decryptByKey a předáme jí název sloupce
--Nemusíme specifikovat GUID klíče, protože SQL se podívá
--na všechny právě otevřené klíče a použije odpovídající klíč automaticky
OPEN SYMMETRIC KEY HR_User_Key DECRYPTION BY PASSWORD='DalšíSložitéHeslo'
GO

SELECT employee,department,
CONVERT(varchar,decryptByKey(salary))
FROM SalaryInfo
GO

CLOSE ALL SYMMETRIC KEYS
GO

--vrátit se zpátky k práci jako sysadmin
REVERT
GO

--Když šifrujete použitím hesla, musíte jej znát
--a předat jej, kdykoliv něco šifrujete
--Alternativně však můžete vytvořit certifikát a umožnit přístup
--uživateli HR User. Tímto způsobem uživatel nemusí předávat heslo
--a v případě, kdy mu potřebujete zabránit přístupu k šifrovaným datům,
--jednoduše odejmete tomuto uživateli certifikát
CREATE CERTIFICATE HRCert1
AUTHORIZATION HR_User
WITH SUBJECT='Certifikát pro použití v personálním oddělení'
```

```
--Otevřít klíč pro umožnění jeho změny
OPEN SYMMETRIC KEY HR_User_Key DECRYPTION BY PASSWORD='DalšíSložitéHeslo'
GO

--Nemůžeme odstranit heslo, protože nesmíme nechat klíč nechráněný,
--a tak nejprve přidáme certifikát
ALTER SYMMETRIC KEY HR_User_Key
ADD ENCRYPTION BY CERTIFICATE HRCert1
GO

--Nyní můžeme z klíče odstranit heslo
ALTER SYMMETRIC KEY HR_User_Key
DROP ENCRYPTION BY PASSWORD= 'DalšíSložitéHeslo'
GO
CLOSE ALL SYMMETRIC KEYS
GO

--Nyní změníme kontext na HR_Login kvůli testu provedených změn
EXECUTE AS LOGIN='HR_Login'
GO

--Všimněte si, že otevíráme klíč bez hesla
--To je možné, protože jsme vytvořili certifikát a dali jsme k němu oprávnění
--explicitně uživateli HR_User
OPEN SYMMETRIC KEY HR_User_Key DECRYPTION BY CERTIFICATE HRCert1
GO

