ČESKÁ ZEMĚDĚLSKÁ UNIVERZITA v Praze

Provozně ekonomická fakulta

Katedra informačních technologií

Diplomová práce

Bezpečnost aplikací na platformě Lotus Notes/Domino

Vít HLAVA

<prázdná strana – vložit zadání DP, první stránka>

<prázdná strana – vložit zadání DP, druhá stránka>

Čestné prohlášení

Prohlašuji, že svou diplomovou práci „Bezpečnost aplikací na platformě Lotus Notes a Domino" jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou citovány v práci a uvedeny v seznamu literatury na konci práce. Jako autor uvedené diplomové práce dále prohlašuji, že jsem v souvislosti s jejím vytvořením neporušil autorská práva třetích osob.

V Praze dne 12. 11. 2013 _____

Poděkování

Rád bych touto cestou poděkoval Ing. Čestmíru Halbichovi za vedení mé diplomové práce, Bc. Lukáši Hanouskovi za pomoc při konfiguraci prostředí IBM Connections, Ing. Petru Titěrovi za vytvoření knihovny pro využití systémových funkcí IBM Domino serveru a dále všem, kteří mě při psaní této práce podporovali.

Bezpečnost aplikací na platformě Lotus Notes/Domino

Security of applications on Lotus Notes/Domino platform

Souhrn

Bezpečnost a ochrana dat je důležitým aspektem každé vytvářené aplikace. Platforma IBM Notes/Domino nabízí v této oblasti široké možnosti, které jsou ovšem velmi často nesprávně využity. Základem této práce je popis bezpečnostního modelu aplikací vytvářených na této platformě, rozbor doporučeného nastavení s popisem možných rizik, pokud nejsou práva nastavena správně.

Bezpečnost aplikací jde velmi často proti pohodlí uživatelů. Jedním z problémů je velký počet různých systému, do kterých je vyžadováno uživatelské přihlášení. To vede většinou k opakovanému využívání stejného slabého hesla, tedy ke snížení bezpečnosti hned několika systému. V rámci práce byl vytvořen bezpečný mechanizmus jednotné autentifikace uživatele do tří systémů IBM Domino, IBM Connections a externího proprietárního webového systému. Mechanizmus využívá tokenu vygenerovaného při prvním přihlášení do externí aplikace. Token je na žádost generován pomocí webové služby na Domino serveru a předáván uživateli, jeho držení pak zajišťuje automatickou autentizaci uživatele vůči IBM serverům.

Summary

In development of any application the security and data protections plays important role. IBM Notes and Domino platform offers robust support of security features which are however not always used according to best practices. This work describes the model of application security, recommended settings of user rights and data protection on selected platform with examples of what wrong implementation can cause.

Application security stays usually against the user comfort. One typical issue is existence of multiple systems with separate authentications which force users to repeatedly use weak password for more applications. In this work the mechanism of single sign on is implemented to access two systems IBM Domino and IBM Connections from proprietary external application. External application requests Domino server to provide LTPA token for authenticated users. Token is provided by web services on Domino server and transferred to user client. The holding of valid LTPA token ensures authentication to both IBM systems without need of typing the password again.

Klíčová slova

IBM Domino, IBM Notes, IBM Connections, Autorská a čtenářská pole, Šifrování, Autentifikace, Web services, Hash, SHA, Base64, LTPA token, SSO.

Keywords

IBM Domino, IBM Notes, IBM Connections, Authors and Readers Fields, Encryption, Authentication, Web services, Hash, SHA, Base64, LTPA token, SSO.

Obsah

Seznam obrázků

Seznam výpisů kódu

Seznam tabulek

1 Úvod

Při tvorbě aplikací se vývojáři velmi často potýkají s více či méně striktními požadavky na zabezpečení. Nejde pouze o autentifikaci vůči systému, ale také o správné nastavení autorizace, tedy přístupových práv k využití různých prostředků systému včetně práva číst a editovat dokumenty.

Stejně jako u mnoha dalších i na platformě IBM Notes/Domino je autentifikace do značné míry zabezpečena nativně. Po spuštění klientské aplikace je uživatel nucen vyplnit jméno a heslo, popřípadě prokázat vlastnictví certifikátu. Pomocí adresáře, ve kterém je uložen otisk hesla, je pak ověřena jeho identita. Složitější je to z hlediska administrátora a správného nastavení oprávnění pro jednotlivé uživatele. Zde je obrovský prostor k tomu, aby vinou nepochopení nebo špatné konfigurace, přestala aplikace splňovat předepsanou bezpečnost. Nejsložitější úlohu z tohoto pohledu mají vývojáři, kteří musí zvážit použití vhodných metod a funkcí pro dosažení požadované úrovně zabezpečení. Bezpečnost každé i sebelepší aplikace je tedy závislá jak na nastavení uživatelských práv administrátorem, tak na přiměřeném použití bezpečnostních mechanizmů a funkcí vývojářem. Vývojář musí zvážit, zda k ochraně dokumentů stačí nastavit autorská a čtenářská pole, šifrování celých dokumentů nebo je nutné šifrovat na úrovni jednotlivých polí s využitím vlastních šifrovacích klíčů, čímž lze např. zajistit, že dokumenty budou v bezpečí i před vlastními administrátory systému.

Na začátku této práce bude představena platforma IBM Notes a Domino, popsán základní bezpečnostní model systému a jeho doporučené nastavení. Přitom bude poukazováno na různá nebezpečí, která mohou ohrozit bezpečnost dat při nedokonalé implementaci.

Ochrana proti neautorizovanému přístupu k dokumentům není jediným předmětem této práce, jak již bylo výše řečeno IBM systémy nabízejí nativní prostředky autentifikace. Nejčastěji se uživatelé do systému přihlašují z tlustého klienta (IBM Notes) pomocí tzv. ID souboru tedy lokálně umístěného certifikátu. Po zadání hesla k ID souboru je spuštěn tlustý klient a uživatel může pracovat s lokálně uloženými aplikacemi. Při pokusu o přístup na Domino server se automaticky použije certifikát uložený v ID souboru a pokud server ověří platnost certifikátu, uživateli je umožněn přístup k aplikacím umístěným na serveru. Kromě tlustého klienta je ovšem možné k aplikací na Domino serveru přistupovat i pomocí webového prohlížeče (tenkého klienta). K tomu je zapotřebí, aby byl na serveru

spuštěn webový server, který umožňuje webový přístup k aplikací (aplikace jsou webovým serverem překládány do HTML). V tomto případě pak autentifikace probíhá klasicky pomocí zadání přístupového jména a hesla. Domino systém jméno a heslo ověří pomocí adresní knihy. Využívat certifikátu není možné, protože webový browser nemá k ID souboru přístup a navíc pro webové uživatele nemusí toto ID vůbec existovat. Tenkým klientem nemusí být pouze webový prohlížeč, může to být např. poštovní klient přistupující přes POP3, IMAP nebo IBM Sametime klient (instant messaging) apod.

V situaci, kdy společnost využívá více než jeden Domino server a webový uživatel potřebuje přistupovat na více serverů, je v základním nastavení nucen se ke každému serveru nejprve pokaždé přihlásit (tedy vyplnit své jméno a heslo). Povinnost hlásit se stejným jménem a heslem k různým serverům stejné Notes domény často způsobuje zkracování a zjednodušování uživatelských hesel a tudíž snižování bezpečnosti celého systému. Proto byl firmou IBM vyvinut mechanizmus tzv. LTPA[1] tokenů, který slouží k ověření uživatele, aniž by musel znovu zadávat přihlašovací údaje. Po prvním přihlášení vygeneruje Domino server LTPA token a pošle jej uživateli v hlavičce HTTP odpovědi. Pokud platný token odešle webový klient v hlavičce dotazu dalšímu Domino serveru, je automaticky autentizován bez nutnosti opětovného přihlašování.

Podobně jako lze zapnout funkci jednoho přihlášení na Domino servery jedné domény, je možné stejný mechanizmus využít i pro přístup k WebSphere serveru, který využívá např. systém IBM Connections. Výzva přichází v okamžiku, kdy je potřeba, aby funkce jednoho přihlášení[2] byla dostupná i z externího systému, který není postaven na IBM technologii. Ve zbylé části práce je popsáno možné praktické řešení funkce jednoho přihlášení v případě, že se uživatelé nejprve logují do externího systému. Řešení spočívá ve vytvoření speciální webové služby, dostupné pouze externímu systému, pomocí které je možné vygenerovat LTPA token pro libovolného uživatele, tento token je pak externím systémem předán uživateli a tím jsou mu zpřístupněny aplikace na Domino serverech.

[1] LTPA – Lightweight Third Party autentication

[2] SSO – Single-Sign-On (funkce jednoho přihlášení)

2 Cíl práce a metodika

Cíl práce

Cílem teoretické části této práce je popis a vysvětlení bezpečnostního modelu systému IBM Notes a Domino s upozorněním na časté chyby, kterých se dopouštějí administrátoři v nastavení uživatelských práv či vývojáři při vlastní implementaci bezpečnostních prvků ve svých aplikacích.

Cílem praktické části je návrh bezpečného mechanizmu funkce jednoho přihlášení mezi externím webovým systémem a webovými aplikacemi postavenými na platformě IBM Domino a WebSphere[3] serveru.

Metodika práce

Metodika řešené problematiky diplomové práce je založena na literární rešerši odborných zdrojů, na vlastních zkušenostech získaných při práci na platformě IBM Notes/Domino a analýze získaných poznatků s ohledem na praktickou využitelnost jednotlivých technologii v aplikacích. V teoretické části práce bylo využito podkladů bakalářské práce autora zpracované na téma aplikace webových služeb na platformě IBM Notes a Domino.

Praktická část práce představuje řešení funkce jednoho přihlášení uživatele z externího systému do platformy IBM. Uživatel autentizovaný v externím systému je automaticky přihlášen do IBM systémů bez nutnosti opakovaného zadání hesla pomocí přenosu autentizačních údajů. Řešení je aplikací technologie webových služeb a šifrovacích algoritmů zajišťujících bezpečný přenos citlivých údajů mezi externím systémem a Domino serverem.

V závěru práce je provedeno zhodnocení zjištěných poznatků a konkrétního příkladu.

[3] Konkrétně aplikací IBM Connections

3 Lotus Notes a Lotus Domino

„Produkty Lotus Notes a Lotus Domino představují širokou softwarovou platformu pro firemní komunikaci, sdílení, publikování dokumentů, tvorbu dokumentově zaměřených aplikací apod. Často se uvádí, že se jedná o groupwarové řešení, což úplně nevystihuje veškeré přednosti této platformy, hlavně možnost jednoduché tvorby vlastních aplikací.“ (Hlava, 2011)

Jedná se o aplikaci typu klient – server. Klientská část se nazývá *Lotus Notes* a serverová část *Lotus Domino*. V nové verzi 9.0 IBM opustila označení Lotus a nahradila je názvem IBM Notes a IBM Domino.

Jednou z největších předností systému IBM N/D je jeho propracovaný bezpečnostní model, jehož popis a následné využití v konkrétní aplikaci bude hlavní náplní této práce.

3.1 Historie

První verzi systému Lotus Notes uvedla firma Iris Software na trh v roce 1989. V roce 1995 tuto společnost koupila firma IBM, která tuto platformu zařadila mezi své prioritní projekty a zasloužila se o celosvětové rozšíření této platformy. Po verzi Lotus N/D 8.5.3. vydané v říjnu 2011, ve které je psána převážná část této diplomové práce, byla v březnu 2013 vydaná devátá verze IBM Notes/Domino s podtitulem Social Edition, který má odrážet přední postavení firmy IBM na trhu se korporátními sociálními systémy.

3.2 IBM Domino

Je serverová část systému, která zabezpečuje všechny systémové funkce (viz níže) a zprostředkovává komunikaci mezi jednotlivými uživateli. Stejně jako u jiných systémů je možné vytvářet rozsáhlou infrastrukturu, ve které je připojeno mnoho domino serverů tvořících společnou doménu.[4]

[4] Typicky je jeden Domino server na každou firemní pobočku, dále bývá často jeden nebo více serverů dedikovaných pro zálohování, pro Lotus Traveler a IBM Sametime.

