VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

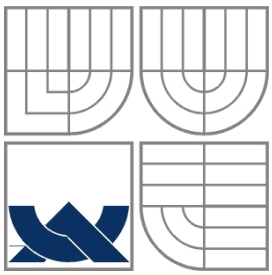
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BEZPEČNÁ AUTENTIZACE VE WEBOVÝCH APLIKACÍCH ASP.NET

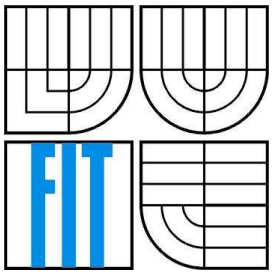
DIPLOMOVÁ PRÁCE
MASTER'S thesis

AUTOR PRÁCE Josef Lukaštík
AUTHOR

BRNO 2007



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BEZPEČNÁ AUTENTIZACE VE WEBOVÝCH APLIKACÍCH ASP.NET

SAFETY AUTHENTIZATION IN WEB APPLICATION ASP.NET

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Josef Lukaštík

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. Vladimír Bartík, Ph.D.

BRNO 2007

Abstrakt

Tato diplomová práce čerpá z oblasti bezpečnosti a to především ve směru webových aplikací (ASP.NET). Zabývá se bezpečným přístupem k aplikacím, které se nacházejí ať už v Ethernetu, nebo Internetu. Hlavní důraz je kladen na přístup k databázi MSSQL a tzv. formulářové ověřování. Toto ověřovaní jsem implementoval i do informačního systému společnosti Daikin Device Czech Republic s.r.o.. Systém je navržen jako webová aplikace v ASP.NET s přístupem k databázi MYSQL. Celý systém je složen postupně z modulů a jedním z modulů je i přístup do databáze MYSQL, kde se používá formulářové ověřování. Zde se snažím o co nejbezpečnější přístup k datům, které jsou uloženy na vzdáleném serveru MYSQL. Výsledky a funkční aplikace běží ve společnosti a jsou podrobeny testovaní funkčnosti.

Klíčová slova

ASP.NET, MSSQL, autorizace, ověřování, SSL, IPSec, RPC

Abstract

The Diploma Thesis deals with security topic and it is focused mostly on its web application (ASP.NET). The thesis concerns itself on the safety access to the applications, that are whether in Ethernet or Internet. Main stress is put on the access to the MYSQL data base and so-called form verification. I implemented this kind of verification into the information system in Daikin Device Czech Republic s.r.o. as well The system is designed as the web application in ASP.NET with access to MYSQL data base.

The whole system consists of modulus and one of them is the access to MYSQL data base where the form verification is used. When speaking about the access to MYSQL data base I endeavour to make the access to the data, that are saved on the distant server MYSQL, as safe as possible. Results and function application is running in the company and they are under the running test.

Keywords

ASP.NET, MYSQL, authorisation, verification, SSL, IPSec, RPC

Citace

Lukaštík Josef: Bezpečná autentizace ve webových aplikacích ASP.NET. Brno, 2007, diplomová práce, FIT VUT v Brně.

Bezpečná autentizace ve webových aplikacích ASP.NET

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením Ing. Vladimíra Bartíka, Ph. D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Josef Lukaštík
28.5.2007

Poděkování

Rád bych poděkoval na tomto místě vedoucímu svého semestrálního projektu Ing. Vladimíru Bartíkovi, Ph. D. za jeho cenné rady a vstřícný přistup

Obsah

1 Úvod

Před několika lety byly počítače na úplném začátku a slova jako bezpečnost nebo kryptografie byly teprve na počátku. Pokud se budeme dívat čím dál víc do minulosti, zjistíme, jak se začala bezpečnost postupně vyvíjet a zlepšovat. V dřívější době bylo zcela běžné, když kterýkoliv člověk umístil na svou doménu internetové stránky, které byly přístupné odkudkoliv z celého světa. S tím se samozřejmě potkáváme i dnes, ale již to nejsou pouze statické stránky. Jedná se o celé aplikace, internetové obchody, informační systémy, internetové bankovnictví a daleko složitější projekty. Již jsem zmínil např. internetové bankovnictví, které se v dnešní době velmi dobře rozvijí a nenajdeme snad žádnou banku, která by tuto službu svým klientům nenabízela. Jedná se jistě o velmi citlivé a osobní informace. Pokud by se snad stal někdy únik informací, mělo by to samozřejmě katastrofální důsledky. Z toho důvodu musela přijít i do světa počítačů, vzdálené komunikace, intranetu, bezdrátové komunikace bezpečnost. Ať se již jedná o bezpečnost intranetu, extranetu internetu, webových služeb, přístupu k datům nebo vzdálené správě, všude tady je potřeba bezpečné uchování informací. Předkládaná práce se zabývá jednou z mnoho možností jak zabezpečit webové aplikace v prostředí ASP.NET. Jistě se spousta lidí a i já setkali v životě se situací, kde se musel vyplňovat vstupní formulář pro přihlášení do aplikace. Zadání přihlašovacího jména a poté i hesla sebou samozřejmě nese i rizika. Ověřovaní uživatelských jmen a hesel v ASP.NET aplikaci, která je propojena s databázi běžící na SQL serveru je taky jednou z možností zajištění bezpečnosti. Formulářové ověření může sloužit k přihlášení do firemního intranetu a poté i k další práci s tímto intranetem.

2 Model zabezpečení aplikace ASP.NET

Tato kapitola přibližuje zabezpečení webových aplikací za použiti platformy .NET Framework a především ASP.NET.

2.1 Webové aplikace založené na technologii .NET

2.1.1 Logické vrstvy

Pokud se podíváme na architekturu aplikace, tak uvidíme tyto 3 vrstvy spolupracující mezi sebou

- uživatelské služby (User Services)
- aplikační služby (Business Services)
- datové služby (Data Services)

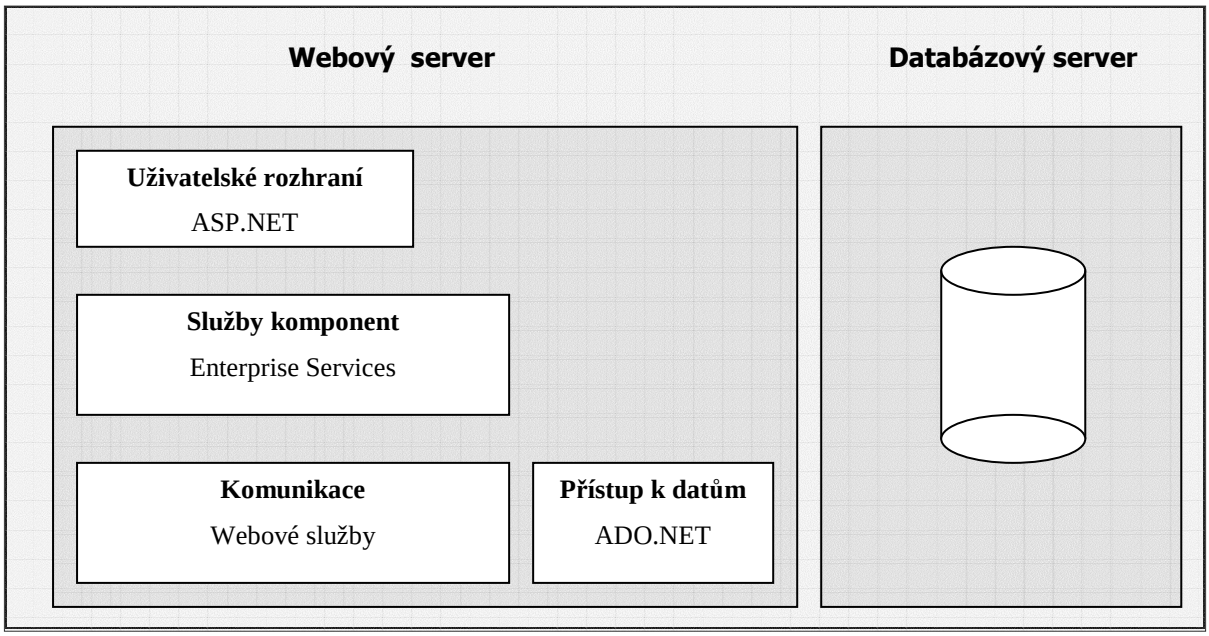
Výsledkem takového rozčlenění je daleko ovladatelnější aplikace.

- **Uživatelské služby.** Tyto služby nám zajišťují komunikaci klienta se systémem. V této službě se setkáváme i s ověřováním a autorizací.

- **Aplikační služby.** Zajištění hlavní funkce systému a zapouzdření aplikační logiky

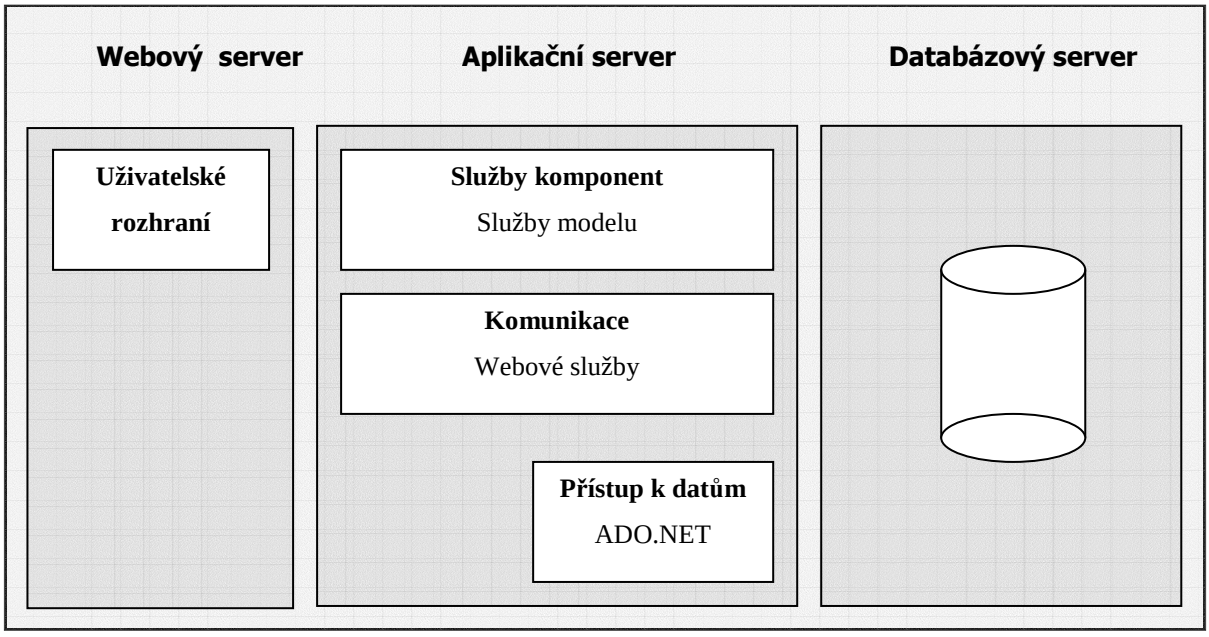
- **Datové služby.** Poskytují přístup k datům, zapouzdřují specifické datové formáty a specifická pravidla pro přístup k nim.

Tyto tři vrstvy můžou být samozřejmě umístěny na jednom počítači, ale zároveň můžou být na prakticky neomezeném počtu počítačů.Umístění aplikační a datové komponenty na webový server může sloužit jako aplikační server. Tímto by se minimalizovaly síťové přenosy a výrazně se zlepšil výkon. Celý webový server jako aplikační server je znázorněn na obrázku 2.1



Obrázek 2.1: Webový server jako aplikační server

Opakem může být situace znázorněná na obrázku 2.2 Zde se jedná o aplikační vrstvu, která je vzdálena. Používá se především v internetových aplikacích.Webová vrstva je zde samostatná a izolována od uživatelů. Mezi aplikační vrstvou a webovou vrstvou je brána, která slouží k odfiltrování paketů.



Obrázek 2.2: Vzdálená aplikační vrstva

2.2 Implementační technologie

Na implementaci jednotlivých služeb můžeme použit následující technologie.

- **ASP.NET.** Tato technologie se používá především k implementaci uživatelských služeb. Dá se jí využít při tvorbě webových stránek.
- **Služby modelu Enterprise Services.**Nabízí aplikacím služby na úrovni infrastruktury. A to především distribuované transakce a služby pro správu prostředků.
- **Webové služby.**Tyto služby umožňují výměnu dat a vzdálené volání aplikační logiky přes bezpečnostní brány firewall.
- **.NET Remoting.**Zajištění vzdálené komunikace modelu .NET.
- **ADO.NET a Microsoft SQL Server 2000.**Technologie ADO.NET poskytuje služby pro přístup k datům. Je spoustu možností, jak spolupracovat s webovými aplikacemi.Služba SQL Server nabízí integrované zabezpečení využívajícího ověřovacího mechanismu operačního systému (Kerberos nebo NTLM). Ověření je zajištěno přihlášením, ale rovněž odstupňovanými oprávněními, jež lze pro jednotlivé objekty nastavit.
- **IPSec (Internet Protocol Security).**Zajišťuje šifrování mezi dvěma koncovými body už na úrovni transakce, jehož i služby ověřování.
- **SSL (Secure Sockets Layer).**Protokol SSL zajišťuje tvorbu zabezpečení komunikačního kanálu mezi dvěma koncovými body.Data posílána takovým kanálem jsou zašifrována.

2.3 Architektura zabezpečení

K jedné z nejdůležitějších akcí patří ověřování a následovaná autorizace.