SELECT employee,department,
CONVERT(varchar,decryptByKey(salary))
FROM SalaryInfo
GO
```

Předchozí příklad popisuje základy šifrování a dešifrování v SQL Serveru 2005. Množství dat v tomto příkladu je z hlediska šifrování a dešifrování nepatrné a zvládl jej i tři roky starý počítač. Výkon související s šifrováním závisí kromě výkonnosti vašeho počítače také na dvou dalších faktorech: na velikosti šifrovaných dat a algoritmu, kterým data šifrujete. Šifruje-li se větší soubor algoritmem RSA2048, bude to trvat déle než šifrování rodného čísla. Není snadné vytvořit graf uvádějící do souvislosti velikost dat a čas pro šifrování, protože to závisí na mnoha faktorech. Nejvhodnější je vytvořit testovací prostředí napodobující vaše pracovní prostředí, kde provedete některé testy výkonu: šifrování a dešifrování různě velkých dat použitím různých algoritmů nebo alespoň algoritmem, který budete používat.

Zajímavým tématem souvisejícím se šifrovanými sloupci je i indexování a vyhledávání. Samotný SQL Server s nimi pracuje jako s binárními sloupci, takže neexistuje účinný způsob, jak v nich vytvořit index, protože nelze předvídat náhodný proud bajtů. V tomto případě je nejlepší pro indexování vytvořit nebo použít jiný nešifrovaný sloupec. Problém však spočívá v tom, že tím můžete bezděčně poskytnout informace o šifrovaných datech. Řekněme, že chceme indexovat mzdy a vytvořit sloupec s názvem rozsah. Kterýkoliv uživatel s oprávněním *SELECT* pro přístup k tabulce si může vyvodit, jaký plat zaměstnanec

pobírá. Pokud potřebujete šifrovaná data indexovat nebo v nich vyhledávat, je nutné být při práci s nešifrovanými sloupci kreativní.

Práce se šifrováním není nikterak jednoduchá a podrobný popis tohoto tématu je nad rámec této knihy. Abyste lépe porozuměli šifrování v souvislosti se zabezpečením SQL Serveru, využijte zdroje dostupné online. Laurentiu Cristofor, jeden z vývojářů společnosti Microsoft stojící za šifrováním SQL Serveru 2005, má na svém blogovém serveru rozsáhlé informace týkající se šifrování: *http://blogs.msdn.com/lcris*. V týmu zabezpečení společnosti Microsoft je také Raul Garcia, který má blog s užitečnými informacemi na adrese: *http://blogs.msdn.com/raulga/*.

Ochrana SQL Serveru 2005

Hackeři stále nacházejí nové způsoby při pokusech o nabourání do systémů a aplikací. Vzhledem k hojnosti informací dostupných v rámci databází se tyto typy aplikací staly primárními cíli pro uživatele se zlými úmysly. V této části se zaměříme na způsoby, jakými SQL Server řeší některé otázky související se zabezpečením. Dozvíte se také, jak mohou hackeři zneužít váš systém.

Zmenšení napadnutelné plochy

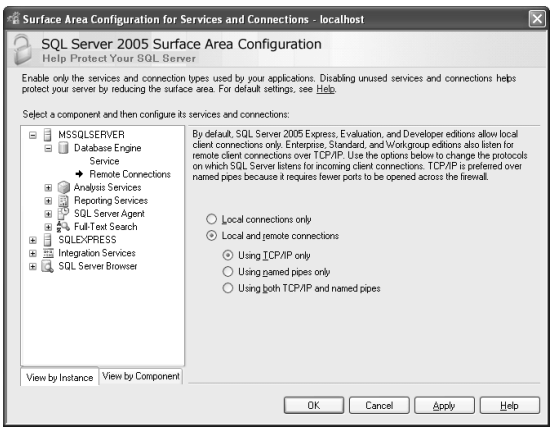
Čím více máte v domě dveří, tím více možností má nezvaný host k prolomení zabezpečení. I když jsou tyto dveře vyrobené z pevné oceli a mají krásné zámky, stále je lze považovat za napadnutelnou plochu. Jednou z možností, jak se vyhnout této zranitelnosti, je neumístit dveře tam, kde nejsou potřeba. Ekvivalent počítačové aplikace spočívá v tom, že pokud některou funkci nepoužíváte, tak ji, je-li to možné, vypněte.

SQL Server 2005 implicitně zajišťuje zmenšení napadnutelné plochy tím, že automaticky vypíná funkce, které jsou volitelné. Mezi tyto funkce patří služba SQL Server Agent, služba SQL Server Browser a různé funkce serveru, včetně *xp_cmdshell*, použití CLR v rámci SQL Serveru a další. Nástrojem umožňujícím správcům tyto funkce vypnout a zapnout je SQL Server Surface Area Configuration. Tento nástroj je instalován jako součást instalace databázového stroje, takže byste jej měli mít dostupný bez ohledu na to, jakou edici nebo které komponenty instalujete.

Když nástroj spustíte, máte na výběr dvě možnosti: Surface Area Configuration for Services and Connections a Surface Area Configuration for Features.

V dialogu Services And Connections (obrázek 7.7) je strom komponent SQL, které jsou instalovány v serveru.

Pod uzlem Database Engine jsou dva panely. První panel, Services, umožňuje uživateli spustit, zastavit a změnit automatická spouštění služby. Druhý panel, Remote Connections, je jedním z přepínačů, které je vhodné si zapamatovat. Vzdálená připojení jsou v edicích SQL Serveru Express, Evaluation a Developer implicitně zakázána. Pokud tedy instalujete edici Express a pokusíte se k ní připojit z jiného klientského počítače, připojení se nezdaří. Chcete-li vzdálená připojení povolit, je nutné vybrat typ vzdáleného připojení, který bude povolen.

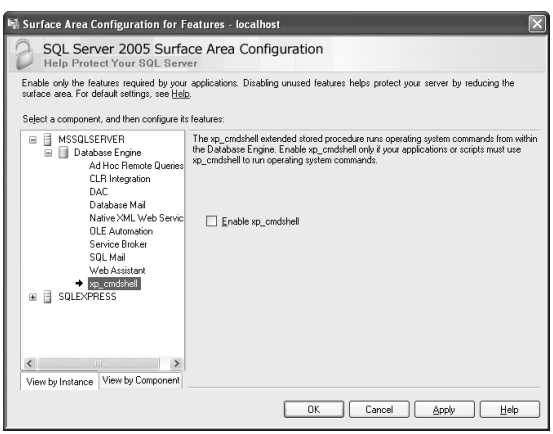


Obrázek 7.7: Panel Remote Connections nástroje Surface Area Configuration