3.2.1 Platformy

„Již od počátku je serverová část podporována celou řadou operačních systémů: Windows, x86 Linux, Solaris, AIX, iSeries, zLinux, z/OS.“ (Hlava, 2011)

3.2.2 Nejdůležitější funkce Domino serveru

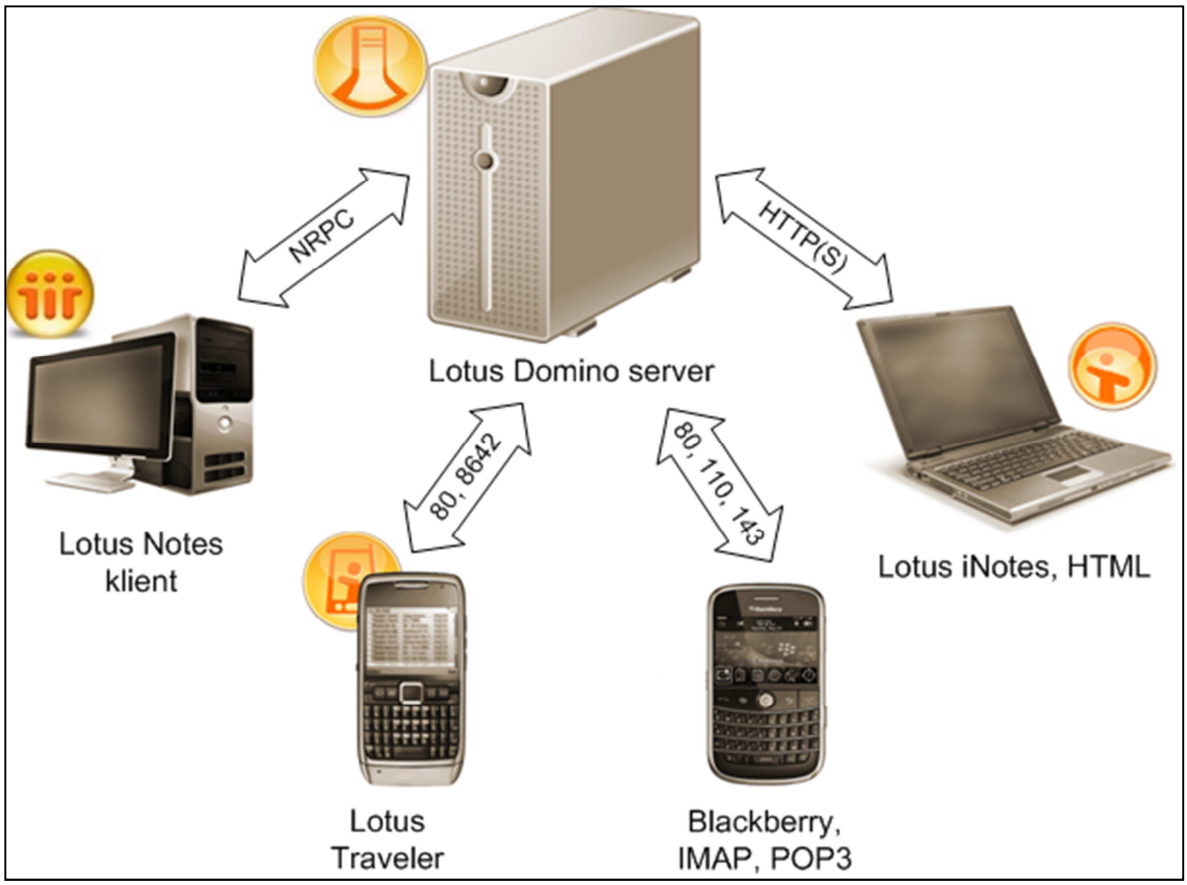
„**Databázový server** - Funkce zajišťující přístup uživatelů k aplikacím a datům uloženým v serverových databázích, synchronizaci dat mezi servery, konzistenci databází atd.

Poštovní server - Server pro příjem a odesílání mailových zpráv. Nejčastěji se pro komunikaci s mailovou schránkou používá přímo klient Lotus Notes, nicméně uživatel může využít i další metody - POP3, IMAP nebo speciální konektory, např. pro MS Outlook.

Aplikační server - Pomocí nástroje IBM Domino Designer je možné vytvářet aplikace (jednotlivé databáze) pro tlustého klienta IBM Notes. Vyvíjí se v jazyce formulí, LotusScriptu popřípadě v Javě. Naprogramovat lze v podstatě cokoli nicméně nejvhodnější typy aplikací jsou aplikace zaměřené na správu dokumentů např. e-mailová databáze, schvalování požadavků, dokumentace, workflow aplikace a podobně. Velkou výhodou je rychlost a jednoduchost programování takovýchto aplikací.

Webový server - Možnost přístupu k databázím pomocí protokolu HTTP (HTTPS). Vhodně navržené aplikace je Domino server schopen převést do HTML podoby a používat je i bez tlustého klienta pomocí webových prohlížečů. Od verze 8.5 je možné v rámci tzv. XPages využít široké škály webových nástrojů pro vytváření dynamických webových stránek např. Dojo frameworku, AJAXu, Javových apletů a dalších.

Další funkce – Po instalaci doplňků je možné přistupovat k poště pomocí mobilních zařízení (IBM Notes Traveler) či Blackberry (BES for Lotus Domino), je možné využívat funkcí okamžitých zpráv (IBM Sametime) včetně přenosu souborů, audia a videa (obdoba aplikací ICQ, Skype na firemní úrovni). Pomocí četných konektorů je možné integrovat datové zdroje z jiných databázových zdrojů, jako jsou SAP, DB2, MS SQL server a další. Adresní kniha Domino serveru sloužící pro autentikaci uživatelů může fungovat také jako LDAP pro integraci s dalšími systémy.“ (Hlava, 2011)



Obrázek 3.1. Komunikace různých klientů s IBM Domino serverem (Zdroj : Autor, 2011)

3.2.3 IBM Notes

Program IBM Notes (do verze 9.0 Lotus Notes) je klientská část systému IBM N/D. Od verze 8.0 je programována v Javě (v prostředí Eclipse). Původní verze byly vyvíjeny v jazyce C a byly výrazně rychlejší a stabilnější. Přechod na Javu přinesl flexibilitu a rozšiřitelnost systému a problémy se stabilitou a výkonem byly nakonec doladěny do uspokojivé úrovně (jistě také samotným rozvojem hardwaru.) V aktuálních verzích je systém podporován na operačních systémech Windows, Mac OS X a Linux.

3.2.4 Funkce IBM Notes

Jako klientská část aplikace se předpokládá, že bude IBM Notes připojen k Domino serveru, ale není to podmínkou. IBM Notes může fungovat i samostatně bez Domino serveru, stejně jako může Microsoft Outlook pracovat samostatně bez serveru Exchange a poštu či jiná data přijímat z jiných zdrojů pomocí standardních protokolů jako POP3, IMAP, RSS ... nebo jiných i proprietárních protokolů. Kromě typických aplikací umožňuje klientské prostředí využívat mnoho dalších funkcí, např. chatování a hlasovou

komunikaci mezi uživateli pomocí IBM Sametime, prohlížení webových stránek, RSS čtečku a lze do něj integrovat další Javové aplikace (widgets).

3.2.5 Silné stránky

Jednou z historických výhod klientského systému je automatický nástroj pro replikace aplikací mezi serverem a klientem. Schopnost replikace je univerzálně použitelná pro všechny databáze postavené na této platformě. Používá se pro zvýšení dostupnosti dat pro domény s více servery, pro zálohování aplikací na jiný server, k šíření nových verzí jednotlivých aplikací a také pro práci off-line na počítači uživatele. Posledně uvedená výhoda je v dnešní době poměrně levného, dostupného a rychlého internetu sporná, nicméně je stále využitelná a využívaná.

Další výhodou je poměrně rychlý vývoj aplikací. Možnost tvorby pohledů a agentů standardními uživateli aplikace, samozřejmě po na nastavení potřebného oprávnění.

„Zásadní předností je i vysoké zabezpečení dokumentů uložených v aplikacích. Na každém dokumentu lze definovat, kdo má právo ho číst a kdo editovat. Systém umožňuje šifrování dat až na úroveň jednotlivých polí dokumentu. Pole zašifrovaná uživatelem nedokáže přečíst ani administrátor systému“ (Hlava, 2011). Více bude tato problematika popsána v následující kapitole.

3.3 Bezpečnost a přístupová práva

Bezpečnost je jednou z nejsilnějších stránek systému IBM Notes a Domino. Vzhledem k výše uvedeným funkcionalitám je zabezpečení systému potřeba nahlížet na různých úrovních:

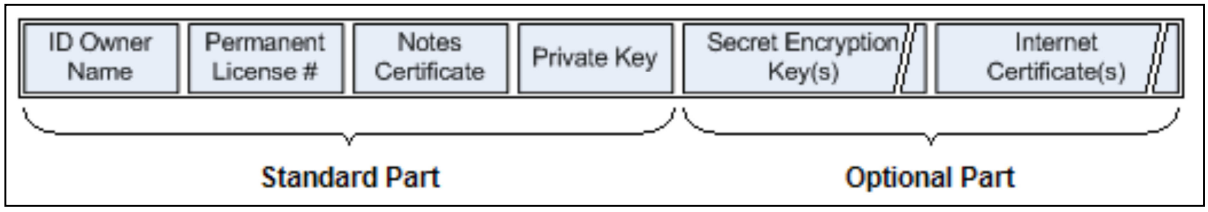
- Zabezpečení IBM Domino serveru, ochrana k datům uloženým na disku,
- Přístup k serveru pomocí IBM Notes klienta nebo jiným způsobem (autentikace a autorizace),
- Zabezpečení klientské části - IBM Notes,
- Přístup k databázím (aplikacím), ochrana dat před zkopírováním (šifrování dat na úrovni databází),
- ochrana uživatelů před administrátory (šifrování jednotlivých dokumentů nebo částí dokumentu v rámci jedné aplikace.

3.3.1 Notes ID, zabezpečení klienta IBM Notes

Nejběžnější přístup k datům na Domino serverech je z IBM Notes klienta, uživatel je po spuštění klienta vyzván k zadání hesla k Notes ID souboru. Heslo slouží pouze pro přístup k tomuto lokálně uloženému souboru nikoli k Domino serveru. K serveru se pak přihlašuje až pomocí certifikátu obsaženému v ID souboru. Tento soubor obsahuje všechny podstatné informace k tomu, aby mohl být uživatel autentizován k Domino serveru (Dahm, Ryan, Schwartz, Smith, & Stalder, 2006):

- Uživatelské jméno (plný hierarchický formát: CN=Vit Hlava/O=CZU/C=CZ)
- Licenční kód (typ licence daného uživatele)
- Certifikáty vzniklé při registraci uživatele (nového serveru). Jedná se digitální podpisy soukromého klíče z certifikačního ID souboru, které ověřují pravost vlastníka ID souboru a souvisejícího veřejného klíče, který je uložen v adresáři na Domino serveru.
- Privátní klíč, který slouží k podepisování odchozích zpráv nebo naopak k přečtení zašifrovaných příchozích.
- Volitelně může ID souboru obsahovat další šifrovací klíče, které pak mohou využívat jednotlivé aplikace. Internetové certifikáty, používané pro podepisování a šifrování odchozím mailových zpráv.

Na obrázku Obrázek 3.2 jsou znázorněny jednotlivé části ID souboru.



Obrázek 3.2 - Struktura Notes ID souboru, zdroj (Dahm, Ryan, Schwartz, Smith, & Stalder, 2006)

Většina informací v souboru je zašifrována pomocí klíče vygenerovaného z uživatelského hesla, veřejné dostupné je pouze uživatelské jméno.

3.3.2 Zabezpečení Domino serveru

Aby mohl být celý systém považován za bezpečný, musí být ochrana zaručena na všech úrovních tohoto systému. Zabezpečení serveru je z tohoto pohledu klíčové, protože bez spolehlivé ochrany všech jeho součástí, není možné zabezpečit ani ochranu dat jednotlivých aplikací a služeb poskytovaných na tomto serveru.

Instalace a souborové uložení jednotlivých databází.

Po instalaci Domino serveru na zvolený operační systém je potřeba zkontrolovat výchozí nastavení zabezpečení tohoto operačního systému.

Musí se zajistit, že neoprávněný uživatel (útočník) nebude moci odcizit data z tohoto serveru. Tzn. nepovolit nezabezpečený vzdálený přístup k tomuto systému, ale také neumožnit nezabezpečený fyzický přístup k hardware.