2.3.1 Ověřování

Máme tyto způsoby ověřování

- Režimy ověřování ASP.NET
- Ověřování službami modelu Enterprise Services
- Ověřování v aplikaci SQL Server

2.3.1.1 Režimy ověřování v prostředí ASP.NET

Ověřování v prostředí ASP.NET zahrnuje režimy Windows, Formuláře, Passport, Žádné

- **Integrované ověřování systému Windows.**V tomto režimu se předpokládá, že služba IIS ověří uživatele a vytvoří přístupový token systému Windows zastupující identitu. Služba IIS nám poskytuje tyto mechanismy ověřování

    - **Základní ověřování.**Identita se prokáže po zadání uživatelského jména a hesla.Jedná se o navržený internetový standard. Můžeme se setkat s ním např. i v Microsoft Internet Explorer.Webový dialog může zobrazit dialog, kde uživatel zadá jméno a heslo k účtu systému Windows.Heslo se před odesláním do systému zakóduje algoritmem Base64.Toto ověření by se mělo používat pouze se zabezpečeným kanálem SSL.
    - **Ověřování algoritmem Digest.**Toto ověřování je použito ve službě IIS 5.0. V podstatě nabízí stejné zabezpečení jako základní ověření.Tento algoritmus přenáší pověření v síti pomocí algoritmu hash MD5 (miniatura zprávy nebo hešový kód). Ověření je tedy bezpečnější , i když vyžaduje přítomnost aplikace (Internet Explorer 5.0 nebo vyšší) na straně klienta a specifickou konfiguraci na straně serveru.
    - **Integrované ověření systému Windows.**Integrované ověřování systému Windows (Kerberos nebo NTLM – v závislosti na konfiguraci klienta a serveru) využívá kryptografickou výměnu dat mezi webovým prohlížečem Internet Explorer a serverem, která zajišťuje potvrzení identity uživatele.S tímto způsobem se můžeme setkat pouze v aplikaci Internet Explorer. Z těchto důvodů je používám pouze v intranetových aplikací.Na webovém serveru lze použít pouze když je zakázán anonymní přístup.
    - **Ověřování certifikátu.**K identifikaci uživatelů je použit klientský certifikát.Klientský certifikát je předáván webovém serveru webovým prohlížečem (nebo klientskou aplikací).Webový server zjistí z přijatého certifikátu identitu uživatele.Tento přístup spoléhá na to, že je klientský certifikát instalován na počítači uživatele.Proto je také nejčastěji používán jako intranetové nebo extranetové řešení se známou uživatelským základnou, kterou lze jednoduše ovládat. Služba IIS může po přijetí klientského certifikátu mapovat certifikát na účet v systému Windows.
    - **Anonymní ověřování.**Umožňuje uživatelům přístup do veřejných oblastí webového serveru nebo serveru FTP, aniž by tito byli vyzváni k zadáni uživatelského jména a hesla.

- **Ověřování pomocí služby Microsoft Passport.**Prostředí ASP.NET využívá v tomto případě ověření Microsoft Passport.Prostředí ASP.NET nabízí pohodlné zapouzdření funkcí poskytovaného sadou Microsoft Passport software Development Kit (SDK), která musí být nainstalována na Webovém serveru.

- **Ověřování pomocí formulářů.**Tento postup využívá přesměrování na straně klienta – neověření klienti jsou automaticky přesměrováni na specifický formulář HTML umožňující zadat pověření uživatele (obvykle ve formě jména a hesla).Tato pověření jsou později ověřována. Je-li pověření uživatele přijato, je generován lístek uživatele (authentication ticket), jenž je odeslán zpět na klientský PC. Lístek obsahuje informace o identitě uživatele.Může taky obsahovat role, jejichž vlastnictví je uživateli uděleno po dobu zahájené uživatelské relace.
- **Žádné ověřování.**Toto nastavení je k neověřování uživatele, prostě nic.

2.3.1.2 Ověřování rozsáhlé služby sítě (Enterprise Services)

Ověřování služeb rozsáhlé sítě je zajištěno nadřazenou transportní infrastrukturou RPC (Remote Procedure Call), která využívá SSPI (Security Service Provider Interface) operačního systému. Prověření má na starost systém Kerberos nebo NTLM.

Komponenta může být uložena buď v knihovně a nebo v serverové aplikaci.Komponenta rozlišuje tyto možné volání.

- **Výchozí (Default)**:Implicitní úroveň zabezpečení
- **Žádné (None):**Klienti nejsou ověřování
- **Připojení (Connect):**K ověření dochází pouze po navázání spojení
- **Volání (Call):**Klient je ověřen na každém volání vzdálené procedury
- **Paket (Packet):**Ověřována a potvrzována jsou veškerá přijatá data
- **Integrita paketu (Packet Integrity):**Je ověřováno a potvrzeno, že by při přenosu nebyly žádná data změněna.
- **Utajení paketu (Packet Privacy):**Pakety jsou ověřovány a šifrovány včetně dat identity uživatele a podpisu.

2.3.1.3 Ověřování v aplikaci SQL Server

Pokud budeme mít toto ověřování, tak se buď můžeme rozhodnout pro integrované ověřování v systému Windows (NTLM nebo Kerberos). Nebo vlastní vestavené schéma ověřování obvykle nazvané ověřování SQL.Zde máme na výběr z dvou možností.

- **Ověřování systémem Windows a službou SQL server.** Jedná se o tzv. režim smíšeného ověřování. Máme na výběr buď přihlášení pomocí Microsoft SQL Server nebo pomocí integrovaného ověřování systému Windows.
- **Pouze integrované ověřování systému Windows**. Uživatel má zde pouze možnost připojení k Microsoft SQL Server pomocí integrovaného zabezpečení systému Windows.

2.3.2 Autorizace

Platforma .NET Framework v systému Windows nabízí tyto možnosti autorizace

- autorizace v prostředí ASP.NET
- autorizace služeb rozsáhlé sítě (Enterprise Services)
- autorizace v aplikaci SQL Server

2.3.2.1 Možnosti autorizace v prostředí ASP.NET

Tato autorizace se využívá především ve webových aplikací typu ASP.NET, webové služby a vzdálené komponenty.

- **Autorizace podle adres URL.** Jde o mechanismus konfigurovaný prostřednictvím konfiguračních souborů počítače a příslušné aplikace.Autorizace je prováděna pomocí IP adres a umožňuje přístup pouze k vybraným souborům a složkám. Lze tedy selektivně odepřít nebo povolit přístup k určitým souborům, či složkám (otevíraných zadáním adresy URL) vyjmenovaných uživatelům.Přístup lze samozřejmě omezit rovněž na základě členství v uživatelských rolích a typem požadavku (GET, POST atd.) Podle této autorizace se musí ověřit existenci identity.Tu lze získat od systému Windows nebo na základě uživatelského lístku.
- **Autorizace souborů.**Tuto službu lze využít pouze v případě, že se používá jen jeden mechanismus ověřování uživatelů v systému Windows a že je v prostředí ASP.NET. Používá se k vybraným souborům na webovém serveru. Oprávnění přístupu lze řídit pomocí seznamů ACL (Access Control List) připojených souborů.
- **Role platformy .NET.** Pokud máme aplikaci, kde se vyskytuje víc uživatelů, tak máme možnost použít role. Jedná se o podobné funkce rolí, jako v systému Windows. Role platformy .NET však nevyžadují přítomnost ověřované identity v systému Windows. Role se používají k řízení přístupu k prostředkům a operacím. Mohou být konfigurovány nejen deklarativně , ale rovněž v programovém kódu.

2.3.2.2 Autorizace v rozlehlé síti (Enterprise Services)

Vše je zde řízeno prostřednictvím členství v rolích služeb rozsáhlé sítě. Tyto role jsou ovšem odlišné od roli .NET, neboť tyto mohou obsahovat skupiny uživatelů systému Windows, nebo přímé uživatelské účty.

2.3.2.3 Autorizace v aplikaci SQL Server

Zde můžeme odstupňovat velmi jemně oprávnění, které lze používat pro jednotlivé databázové objekty. Oprávnění lze založit na členství v roli (SQL server nabízí databázové role, uživatelsky definované role a aplikační role). Máme zde možnost přidělit oprávnění jak jednotlivým uživatelům, tak skupinovým účtům systému Windows.

3 Ověřování a autorizace

Tato kapitola přibližuje návrh strategie ověřování a autorizace na platformy .NET Framework a především ASP.NET.

3.1 Prostředky

Prostředky jsou objekty, které musí aplikace poskytovat klientům. Patří zde

- **Prostředky webového serveru**. Patří zde např. webové stránky, webové služby, statické stránky (HTLM a obrázky)
- **Databázové prostředky**. Osobní data jednotlivých uživatelů nebo sdílené data
- **Síťové prostředky.** Prostředky vzdáleného souborového systému a data z adresových úložišť. (Active Directory)

3.2 Způsoby autorizace

Máme 2 základní možnosti autorizace

- **Autorizace založena na rolích.** Přístup k operacím je zabezpečen na základě členství volajícího klienta v roli. Role jsou kategorie uživatelů definované pro určení přístupových oprávnění k prostředkům aplikace. Role můžou přidělovat vývojáři komponentám, rozhraním a metodám.Uživatelům přiděluje role správce, který tak uživateli v rámci dané role umožní přístup ke všem prostředkům k této roli. Uživatelé jsou rozděleni do logických rolí určeným vývojářem aplikace.Členové určité role sdílí s aplikací stejné privilegia. Přístup k operacím je autorizován na základě členství v příslušné roli.
- **Autorizace orientovaná na prostředky.** V tomto případě aplikace nejdříve příjme roli volající klienta a teprve pak se pokusí o přístup k prostředkům. To umožní operačnímu systému vykonat standardní kontrolu přístupu. Všechny přístupu k prostředkům jsou realizovány pomocí kontextu zabezpečení původního volajícího klienta. Popisovaní zosobnění však velmi významně ovlivňuje škálovatelnost aplikace, neboť znamená, že ve střední vrstvě aplikace nelze efektivně využívat funkci seskupování připojení. Pokud budeme mít aplikaci .NET Framework, je zcela neodmyslitelnou vlastností škálovatelnost. Autorizace je založena na rolích je tedy pro ně nejlepší volbou.

Budeme mít třeba příklad, kdy budeme muset udělat autorizaci založenou na rolích. Nejdříve by jsme měli ověřit uživatele v uživatelské části webové aplikace. Poté uživatele na role. Pokud máme členství v roli, můžeme autorizovat přístup k operacím. Nyní máme nezbytný přístup dalších systémů, to lze pouze za pomocí pevných identit služeb.

3.3 Volba identit pro přístup k prostředkům

Můžeme si vybrat z těchto identit:

- **Identita původního uživatele.** V tomto případě předpokládáme existenci modelu zosobnění (nebo delegování), v němž lze identitu původního volajícího klienta zjistit a předávat mezi jednotlivými vrstvami systému. Možnost delegování je klíčovou podmínkou používanou k určení mechanismu ověřování.
- **Identita procesu.** Výchozí přístup (bez specifického zosobnění). Přístup k místním, jakož i k dalším prostředkům je omezen na identitu aktuálního procesoru. Proveditelnost tohoto pojetí závisí na procházených hranicích, neboť identita procesoru musí být zjistitelná především pro cílový systém.
- **Místní pevný účet.** Tento přístup využívá (pevný) účet služby.

    - pro přístup k databázi to může být pevné uživatelského jméno SQL a hesla použitého komponentou připojující se k databázi
    - je-li vyžadována pevná identita v systému Windows, použijte serverovou aplikaci modelu Enterprise Services

- **Vlastní identita.** Nemáte-li k dispozici vhodný účet systému Windows, můžeme vytvořit vlastní identity. Tyto identity mohou obsahovat podrobnosti odpovídající našemu vlastnímu specifickému kontextu zabezpečení. Mohou například zahrnovat seznam rolí, jedinečných identifikátorů nebo jakýkoliv jiný typ informace.

3.4 Volba mechanismu ověřování

- **Identita.** Mechanismus integrovaného ověřování systému Windows je vhodný pouze v případě, že uživatelé naší aplikace mají účty v systému Windows, které lze ověřovat prostřednictvím důvěryhodné autority dostupné z webového serveru naší aplikace.
- **Správa pověření.** Jednou z výhod integrovaného ověřování systému Windows je fakt, že je možné přenechat starost o správu uživatelských pověření operačnímu systému. U jiných mechanismů, např. formulářového ověřování, musíme nejprve skutečně pečlivě zvážit, kde a jak se budou uživatelská pověření ukládat. Nejčastěji máme tyto dva způsoby:

    - databáze služeb SQL Serveru
    - uživatelské objekty ve službě Active Directory

- **Přenášení pověření.** Potřebujeme implementovat schéma zosobnění a delegování, abyste mohli mezi jednotlivými vrstvami aplikace předávat pověření klienta na úrovni operačního systému. Pokud ano, musíte být schopni zosobnit klienta a delegovat jeho kontext zabezpečení.
- **Typ prohlížeče.** I na tomto velmi záleží protože ne všechny prohlížeče podporují všechny možnosti ověřování.

Ověřování certifikátů, základní, passport, formulář nevyžaduje Internet Explorer. Metody jako je integrované ověřování systému Windows, nebo metoda Digest vyžaduje Internet Explorer.

4 Zabezpečená komunikace

Řada aplikací předává citlivá data týkající se zabezpečení mezi jednotlivými uživateli v síti, nebo mez jednotlivými body aplikace umístěny mezi nimi. Tato data mohou obsahovat pověření používaná k ověřovaní nebo také jiné údaje, jako jsou čísla kreditních karet nebo podrobnosti o bankovních transakcí. Kvůli ochraně informací před nechtěným prozrazením a dat před neoprávněnými úpravami během přenosů musí být kanál mezi komunikujícími koncovými body zabezpečen.

Zabezpečení komunikace lze zajistit dvou funkcí.

- **Utajení.** Funkce zaručeného utajení se stará o to, aby data zůstala utajená a důvěryhodná, aby je nemohli tajně odposlouchávat ani uživatelé vybavení softwarem pro monitorování sítě. Utajení je obvykle zaručeno díky šifrování.
- **Integrita.** Kanály zabezpečené komunikace musí rovněž zaručit, aby byla data chráněna před náhodnými nebo cílenými (zlomyslnými) změnami během přenosů. Integrita je obvykle zajišťována pomocí kódů MAC (Message Authentication Codes)

4.1 Model zabezpečené webové aplikace

Spojení s klientem a webovým serverem může vést Internetem nebo firemním intranetem a typicky používá protokol HTTP. Zbývající dva komunikační kanály spojují servery uvnitř firemní domény. Přesto jsou všechny 3 kanály potenciálním bezpečnostním rizikem. Mnoho čistě intranetových řešení přenáší mezi vrstvami citlivá data o zabezpečení. Na dalším obrázku jde vidět, kde by se případně mohly zabezpečit jednotlivé kanály. Můžeme použít tyto protokoly: SSL, IPSec, šifrování RPC.

4.1.1 Protokoly SSL/TLS

Protokoly SSL/TLS se používají k vybudování zašifrovaného komunikačního kanálu mezi klientem a serverem.

- Je-li použit protokol SSL, používá klient protokol HTTPS (a používá adresu https://URL). Server naslouchá na portu TCP 443
- Povolíme-li užití protokolu SSL, měli by jsme monitorovat výkon své aplikace.Protokol SSL využívá k šifrování a dešifrování dat velmi složité kryptografické funkce, což může mít vliv na výkon ovlivněných aplikací.
- Všechny stránky. Které používají protokol SSL, by jsme měli optimalizovat. Tyto stránky by měli obsahovat minimum textu a velmi jednoduchou grafiku.