POZNÁMKA

Povolíte-li vzdálená připojení a máte potíže se připojit, zkontrolujte svá nastavení brány firewall. Pravděpodobně bude nutné přidat výjimku pro proces Sqlsvr.exe nebo otevřít specifický port pro SQL Server.

Další důležitou funkcí tohoto nástroje je možnost konfigurovat, které funkce mají být zapnuté nebo vypnuté. Po klepnutí na Surface Area Configuration For Features se otevře dialog, který vidíte na obrázku 7.8.



Obrázek 7.8: Panel Features nástroje Surface Area Configuration

Na obrázku je vidět, že nástroj vám umožňuje povolit nebo zakázat použití rozšířené uložené procedury xp_cmdshell. Další funkce, které lze zapnout nebo vypnout jsou CLR, příležitostné (ad hoc) vzdálené dotazy a služba Service Broker.

Každou z těchto možností lze také programově vypnout a zapnout použitím uložené procedury *sp_configure*. Pro zobrazení těchto možností nejprve použitím *sp_configure* nastavte hodnotu vlastnosti *show advanced options* na 1. Existuje také nový systémový

pohled s názvem *sys.configurations*, který v podstatě zobrazuje stejné informace jako uložená procedura *sp_configure* se zapnutou vlastností *show advanced options*.

Jak hackeři napadají SQL Server

Na začátku této části musíme uvést upozornění. V této části nejsou obsaženy všechny možné způsoby, kterými se hackeři mohou pokusit ohrozit SQL Server. Účelem této části je představit různé metody zneužití a doporučit vám zvážení dostupných prostředků zabezpečení při navrhování databázových aplikací a konfiguraci vašeho systému. Nejprve se zaměříme na témata týkající se způsobu konfigurace SQL Serveru.

Přímé připojení k Internetu

Přímé vystavení operačního systému či aplikace síti Internet bez použití brány firewall je riskantní bez ohledu na to, zda používáte Linux, UNIX, Windows nebo jiný operační systém. Je to jako hrát hru, kdy někdo sedí na kraji bazénu a ostatní se do něj snaží strefit míčem. Když se jim to podaří, člověk na kraji bazénu spadne do vody. Proč byste měli svůj systém vystavovat takovýmto pokusům a čekat, až vás někdo potopí? Microsoft učinil mnohá opatření implicitně chránící operační systémy a aplikace. Když dojde k pokusům o prolomení zabezpečení, rychle se těmto potížím věnují a nabízejí řešení.

Tato opatření však nestačí. S tolika přepínači a režimy zabezpečení pro různé produkty je tedy pravděpodobné, že správce nebo uživatel něco špatně nakonfiguruje, a vystaví tak systémy napadením. Aby se tyto potíže zmírnily, je pro uživatele velmi důležité izolovat tyto systémy od Internetu pomocí bran firewall a použít další metody izolace.

Slabá hesla správců systému

Jedním z nejsnazších způsobů, jak někomu předat klíče k SQL Serveru, je použít pro účet správce systému slabé heslo. V předchozích verzích SQL Serveru bylo možné pro účet správce systému bez větších potíží zadat prázdné heslo. SQL Server 2005 pracuje s mnohem rozšířenějšími zásadami hesel a jejich vynucováním. V této kapitole jsme se tímto tématem zabývali v souvislosti s dodržováním zásad skupin domény Windows při ověřování účtů SQL Serveru. Konfigurace bezpečné délky hesla a uzamykání účtů ve vaší doméně zajistí, že všichni uživatelé SQL Serveru zadávají hesla, jejichž zjištění je obtížnější.

Služba SQL Server Browser

Názvy a čísla portů instance SQL Serveru požadované klientem vrací SQL Server prostřednictvím portu UDP 1434. Před několika lety byl tento výčet klíčem pro vir DoS „SQL Slammer“. Neustálým odesíláním požadavků o výčet portů určitému serveru tento virus způsobil, že server byl příliš vytížen na to, aby zpracovával ostatní požadavky. V SQL Serveru 2005 tuto funkci výčtu obsahuje samostatná služba s názvem SQL Server Browser service. Tato funkce již nepracuje v oblasti procesů SQL Serveru a lze ji vypnout a zapnout, aniž by byl ovlivněn SQL Server. Pokud službu SQL Server Browser nechcete použít, lze se připojit také k jiným instancím vašeho serveru, ale propojovací řetězec musí obsahovat další informace (jako například konkrétní číslo portu v případě připojení TCP). Chcete-li ve své organizaci použít službu Browser, lze riziko dalších napadení snížit blokováním portu UDP 1434 v bráně firewall.

Útok typu „SQL injection"

„SQL injection" je proces, jímž uživatel se zlými úmysly zadá příkazy SQL namísto platných vstupních údajů. Webový server například požaduje uživatelské jméno. Namísto toho, aby uživatel zadal uživatelské jméno, zadá *blah'; DROP TABLE sales;--*. Webový server uživatelovy vstupní údaje bez potíží přijme a předá je své střední vrstvě, kde jsou údaje provedeny v kódu následujícím způsobem:

```
SqlCommand cmd = new SqlCommand("SELECT * FROM sales WHERE name='" +
s_Customername + "'", myConnection)
```

Pro SQL Server tento příkaz vypadá následovně:

```
SELECT * FROM sales WHERE name='blah'; DROP TABLE sales;--'
```

Po provedení tohoto příkazu se smaže tabulka sales. Vidíte, jak může být snadné pro uživatele se zlými úmysly způsobit potíže a vrátit potenciálně citlivé informace pomocí jednoduchých vstupních údajů na webové stránky či aplikace, které uživatelské vstupní údaje přímo využívají. Abyste snížili riziko spojené s tímto problémem, lze uživatelské vstupní údaje přidat jako parametr do příkazu *SqlCommand*, jak vidíte níže:

```
Using (SqlCommand cmd = new SqlCommand("SELECT * FROM sales WHERE name=
@s_CustomerName", myConnection));
{
cmd.Parameters.Add("@s_CustomerName", s_CustomerName);
...
```

Cokoli uživatel zadá, bude SQL Server považovat pouze za hodnotu parametru v klauzuli *where*.

Inteligentní pozorování

S příchodem výkonných vyhledávacích strojů, jako je Google a MSN Search, je hledání údajů na webu poměrně snadné. Webový robot těchto vyhledávacích strojů shromažďuje klíčová slova a umísťuje je do vlastní interní databáze. Tato klíčová slova se používají v rámci vlastních vyhledávacích algoritmů, takže pokud chcete vyhledat určitý údaj, vyhledávací stroj snadno vrátí seznam odpovídajících možností. Robot vyhledává a ukládá nejen informace například o webových serverech pizzerií, ale získává také různé typy chybových zpráv vrácených z webových serverů. Tyto údaje jsou pro hackera velmi cenné. Zadá-li hacker například *invalid password access denied* do vyhledávacího řetězce MSN, získá seznam různých témat, která obecně nejsou tak zajímavá. Jedna položka však zobrazí tento řetězec: *Warning: mysql_pconnect(): Access denied for user 'root'@'localhost' (usingpassword: YES) in /home/vhosts/«z právních důvodů odstraněno»/docs/citeheader.inc.php on line 2*. Hacker ví, že tento server pracuje s MySQL a PHP, a nyní zná část struktury složek webového serveru */home/vhosts/«z právních důvodů odstraněno»/docs*. Nyní se hacker může prostřednictvím vlastního internetového prohlížeče pokusit dotázat na samotnou cestu adresáře, aby zjistil, zda se mu podaří odkrýt další užitečné informace – například soubor skriptu. Nalezne-li v této složce soubor skriptu a vývojář pevně zakódoval pověření k přihlášení do databáze, nachází se velmi blízko bodu, kdy může databázi ohrozit.

Z toho vyplývá, že vyhledávací stroje jsou velmi dobrým nástrojem hackerů. Nikdy do souborů skriptů pevně nezakódujte hesla a vždy zajistěte přesměrování webových stránek v případě chyb v rámci vaší webové aplikace. Vždy věnujte zvýšenou pozornost webovým aplikacím přijímajícím vstupní údaje. Zajistěte, aby tyto typy dat byly chráněny před útoky „SQL injection".

Shrnutí

Zabezpečení je jedním z nejvýznamnějších faktorů jakéhokoliv projektu a čas věnovaný studiu možností zajištění SQL Serveru a vašich aplikací se z dlouhodobého hlediska vyplatí. Microsoft učinil opatření ke zvýšení výchozího zabezpečení SQL Serveru 2005 vypnutím funkcí, jako je *xp_cmdshell*, služby SQL Server Agent a mnoha dalších volitelných služeb v rámci tohoto produktu. Rozšířil také sadu funkcí zabezpečení mimo jiné také přidáním možnosti snadno šifrovat data při jejich přenosu a při ukládání v databázi, využitím podrobnějších oprávnění a povolením snadného přepínání kontextu.

V této kapitole jsme představili některé ze stěžejních pojmů zabezpečení v SQL Serveru 2005. Mnoho pramenů online (jako například SQL Server Books Online) s těmito a dalšími tématy pracuje podrobněji. Při práci na vaší další aplikaci nikdy nezapomeňte na možnost pokusů o neoprávněné získání přístupu k vašim datům či aplikacím. V dnešní době pouhý zámek na dveřích chránící před lidmi se zlými úmysly nestačí. Každý profesionál a vývojář v oblasti IT musí vždy brát v potaz význam zabezpečení.