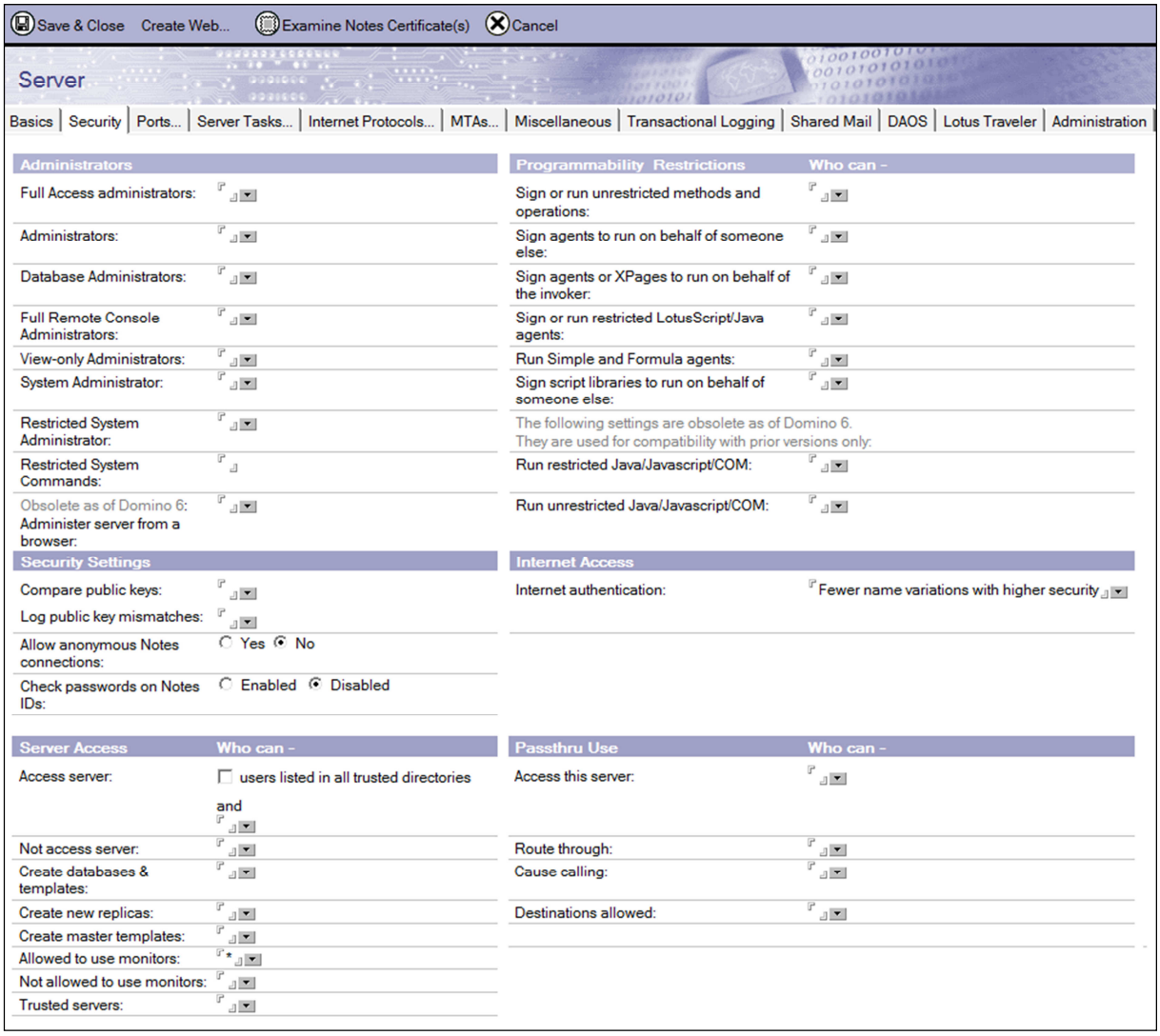
Druhou možností je šifrování souborů na úrovni operačního systému nebo na úrovni IBM Domino serveru, ani jedna se ale v praxi moc často nepoužívá, protože má své nevýhody: nižší výkon, problematické zálohování (inkrementální), archivace, napojení dalších služeb a externích aplikací. Všechny tyto nevýhody jsou sice řešitelné, ale znamenají vícenáklady, které je potřeba zvážit.

Pro administraci Domino serveru je potřeba mít vygenerovaný administrátorský účet. Standardně se tento účet vytváří při instalaci Domino serveru, ale později je možné stejná práva přidělit i libovolnému jinému účtu.

Základní nastavení, ale zdaleka ne jediné, se definuje na tzv. *server dokumentu*. Na záložce *Basic* lze nastavit jaké adresní knihy se používají k autentizaci a autorizaci k systému IBM Domino. Záznamy v adresní knize jsou několikerého typu: Osoba, Server, Skupina osob, Skupina serverů nebo bez udaného typu. Skupiny mohou obsahovat přímo osoby nebo servery nebo další skupiny (*vnořování skupin*). Výpočet konečných práv pro účty zadané pomocí skupin bude vysvětlen v dalších kapitolách.

Nejdůležitější serverové nastavení je na záložce *Security.* Zde se mimo jiné nastavuje:

- kdo smí resp. nesmí přistupovat na server
- kdo smí vytvářet nové aplikace, vytvářen nové repliky aplikací,
- kdo může administrovat server, vidět a zadávat příkazy na konzoli
- kdo může spouštět automatické skripty (dle typu skriptu a úrovně práv)
- a další



Obrázek 3.3 - Server dokument – Security záložka (zdroj Lotus Notes v 8.5.3)

Doporučené nastavení oprávnění

Pro správné nastavení práv k administraci serveru je potřeba si uvědomit, že konfigurační server dokument je také dokument uložený ve standardní základní aplikaci – v adresní knize (Domino directory), která je na každém serveru a souborově se nazývá names.nsf. Ten kdo má práva editace a vytváření dokumentů v této aplikaci má zároveň možnost si nastavit libovolné práva pro administraci serveru. Dokonce i uživatel, který do této aplikace nemá přístup, ale má nepřiměřeně vysoká jiná práva si může tato práva přidělit. Např. právo podepisovat skripty serverovým certifikátem, lze zneužít tak, že se napíše skript, který otevře adresní knihu a přidá vybraný účet mezi administrátory systému. Pak stačí skript naplánovat ať se spouští např. každých pět minut, podepsat ho serverem a počkat až ho server se svými právy alespoň jednou provede.

Výsadní právo *Full Administrator* je dalším silným prostředkem pro administrátory. Při aktivaci plné administrace může administrátor obejít bezpečnostní nastavení databázích i zabezpečení definované na jednotlivých dokumentech. Jedinou „ochranou“ uživatelů před zvídavými administrátory s příliš vysokými právy je pak šifrování. Doporučení pro nastavení plné administrace je vygenerovat speciální účet Full Administrátor a tomuto účtu nastavit práva pro plnou administraci. Pro přihlášení k tomuto účtu nastavit nutnost zadání alespoň dvou ze tří hesel a každé heslo sdělit jinému uživateli (např. vedoucímu IT a provoznímu a finančnímu řediteli firmy). Bez znalosti minimálně dvou z těchto hesel pak není možné se k tomuto účtu přihlásit.

Pro nastavení bezpečnosti ostatních služeb Domino serveru se používají další konfigurační dokumenty a záložky server dokumentu. Například seznam aliasů, které uživatel může použít pro autentizaci k webovému serveru, IMAP, POP3 a ostatním, lze např. povolit nebo zakázat aby se webový uživatel mohl logovat pomocí své e-mailové adresy. Lze vynutit, aby logovací webový formulář byl přenesen pomocí bezpečnějšího HTTPS i když je webová session vytvořena přes HTTP.

Pomocí administračních politik lze nastavit minimální úrovně síly hesel, pravidelné vynucení změny hesla, možnost hlídání konzistence hesel v případě, kdy uživatel používá IBM Notes klienta na více počítačích. Pokud se změní heslo k certifikátu na jednom počítači, server pro přilogování rozpozná změněné heslo (haš tohoto hesla a datum změny si uloží). Při pokusu o připojení se starým heslem (z druhého počítače uživatele) bude uživatel odmítnut s hláškou, že heslo bylo změněno. Při prozrazení hesla stačí změnit heslo tohoto certifikátu na jiné, pokud byl odcizen i počítač, je potřeba obnovit certifikát ze zálohy a změnit heslo, pak se již útočník kvůli nekonzistenci hesla na server nedostane. Dostane se ovšem k datům, která jsou pomocí replikace uloženy na lokálním počítači.

3.3.3 Zabezpečení na úrovni aplikací

Systém přístupových práv je potřeba při vývoji aplikace pečlivě navrhnout a nastavit, bezpečnost systému může být snížena nejen špatným nastavením systému administrátory ale již samotnými vývojáři.

Přístupová práva se definují v *Access Control Listu* (ACL) každé aplikace a práva uživatelů jsou rozdělena do následujících sedmi základních úrovní, viz Obrázek 3.4. Přístupová práva se přidělují konkrétnímu uživateli, serveru, skupině uživatelům či skupině serverů.

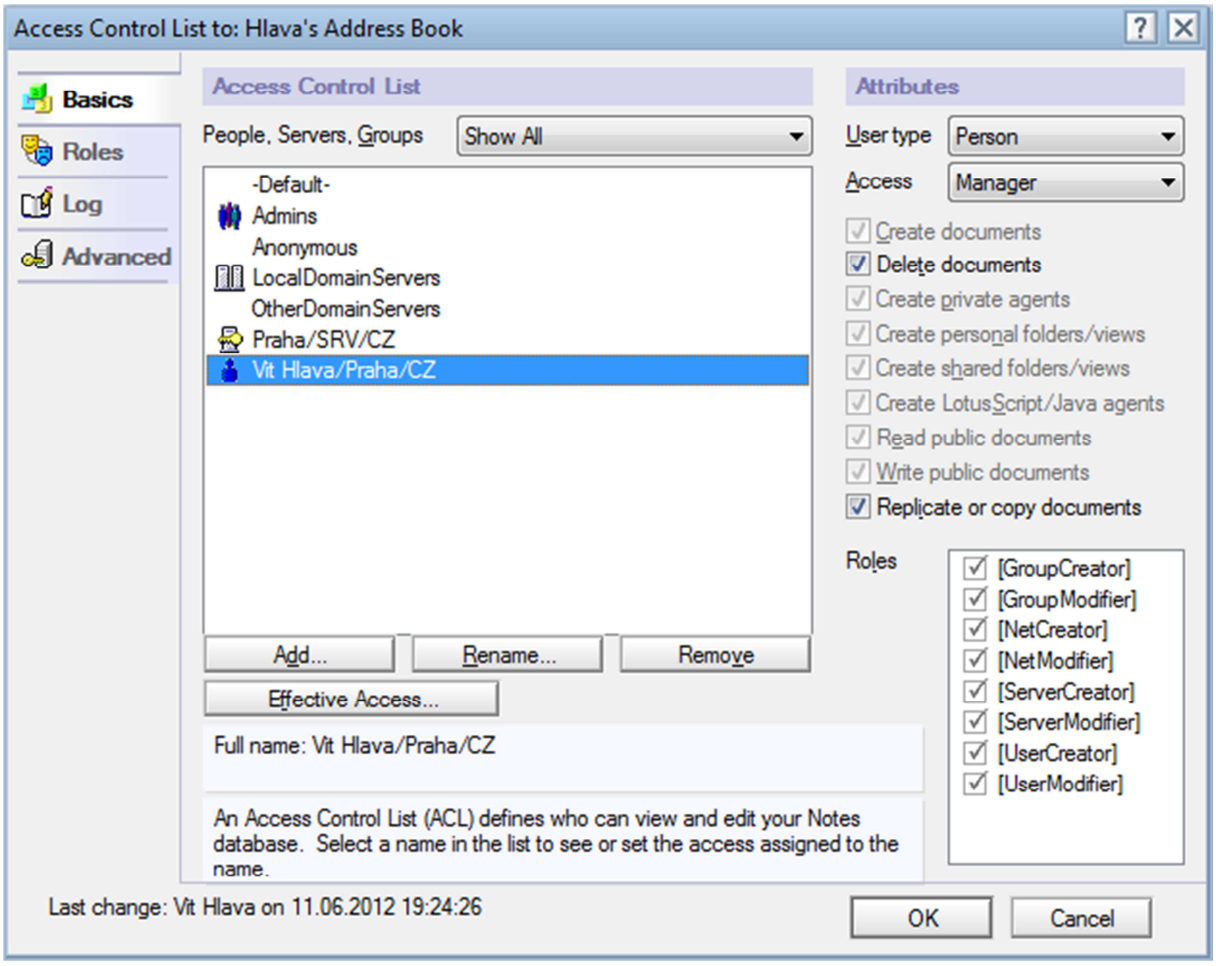
- No Access – bez přístupu k databázi
- Depositor – Uživatel může vytvořit dokument v databázi, ale nevidí ho. Příkladem může být ACL aplikace pro odchozí poštu mail.box
- Reader – Uživatel má právo číst dokument, ale nemůže je editovat.
- Author – Uživatel má právo pro čtení dokumentů a právo editovat dokumenty, ve kterých je uveden jakou autor.
- Editor – Právo číst a editovat dokument. Tato úroveň je nejvyšší jakou by měl standardní uživatel nastavenou.

Další dvě úrovně jsou doporučeny pro administrátory a vývojáře:

- Designer – Uživatel má právo měnit speciální dokumenty obsahující aplikační logiku. Tzv. design elementy.
- Manager – Plná práva na dokumenty, plus právo smazat celou databázi a nastavovat oprávnění.

Pro každou zvolenou úroveň práv je možné dodefinovat další privilegia. Zásadní pro většinu aplikací je privilegium pro vytváření dokumentů. Bez tohoto nastavení není uživatel v aplikaci schopen vytvořit nový dokument. Podle úrovně ostatních práv může pouze číst nebo editovat existující dokumenty.

V nastavení ACL se dělají často ještě větší chyby než v nastavení serveru. Výsledkem špatného nastavení jsou pak situace, kdy je běžný uživatel (bez dalších privilegií) schopen vymazat celou aplikaci ze serveru.



Obrázek 3.4 - ACL databáze (zdroj - Lotus Notes v 8.5.3)

3.3.4 Nastavení ACL

Nastavení ACL aplikace může provádět pouze uživatel, který má v ACL nastaven manažerský přístup nebo full administrátor, který nemusí být v nastavení vůbec uveden.