- Měli by jsme rovněž zajistit, aby počáteční výměna klíčů nepřekročila limit – Měli by jsme mít pořád na mysli, že užití protokolu SSL zpomaluje běh aplikace.
- Protokol SSL vyžaduje, aby byl certifikát serveru instalován na webovém serveru

4.1.2 Protokol IPSec

Tento protokol se používá při zabezpečení mezi dvěma počítači. Např. mezi aplikačním a databázovým serverem. Protokol komunikuje a s dalším aplikacemi běžným způsobem prostřednictvím obvyklých portů TCP a UDP.

- Zajistit důvěrnost zprávy šifrováním veškerých dat posílaných mezi dvěma počítače.
- Zajistit integritu zprávy posílané z jednoho počítače na druhý (bez nutnosti šifrovat data)
- Zajistit oboustranné ověřování mezi dvěma počítači (nikoli ověřování uživatelů)
- Můžeme omezit počet počítačů, které mohou vzájemně komunikovat. Dále lze komunikaci omezit jen na vybrané protokoly IP a na specifické porty TCP/UDP.

4.1.3 Šifrování RPC

Protokol RPC je základním transportním mechanismem používaný v modelu DCOM. Poskytuje řadu konfigurovatelných úrovní ověřování (a nenulovou ochranu dat), počínaje a plným šifrováním stavu parametrů konče. Nejčastěji se používá v případech, kdy webová aplikace komunikuje se serverovými komponentami modelu Enterprise Services, která je spuštěna na vzdáleném počítači.

4.1.4 Porovnání protokolů SSL a IPSec

Pokud začneme jednotlivé protokoly porovnávat, tak

- Protokol IPSec je především určen mezi dvěma počítači, kdež to protokol SSL je typický pro jednotlivé aplikace
- Protokol IPSec je nastavení, pro celý počítač.Server jde ale rozdělit, aby jedna jeho část protokol SSL používala, zatímco druhá ne. Pokud se rozhodneme k navázání spojení s aplikací SQL Server protokol SSL, můžeme se pro použití tohoto protokolu rozhodnout v klientské aplikaci podle jednotlivých uživatelů.
- Protokol IPSec lze použít společně se zabezpečenými protokoly, které jsou spuštěny nad protokolem IP. Jsou to třeba protokoly http, FTP, smtp.

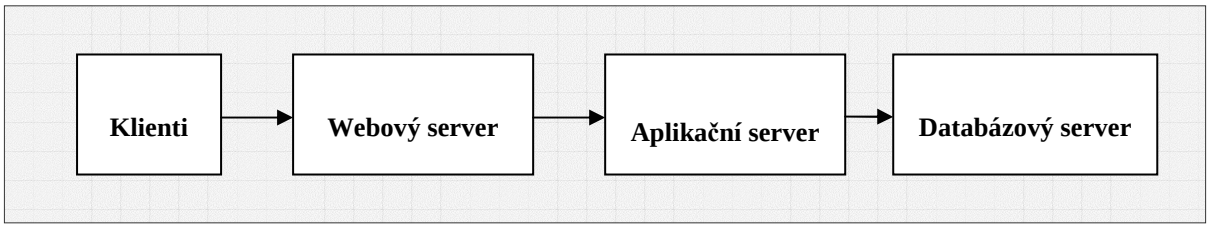
- Protokol IPSec vyžaduje, aby na obou komunikujících počítačích byl instalován systém Windows 2000 nebo novější.
- Protokol SSL může fungovat společně s bezpečnostními branami používající překlad síťových adres (NAT) . Toto protokol IPSec nemůže.

5 Zabezpečení intranetu

Použití ASP.NET aplikace ve spojení s aplikací SQL Server se používá v této době, čím dál častěji.

5.1 Komunikace mezi prostředím ASP.NET a aplikací SQL Server

Aplikace používá schéma důvěryhodného podsystému a spouští volání jménem spouštějících uživatelů. Ověřuje spouštějící uživatele pomocí integrovaného ověřování systému Windows a s databází komunikuje prostřednictvím identity procesu ASP.NET. S ohledem na důvěryhodnou povahu přenášených dat se k zabezpečení komunikace mezi webovým serverem a klienty používá protokol SSL. Na obrázku 5.1 je znázorněna komunikace mezi ASP.NET a SQL Server.



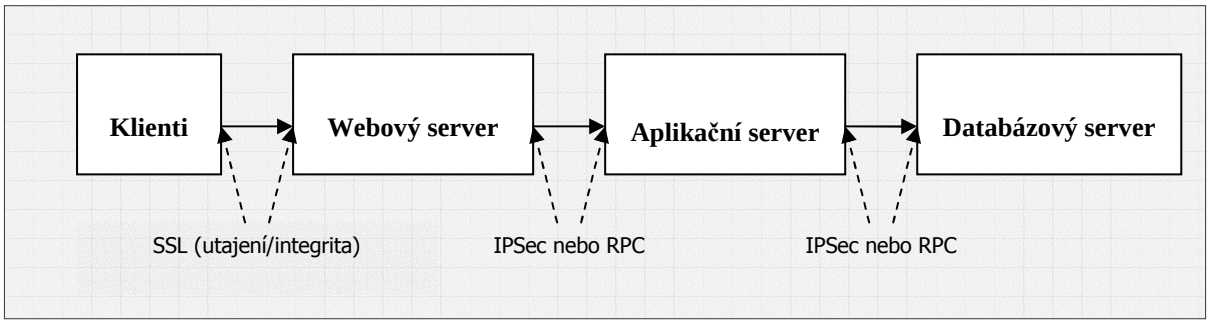
Obrázek 5.1: Typický model webové aplikace

5.1.1 Základní charakteristika

- Klienti musí být vybavení aplikací Internet Explorer
- Uživatelské účty musí být uloženy v adresáři služby Microsoft Active Directory
- Aplikace obstarává důvěrná osobní data
- Přístup k aplikaci mohou získat pouze ověření klienti
- Aplikace komunikuje s databází jménem svých uživatelů
- Aplikace Microsoft SQL Server používá pro autorizaci jen jednu databázovou uživatelskou roli

Webový server ověřuje spouštějícího uživatele a na základě zjištěných údajů. Podle toho mu omezuje přístup k místním prostředkům. Databáze ověřuje pouze implicitní identitu procesu ASP.NET, což je nejméně privilegovaný účet (databáze tedy důvěřuje aplikaci ASP.NET)

Pokud si vezmeme jednotlivé kategorie postupně, měli by jsme dojít k něčemu podobnému jako vidíme na obrázku 5.2.



Obrázek 5.2 Typický model webové aplikace se zabezpečenou komunikací

5.1.1.1 Ověřování

Především by jsme měli poskytnout webovém serveru silné prostředky pro ověřování identit původních uživatelů. Jako ideální se jeví k tomuto kroku integrované ověřování systému Windows. Integrované ověřování systému Windows požívané v internetové informační službě (IIS)se jeví jako ideální, protože všichni uživatelé mají účty v systému Windows a používají Microsoft Internet Explorer.Výhodou integrovaného ověřování systému Windows je skutečnost, že hesla uživatelů nejsou nikde odesílaná do sítě. Přihlášení je navíc z pohledu transparentní, neboť systém Windows používá přihlašovací relaci aktuálního interaktivního uživatele. Při použití tohoto ověřování systému Windows není třeba ukládat pověření do souborů ani je prostřednictvím sítě posílat do databáze. V prostředí ASP.NET použijeme integrované ověřování systémem Windows (bez zosobnění). Účet ASP.NET je nejméně privilegovaným účtem, což minimalizuje ohrožení z jeho strany. Určitě by jsme měli zabezpečit spojení mezi SQL Serverem a to integrovaným ověřováním systému Windows. Databáze důvěřuje voláním aktivního procesu ASP.NET. Identitu procesu ASP.NET ověřuje v databázi.

5.1.1.2 Autorizace

Na webovém serveru je nutné nastavit seznam oprávnění pro přístup z účtu spouštějícího uživatele. Pro jednodušší zprávu by měli být uživatelé seskupeni do skupin systému Windows. Webová aplikace by měla kontrolovat členství spouštějícího uživatele v příslušné roli a na základě výsledku by měl být omezen přístup k vybraným stránkám.

5.1.1.3 Zabezpečená komunikace

Především zabezpečit citlivý data posílaná mezi webovým serverem a databází a taky mezi uživateli a webovou aplikací.

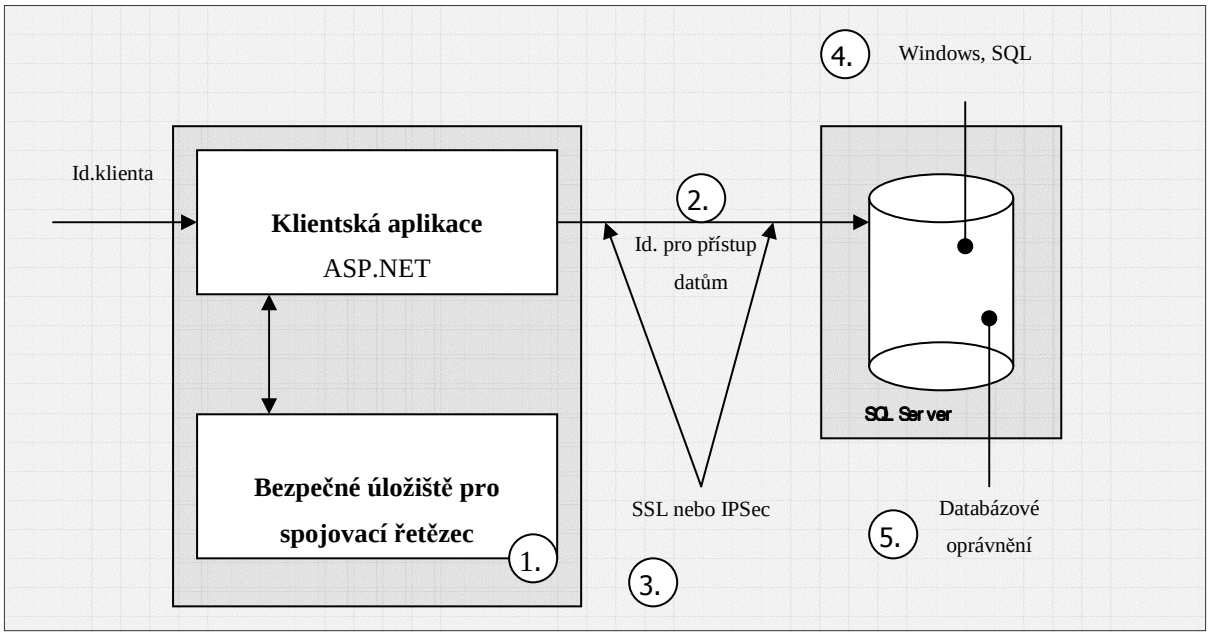
- Existence stejných účtů v systému Windows a v aplikaci SQL server zvyšuje nároky na údržbu. Dojde-li ke změně hesla na určitém počítači, musí být účty synchronizovány a aktualizovány rovněž na všech ostatních počítačích. V určitých scénářích je kvůli snazší správě nutné použít účet s nejmenšími oprávněnými.
- Cesta duplikovaného místního účtu navíc funguje rovněž za přítomnosti bezpečnostní brány (firewall), která uzavírá porty nezbytné k integrovanému ověřování systému Windows. To znamená, že nelze použít integrované ověřování systému Windows, ale ani doménové účty.
- Heslo by mělo být uloženo na bezpečném místě

6 Zabezpečení přístupu k datům

V této kapitole se budeme věnovat výhradně službami SQL Server

6.1 Úvod do zabezpečení přístupu k datům

Na obrázku 6.1 je znázorněno zabezpečení přístupu k datům.



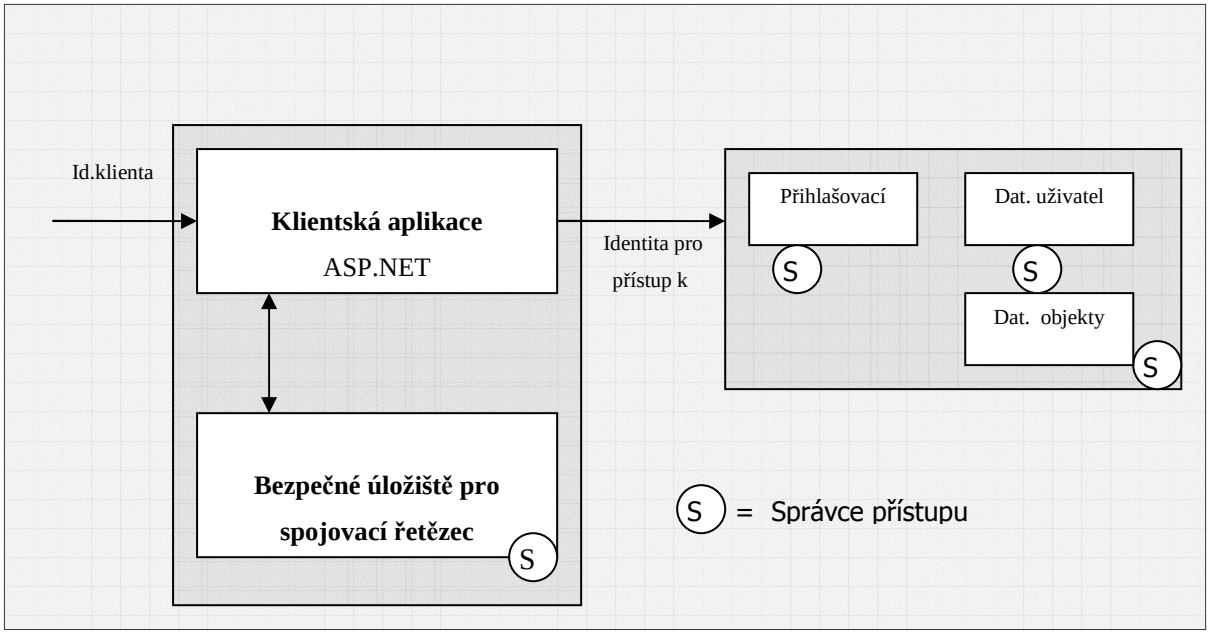
Obrázek 6.1:Klíčové aspekty zabezpečení přístupu k datům

Na co by jsme si měli dávat pozor:

- **Bezpečné ukládání databázových spojovacích řetězců.** Je důležité především v případě, že naše aplikace používá připojení k databázím ověřování službou SQL Server, nebo pokud se připojuje k databázím jiných dodavatelů (než firmy Microsoft), jež od klientů vyžadují přihlašovací pověření. V těchto případech obsahují spojovací řetězce uživatelské jména a hesla jako prostý text.
- **Užití vhodné identity nebo identit pro přístup k databázi.** Pro přístup k datům lze použít identitu volajícího procesoru, některou z identit používaných ke spouštění služeb nebo identitu spouštějícího uživatele. Možné volby jsou předem určeny volbou modelu přístupu k datům – použít můžete model důvěryhodného systému

- **Zabezpečení dat odesílaných do sítě.** Zabezpečení přihlašovacích pověření a citlivých dat odesílaných serveru SQL Serveru. Služba SQL Server 200 podporuje protokol SSL a serverové certifikáty.
- **Ověřování volajících v databázi.** SQL Server podporuje integrované ověřování systému Windows (metodou Kerberos nebo NTLM) a ověřování službou SQL Server.
- **Autorizování volajících v databázi.** Oprávnění jsou přidružena k jednotlivým databázovým objektům. Lze je spojovat s uživateli, skupinami a rolemi

6.2 Správci přístupu SQL Serveru



Obrázek 6.2: Správci přístupu SQL Serveru

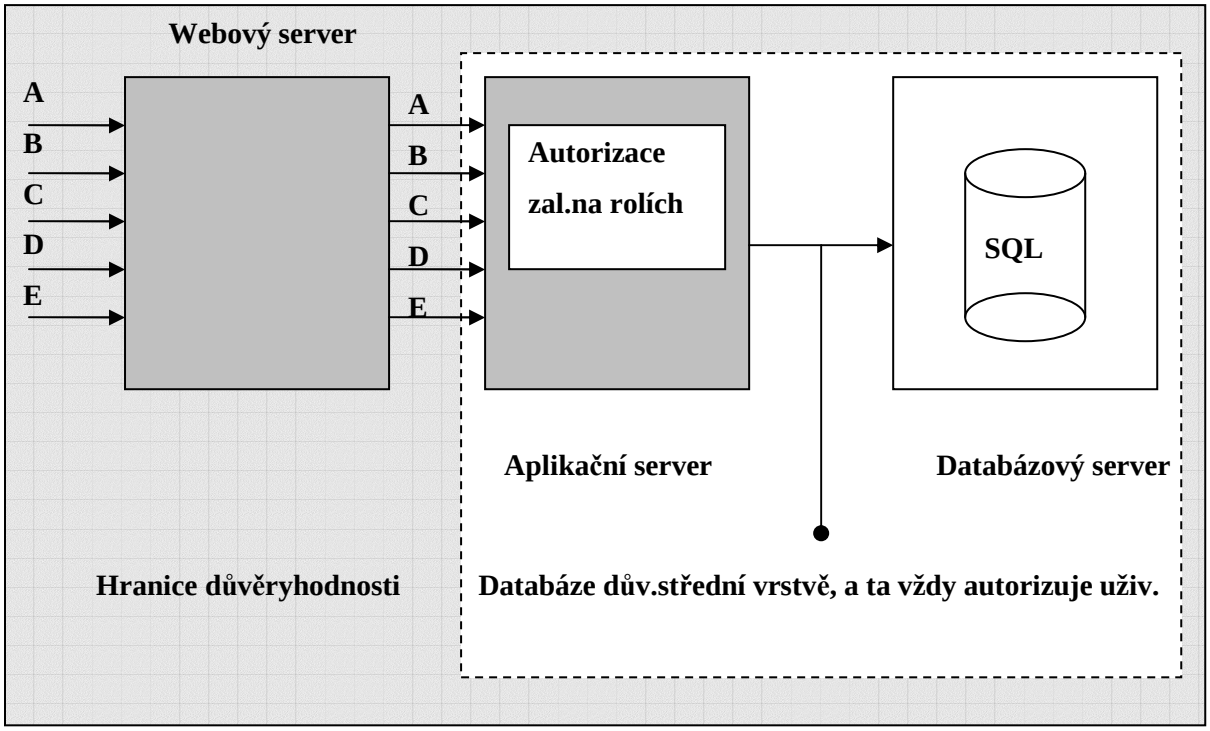
Klíčovými správci přístupu jsou

- vybrané úložiště používané k ukládání databázových spojovacích řetězců
- přihlašovací údaje služby SQL Server (serverový název určený ve spojovacím řetězců)
- uživatel databáze a přidružené databázové role
- oprávnění přidružená k jednotlivým databázovým objektům

6.2.1 Důvěryhodný model zosobnění versus delegování

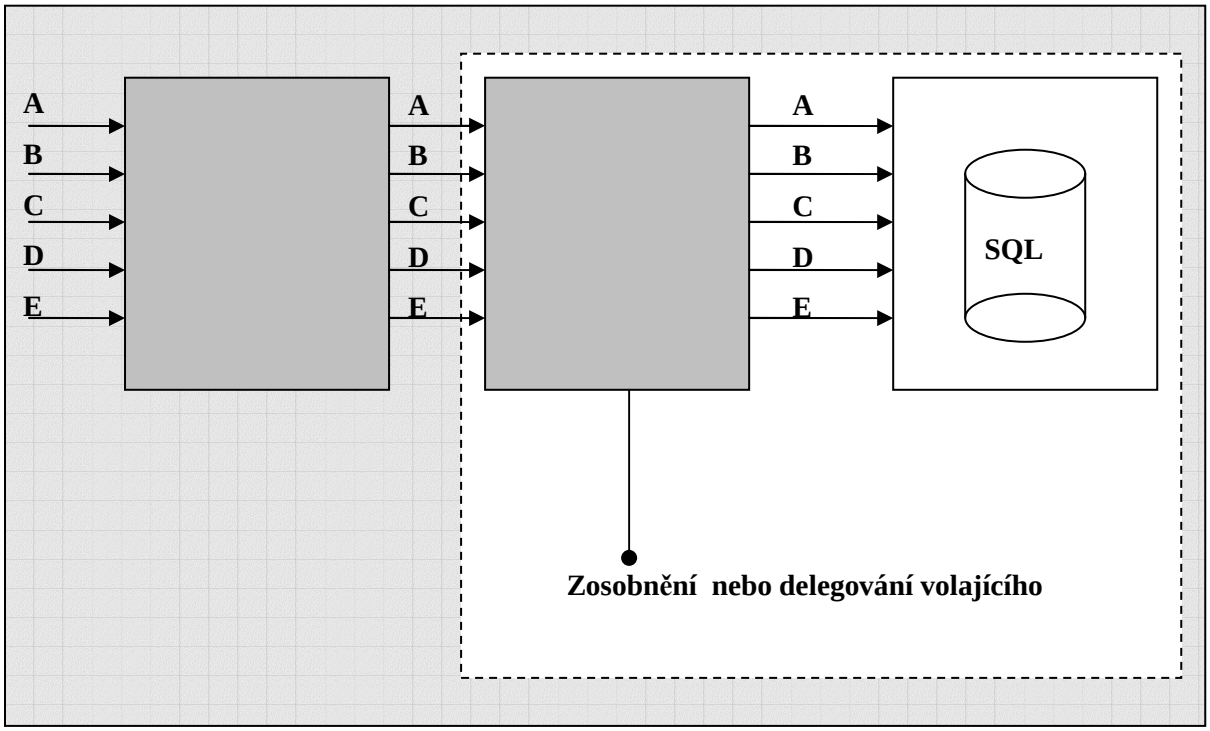
Členění přístupu k databázi je klíčovým faktorem. Potřebujeme v databázi autorizovat jednotlivé uživatele, což vyžaduje existenci modelu zosobnění a delegování.Můžeme k autorizaci uživatelů využít logiku aplikační role ve střední vrstvě.Tady jde opět uplatnit model důvěryhodného podsystému.

Vyžaduje-li databáze autorizaci na úrovni uživatelů, je třeba zosobnit spouštějícího uživatele. I když můžete využít model zosobnění a delegování, doporučuje se model důvěryhodného podsystému.



Obrázek 6.3:Model důvěryhodného podsystému

V něm je spouštějící uživatel ověřen branou IIS/ASP.NET, v niž je mapován roli a potom na základě členství v roli autorizován.Systémové prostředky aplikace jsou dále autorizovány na úrovni aplikace nebo role.K tomu se používají účty služeb nebo identita procesu aplikace (např. ASP.NET)



Obrázek 6.4:Zosobnění nebo delegování volajícího

Měli by jsme si dát pozor především na tyto věci:

- **Jaký typ ověřování používáme.** Integrované ověřování systému Windows nabízí vylepšené zabezpečení, ale zabezpečení, ale bezpečnostní brány (firewalls) a nedůvěryhodné domény Vás mohou nutit k ověřování službou SQL Server.V takových případech by jsme měli zjistit, aby naše aplikace používala tento způsob ověřování co nejbezpečnější cestou.

- **Jedna role proti několika.** Potřebuje naše aplikace přístup pomocí jediného účtu s pevnou sadou oprávnění, nebo vyžaduje několik na rolích založených účtů vyžadující různé úrovně oprávnění.

- **Identita volajícího.**Musí databáze znát identitu spouštějícího uživatele, aby mohla zajistit správnou autorizaci a vhodné auditovaní.Nebo může používat jedno, případně více důvěryhodných připojení a předávat identity volající na úrovni aplikace. Pokud chceme, aby byla identita volajícího předávána na úrovni operačního systému, musíme ve střední vrstvě povolit zosobnění a delegovaní. Tím se velmi výrazně snižuje efektivita sdružování databázových připojení. Tato funkce je sice stále povolena, ale výsledkem jsou velmi malé fondy (pro každý kontext zabezpečení) a malými možnostmi opětovného použití.

- **Citlivá data.** Pokud používáme integrované ověřování systému Windows, nemusíte při komunikaci s databázi odesílat pověření do sítě. Jsou-li citlivá rovněž data aplikace (výplatní listina), měly by jsme komunikační kanál zabezpečit protokoly IPSec nebo SSL.

6.2.2 Předefinované identity v ASP.NET

Pokud používáme v aplikaci předdefinované identity, aplikaci ASP.NET pro zosobnění specifické předdefinované identity. Užití pevné zosobněné identity se v platformě .NET FrameWork 1.0 na serverech Windows se neporučuje. V takových případech je totiž udělit procesoru ASP.NET velmi silné oprávněni „jednat jako část operačního systému“.Toto oprávnění vyžaduje proces ASP.NET, protože volá metodu LogonUser s vašimi pověřením.

Platforma .NET FrameWork 1.1 nabízí rozšíření popisovaného scénáře pro systém Windows. Přihlášení už nezajišťuje proces ASP.NET, ale proces IIS,takže proces ASP.NET už nevyžaduje oprávnění „jednat jako část operačního systému“.

Užití nabízených komponent

Serverové komponenty můžete záměrně vyvíjet, aby obsahovaly kód pro přístup k datům.Pomocí serverových komponent můžeme databázi používat ze serverové aplikaci modelu Enterprise Services spuštěné pomocí specifické identity nebo můžete napsat vlastní kód, v němž zosobnění zajistíme voláním funkce API LogonUser.

Užití serverových komponent spuštěných v odděleném paměťovém prostoru zvyšuje zabezpečení, neboť ztěžuje úlohu útočníka. To platí zejména o případech, kdy jsou oba procesy spouštěny pomocí odlišných identit. Další výhodou je možnost izolace kódu vyžaduje širší oprávnění nebo zbývající části aplikace.

Použití identity spouštějícího uživatele

Tento způsob bude fungovat v případě, že při komunikaci s databází používáte delegování a zosobnění metodou Kerberos, bez ohledu na to, zda voláte databázi přímo z aplikace ASP.NET, nebo serverové komponenty.

Použít účet hosta v internetu

Jinou variantou předchozího přístupu, především u scénářů. Kdy aplikace používá formulářové ověřování nebo ověřování službou Passport (což předpokládá anonymní ověřování ve službě IS), je povolit v konfiguračním souboru aplikace Web.config zosobnění, abychom mohli přístup k databázi použít účet hosta v Internetu (neboli účet anonymního internetového uživatele).

Je-li ve službě IIS povoleno anonymní ověřování, bude kód naší aplikace spouštěn pod účtem hosta v Internetu a s tokenem zosobnění této identity. Ve Webovém prostředí to má nespornou výhodu v auditovaní a sledování přístupu do databáze z mnoha webových aplikací.

Za určitých okolností nelze integrované ověřování systému Windows použít

- databázový klient a databázový server jsou od sebe odděleni bezpečnostní bránou, která neumožňuje integrované ověřovaní systému Windows

- naše aplikace se musí k databázi musí připojovat pomocí více identit

- připojuje se k jiné databázi než SQL Server

- neexistuje způsob, jímž by šlo v aplikaci ASP.NET spustit kód jako vybraný uživatel systému Windows. Nemůžeme odesílat kontext zabezpečení spouštějícího uživatele a raději dáte přednost vyhrazeného účtu služby před přidělováním přihlašovacích údajů koncových uživatelům.

V těchto scénářích musíte použít ověřování službou SQL Server (nebo přirozený mechanismus ověřování databáze). Kromě toho ještě musíme:

- chránit pověření uživatele databáze na databázovém serveru

- chránit pověření uživatele databáze během jejich přenosu ze serveru do databáze

Pokud použijeme ověřování službou SQL Server, nesmíme zapomenout, že existuje mnoho způsobů, jak tento způsob ověřování mnohem lépe zabezpečit.

6.2.3 Ověřování službou SQL Server

Pokud musí naše aplikace z určitých důvodů použít ověřování službou SQL Server, tak by měla splňovat tyto argumenty:

- po připojení k databázi SQL serveru používáme učet s nejmenšími oprávněními

- pověření jsou odeslána do sítě, kde je mohou číst jiní uživatelé. Pověření je tedy třeba zabezpečit

- spojovací řetězec SQL (obsahující pověření) je třeba zabezpečit

Volba účtu pro připojení

Pro připojení k databázi rozhodně nepoužíváme vestavěný účet sa nebo jakýkoliv jiný účet. Jenž je členem předdefinované aplikační role služby SQL Server sysadmin nebo předdefinované role db_owner. Členové role db_owner mají v databázi neomezené privilegia. Místo toho abyste používat účty s nejmenšími oprávněními a se silnými hesla.Toto nám ale pořád neřeší problém s ukládáním pověření jako prostého textu v konfiguračních souborech Web.config. Prozatím jsme omezili rozsah možných škod na možnosti účtu s nejmenšími oprávněními. Další možností jak lépe zabezpečit je šifrování pověření.

6.3 Ověřování klientů služby SQL Server

6.3.1 Integrované ověřování systému Windows

Toto ověřování je v základu bezpečnější než ověřování službou SQL Server. Je to především z těchto důvodů:

- Systém se stará o správu pověření a především nejsou pověření odesílána do sítě.
- Nemusíte vkládat uživatelská jména a hesla do spojovacích řetězců.
- Bezpečnost přihlášení je vylepšeno expiračními periodami, minimální dobu platnosti a uzamčení účtu po několikerém neplatném pokusu o přihlášení.