Doporučené nastavení je takové, aby manažerskou úroveň měl pouze administrátor serveru a samotný server, na kterém je aplikace umístěna. Server totiž nad databází provádí úkony, ke kterým jsou tyto práva zapotřebí (údržbu konzistence, indexaci pohledů, kompaktaci a další). Pokud se jedná o aplikaci, která se synchronizuje (replikuje) na více Domino serverů, pak je samozřejmě potřeba, aby i tyto další servery měli do aplikace patřičný přístup.

Pro vývojáře je vyhrazena úroveň designer, která jim umožňuje vytvářet návrh aplikace. Tj. vytvářet formuláře, pohledy, složky, skriptové knihovny, agenty atd. Nemohou tedy nastavovat oprávnění jiným uživatelům nebo si omylem smazat aplikaci. Vývojáři navíc standardně pracují na pomocných .ntf aplikacích, které potom slouží jako šablony pro jednotlivé pracovní aplikace. Do cílových aplikací již designer nemusí mít

vůbec žádný přístup. Výhoda je i v tom, že z jedné šablony je pak možno vytvořit a pravidelně aktualizovat návrh mnoha stejných aplikací (typicky mailové aplikace uživatelů pocházejí z jediné šablony). Případná úprava šablony se při vhodném nastavení distribuuje do všech aplikací, které mají nastaveno dědění (*inherit design from master template*) z této šablony.

Pro všechny standardní uživatele včetně nejvyššího generálního ředitele by měla být nastavena práva maximálně na úrovni editorské. Takový uživatel pak nemůže svým neodborným zásahem znehodnotit nebo smazat celou aplikaci sdílenou celou společností. Výjimka z tohoto doporučení může být obhajitelná pouze pro některé specifické databáze, např. pro e-mailovou databázi nebo soukromou adresní knihu uživatele.

3.3.5 Role

Dalším prvkem na pomezí administrace a vývoje aplikace jsou Role. Každý uživatel nebo skupina uživatelů, může mít přiřazeny tzv. Role, které vyjadřují jakou funkci v aplikaci uživatelé zastávají (např. [Schvalovatel]). Role je pak možné využívat i při návrhu aplikace. Vývojáři mají k dispozici nativní funkce na zjištění rolí aktuálního uživatele a na základě vlastnictví příslušné role pak mohou skrýt nebo naopak zobrazit určitá pole nebo tlačítka jednotlivým uživatelům. Např. uživatele s rolí [Schvalovatel] budou mít v aplikaci možnost vidět tlačítko na schválení dovolené. Ostatní budou naopak vidět tlačítko na odeslání žádosti ke schválení.

3.3.6 Výpočet práv pro uživatele

Pokud je uživatel definován v ACL jmenovitě, tzn. Pomocí hierarchického jména shodného s hierarchickým jménem certifikátu, pod, kterým je do Lotus Notes přihlášen, platí pro něj přesně takto definovaná práva i kdyby byl členem skupiny, která by v ACL aplikace byla zadána s jiným stupněm oprávnění. Explicitní zadání má tedy maximální přednost.

Situace je složitější pokud uživatel není explicitně zadán. Pak se přístupová práva vypočítají dle jeho členství v některé skupině, které jsou v ACL uvedeny. Pokud je uveden ve více skupinách, jeho práva se tzv. sčítají a dostává maximální úroveň oprávnění z obou nebo více skupin, ve kterých je členem. Zároveň má všechny role, přiřazené jednotlivým skupinám. Pokud uživatel není v ACL uveden vůbec vztahuje se pro něj

přístup, který je uveden v položce Default. V této položce by rozhodně neměl být nastaven přístup Manager.

3.3.7 Bezpečnost na úrovni jednotlivých dokumentů

Pro jednotlivé dokumenty v aplikacích (databázích) lze dodefinovat další zabezpečení. Jedná se o tzv. Autorská (Authors) a čtenářská (Readers) pole.

Readers pole

Pokud je na dokumentu uloženo pole typu Readers, vidí tento dokument v této aplikaci pouze uživatel, který má v ACL databáze nastavenu tuto hodnotu. Platné hodnoty v Readers a Authors polích jsou hierarchická jména uživatelů, role a jména skupin z adresní knihy – tato adresní kniha musí být nastavena tak, aby byla využívána pro kontrolu přístupových práv. Další možností je využívat zástupného znaku * pro hierarchická jména. Bude-li v Readers poli nastavena pouze *, uvidí tento dokument každý, bude-li tam hodnota */Firma/CZ, uvidí tento dokument každý, jehož hierarchické jméno končí /Firma/CZ.

Readers pole se vztahují na všechny uživatele aplikace včetně vývojářů a manažerů, pokud administrátor s manažerskou úrovní není v Readers poli, tento dokument neuvidí. Tímto mechanizmem je tedy možné zabránit dokonce i administrátorům, aby viděli obsah těchto dokumentů. Readers pole se nevztahují na Full Administrátora, uživatel s tímto oprávněním vidí všechny dokumenty nezávisle na nastavení ACL aplikace a obsahu Readers polí na jednotlivých dokumentech. Jediné, co neuvidí, je obsah zašifrovaných polí.

Při návrhu aplikací je potřeba pamatovat i na to, že použití čtenářských polí může mít určitý dopad na výkon aplikace. V případě, že aplikaci využívá více uživatelů, vidí každý uživatel jinou podmnožinu dokumentů v závislosti na svých právech. Při otevírání pohledu musí server tedy pro daného uživatele napočítat, které dokumenty vidí a které nikoli. Pokud je v databázi těchto dokumentů velké množství[5], může zobrazení dokumentů v pohledech chvíli trvat a kazit tak uživatelský dojem.

[5] V systému IBM Notes Domino znamená „mnoho“ řád statisíců a milionové databáze jsou již opravdu nevýkonné.

Authors pole

Pokud je na dokumentu uloženo pole typu Authors, může tento dokument upravovat pouze uživatel, jehož *hierarchické* jméno je uvedeno v tomto poli nebo je členem *skupiny*, která je v tomto poli nebo má nastavenu *Roli*, která je uložena v tomto poli.

Autorská pole se vztahují pouze na uživatele, kteří mají v ACL databáze nastavenou přístupovou úroveň Authors, pro uživatele s právy vyššími je tato hodnota irelevantní. Autorské pole zároveň garantuje i čtenářský přístup (v některých dokumentech je toto chybně uváděno.)

Autorská pole se s výhodou používají v aplikacích typu workflow, kdy je potřeba měnit oprávnění na dokumenty v jednotlivých stavech jeho života.

Slabá místa Readers a Authors polí

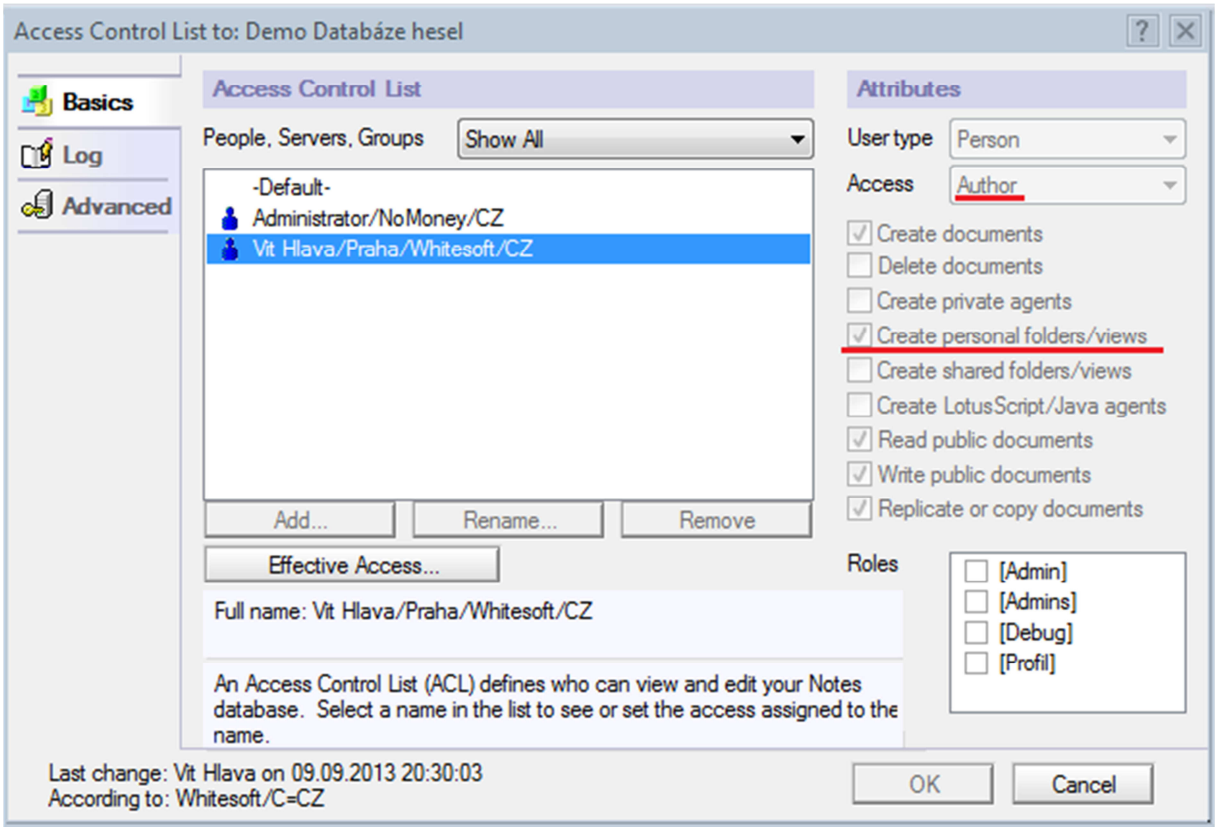
Je potřeba zdůraznit, že Readers a Authors pole nechrání dokumenty před uživatelem s Full Administrátorskými přístupovými právy. Samozřejmě, že zapnutí Full Administrátorské funkce se loguje a je možné jeho zapnutí notifikovat zodpovědným uživatelům např. SMS zprávou, lze nastavit auditování na otevřené dokumenty, je tedy možné dohledat jaká data a kdo četl, ale vše lze obejít jednoduchým trikem. Administrátor si zkopíruje databázi na svůj počítač a v off-line režimu se do databáze podívá, popřípadě si databázi přenese na jiný domino server, kde si nastaví práva jaká potřebuje. Dokonce je možné si na pomocném serveru zaregistrovat uživatele se stejným hierarchickým jménem a ukrást tak informace, aniž by se o tom kdokoli dozvěděl.

Odvození hodnot polí dokumentů bez oprávnění pro čtení

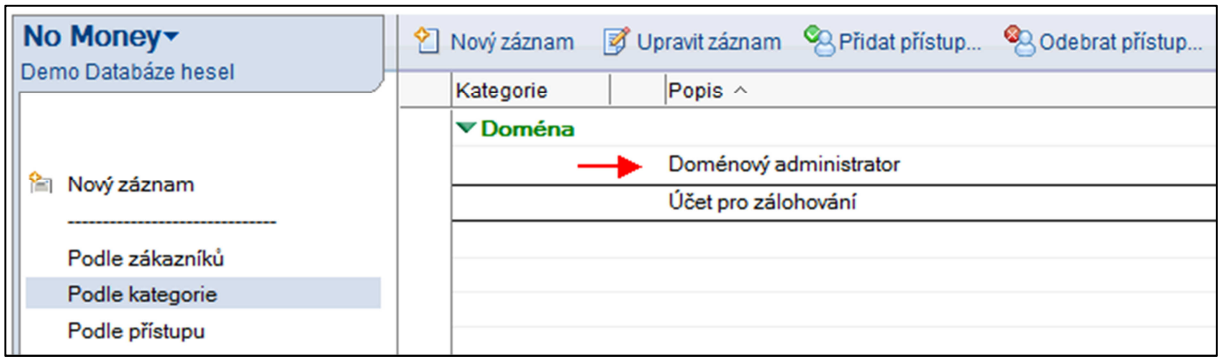
Další nedokonalostí je možnost částečného odvození hodnot polí na dokumentech, které nevidíme a to pomocí tzv. součtových sloupců v pohledech. Tyto sloupce umějí sečíst hodnoty daného pole pro jednotlivé kategorie dokumentů a sčítají je nezávisle na tom zda dokument uživatel vidí nebo ne. Pokud tedy budu mít jako uživatel právo vytvářet soukromé pohledy v aplikaci (obvyklé nastavení i pro úroveň Author), jsem schopen metodou vytvořit vhodně formátovaných pohled, ze kterého odvodím hodnoty na dokumentech, které nevidím. Obdobného výsledku lze použít pomocí kategorizovaných pohledů, kdy je kategorie (která je daná opět konkrétní hodnotou pole) vidět i když na žádný dokument v této kategorii nemá uživatel právo na čtení.

V následující jednoduché aplikaci pro správu hesel je ukázáno, jak je běžný uživatel s možností vytvářet soukromé pohledy schopen vyčíst heslo z dokumentu, ke kterému nemá vůbec přístup.