Pokud používáme pro připojení k databázovém serveru SQL Server z webové aplikace ASP.NET integrované ověřování systému Windows, máme tyto možnosti:

- můžeme použít identitu procesu ASP.NET
- můžeme použít v aplikaci ASP.NET předdefinované identity
- můžeme použít serverové komponenty
- můžeme použít identitu spouštějícího uživatele
- můžeme použít účet anonymního uživatele Internetu (hosta v Internetu)

Neměli by jsme zapomenout nakonfigurovat identitu místnímu procesu ASP.NET. Stačí k tom pouze změna hesla na vlastní heslo na webovém serveru a vytvoření duplicitního účtu na databázovém serveru.

6.3.1.1 Identita procesu ASP.NET

Při připojení přímo na SQL Server pomocí ASP.NET se používá pouze identita procesu. To je běžný postup.. Aplikace v něm definuje hranice důvěryhodnosti. Databáze tedy důvěřuje účtu ASP.NET a povoluje mu přístup k databázovým objektům.

- můžeme použít duplicitní místní účty ASP.NET
- můžeme použít duplicitní vlastní účty
- můžeme použít vlastní doménový účet

Duplicitní místní účty ASP.NET

Jedná se o nejjednodušší postup. Používáme ho především tam, kde máme kontrolu nad cílovou databází. Můžeme zde použít privilegovaný účet ASP.NET, což je místní účet používaný ke spouštění procesů ASP.NET. Tento účet vytvoříme znovu na databázovém serveru.

Vlastní doménový účet

Když máme tento postup, tak předpokládáme, že klientský a serverový PC jsou ve stejné doméně nebo důvěryhodných doménách. Hlavní výhodou je fakt, že oba počítače nesdílejí uživatelská pověření, ale jednoduše umožňují přístup doménového účtu.

7 Rezervační systém

Ve společnosti Daikin Device Czech Republic s.r.o.(DDC) se sídlem v Brně, vznikl požadavek na rezervační systém. Tento rezervační systém by měl sloužit k rezervaci potřebných věcí ve společnosti DDC. Mezi tyto věci patří především firemní auto, projektory, a především místnosti.

7.1 Analýza situace

Ve společnosti DDC je vybudována počítačová síť na platformě Microsoft. Veškeré PC běží na platformě Microsoft. Každý uživatel je připojen do počítačové sítě k jednotlivým serverům. Na jednom z těchto serverů běží i rezervační systém. Celá aplikace je navrhnutá v ASP.NET. Čili programovací jazyk C#, Microsoft Framework, vývojové prostředí Microsoft Visual Studio 2005. Samozřejmě, že v aplikaci je použita databáze a jednotlivý přístup k databázi. Jako databáze je použita MicroSoft SQL Server 2005. K aplikaci se přistupuje přes webové prostředí a to jak přes MicroSoft Internet Explorer tak i přes Mozila Firefox nejnovějších verzí.Samotná aplikace je plně dostačující, ale především mě šlo o zabezpečení přístupu k tomuto rezervačního systému. Zde jsem sáhnul k nejrozšířenějšímu zabezpečení SSL a IPSec. Za pomocí tohoto zabezpečení jsem implementoval do programu tyto zabezpečení a jako pokračovaní jsem uvedl možnost vlastního zabezpečení a zvýšení úrovně bezpečnosti. Samozřejmě, že jsem v průběhu přicházel na různé problémy a právě s těmito problémy bych se zabýval dal při popisu aplikace a její funkce.

7.1.1 Instalace Microsoft SQL Server 2005

Hned při tvorbě aplikace jsem narazil na problém s vývojovým prostředím. Jelikož, jsem začal používat Microsoft Visual Studio 2005, tak jsem byl nucen na serveru přeinstalovat starou verzi SQL serveru. Důvodů jsem během implementace systému našel spoustu. Především mě zaujala lepší podpora vzhledem k formulářovém ověřovaní. Jelikož mi především o toto ověřovaní jde, tak jsem s radostí sáhnul k tomuto řešení. Instalace nového serveru mě docela šla a nesetkal jsem se s žádným problémem, kromě uživatelských instanci na SQL Server 2005.

7.1.1.1 Uživatelské instance na SQL Server 2005

Při vývoji a testování si zaplevelujete server různými testovacími databázemi, pro jednouživatelské desktopové aplikace zase klasické použití SQL Serveru znamená složitější nasazení a správu.
Proto Microsoft SQL Server Express 2005 obsahuje novinku zvanou "uživatelské instance" (user instances). Zdůrazňuji, že vše níže napsané se týká jenom edice Express, v tomto případě neplatí, že vyšší edice mají veškerou funkcionalitu edic nižších. User instances zdánlivě boří shora uvedené paradigma, protože místo databáze zadáváte fyzickou cestu k MDF souboru.

Postup je zhruba následující: Při otevření spojení se vytvoří a nastartuje nová instance SQL Serveru. Jedná se o dočasnou instanci drženou v paměti, která je dostupná jenom pro oho uživatele, který ji vytvořil. Specifikovaný databázový soubor a jeho log se připojí jako jediná databáze a uživatel je do ní odkázán. Po uzavření spojení se databáze zase odpojí a instance se zruší.

Connection string vypadá v uvedeném případě takto: Data Source=.\SQLExpress; Integrated Security=True; User Instance=True; AttachDBFilename=X:\cesta\k\souboru.mdf.

Možnosti SQL Serveru jsou v tomto režimu práce omezené, nefunguje údržba, je omezen systém uživatelských práv atd. Proto je použitelnost uživatelských instancí omezená. Jsou však výhodné ve dvou základních případech.Prvním případem je vývoj aplikací (i webových). Databáze je součástí struktury projektu a je dynamicky připojena v případě, když je to zapotřebí. Je fakticky součástí zdrojového kódu. Pro nasazení na ostrém produkčním serveru se pak databáze vytvoří nebo připojí běžným způsobem.

Druhým případem je tvorba jednouživatelské desktopové aplikace, která chce využívat SQL Server pro ukládání dat. S datovými soubory je možno (pokud není aplikace spuštěna) zacházet běžným způsobem, aniž by bylo nutno se vzdát většiny výhod SQL Serveru. Rovněž není nutno SQL Server Express nijak zvlášť konfigurovat, např. v rámci setupu aplikace a podobně.

S tímto problémem jsem bojoval docela hodně dlouho, protože se mě nedařil přistup do mojí databáze a jako ošetření do budoucna, teď vím, co je potřeba pro příště upravit. Jedná se o to, že se nedoporučuje takový přístup k databázi.

User instance jsou ovšem naprosto nevhodné pro ostrý běh webových aplikací na serveru. Důvodů pro to je několik.

- Vytvoření nové instance, její nastartování a připojení databáze znamená poměrně velkou režii. První požadavek na aplikaci po delší době nečinnosti pak trvá velmi dlouho, protože tohle všechno musí provést.

- Použití více instancí SQL Serveru znamená pro server větší zátěž, je mnohem efektivnější mít jednu instanci a v ní různé databáze.

Takovýmto způsobem používaná databáze se prakticky nedá zálohovat. Souborově ji nezkopírujete (protože soubor je zpravidla otevřen instancí a používá se), plánované zálohování SQL Serveru ale nelze použít. Totéž se týká jakýchkoliv změn v databázi. Uživatelská instance je viditelná jenom pro toho, kdo ji vytvořil, nedá se na ni nijak zvlášť připojit a například modifikovat strukturu tabulek.

Jediný způsob jak zazálohovat databázi nebo provést změny v její struktuře (pokud tak samozřejmě neučiní aplikace sama z kódu) je shodit aplikaci, například pomocí triku se souborem app_offline.htm a zkopírovat přímo datové soubory.

Právě z tohoto důvodu je možno user instance používat pouze u Express edice SQL Serveru, nikoliv u edic vyšších.

Shrnuto kromě tohoto problému se instalace povedla a já jsem se mohl pustit do vývoje aplikace na rezervačního systému.

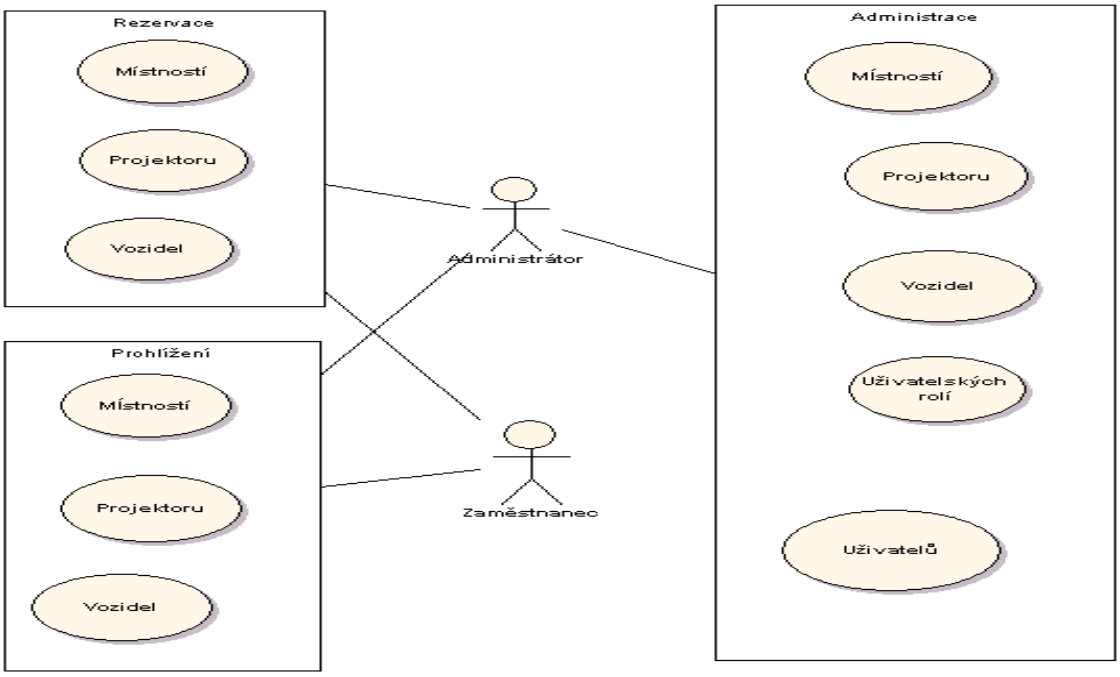
7.2 Vývoj Rezervačního systému

Aplikaci jsem vyvíjel v ASP.NET za pomoci Microsoft Visual Studia 2005. Pokud aplikaci umístíme kdekoliv na server není problém s kteréhokoli PC ve společnosti přes webové rozhraní připojit.

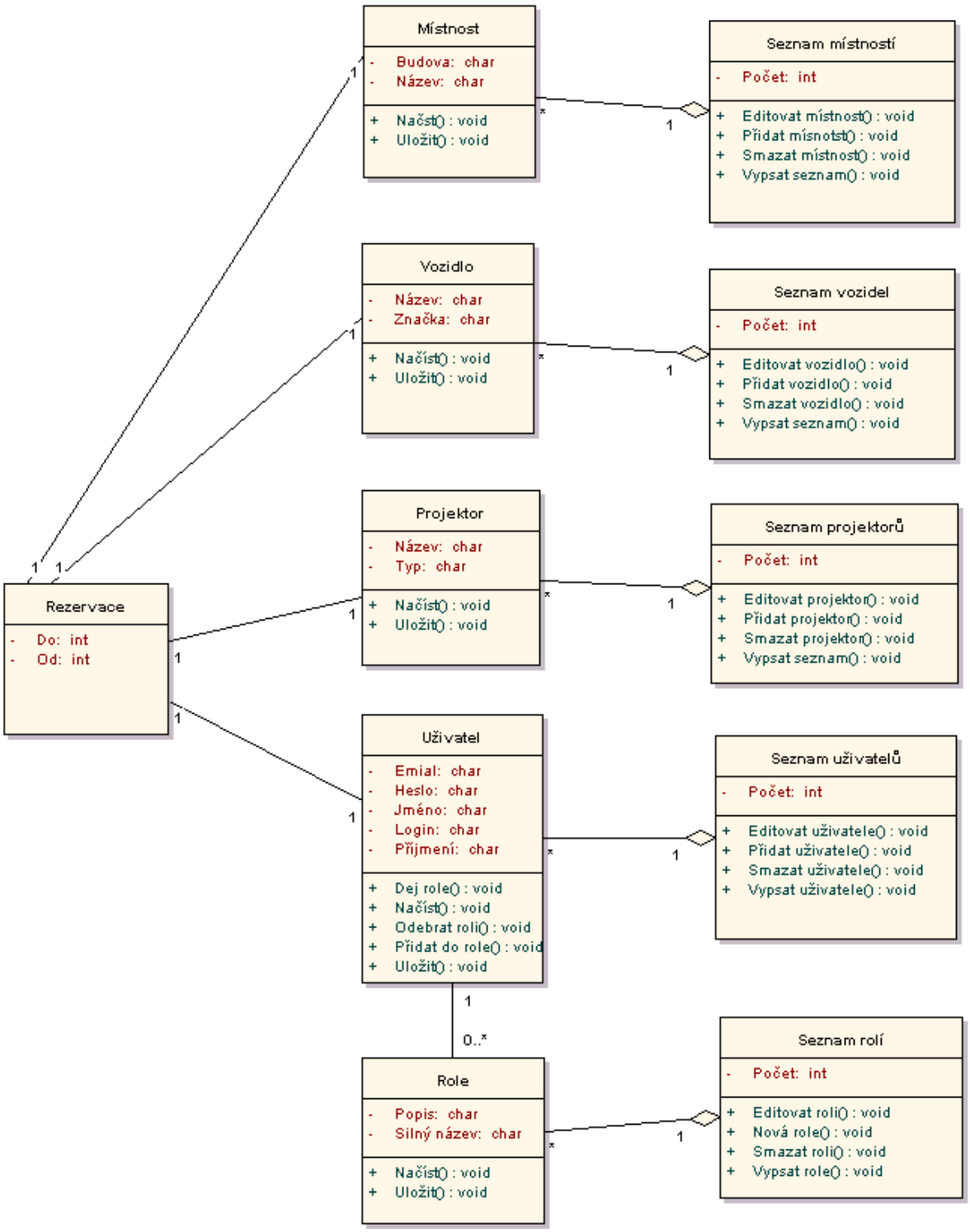
7.2.1 Přihlašování uživatelů

Pokud zadáme do vyhledávače správnou adresu, tak nám se připojíme přímo na server a na rezervační systém. V tuto chvíli jsme vyzváni, abychom zadaly svůj login a heslo. Každý uživatel ve společnosti je identifikovaný svým loginem a heslem. Tyto potřebné údaje uchovávám v databázi, která je umístěna na serveru. Jedná se o jednoduchou tabulku, kde se uchovává login a heslo. Nejlepší pro tento případ bude, se podívat na UML diagram, který se nachází níže. V tomto UML diagramu, je zcela jasné jaké jsou návaznosti v jednotlivých tabulkách.