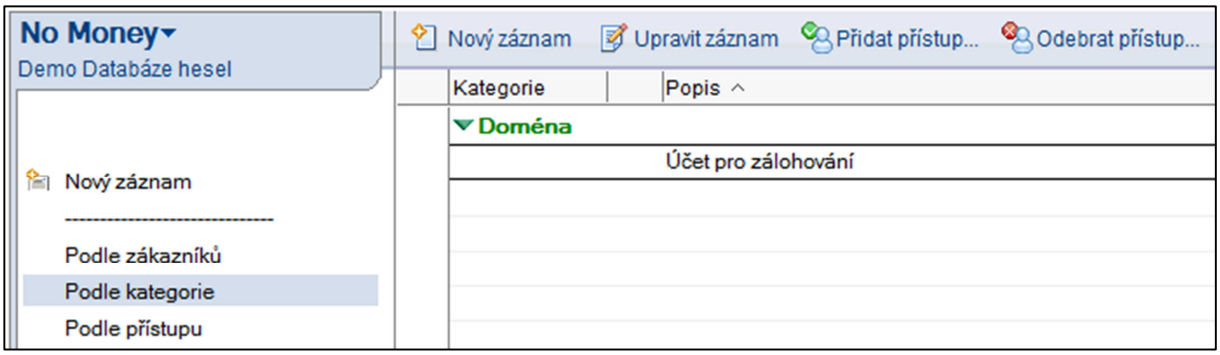
Při návrhu aplikace je vždy vhodné zhodnotit, jaká data je potřeba vidět v pohledech a zvážit, zde určité hodnoty (pole) dokumentu více nezabezpečit. Jednou z možností je kritická pole nastavit, tak aby nebyla typu *Summary*, tento typ polí pak v pohledech nelze zobrazovat vůbec nebo daná pole jednoduše zašifrovat.



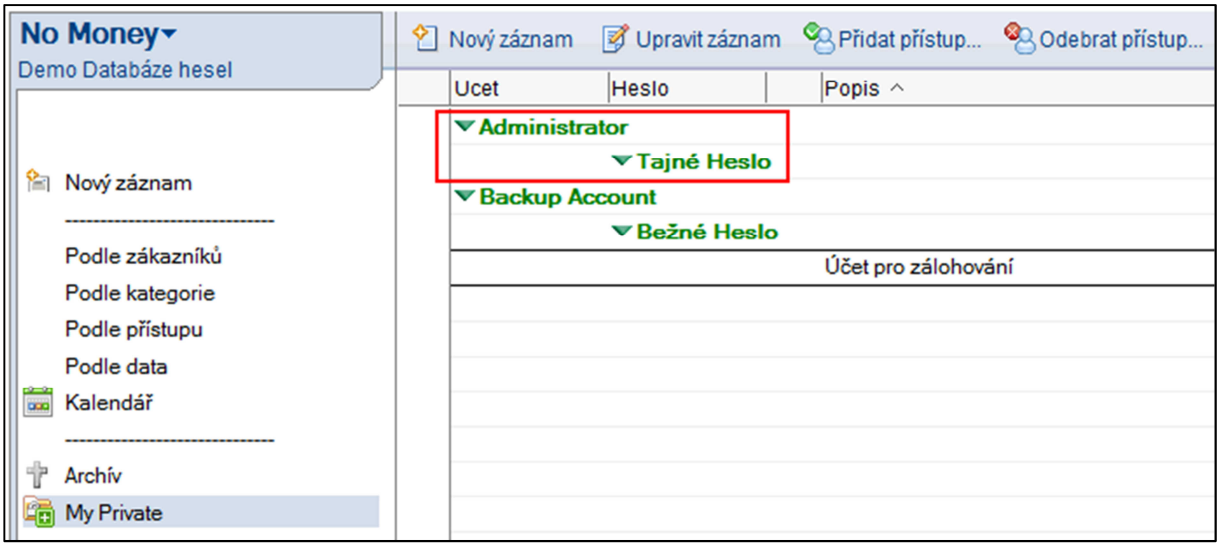
Obrázek 3.5 – Databáze hesel - Nastavení přístupových práv



Obrázek 3.6 - Databáze hesel z pohledu Administrátora



Obrázek 3.7 - Databáze hesel z pohledu běžného uživatele



Obrázek 3.8 - Databáze hesel - soukromý pohled běžného uživatele

3.4 Šifrování

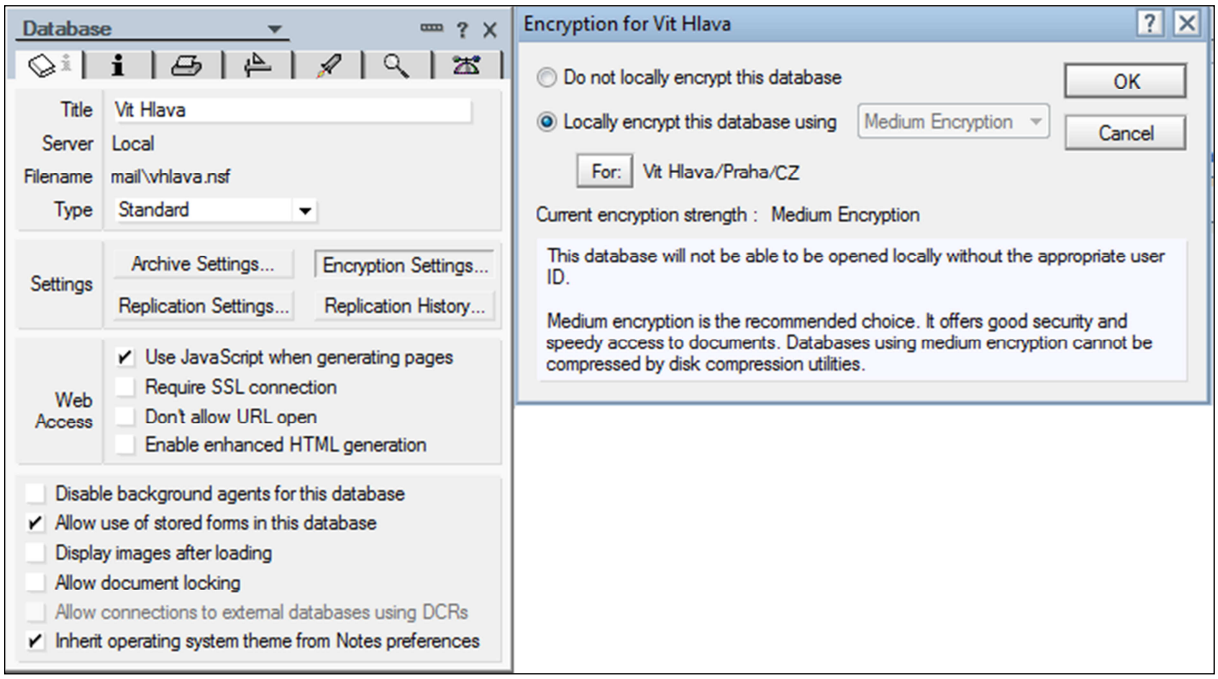
Častým požadavkem na aplikace schraňující citlivé údaje jako např. databáze hesel, smluv, osobních údajů apod. je možnost zabezpečit dokumenty nejen před cizími útočníky ale i před ostatními uživateli včetně administrátorů. V těchto situacích je nastavení práv pomocí Autorských a Readers polí nedostatečné a je nezbytné citlivé informace na dokumentech šifrovat. Systém IBM Notes nabízí možnost šifrování jednotlivých polí nebo celých dokumentů v aplikaci pomocí veřejného klíče z adresní knihy, popřípadě více veřejných klíčů nebo také pomocí klíče vygenerovaného nebo importovaného samotným uživatelem. Další možností je zašifrování celé aplikace certifikátem uživatele. Šifrování na úrovni aplikací uchrání data před cizím útočníkem ale nikoli před administrátorem, viz níže.

3.4.1 Šifrování na úrovni aplikací

Ve vlastnostech aplikace může oprávněný uživatel nastavit úroveň šifrování aplikace. Lotus Notes umožňují čtyři základní úrovně:

- Žádné šifrování
- Nízká úroveň šifrování
- Střední úroveň šifrování
- Vysoká úroveň šifrování

Jednotlivé úrovně se liší délkou klíče, který se k šifrování používá a tím pomocí kterého veřejného klíče (klíče kterého uživatele) se šifruje.



Obrázek 3.9 – Šifrování celé aplikace

Pokud si útočník zkopíruje zašifrovanou databázi, neudělá s ní nic. (dokud tedy nebude prolomeno šifrování založené na soukromých a veřejných klíčích). Každopádně pokud je útočníkem administrátor, který Vás před dvěma lety do systému IBM Notes registroval, stačí mu k prolomení vyhledat ve svých zálohách certifikát uživatele, pod kterým je aplikace zašifrovaná. Navíc od verze Lotus Domino R7 mají administrátoři k dispozici i nástroje na reset zapomenutých hesel a obnovu ztracených certifikátů, takže jim nic nebrání si pomocí těchto nástrojů nechat obnovit potřebný certifikát.

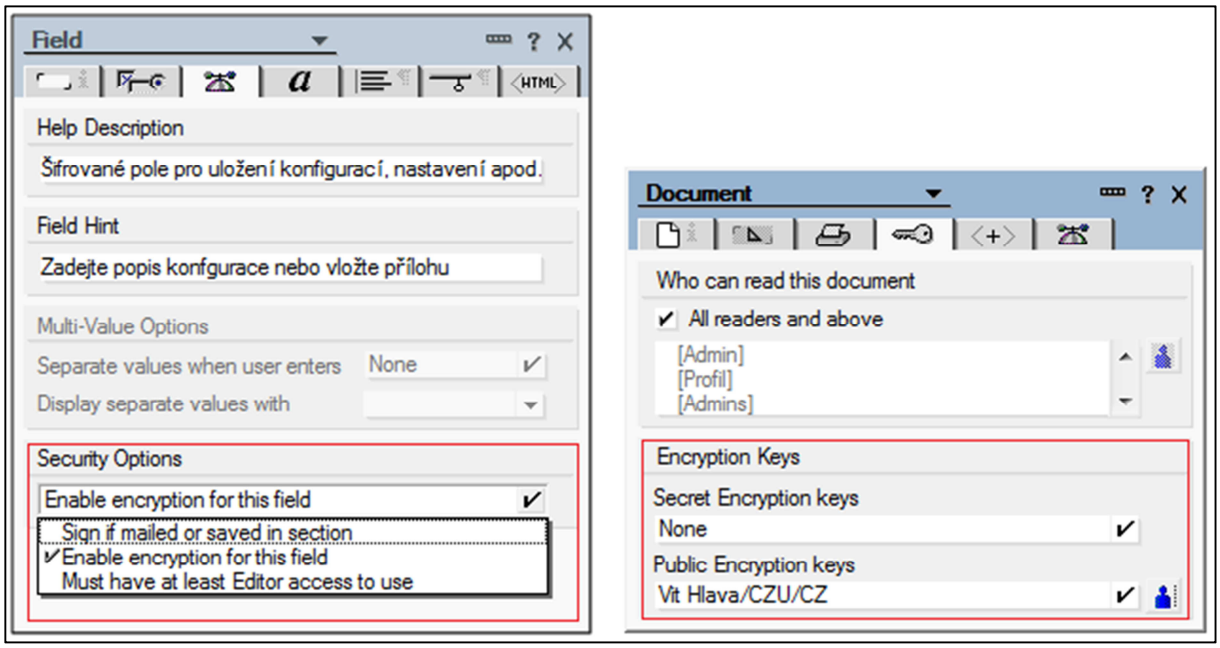
3.4.2 Šifrování na úrovni dokumentů

Šifrování na úrovni dokumentů je vhodné pro ochranu opravdu důležitých záznamů. Je zde možnost šifrovat pomocí veřejného klíče z adresní knihy nebo pomocí certifikátů vygenerovaných nezávisle na certifikátu uživatele vzniklým při registraci. Vzhledem k tomu, že v prvním případě stačí k prolomení obnovený uživatelský certifikát (ať už ze zálohy z *ID Recovery* ve verzi R7 nebo ID Vault od R8) je jedinou opravdovou ochranou šifrovat dokumenty pomocí klíče, který si uživatel vygeneruje sám. To může udělat každý uživatel v rámci svého Notes klienta. Klíč je pak uložen v lokální kopii certifikátu na uživatelském počítači, kterou je tedy nutné ochraňovat před odcizením, ale i před ztrátou. Protože je tato ochrana z hlediska IBM Notes absolutní, je se ztrátou klíče spojena i nevratná ztráta dat.

Ačkoli tuto ochranu IBM Notes poskytují, musí být pro její použití aplikace připravena. Tento mechanizmus musí být vývojáři do aplikace implementován, nejedná se tedy na rozdíl od šifrování na úrovni databáze o administrátorem nebo uživatelem nastavitelnou funkci.