V tomto přihlašování používám i formulářové ověřovaní, Které je právě hlavním důvodem mé aplikace a praktické části. Hned ze začátku jsem se bohužel setkal s problémem ohledně SSL zabezpečení. Získat certifikát zabezpečení SSL pouze pro tuto aplikaci, nebylo vůbec jednoduché. Sice jsem mohl použít certifikát, který ve společnosti DDC již dávno běží a používá se s japonským spolupracovníkem ohledně výměny dat. Ale vzhledem k testování a vyvíjení aplikace jsem volil jinší možnost a to koupení licence na jeden rok s možností prodloužení tohoto certifikátu.



Obrázek 7.1:UML Usecase



Obrázek 7.2:UML diagram tříd

7.2.1.1 Certifikát SSL na SQL serveru 2005

Instalace certifikátu na server bych nazval jedno velké dobrodružství. Trápil bych se s ním asi hodně dlouho, až nakonec se podařilo.

SSL (Secure socket layer) vyvinula v roce 1996 firma Netscape jako nekomerční otevřený protokol. Používat ho může kdokoli pro soukromé i komerční účely.SSL je vrstva/protokol zabezpečující data na přechodu mezi aplikační a transportní vrstvou (protokolem TCP/IP). Lze zajistit šifrování přenášených dat a autentizace serveru pomocí digitálních certifikátů. SSL není nijak omezeno pouze na protokol HTTP. SSL je možno použít i pro bezpečné připojení prostřednictvím FTP, NNTP ale i k poštovním službám přes SMTP, POP3, IMAP4 a řadu dalších protokolů. Oproti klasickým protokolům se budou zabezpečené lišit jen o písmeno "s" (například FTPS).Pro použití SSL je třeba mít na straně serveru nainstalovanou podporu SSL a také jej musí podporovat náš prohlížeč, což v současnosti podporují téměř všechny.

To, že jsme se připojili na webové stránky zabezpečené pomocí SSL, poznáme podle adresy (obsahuje navíc písmeno s, např. https://server.cz) nebo podle indikace prohlížeče. Obyčejně je zabezpečení přenosu indikováno ikonkou zamčeného zámku ve stavovém řádku okna prohlížeče. V závislosti na nastavení nás prohlížeč informuje o přechodu mezi zabezpečeným a nezabezpečeným režime.
To že jsme v aplikaci úspěšně připojeni k serveru zabezpečeně, nás informuje prohlížeč. Po úspěšné autentizace se úspěšně připojíme k databázi.

7.2.1.2 Autentizace pomocí digitálních certifikátů

Při autentizace ověřujeme pravost klienta (resp. serveru) s nímž komunikujeme. Při tomto procesu se používá asymetrické šifrování např. algoritmus RSA, tzv. digest a certifikáty.

- Digest je výběr znaků z původní zprávy nějakou funkcí. Tato funkce je vybrána tak, aby nebylo možné nebo alespoň maximálně obtížné sestrojit k ní funkci inverzní a aby pro různé zprávy vracela různý digest, tj aby nebylo možné, že by byl digest stejný pro různé zprávy.
- Certifikát vydává nezávislá certifikační autorita a podepisuje jej svým soukromým klíčem. Pomocí veřejného klíče certifikační autority si pak může kdokoliv ověřit pravost certifikátu. Certifikát obsahuje jméno certifikační autority, jméno subjektu, pro který byl certifikát vystaven, veřejný klíč subjektu a údaje o časové platnosti.

Předpokládejme, že spolu chtějí komunikovat dva objekty "I." a "II.". Oba mají k dispozici svůj pár klíčů - soukromý a veřejný.

- I. ověřuje, že komunikuje s II.a pošle II. inicializační zprávu.
- II.odpoví na inicializační zprávu spolu se svým certifikátem.
- I. ověří pravost certifikátu.
- II. odešle I. nezašifrovanou zprávu spolu s tzv. digestem (výběrem znaků této zprávy), který zašifruje pomocí svého soukromého klíče.
- I. rozšifruje pomocí veřejného klíče II. poslanou zprávu - získá tak digest. Tu samou funkci jakou použil objekt II. pro výpočet digestu aplikuje na nezašifrovanou část zprávy a obdrží další digest. Oba dva digesty porovná. Rovnají-li se, pak komunikuje opravdu s tou správnou osobou

Pokud by mezi komunikující objekty I. a II. vstoupil další III. objekt (tzv. "man in the middle"), mohl by podvrhnout svůj veřejný klíč a vydávat se za objekt II. Použitím certifikátu je zaručeno že posílaný veřejný klíč patří opravdu objektu II. Podobně také v posledním kroku je nutné ověření. Pouze objekt, který má správný soukromý klíč (nejen ten veřejný) úspěšně projde autentizace.

7.2.1.3 Navazování spojení, šifrování přenášených dat

Při inicializaci spojení a pro zasílání klíče celé relace se v SSL používá opět používá algoritmus RSA.

- Klient pošle serveru požadavek Client.Hello. Spolu s tímto požadavkem posílá i svůj veřejný klíč (tento je obvykle generován v procesu instalace prohlížeče podporujícího SSL).
- Server přijme požadavek Client.Hello a odpoví Server.Hello. Odpověď zašifruje pomocí veřejného klíče prohlížeče. V této odpovědi posílá i veřejný klíč serveru.
- Po úspěšném přijetí zprávy Server.Hello odešle prohlížeč serveru žádost o klíč, kterým bude šifrována celá relace. I tato zpráva je zašifrována veřejným klíčem serveru.
- Jako odpověď server zasílá klíč relace. Tato zpráva je zašifrována veřejným klíčem prohlížeče.
- Když klient dostane požadovaný klíč relace, šifruje se veškerá další komunikace tímto klíčem, tj. v případě HTTP přenosu se šifrují všechny HTTP požadavky.

Dohodnuté šifrování zůstává v platnosti pro více po sobě následujících spojení. Nové klíče se generují pro každý přenos. Používá se 128 bitový klíč, v USA pak 512 bitový.

7.2.1.4 Užití SSL v aplikaci na webu

Pokud náš server podporuje SSL, je použití úplně jednoduché - stačí prostě přistupovat k naší aplikaci přes protokol HTTPS. Pomocí PHP můžeme lehce ověřit, zda naše aplikace běží v zabezpečeném režimu, aby například někdo nepřistupoval k našim stránkám v nezabezpečeném režimu. Na začátek naší aplikace přidáme tento kód:

```
<?PHP
$SSL_Port=443; // port SSL komunikaci, administrátor serveru může nastavit
i jiný
if ($SERVER_PORT!=$SSL_Port) { // ověřit, zda jde o protokol https
  if (empty($mustbesecure)) // je-li prázdná testovací proměnná,
přesměrovat na zabezpečenou stránku s nastavením této proměnné
    header("Location:
https://$HTTP_HOST$SCRIPT_NAME?$QUERY_STRING&mustbesecure=1");
  else
    echo "Zabezpečené spojení nelze navázat"; // ani po přesměrování se
nezdařilo zabezpečený přenos navázat
  exit;
}
?>
Zabezpečeno
```

Tímto jsem si ověřil , že na serveru plně funguje ověřování SSL a že přihlašování funguje, tak jak má a že využívá přihlašování, tak jak má. Došel jsem také k tomu, že výměna ověřovacích informací totiž může nepříjemně zpomalit naši aplikaci.

7.2.1.5 Vlastní certifikát

Laicky řečeno, potřebujeme dát k dispozici klientům jejich vlastní certifikát pouze tehdy, pokud potřebujeme zajistit, že s naší aplikací komunikuje určitý konkrétní objekt (například klient přistupující ke svému bankovnímu účtu). Pokud nám je lhostejné, kdo s aplikací pracuje, jen chceme zabezpečit, aby informace, které si klient s aplikací vymění (například některý ze správců databáze, redakčního systému) nemohly uniknout, certifikát není potřebný. Při navázání spojení se serverem přes https nabídne prohlížeč k použití certifikát serveru (ten musí být k dispozici, zajistí to správce serveru) a žádný další není potřebný.Vlastní certifikát jsem nepoužíval v aplikaci na rezervační systém. Nicméně jako další pokračování, ho lze v pořádku použít.

7.2.2 Autentizace IpSec

Další věcí kromě šifrování v aplikaci na rezervaci kromě SSL je bezpečnostní rozšíření na IP protokol.V architektuře OSI se jedná o zabezpečení již na síťové vrstvě. Toto rozšíření je tak nezávislé na dalších (vyšších) protokolech TCP/UDP. Je definován v několika desítkách RFC, ale základní jsou 2401 a 2411. Vytváří logické kanály - security agreements (SA), které jsou vždy jednosměrné, pro duplex se používají dva SA.

Bezpečnostní rozšíření vypadá následovně:

- **Ověřování** - při přijetí paketu může dojít k ověření zda vyslaný paket odpovídá odesilateli či zda vůbec existuje.
- **Šifrování** - obě strany se předem dohodnou na formě šifrování paketu. Poté dojde k zašifrování celého paketu krom IP hlavičky.

Základní protokoly:

- **Authentication Header (AH)** - zajišťuje autentizace odesílatele a příjemce, integritu dat v hlavičce, ale vlastní data nejsou šifrována.
- **Encapsulation payload security (ESP)** - přidává šifrování paketů

7.2.2.1 IPSec a rezervační systém

V rezervačním systému jsme už zkoušeli implementovat zabezpečení SSL. Nyní zkusíme implementovat IPSec a uvidíme jakých výsledků dosáhneme oproti SSL.

Pojem IPsec(IP Security Protocol) definuje přidání bezpečnostního mechanismu do standardní IP vrstvy. Bezpečnostní mechanismy, které definuje IPsec jsou dva:

- autentifikace

Definuje vlastní původ dat. Příjemce si muže ověřit, že právě přijatý IP paket pochází opravdu od toho, kdo paket vyslal.

- kryptování

Všechno kromě vlastní hlavičky IP paketu je zakryptováno pomocí předem domluveného algoritmu. Příjemce musí umět tento paket dešifrovat. To znamená, že před vlastním přenosem dat se musí obě komunikující strany domluvit na způsobu šifrování dat.

Jak již bylo řečeno výše toto rozšíření je definováno na IP úrovni. Je tedy nezávislé na protokolech vyšších vrstev. Aplikace nemusí podporovat žádné speciální komunikační metody, aby mohla komunikovat přes IPsec. Mohou se vytvářet šifrované tunely (VPN) nebo se může jenom šifrovat komunikace mezi dvěma počítači.

Instalaci a rozjeti IPSecu se mě nejlépe dělalo pod Linuxem. Bylo to z důvodů plné podpory a instalace. Jako zkoušecí verzi jsem měl nainstalovanou verzi Gentoo na svém notebooku.

- Jako první jsem aktualizoval indexy zdrojů pro instalaci balíčků.
- Poté instalace záplat pod Linuxem.
- Instalace zdrojů jádra příslušné verze
- Rozbalení
- K vytvoření balíčku jsme museli doinstalovat nástroj make-kpkg
- Přeložení a vytvoření balíčku
- Instalace vytvořeného jádra
- Instalace obslužných programů, konfiguračních souborů, ...
  apt-get install freeswan

Zvolil jsme automatické vygenerování klíčů „Plain RSA“ čímž mi byly jednotlivé komponenty klíče uloženy přehledně textově v šestnáctkové soustavě do souboru /etc/ipsec.secrets.

- Na závěr jsme upravili nastavení sítě v souboru /etc/network/options:

ip_forward=no

spoofprotect=no

syncookies=no

- Úplně posledním krokem je nastavení firewallu pomocí iptables.
- Povolení přístupu protokolu UDP na port 500
  iptables -A INPUT -p UDP -i ipsec0 --dport 500 -j ACCEPT

- Povolení protokolu 50 (ESP) a 51 (AH)

iptables -A INPUT -p 50 -j ACCEPT

iptables -A OUTPUT -p 50 -j ACCEPT

iptables -A INPUT -p 51 -j ACCEPT

iptables -A OUTPUT -p 51 -j ACCEPT

Tady je v krátkosti popsán způsob, jakým jsem docílil instalace protokolu IPSec.

7.2.2.2 Konfigurační soubor ipsec.conf

Spojení mezi dvěma počítači se vytvoří přidáním sekce conn název_spojení do souboru /etc/ipsec.conf, ve které se pomocí parametrů a jejich hodnot nastaví vlastnosti spojení. Spojení mezi počítači s adresami xxx.xxx.xxx.xx1 a xxx.xxx.xxx.xx2 se vytvoří následovně:

```
conn MicPet
            left= xxx.xxx.xxx.xx1
            right= xxx.xxx.xxx.xx2
            auth=ah
            auto=route
            leftrsasigkey=0sAQN1+...
            rightrsasigkey=0sAQNgT...
```

Parametry left a right definují IP adresy levé a pravé strany spojení.
Parametr auth říká, jestli se má použít protokol AH.
Parametr auto ovlivňuje, co se s tímto spojením bude dít při bootování systému. V tomto případě se spojení aktivuje. Parametry leftrsasigkey a rightrsasigkey jsou veřejné klíče levé a pravé strany spojení.

7.2.2.3 Vyhodnocení protokolu IPSec

Výsledkem našeho snažení je bezpečná komunikace mezi dvěma stroji přes rozhraní „ipsec0“. Používá se protokol ESP, autentifikace zajišťuje hlavička AH. Na obou stranách (levé i pravé) je soubor /etc/ipsec.conf, který obsahuje konfiguraci spojení conn. Ta je shodná pro obě strany, obsahuje mj. ip identifikace a veřejné klíče. Klíče jsme „nakopírovali“ manuálně ze souborů /etc/ipsec.secret.

Kryptovaný přenos dat jsme otestovali zachycením paketů programem Ethereal na „cizím“ stroji, zatímco jsme přenášeli pomocí protokolu ftp textový soubor. Byly zachyceny pouze ESP pakety, které v sobě samozřejmě obsahují i datovou část ftp paketů, ovšem v zašifrované podobě.

Tímto ozkoušením jsme zjistily jak funguje IPSec vzhledem rezervačního systému. Z výsledků je patrné, že i při zabezpečení, které je implementováno na fyzické vrstvě, jde docílit slušného zabezpečení a to jak na straně serveru i klienta. Zde jsme použily jako jeden PC server a jako druhy klientskou stanici na které běžel vzdáleně rezervační systém. Určitě bylo zajímavém se do budoucna

zamyslet, jak by bylo těžké se dostat přes tuto ochranu a pak ji porovnat s ochranou SSL, která je v této době více rozšířena.

7.2.3 Formulářové ověřování

V této aplikaci, používám formulářové ověřování. Ukládají se v tomto případě uživatelská pověření (uživatelská jména a hesla) společně s názvem skupin, k nimž dotyčný uživatel patří, tj. do databáze SQL Serveru.

Při ukládání jsem se řídil především těmito zásadami:

- **ukládání miniatur (hešových kódů) hesel**.Z bezpečnostních důvodů by neměly být uloženy v databázi jako prostý text. Zde jsem se zabýval uložením jednosměrného hešového kódu uživatelského hesla.Podle mě je tento způsob lepší než ukládání zašifrovaného hesla, neboť odpadají problémy se správou kryptografických klíčů. Vyskytl se zde i problém a to v možnosti zapomenutí hesla, které nelze již obnovit. Aplikace může nabídnout možnost užití nápovědy hesla. Ale určitě nedokáže heslo ukázat z důvodu heše. Dále je použit tzv. salt, což je z důvodu lepšího zabezpečení. Jedná se o to, že se k heslo připojí před šifrováním náhodné číslo.

- **validace uživatelského vstupu.** V aplikaci můžeme vstup předávat do příkazů SQL, například jako textové literáty (doslovné hodnoty) požívané při porovnáních nebo ve výrazech pro rozpoznání vzorů, je třeba věnovat velkou pozornost validaci vstupu. Jednak je třeba zjistit, aby výsledné příkazy neobsahovaly syntaktické chyby, jednak aby útočník nebyl schopen pouštět libovolné příkazy SQL. Validace zadaného uživatelského jména během přihlašování je životně důležitá, neboť model zabezpečení aplikace je zcela závislý na správném a bezpečném ověřování uživatelů.

Pro formulářové ověřování je asi nejdůležitější aplikace, která se při spuštění přihlásí přihlašovací stránkou.Ta obsahuje především přihlašovací formulář, v němž uživatelé zadávají svoje přihlašovací jména a hesla. Je zde samozřejmě implicitní stránka zobrazující jméno identity přidružené k aktuálnímu webovém požadavku a informace o členství identity ve skupinách.

Pro formulářové ověřování byla nutná i konfigurace webové aplikace. To se především odehrávalo v souboru Web.Config, který je součástí přílohy, protože jsou zde velmi důležité nastavení ohledně zabezpečení.

Nutné bylo i vytvořit ve webové aplikaci dvě pomocné metody. Jedna generuje náhodnou hodnotu přísady, druhá hešový kód (miniaturu) založený na hesle a přídavné hodnotě (přísadě).

Tvorba databáze uživatelských účtů byla nutná, protože obsahuje tabulku users se všemi uživateli.Také je vytvořena procedura pro zadávání dotazů.

Uložení podrobností o účtu do databáze pomocí ADO.NET, kde se projevuje uložením zadaného uživatelského jména, miniatura hesla a kryptografickou přísadu do databáze.

Poté je nejdůležitější fáze a to ověření uživatelských pověření na základě databáze. Zde jsem použil třídy rozhraní ADO.NET. Tento kód vyhledá v databázi zadané uživatelské jméno a porovná miniatury hesla s miniaturou uloženou v databázi zkontroluje platnost zadaného hesla. Při formulářovém ověřovaní autorizaci založenou na rolích modelu .NET, můžeme z databáze společně s podrobnostmi o uživateli také seznam rolí, v nichž je uživatel registrován. Tyto informace používám k tvorbě objektu typu GenericPrincipal, jenž je kvůli ověření přidružen k ověřeným webovým požadavkům.

Formulářové ověřovaní funguje tak, že registruje uživatele. Uživatelské jméno, miniatura hesla a kryptografická přísada jsou tedy uloženy do tabulky users v databázi. Poté se přihlásíme pomocí stejných uživatelských pověření a zjistíme funkci rutin používaných k verifikaci hesla.

7.2.4 Formulářové ověřování pomocí Generic Principál

V této aplikaci jsem také implementoval formulářové ověřování pomocí třídy GenericPrincipal (společně s objekty typu FormsIdentity).Tyto objekty jsou určeny k tvorbě schématu autorizace nevyžadující přítomnost systému Windows na obou stranách komunikace, ani domény systému Windows.

Lze to využit například:

- používat formulářové ověřování pro získání pověření uživatelů (uživatelské jméno a heslo)
- kontrolovat platnost zadaných pověření podle informací načtených z datového úložiště
- (například z databáze nebo z adresářového stromu adresářové služby Microsoft Aktive Directory)
- vytvářet objekty GenericPrincipal a FormsIdentity podle informací načtených z datového úložiště (zmiňovanými údaji může být například seznam uživatelových rolí)
- používat uvedené objekty k autorizaci

V tomto formulářovém ověřování se ověřuje uživatel a vytváří se tzv. formulářový ověřovací lístek obsahující informace o uživateli a jeho rolích. Je zde použito mapování informací do objektů typu GenericPrincipal a FormsIdentity, ale taky způsob, jak tyto objekty připojit ke kontextu webovém požadavku http, aby je bylo možné požít k ověřování v aplikaci.

Nejdůležitější je zpracování formulářového ověřovacího lístku.A také zpracovaní jednotlivých tříd GenericPrincipal a FormsIdentity.

Aplikace s přihlašovací stránkou obsahuje přihlašovací formulář, v němž uživatelé zadávají svoje přihlašovací jména a hesla. Samozřejmě, že opět obsahuje implicitní stránku, která zobrazí jméno identity přidružené k aktuálním webovém požadavku a informace o členství ve skupinách.

Konfigurace aplikace je stejná jako u výše popsaného formulářového ověřování.

Tvorba lístku je ta nejdůležitější věc. Ověřovací lístek je specifický typ souboru cookie používaný komponentou ASP.NET FormsAuthenticationModule.Ověřovací kód funguje na principu, že vyhledá zadané uživatelské jméno a heslo nebo v adresářovém stromu adresářové služby.

Tvorba ověřovacího lístku pro ověřené uživatele

Nastavíme potvrzení platnosti zadaných uživatelských pověření. Zajistíme načtení rolí.. Poté se pomocí přihlašovacího formuláře předá uživatelské jméno a heslo. Pokud se uživatel úspěšně ověřil, tak ze seznamu rolí lze, zjistit kde uživatel vystupuje. Poté stačí vytvořit nový ověřovací lístek obsahující uživatelské jméno, dobu platnosti a seznam rolí v nichž uživatel vystupuje. Seznam rolí uživatele je uložen ve vlastnosti UserData.. Nyní stačí zašifrovat řetězec obsahující lístek. Nesmíme samozřejmě povolit cookie.

7.2.5.Delegování pomocí metody Kerberos

Operační systém MicroSoft Windows implicitně používá k ověřování protokol Kerberos. Jedná se o velmi výkonnou funkci, která umožňuje serveru zosobňující volajícího klienta přistupovat ke vzdáleným prostředkům jménem tohoto klienta.

Delegování je velmi výkonná funkce a v systému Windows není ničím omezena. Měla by tedy být používaná s rozvahou. Přístup k počítačům konfigurovaným pro delegováni by měla být pod kontrolou., aby nedošlo k nežádoucím zneužití této funkce.

Pokud budeme chtít zosobňovat server klienta, generuje protokol Kerberos token delegování (který lze použít k odpovědím na ověřovací výzvy ze vzdálených počítačů.) K tomu jsem musel splnit, tato následující podmínky:

- účet zosobňovaného klienta nesmí být v adresářovém stromu adresářové služby Active Directory označen jako citlivý a nevhodný pro delegování

- účet serverového procesu (uživatelský účet, pod nímž je spuštěn serverový proces, nebo účet počítačem je-li proces spuštěn pod místním systémovým účtem) je v adresářové službě označen jako důvěryhodný pro delegování.

Delegování, může mít ale i úskalí a to, zda jsou všechny PC (klienti a servery) součástí jednoho lesa adresářové služby. A také na serveru povoleno delegování a pokud chceme přenášet kontext zabezpečení spouštějících uživatelů mezi vrstvami

7.3 Výsledky testovaní rezervačního systému

Během testování na serveru zabezpečení SSL a IPSec se mě podařilo rozeběhnout obě služby jako základ zabezpečení.

Snažil jsem se především o zabezpečení přenosu mezi SQL Server 2005 a webovým Serverem IIS 6.0. Zde jsem implementoval jak zabezpečení SSL, tak i IPSec.

Ohledně zabezpečení SSL, jsem musel získat k testovaní SSL certifikát, který má omezenou platnost. Jedná se o měsíční verzi. SSL certifikát běží v tuto chvíli na serveru a je plně funkční, akorát ho můžu používat pouze na instalovaném serveru. Pokud se začnu přihlašovat z jinšího PC, tak má omezené funkce. Nicméně funkce na serveru plně funguje a je možná vzdáleným přístupem kdykoliv otestovat.

Ohledně zabezpečení pomocí protokolu IPSec. Který jsem podrobil testovaní pomocí PC s nainstalovaným operačním systémem Linux. Zabezpečení bylo otestováno po mocí programu a funguje.

Zabezpečení přenosu dat mezi klientem a webovým serverem. Zde používám opět SSL certifikát s omezenou platností.Certifikát bez problému funguje, a je plně vyhovující podmínkám společnosti DDC.

Nejdůležitější část této práce se točí kolem ověřování uživatelů. Naimplementoval jsem více zabezpečení resp. Ověřovaní uživatelů.Důležitou věcí je to, že ve výchozím stavu, může být použita jen jedna varianta z uvedených.

- **Formulářové ověřování**

  Pokud se přihlašujeme do aplikace, tak je potřeba vyplnit základní údaje o uživateli a to jméno a heslo. Ověření může fungovat v těchto způsobech

  - **Přihlašovací informace jsou uloženy v databázi**

    Jedná se o jednoduché ukládáni přihlašovacích údajů v databázi (jméno a heslo). Pokud se uživatel snaží přihlásit do systému, tak se pomocí SQL dotazu odešle požadavek o tyto informace a pomocí protokolu SSL se zabezpečeným přístupem se tyto informace pošlou

- **Hashovaní hesel**

  Musíme rozlišovat dvě možnosti a to jednak ukládání a ověřování. Setkáváme se zde taky s pojmem Salt.Salt je náhodně generované číslo, nebo posloupnost znaků, které se připojí k heslu a to je následovně zaheshováno.Toto se přesně děje při ukládání.Při ověřování se načte heslo a náhodně vygenerovaný salt ten se zaheshuje. Tyto dva hashe se porovnají a pokud jsou shodné, tak se jedná o stejného uživatele a uživatel je autorizován.Nastává otázka, co tedy Salt zabezpečuje? Podle mě je to pouze vygenerování něčeho, aby heslo nebylo tak jednoduše objevitelé.

- **Windows autentizace**

  Zabezpečení Kerberos nám zajišťuje kompletně zabezpečení o které se stará operační systém Windows. Je plně nastavitelné a já ho mám naimplementováno v systému, kde ověřuji uživatele při přihlášení do adresářové struktury. Jedná se o jednoduché ověření.

- **Bezpečné ukládání spojových řetězců**

  Zde můžeme a rozlišujeme dvě varianty uložení řetězců.

  - **spojovací řetězec je uložen v registrech systému Windows**

    Zde ukládám řetězec pouze do registrů a když je potřebuji, tak jsou uloženy v registrech.

  - **spojovací řetězec je uložen zašifrované formě**

    Řetězec není uložen v podobě pouze čistého textu, ale v zašifrované podobě.

8 Závěr

V této práci jsem se zabýval především bezpečností aplikací v Microsoft ASP.NET a přístupem k databázi MYSQL přes webové formuláře. V aplikaci, kterou jsem navrhl ve společnosti Daikin Device Czech Republic využívám plně přístupu k databázi MYSQL. Tato databáze se nachází na serveru, který je umístěn v speciální místnosti pro servery. Aplikace je přístupná ve firemní síti (Ethernetu), přistupuje se zde přes Internet Explorer. Stačí pouze zadat adresu aplikace a jsme automaticky přeneseny na úvodní stránky aplikace. Zde je potřeba zadat login uživatele a heslo. Zde se poprvé setkáváme s formulářem pro přihlašování. Určitě se mi ověřilo, že nejlepší je využívat integrované ověřovaní ve Windows. V databázi je vhodné používat především účty s nejmenším oprávněním. Pokud jsem se pokoušel spouštět aplikaci a přistupoval na SQL server, používal jsem učet s nejmenšími právy. Pro autorizaci pouze vlastní role. Pokud jsem přistupoval k SQL Serveru, jednoznačně používat silné hesla. Spojovací řetězce šifrované, kvůli ochraně dat. Při formulářovém ověřovaní podle údajů uložených v databázi SQL, chráním se před útokem vloženým kódem SQL. Pokud přenášíme data po síti, tak pomocí integrovaného ověřování systému Windows chrání uživatelská pověřování, nikoli data aplikací.

Jako pokračovaní vývoj této aplikace se mě jeví především možnost rozšíření aplikace o Active Directory. Tato služba mě pořád chybí na serveru. Pokud by se mě podařilo rozjet tuto službu na serveru, pak bych mohl zabezpečit daná data ještě lépe a účinněji. Nabízí se celá řada možností, jak server se službou Active Directory zabezpečit

Jako přínos této práce oceňuji především možnost seznámení s metodami, které jsem implementoval a zabezpečení firemního intranetu před nevhodnými uživateli, kteří nemají povolený přístup k citlivým (transakčním) datům.

Literatura

[1] Kiszka B., *Vytváříme zabezpečené aplikace v Microsoft ASP.NET*. Computer Press, Brno 2004

[2] G. Andrew Duthie: *Microsoft ASP.NET krok za krokem*. Microsoft, Praha, 2003

[3] Rebecca M. Riordan, *Microsoft ADO.NET krok za krokem*. Microsoft, Praha, 2002

[4] Jeff Prosise, *Programování v Microsoft.NET, Webové aplikace v .NET Framework, C++ ASP.NET*. Microsoft, Praha, 2002

Seznam příloh

Příloha 1. Manuál

V této příloze přikládám návod na ovládání programu

Příloha 2. Zdrojové texty

Zdrojové texty jsou přiloženy na CD.