Pro umožnění šifrování pole na dokumentu je zapotřebí povolit u příslušného pole v návrhu formuláře šifrování nastavení *Security Options* na hodnotu *Enable encryption for this field,* viz Obrázek 3.10. Tím zajistíme, že při každém uložení dokumentu je obsah tohoto pole zašifrován. Bez dalšího upřesnění ale nedefinujeme pomocí jakého klíče se bude šifrovat a proto bude dokument uložen v nezašifrované podobě. Aby vše proběhlo tak, jak má, je potřeba na dokumentu specifikovat jakým klíčem popřípadě klíči chceme data zašifrovat.

Jednodušší je použití veřejných klíčů uživatelů, které jsou součástí dokumentu uživatele v serverové adresní knize. Tím z hlediska bezpečnosti zajistíme, že obsah tohoto pole může přečíst pouze osoba, která má přístup k privátnímu klíči, který je uložen v uživatelském v ID souboru. Seznam osob z adresní knihy je možné nastavit pomocí Security záložky na vytvářeném dokumentu jednoduchým výběrem z adresní knihy nebo programově vyplněním systémového pole PublicEncryptionKeys. Pokud je v tomto poli vyplněno více osob, mohou dokument rozšifrovat všichni uvedení (tím se samozřejmě zvyšuje velikosti šifrovaného obsahu).



Obrázek 3.10 – Povolení šifrování pole, nastavení šifrovacího klíče na dokumentu

Druhou možností je šifrování pomocí jiného klíče, ke kterému nemusí mít přístup ani administrátor systému. Tento pomocný klíč je potřeba nejprve vygenerovat a rozeslat všem uživatelům, kteří mají mít možnost dokument rozšifrovat. Vygenerování a rozeslání šifrovacího klíče je možné provést přímo z prostředí IBM Notes klienta z nabídky: Soubor –> Zabezpečení –> Zabezpečení uživatele, viz Obrázek 3.11. Je také možné importovat externí šifrovací klíč.

Název šifrovacího klíče je potřeba obdobně jako v případě veřejných klíčů z adresní knihy nastavit na konkrétním dokumentu, popř. programově vložit do systémového pole SecretEncryptionKeys. Obě přechozí možnosti lze zkombinovat. Pak je pro přečtení dokumentu vyžadován jak privátní klíč z ID souboru tak i pomocný šifrovací klíč.

Obrázek 3.11 - Správa šifrovacích klíčů

3.5 IBM Connections

„IBM Connections je podniková platforma pro sociální software, která může vaší organizaci pomoci zapojit ty správné osoby, urychlit inovace a dospět k reálným výsledkům. Tato integrovaná a kvalitně zabezpečená platforma pomáhá k propojení se sítěmi odborníků v kontextu nejdůležitějších podnikových procesů.“ (IBM, 2013)

Jednotlivé aplikace tohoto systému jsou navrženy tak, aby na podnikové úrovni zlepšili spolupráci uživatelů. V rámci IBM Connections lze vytvářet dynamické skupiny uživatelů podle aktuálních potřeb a projektů a jednotlivým uživatelům těchto skupin umožňují snadné sdílení aktivit, dokumentace, znalostí a úkolů.

Hlavními přednostmi této platformy jsou:

- Dostupnost z mnoha typů různých zařízení od standardních počítačů po mobilní telefony.
- Dovoluje tematicky zpracovávat informace a využívat znalostí všech spolupracovníků a tím zvyšovat know-how podniku.
- Umožňuje integraci s celou řadou dalších produktů, které umožnuje efektivní práci uživatelů. Je možné upravovat vzhled, loga a tím se spolupodílet na podnikové kultuře.
- Jedná se o vysoce škálovatelné řešení, které lze využívat v malých firmách s několika málo uživateli ale lze přizpůsobit potřebám organizací se stovkami tisíc uživatelů.
- Produkt IBM Connections lze implementovat na vlastní infrastruktuře, v hybridní variantě nebo ve standardním Cloudu.

Jednotlivé aplikace, ze kterých se IBM Connections skládá jsou popsány v následujících kapitolách.

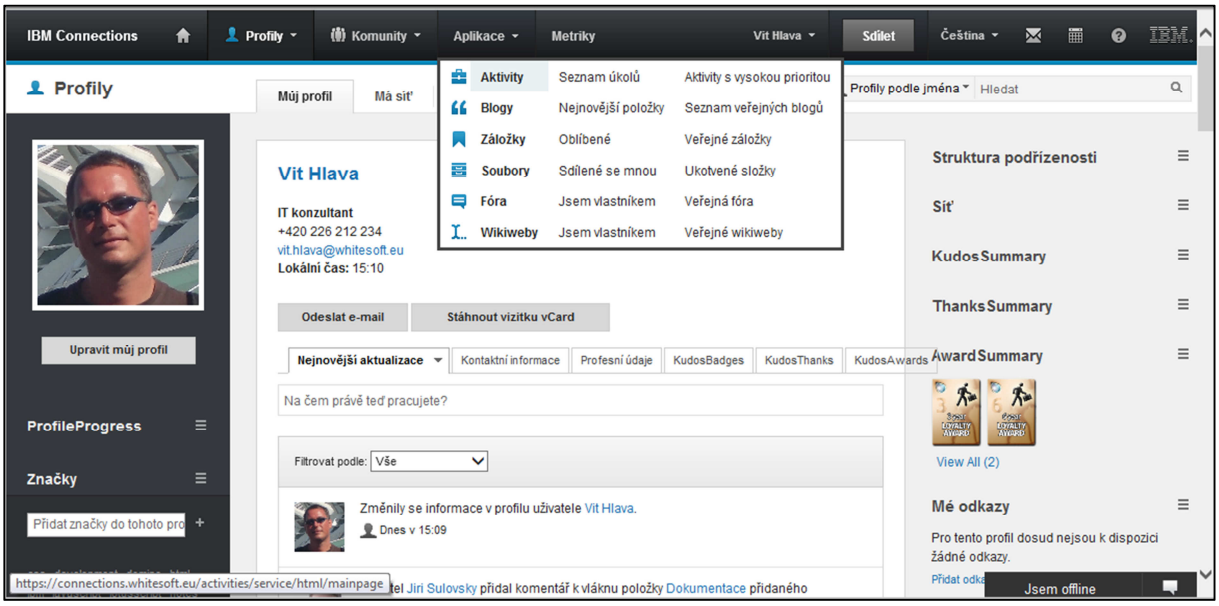
Activities

Tato aplikace je určena pro sdružování informací o probíhajících projektech. V rámci aktivit lze vytvářet a přiřazovat aktivity (úkoly) jednotlivým členům týmu. Vkládat potřebné podklady, tykající se daného projektu a diskutovat nad jednotlivými položkami.

Jednotlivé položky aplikace lze logicky rozdělit na záznamy (*Entries*), úkoly (*To-do items*) a sekce (*Sections*). Záznamy umožnují vkládat richtextový obsah např. zadávací dokumentaci daného projektu. Každý záznam je možné opatřit značkou (tag) a tím ji

zviditelnit případným zájemcům ve správném kontextu. Úkoly slouží pro plánování a přiřazování úloh jednotlivým řešitelům. Úkolům je možné navíc zadat prioritu, termín a další nastavitelné údaje. Sekce jsou pomocnou strukturou, která umožňuje přehledně slučovat úkoly a záznamy do menších logických celků.

Vytváření nových aktivit, lze usnadnit použitím tzv. šablon, tedy prázdných aktivit, které již mají připravenou strukturu a nastavení poplatné určitému typu aktivity. Tím je možné pro opakující se projekty využívat jedno výchozí nastavení.



Obrázek 3.12 - IBM Connections uživatelský profil

Blogs

Tato aplikace umožňuje jednotlivým uživatelům vytvářet osobní rich-textový obsah, který mohou sdílet s ostatními uživateli. Blog[6] může být jednoho ze tří základních typů, jež se od sebe neliší svou funkcionalitou ale tím, kdo může vytvářet nové záznamy a reagovat na ně:

- Uživatelský (*Personal*) blog spravovaný pouze svým vlastníkem.
- Komunitní (*Community*) blog vyhražený celé komunitě nebo
- Ideový (*Ideation*) blog věnované jedné myšlence, nápadu. Tento blog je vždy součástí určité komunity a slouží k rozvíjení daného tématu. Ostatní uživatelé mohou hlasovat a komentovat jednotlivé ideje. Idea, kterou administrátor

[6] Blog – složenina slov ze slov web a log, jedná se formu webového deníku sprovanou jednou osobou, firmou nebo komunitou.

vyhodnotí jako dostatečně podpořenou (hlasy nebo komentáři) může být převedena (*graduated*) do role nové aktivity.

Bookmarks

Tato jednoduchá aplikace poskytuje uživatelům možnost ukládat si odkazy na nejrůznější webový obsah mimo platformu IBM Connections. Záložky lze opatřit názvem a popisem, organizovat a dohledávat pomocí značek (tagů), komentovat a hodnotit jejich popularitu. Bookmark může být nastavený jako soukromý nebo veřejný a lze ho přidávat do aktivit, blogů a komunit.

Communities

Podobně jako v jiných intranetových aplikací lze v této aplikaci definovat virtuální týmy uvnitř organizace, které sdružují uživatele se stejnými zájmy. Každá komunita má svou webovou stránku s možností úprav vzhledu. V rámci komunity je možné vytvářet v podstatě všechny typy obsahu: Activities, Blogs, Bookmarks, Files, Wikies ale také další vnořené sub-komunity. Kromě toho je navíc možné vytvářet různé události např. setkání členů komunity.

Files

Součást IBM Connections, která slouží jako webové uložiště souborů. Soubory mohou být sdíleny v rámci komunit, popřípadě může být přístup omezen pouze pro zadané uživatele nebo úplně zveřejněn. Soubory je možné umísťovat do složek, verzovat a komentovat. Tento modul je možné užívat bez nutnosti pořízení licencí IBM Connections v rámci licence IBM Domino.

Forums

Aplikace, která slouží jako diskusní panel. Uživatelé zde mohou vytvářet vlákna s dotazy, myšlenkami, které mohou ostatní dále rozvíjet. Vytvářené téma může být nastaveno jako otázka, odpovědi pak může autor tématu upravovat a tímto způsobem vytvářet znalostní bázi (nejčastější dotazy). Vlastník tématu může kdykoli téma uzamknout a znemožnit tak vkládání dalších reakcí.

Profiles

Jedná se o adresář, který obsahuje základní informace o uživatelích IBM Connections. Uživatelé zde mohou vyplnit informace o své osobě, fotku, kontaktní informace, své

pracovní zkušenosti, aktualizují zde svůj stav a mohou spravovat síť svých kontaktů. Tento modul je podobně jako modul files možné využívat i s licencí IBM Domino.

Wikis

Tento modul je vhodný k tvorbě webových stránek věnovaných určitému tématu v rámci týmu nebo celé komunity. Při vytváření *wikiwebu* autor specifikuje název a popis webu a určí uživatele, kteří budou mít právo spravovat jeho obsah. Jednotlivý uživatelé mohou upravovat a přidávat další podřízené stránky.

4 Shrnutí použitých technologií

V níže popisovaném řešení bylo využito obecných technologií, které v této kapitole stručně shrneme.

4.1 LDAP

The Lightweight Directory Access Protocol (LDAP) je internetový protokol vytvoření k přístupu k distribuovaným adresářovým službám. Je specifikován v RFC 4511 (J. Sermersheim, 2006).

Adresářovou službou se rozumí systém, který spravuje a poskytuje informace o objektech nejrůznějšího typu (uživatel, počítač, datový zdroj…). Typicky se tyto informace velice často poskytují, ale málo kdy se mění.

Každý objekt je v adresáři identifikován jednoznačným rozlišovacím jménem (DN - Distinguished name) a pomocí tzv. atributů jsou uchovávány jeho specifické vlastnosti. Místo označení „Rozlišovací“ se v systéme IBM Notes setkáme s tzv. hierarchickým jménem (Např. CN=Vit Hlava/OU=Praha/C=CZ).

Protokol LDAP pro výměnu adresářových informací musí splňovat předepsanou specifikaci tzn. musí obsahovat všechny povinné elementy požadavků a odpovědí, nicméně je možné základní normu volitelně rozšiřovat pomocí dalších (např. norma RFC4513 specifikuje metody autentikace a bezpečnostní mechanizmy pro použití tohoto protokolu).

Pokud klientská aplikace potřebuje ověřit existenci uživatele daného systému, odešle adresářovému serveru pomocí tohoto protokolu požadavek obsahující rozlišovací (hierarchické) jméno daného objektu a čeká, až Server tento požadavek zpracuje a předá zpět požadovanou odpověď.

4.2 Protokol HTTP

Hypertext Transfer Protocol (HTTP) popsaný v RFC 2616 (Fielding, Gettys, Mogul, & at al, 1999) je základní protokol pro komunikaci webových aplikací, tedy způsob datového přenosu mezi dvěma popř. více systémy. Aktuální verze protokolu 1.1 více upřesňuje původní verzi 1.0 z roku 1990, která nabízela pouze jednoduchý přenos binárních dat, tak aby bylo dosaženo větší spolehlivosti při implementaci jeho funkcí.

Komunikace mezi dvěma systémy je zprostředkována pomocí předávaných datových zpráv. Rozlišujeme zprávy dvou typů:

- HTTP-message = Request (požadavek) a
- HTTP-message = Response (odpověď).

Oba typy se skládají ze startovacího řádku (Request line resp. Status line), hlavičky (Header), prázdného řádku (CRLF) a těla zprávy (message-body).

Request line

Identifikuje metodu, kterou má být požadavek zpracován, identifikací zdroje (tzv. URI – Uniform resource identificator, typicky http:// adresou cílového serveru včetně cesty k danému zdroji) a verzí HTTP protokolu (kvůli kompatibilitě verze 1.0 a 1.1.). Existuje skupina osmi základních metod OPTIONS, GET, HEAD, POST, PUT, DELETE, TRACE a CONNECT, kterou je možné dále rozšiřovat pomocí tzv. extension-methods. Mezi nejzákladnější a nejpoužívanější metody patří povinná metoda GET a metoda POST, které budou podrobněji popsány v následujícím textu.

Status line

Je prvním řádkem odpovědi a obsahuje verzi protokolu, třímístným status kódem a textovým popisem tohoto kódu. Status cody jsou rozděleny do pěti základních skupin:

- „1xx“ – Informativní kódy oznamující, že požadavek byl přijat a pokračuje zpracování.
- „2xx“ – Požadavek byl úspěšně doručen, pochopen a přijat.
- „3xx“ – Přesměrování, je potřeba podstoupit další akci pro dokončení požadavku.
- „4xx“ – Chyba klienta - neplatný požadavek: nesrozumitelný, obsahuje špatnou syntaxi nebo není na dané adrese požadovaný zdroj nalezen.
- „5xx“ – Chyba serveru – ačkoli byl požadavek validní nepodařilo se jej serveru požadovaným způsobem zpracovat.

Směrování komunikace

V nejjednodušším případě odesílá iniciátor spojení (*client*) požadavek na *server* definovaný pomocí URI v Request line, který na něj reaguje odesláním odpovědi. Ve složitějších případech může být požadavek i odpověď předávána pomocí prostředníků

(*intermediary*). Těmito prostředníky může být proxy, gateway nebo tunel a od sebe se liší mírou, kterou upravující původní zprávu. Proxy v podstatě vygeneruje celou zprávu znovu, gateway pracuje na vyšší úrovni a zprostředkovává zprávy serverovým protokolům, Tunel zprávu beze změny předává zprávu dál, aniž by se ji snažil interpretovat obsah přenášené zprávy. Celá implementace protokolu včetně cáchování a bezpečnosti je mnohem složitější a přesahuje rámec této práce.

GET

Je jednou ze základních metod HTTP protokolu. Umožňuje získat informace (data) pouze identifikací zdroje. Pokud zdroj odkazuje na akci generující výstupní data, pak jsou vrácena data nikoli zdrojový kód této akce.

Použitím speciálních výrazů If-Modified-Since, Is-Unmodified-Since, If-Match, If-None-Match nebo If-Range v hlavičce požadavku lze minimalizovat množství dat vracených v odpovědi. Nutným předpokladem je cáchování jednotlivých entit dat z předchozích požadavků, tedy i splněných všech požadavků na cachování. Dalším mechanizmem, kterým je možné optimalizovat nároky na přenos je tzv. částečný přenos, kdy je aktualizována pouze dosud nenačtená část dat.

POST

Je další významnou metodou HTTP protokolu, která naopak dává klientovi možnost odeslat koncovému serveru další data obsažená v těle zprávy. Pomocí metody Post lze:

- Odesílat blok dat (např. při odeslání vyplněného formuláře) procesu, který se stará o jejich zpracování.
- Aktualizovat data v databázích.
- Odesílat komentáře, příspěvky, články do různých content management systémů.

Zpracování Post požadavku závisí pouze na serveru (specifikovaným obvykle v Request-URI). Pokud není uvedeno v hlavičce dotazu jinak, pak se data vracená touto metodou necachují, popřípadě to lze umožnit přesměrováním na cachovatelný zdroj pomocí stavového kódu 303 (See Other).

Cookies

HTTP protokol je bez stavový, tzn. že server na každý požadavek od klienta odpovídá bez přihlédnutí ke kontextu možné předchozí komunikace. To může být pro mnoho webových aplikací velice limitující. Technologie cookies je rozšíření HTTP, které toto omezení pomáhá odstranit.

Standard RFC 2965 - HTTP State Management (D. Kristol, 2000) popisuje mechanizmus schopný přenášet stavové informace a udržovat informaci o konkrétní komunikaci. Kontext celé komunikace dvou účastníků označujeme jako *Session.* Základem jsou nové položky hlavičky HTTP zpráv: Cookie, Cookie2, Set-Cookie2.

Cílový server pomocí pole Set-Cookie2 hlavičky odpovědi vytvoří session. Hlavička Set-Cookie2 se skládá z klíčového slova Set-Cookie2 a po dvojtečce následují páry konkrétních cookie Nazev = Hodnota oddělěných čárkami, po posledním páru mohou následovat páry Atribut = Hodnota oddělené středníky, které upřesňují platnost, doménu, port, cestu, verzi atd. přenášených cookies.

Klient (typický webový browser) interpretuje hlavičku odpovědi a příslušné cookies si uloží odděleně pro každou doménu a port. Doména omezuje platnost cookie, není tudíž možné sdílet tutéž cookie mezi servery z více domén. Webový klient podle specifikace vždy předává v hlavičce svých HTTP požadavků všechny relevantní cookies (tedy cookies pro danou doménu), není-li dáno jinak. Server naopak cookies ve svých odpovědích posílat nemusí. Pokud klient očekávanou cookie v hlavičce nepošle (nechce nebo není tato funkcionalita povolena), pak cílový server může odmítnout další komunikaci, popřípadě klienta přesměrovat jinam např. zobrazit výzvu, aby si uživatel povolil ve svém prohlížeči použití cookies. Navázanou session může server kdykoli ukončit odesláním cookie atributu Max-Age = 0 v hlavičce odpovědi.

4.3 Webové služby

„Technologie webových služeb byla vyvinuta pro komunikaci mezi dvěma nezávislými, samostatně fungujícími počítačovými systémy. Inteligence a porozumění „živých“ uživatelů, kteří jsou schopni inteligentně klást i zodpovídat dotazy různých forem a tvarů, je nahrazena striktní definicí volání poskytované služby a pevně danou strukturou vstupních parametrů i výstupních dat“ (Hlava, 2011).

„V širším pojetí je webová služba libovolná internetová technologie poskytující služby nezávislé na operačním systému a programovacím jazyce a komunikující pomocí zpráv formátovaných ve standardizovaném XML tvaru“ (Ethan, 2002, s. 11).

Systém (zpravidla server) zprostředkovávající danou webovou službu nazýváme poskytovatelem (provider) a naopak systém využívající této služby označujeme jako konzumenta (může se jednat o klientský systém nebo další server). Poskytovatel musí být stále spuštěn a čeká na požadavky (zprávy) přicházející od konzumentů webové služby. Přicházející zprávy zpracuje a ve většině případů vrací konzumentům zprávu s výstupními daty popř. chybou.

Pro praktické využití chceme, aby webová služba měla určité minimální vlastnosti. Podle (Ethan, 2002) má webová služba být:

Sebepopisná

Kromě vlastní implementace metod je součástí webové služby XML dokumentace popisující seznam všech jejich veřejných metod, jejich argumentů a vrácených hodnot. Při návrhu systém, který bude službu využívat, je ve většině programovacích nástrojů možné vygenerovat pomocí této definice kostru funkcí automaticky.

Standardizovaná

Dostupná v internetové nebo intranetové síti pomocí standardních protokolů a komunikující pomocí standardizovaného na XML postaveného systému zpráv (SOAP).

Nezávislá

Nezávislá na platformě (operačním systému) a či jazyce, ve kterém je služba implementována. Nezávislost je posílena právě použitím obecně uznávaných a podporovaných standardů vyžadovaných pro definici samotné služby, přenos i obsah dat.

Vyhledatelná[7]

Byl navržen standardizovaný způsob, kterým se dostupné webové služby zveřejňují, tak aby případní konzumenti byli schopni vyhledat službu požadovaných vlastností, získat popis jejího rozhraní a adresu, na které je služba k dispozici.

4.3.1 XML

„Koncept webových služeb je vybudován na několika standardech založených na eXtensible Markup Language (XML). Jedná se o jazyk patřící stejně jako např. HTML do rodiny tzv. značkovacích jazyků. Dokumenty se skládají z vlastního textu, jenž je uzavírán mezi páry značek, které nesou informaci o významu uzavíraného textu nebo způsobu jeho zpracování. Obsah této kapitoly je převzatý z bakalářské práce autora (Hlava, 2011).

Element

Element je základním stavebním prvkem XML dokumentů a může být definován jako:

- Dvojice tagů – výrazy uzavřené ve špičatých závorkách obsahující jméno elementu např: otevírací tag `<dotazy>` a koncový tag `</dotazy>`. V rámci otevíracího tagu je možné definovat atributy ve tvaru jméno = “hodnota“. Vícenásobné atributy oddělujeme od sebe mezerou, např: `<dotazy dotaz_pocet="1" tazatel="Vít Hlava">`.
- Případně „prázdný” tag `<prazdny/>`, který nahrazuje zápis `<prazdny></prazdny>`.
- Nebo speciální výraz: např. komentář, `<!DOCTYPE ...>, atd.`

Element obsahuje seznam atributů nebo další element – potomka. Potomek je buď:

- element
- textová hodnota (sekvence Unicode znaků)
- komentář `<!-- Komentář -->`
- nebo jiný speciální výraz

[7] „Tento princip není dodržován, zvláště když konkrétní realizace webové služby slouží jako integrátor dvou interních podnikových systémů. Podnik tak ani nemá zájem o zveřejnění rozhraní této webové služby. Tento princip má svůj původ v obecnějších požadavcích SOA“ (Hlava, 2011).

„Z této rekurzivní definice vyplývá základní pravidlo o vnořování elementů. Jestliže jeden element začíná uvnitř jiného elementu, musí také uvnitř tohoto elementu skončit (elementy se nesmějí překrývat). Jinými slovy tvoří elementy dokumentu stromovou strukturu. Až na kořenový element má každý element právě jednoho rodiče a žádného nebo více potomků. U názvů elementů i atributů rozlišujeme velikost písmen[8], tzn. element `<dotazy>` je jiný než element `<DoTaZy>`“ (Hlava, 2011).

Názvy elementu nesmějí obsahovat mezeru, jsou tedy jednoslovné, povolené jsou alfanumerické znaky a kromě tečky, dvojtečky, pomlčky a podtržítka nesmí obsahovat žádné další speciální znaky.

Kompletní omezení ve jménech elementu je uvedeno ve specifikaci (W3C, 2006).

„Vzhledem ke speciálnímu významu určitých znaků bychom při pokusu o jejich použití uvnitř elementu porušili XML syntaxi. Např. při zápisu nerovnice 10<x by zdrojový text vypadal následovně: `<nerovnost>10<x</nerovnost>` a parser by při čtení takového dokumentu reportoval nepovolené znaky „`</`“ v názvu elementu `<x</nerovnost>`. Proto bylo potřeba definovat zástupné znaky (speciální entity). Nejdůležitější zástupky jsou v níže uvedené tabulce. Správný XML zápis uvedené nerovnice je tedy `<nerovnost>10&lt;x</nerovnost>`.“ (Hlava, 2011)

Tabulka 4.1 Zástupné znaky v XML

<	&lt;	“	&quot;
>	&gt;	‘	&apos;
&	&amp;		

XML dokument

„Dokument vždy obsahuje právě jeden kořenový (dokumentový) element. Zápis dokumentu začíná řádkem definujícím verzi, kódování či jiné vlastnosti dokumentu. Dále může obsahovat komentář a další speciální položky, např. odkaz na validační dokument a právě jeden dokumentový element. V našem případě element `<dotazy>`, viz Obrázek 4.1.