Příloha 3. CD/DVD

Kompletní program s zdrojovými kódy

Příloha 1

1.1 Manuál k rez. systému společnosti DAIKIN

1.1.1 Popis aplikace

Cílem aplikace je nastudování a implementace bezpečnostních mechanismů v prostředí ASP.NET. Aplikace je vybudována nezávisle na informačním systému společnosti a to z důvodu ochrany dat. Implementované a otestované bezpečnostní mechanismy boudou zapracovány do hlavního informačního systému.

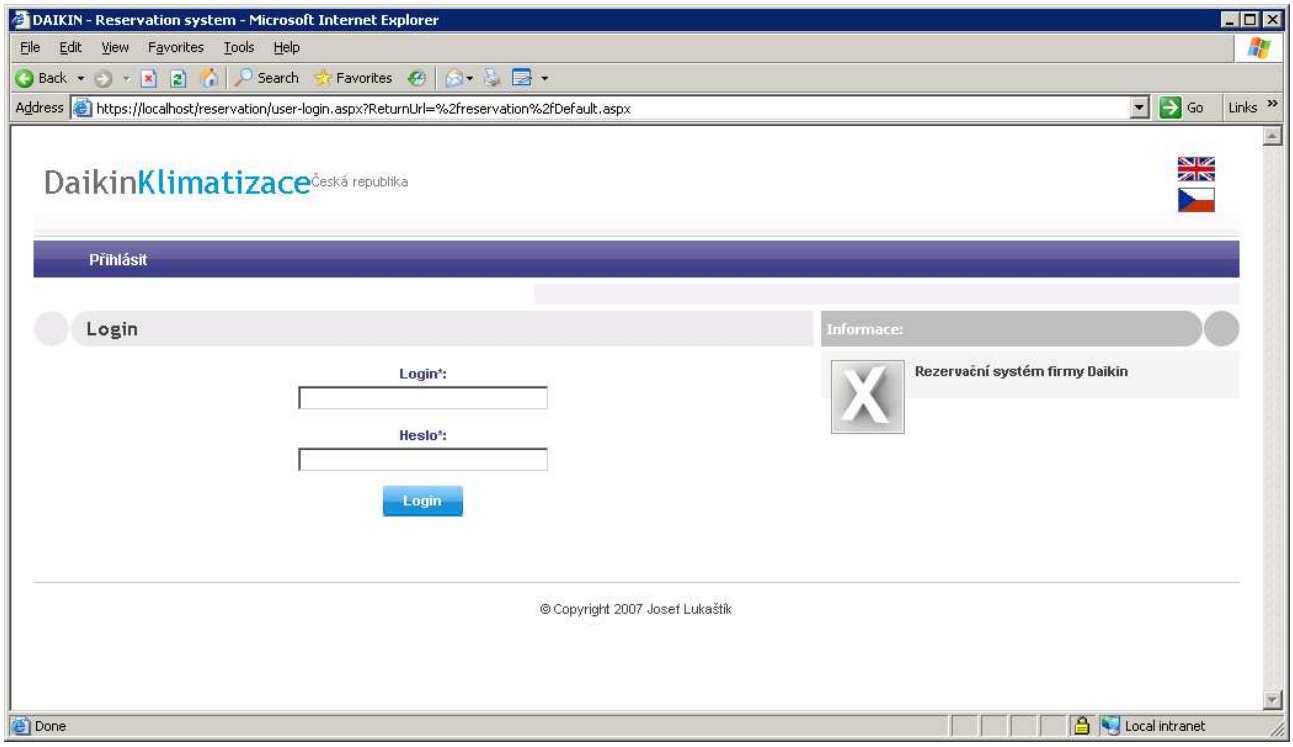
Pro aplikaci zbylo zvoleno formulářové ověřování uživatele s uložením jeho informací v databázi MSSQL 2005. Funkční stránky aplikace jsou chráněny autentizace a všechny požadavky neoprávněného uživatele jsou přesměrovány na přihlašovací stránku.

Aplikace je navržena jako více jazykový web. Snímky obrazovek byly pořízeny v českém nastavení lokalizace aplikace. Proto pro správné porozumění názvům položek je doporučeno mít nastavenou také českou lokalizaci.

Struktura aplikace je rozdělena do 3 hlavních sekcí. První sekce je logo v záhlaví stránky včetně menu. Menu aplikace se mění podle toho, zda je uživatel přihlášen, nebo nikoliv. Až po přihlášení jsou uživateli zobrazena všechna pole menu. Druhou částí je pravý sloupec. Jsou zde zobrazeny aktuality pro uživatele. Spuštění nové verze, popis nových vlastností aplikace a podobné články. Třetí a nejdůležitější částí je hlavní obsah. V této části se na stránkách zobrazují funkční prvky pro editaci položek rezervace, správy uživatelů a vytvoření samotné rezervace.

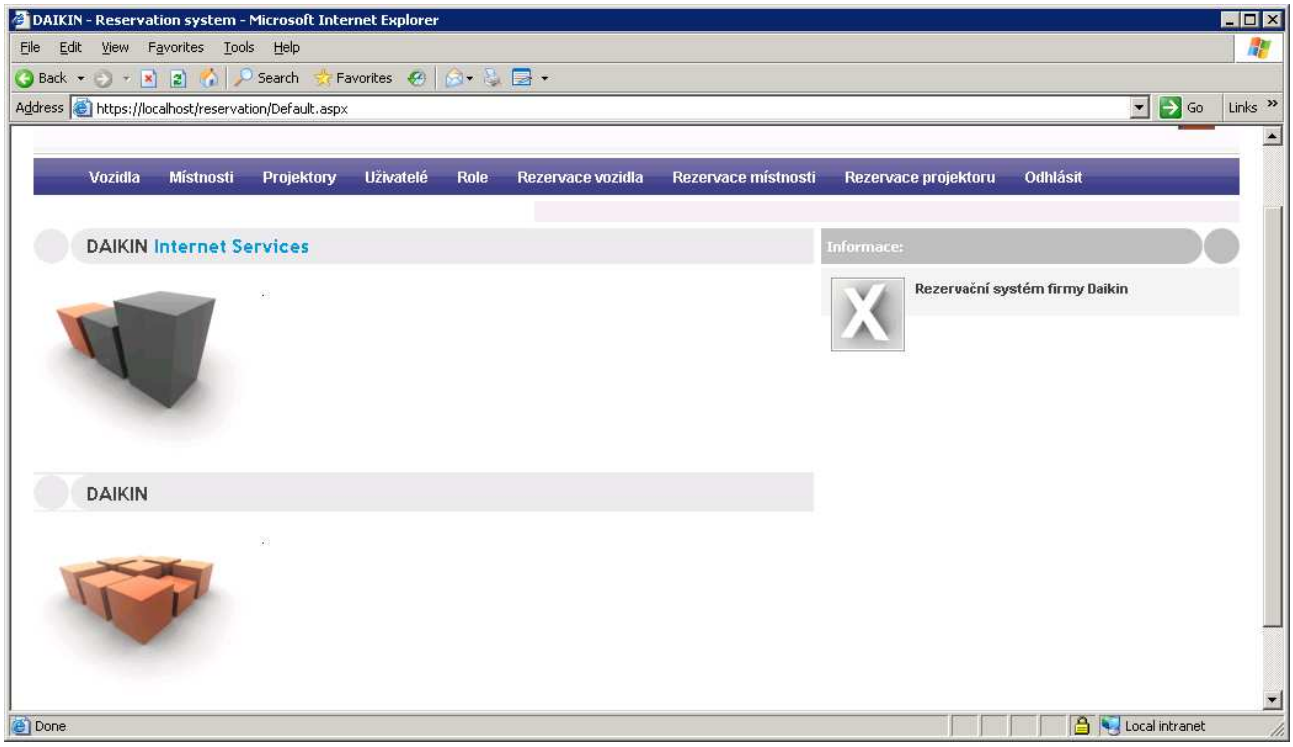
1.1.1.1 Přihlášení do aplikace

Uživatel zadá jméno a heslo do textových polí formuláře na stránce. Pro přihlášení stiskne tlačítko „Login“.



1.1.1.2 Výchozí stránka po přihlášení

Po přihlášení je uživatel přesměrován a stránku, o kterou původně požádal. Ve většině případů uživatelé požadují výchozí stránku aplikace. Na výchozí stránce je zobrazí aktuality určené pouze pro přihlášení uživatele. Z důvodu ochrany dat společnosti jsou tato pole prázdná. Viz obrázek níže.

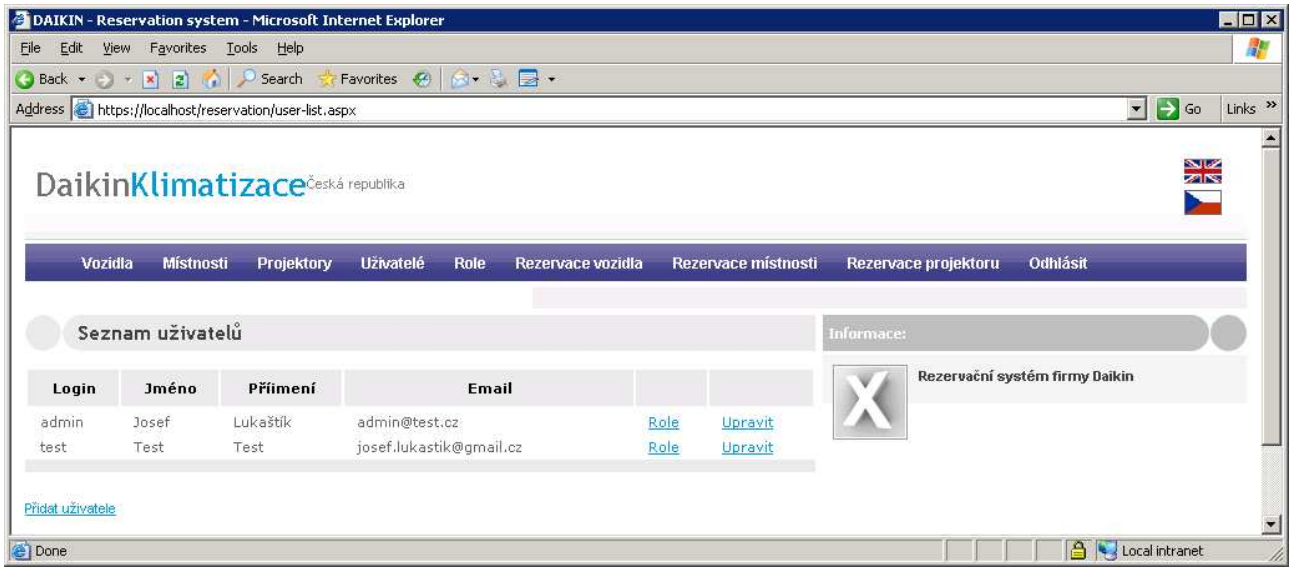


1.1.1.3 Editace položek pro rezervaci

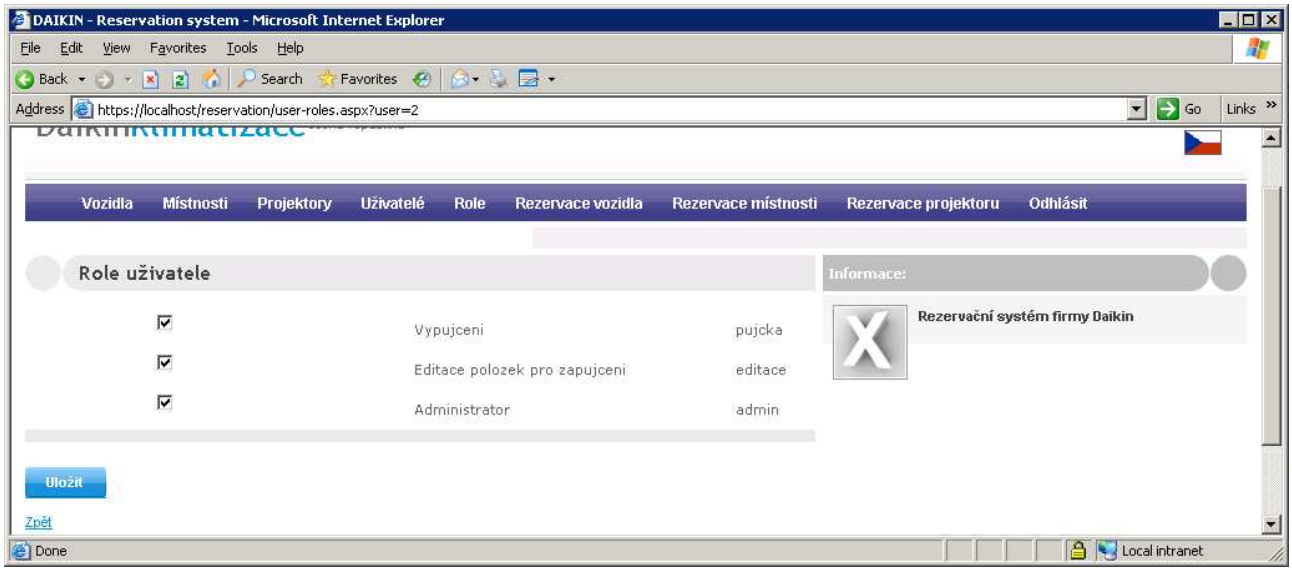
První tři odkazy menu vedou na stránky editace položek pro rezervaci. Pro možnost editace je nutné, aby měl uživatel nastavenou roli „editace“. Pouze v tomto případě může provádět změny.

1.1.1.4 Správa uživatelů a rolí

Pro editaci administrátorských úkonů je nutné, aby měl uživatel přidělenou roli „admin“. Do editace uživatelů se dostanete přes odkaz „Uživatelé“ v menu aplikace. Zobrazí se seznam uživatelů viz obrázek níže. Pro zobrazení rolí uživatele klikněte na odkaz „Role“ u daného uživatele.



Je zobrazena následující stránka, kde uživateli zatrhneme role, které zastarává. Stisknutím tlačítka „Uložit“ jsou informace uložení do databáze.



Poznámka:

Změna oprávnění uživatele se změní až po opětovném přihlášení uživatele do systému.

1.1.1.5 Rezervace položky

Pro možnost rezervace položky je nutné mít opět nastaveny patřičné oprávnění. Tentokrát je k tomu potřeba přidělena role „půjčka“.

Kroky pro rezervaci místnosti:

- ve výběrovém poli typu „combobox“ vybereme místnost, kterou chceme registrovat
- v poli kalendáře zvolíme požadované datum
- nyní v seznamu ne spodní části stránky vidíme, kdy je už místnost reservována
- do textových polí od do zadáme časové omezení, kdy chceme místnost rezervovat. Čas zadávejte ve formátu hh:mm
- pokud byly všechny potřebné údaje vyplněny, přibyla registrace ve spodní části stránky