[8] Jedním z důvodů je náročný převod Unicode kódování na malá písmena nebo různé převody některých písmen v různých národních abecedách (ačkoli původně měl být „monocase“.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Komentář -->
<dotazy dotaz_pocet="1" vystup_format="XML">
        <dotaz>
                <hledany_text>Most</hledany_text>
        </dotaz>
</dotazy>
```

Obrázek 4.1 Příklad jednoduchého XML dokumentu.

Jmenné prostory (Namespaces)

Vzhledem k jednoduchosti definice vlastního jazyka založeného na XML vzniká velké množství seznamů jmen elementů a atributů. Pokud bychom v jednom dokumentu zkombinovali více schémat či jiných jazyků, mohlo by se snadno stát, že by byl element daného jména popsán v několika dokumentech a tím by vznikl konflikt při validaci takto pojmenovaných elementů.

Aby se zajistila jednoznačnost jmen všech elementů a atributů, přiřadí se lokální názvy do příslušných jmenných prostorů. Nejprve nadefinujeme jmenný prostor pomocí atributu xmlns kořenového elementu dokumentu. Syntaxe přiřazení je následující:

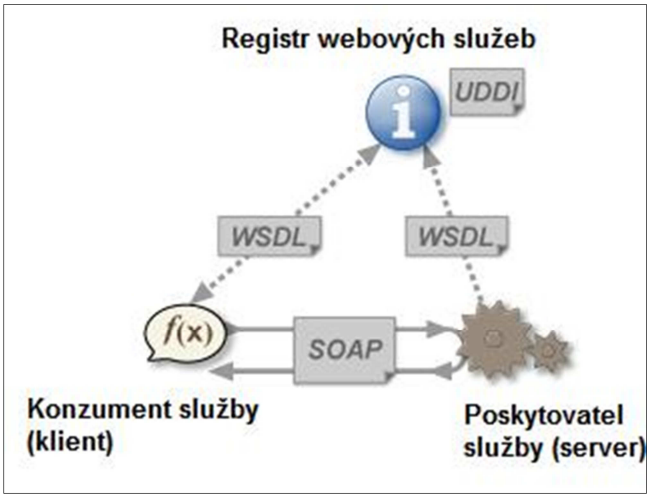
xmlns:jmeno_prostoru="http://pef.czu.cz/priklad". Názvy elementů a atributů patřících do kontextu tohoto jmenného prostoru opatříme prefixem jmeno_prostoru:název_elementu. URL tvary pro jména prostorů se používají, protože jsou jednoznačné, čímž je zabezpečena nezaměnitelnost názvů.[9]

Architektura webových služeb

Webové služby patří svou architekturou mezi tzv. *klient - server* aplikace. Viz **Chyba! Nenalezen zdroj odkazů.**. Serverová aplikace poskytuje své služby jednomu nebo několika klientům. Komunikace je zprostředkována výměnou zpráv založenou na protokolu SOAP.

[9] Ve skutečnosti tento URL nemusí ani existovat. Slušní programátoři na URL umísťují dokumentaci vztahující se k danému XML dokumentu.

Dalším subjektem v architektuře je centralizovaný registr webových služeb, v němž je k dispozici popis webových služeb, odkaz na popis rozhraní – WSDL a případná další omezení. Nicméně pokusy o centralizaci na vyšší než podnikové úrovni se v praxi neujaly.



Obrázek 4.2 Architektura webové služby, Zdroj: (Ethan, 2002)

Pomocí mechanizmů UDDI se poskytovatel webové služby při vzniku zaregistruje do registru a stává se tak dostupným pro konzumenty těchto služeb. Naopak potenciální konzument webové služby může v registru vyhledat vhodnou službu a získat síťovou adresu, na které je dostupná a definici jejího rozhraní.

Aplikační vrstva		
WS-Reliable messaging	WS-Security	**Kvalita, dostupnost**
WSDL	WS-Policy	**Popis služby (Description)**
SOAP	WS-Addressing	**Zprávy (Messaging)**
HTTP, SMTP, TCP/IP		**Transportní vrstva**

Obrázek 4.3 Vrstvy webové služby, zdroj (Weerawarana, Cerbera, & et al, 2005)

Z hlediska závislosti funkčních prvků můžeme webovou službu rozdělit do pěti základních vrstev: (Weerawarana, Cerbera, & et al, 2005). Viz **Chyba! Nenalezen zdroj odkazů.**.

Transportní vrstva (Transport)

Funkce na nejnižší úrovni zabezpečují transport zpráv mezi klientskou a serverovou aplikací. Přenos dat může probíhat na libovolném komunikačním protokolu. Nejčastěji se používá protokol HTTP (HTTPS).[10]

Definice zpráv (Messaging)

V této vrstvě se za pomocí W3C standardu SOAP definuje struktura zpráv posílaných mezi klientem a serverem a zároveň způsob adresování (doručení) zprávy na místo určení.

Popis služby (Description)

Na této úrovni se pomocí jazyka WSDL definuje samotná webová služba: seznam funkcí, parametrů a návratových hodnot. Patří sem i další specifikace webových služeb, které nejsou zahrnuty ve WSDL a popisují se pomocí politik (WS-Policy).

Kvalita služeb (Quality Of Service)

Popisuje další obecnější požadavky kladené na webovou službu, jimiž jsou bezpečnost a spolehlivost. Toto rozšíření není v aktuální verzi IBM Notes/Domino podporováno.

Aplikační vrstva

Představuje vlastní aplikaci či komponentu poskytující nebo využívající webovou službu. Obsahohuje logiku pro zpracování konkrétních funkcí. “ (Hlava, 2011)

4.3.2 SOAP

„SOAP je základním stavebním prvkem webových služeb. Je to prostředek pro výměnu zpráv mezi počítači přes síťové rozhraní. Je kompletně založený na XML a právě nezávislost na platformě a programovacích nástrojích byla rozhodující výhodou proti podobným technologiím (CORBA, DCOM) a důvodem jeho rychlého rozšíření.

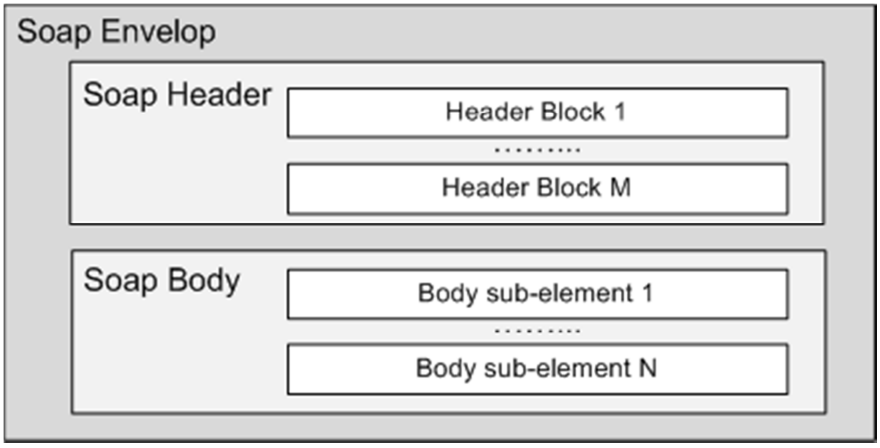
SOAP je definován jako souhrn pravidel pro tvoření správně formátovaných zpráv a jejich zpracování. Tato pravidla určují chování jednotlivých SOAP uzlů, přes které je zpráva doručována ke svému koncovému příjemci. Uzly jsou zařízení nebo aplikace, jež

[10] Ve verzi IBM Notes/Domino 9.0.1 nelze výběr protokolu ovlivnit. Podporován je pouze HTTP.

obsahují logiku potřebnou pro vytvoření, přijetí, zpracování nebo přeposlání SOAP zprávy.

Struktura SOAP zprávy

Struktura SOAP zprávy je složitější než u předchozího modelu, podporuje použití XML Schema, jmenné prostory a další doplňky jako WS-Addressing. Zpráva se skládá z obálky (Envelop), která obsahuje hlavičku (Header) a tělo (Body). Hlavička je rozdělena na bloky (nula, jeden nebo více), které obsahují informace o jednotlivých příjemcích zprávy: adresu příjemce, identifikátor zprávy, časovou značku a případně další informace dle implementace SOAP. Tělo obsahuje vlastní data, tj. volání funkce nebo XML dokument ke zpracování.



Obrázek 4.4 Struktura SOAP (Zdroj: Weerawarana, Cerbera, 2005)

SOAP podporuje dva způsoby předávání vlastních dat v těle zprávy. První způsob je předání samostatného XML dokumentu (document literal), příjemce dokument zpracuje a v odpovědi vrací nový XML dokument. Druhým způsobem je RPC (remote procedure call), kdy tělo zprávy, tedy element <env:Body>, obsahuje jediný vnořený element s názvem volané funkce a v něm parametry funkce (anlogicky jako v XML-RPC modelu). V těle odpovědi se pak vrací výsledek volané funkce. Ve webových službách preferujeme předávání celých dokumentů před programově zaměřeným voláním funkcí.

Komunikační vzory SOAP

SOAP je ze své podstaty bez stavový, jednosměrný systém předávání zpráv od odesílatele k příjemci. Tuto základní funkcionalitu lze rozšířit a docílit komunikace typu *požadavek – odpověď*, typu *odpověď* (analogicky HTTP GET request) nebo složitějších

komunikačních vzorů. Rozšíření je možné provést pomocí doplňků implementovaných v hlavičce SOAP zprávy (WS-Addressing), nebo využitím transportního protokolu, kterým je požadavek přenášen (např. použití transportního protokolu HTTP), nebo přímo na aplikační úrovni. Nejčastěji je přenos zprávy zabezpečen pomocí HTTP.

Přenos SOAP zprávy pomocí HTTP

Specifikace SOAP 1.2 dokumentuje dva způsoby využití HTTP k přenosu zpráv.

Prvním je vložení SOAP zprávy do těla HTTP POST requestu a výsledné SOAP zprávy do těla HTTP response. HTTP POST request a HTTP response jsou přenášeny za použití jediného TPC připojení a tedy přirozeně fungují jako komunikační vzor typu *požadavek – odpověď*. Jak vypadá struktura požadavku je znázorněno na obrázkuObrázek

```
POST /GetListofStreets HTTP/1.1
Host: nomoney.cz
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>  ...  </env:Header>
  <env:Body>  ...  </env:Body>
</env:Envelope>
```

Obrázek 4.5 Schéma HTTP POST požadavku

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset="utf-8"
Content-Length: nnnn
<?xml version='1.0' ?>
<env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope" >
  <env:Header>  ...  </env:Header>
  <env:Body>  ...  </env:Body>
</env:Envelope>
```

Obrázek 4.6 Schéma HTTP response

4.5.

Druhým specifikovaným způsobem je metoda volání GET, která odpovídá komunikačnímu vzoru *odpověď*. V hlavičce HTTP GET se uvede, že výsledek má být předán v těle HTTP response jako SOAP zpráva a to nastavením parametru Accept: application/soap+xml. Strukturu odpovědi ukazuje obrázek Obrázek 4.6.

Rozdíly mezi SOAP 1.1 a SOAP 1.2

SOAP verze 1.1 je popsána v jediném dokumentu, ve kterém je definován model zpracování SOAP zprávy, vzory pro volání funkcí RPC, vazba na HTTP protokol a kódování dat.

Specifikace Verze 1.2 je rozdělena do tří dokumentů. Nevyžaduje již serializovaný XML dokument, ale pracuje s obecnějším datovým modelem - XML Infoset. V první části je přehled standardu včetně popisu přenosu příloh. Ve druhé části je definováno vlastní zprácování zpráv a možná propojení SOAP zpráv s jinými transportními protokoly. Ve třetí části je specifikace volitelných rozšíření komunikačních vzorů včetně ukázky komunikačního vzoru typu *požadavek – odpověď* pomocí protokolu HTTP.

Vzhledem k rozšíření SOAP 1.1 se nepředpokládá převod aplikací kompatibilních s verzí 1.1 na verzi 1.2“ (Hlava, 2011).

4.3.3 WSDL

„Web Service Description Language - WSDL je skriptovací jazyk, který se používá k popisu webové služby, jejího volání, způsobu serializace dat a upřesnění dalších vlastností. Popisuje syntaxi a strukturu jednotlivých funkcí služby včetně struktury vstupních a výstupních parametrů jednotlivých funkcí, adresování služby (kde je služba dostupná). WSDL nepředepisuje vnitřní uspořádání a implementaci metod, programovací jazyk či platformu. Konzument webové služby je schopen pomocí WSDL dokumentu předepsaným způsobem komunikovat s poskytovatelem. Konzument není závislý na poskytovateli služby ve smyslu jeho vnitřní struktury. Všechny případné změny poskytovatele jsou pro něj transparentní, pokud nedojde ke změně WSDL dokumentu. Díky tomu může např. poskytovatel přejít na novější verzi nebo na úplně jinou platformu bez dopadu na funkčnost klienta“ (Weerawarana, Cerbera, & et al, 2005).

Struktura WSDL

Jedná se o značkovací jazyk založený na XML. To umožnuje validaci WSDL dokumentů pomocí standardních nástrojů např. DTD, XML Schema apod. Datové typy se většinou odvozují od definic datových typů z XSD. Pro potřeby budoucích nebo složitějších aplikací mohou být základní datové typy neomezeně rozšiřovány. Validace se doporučuje vždy když nevytváříme aplikaci, ve které je konzistence dat zajištěna jiným interním mechanizmem, nebo když může mít velký dopad na celkový výkon aplikace (např. jednoduchá webová služba používaná velkým počtem konzumentů).

„WSDL dokument se obecně skládá ze dvou logických částí uzavřených do kořenového elementu <definitions> :

- Abstraktní část– popisující seznam funkcí, které služba poskytuje. Tato část odpovídá na otázku: *„Co?“* služba poskytuje. (ve WSDL 1.1. je odpověď obsažena v elementech <type>, <message> a <portType> a ve WSDL 2.0 uvnitř tagu <interface> .)
- Konkrétní – popisující konkrétní způsob transportu zpráv, jejich formát a adresu webové služby. Tato část odpovídá na otázku: *„Jak?”* se lze k webové službě připojit, pomocí jakého protokolu a odpovídá elementu <binding>. Dále odpovídá na otázku: *„Kde?“*, na jaké adrese je služba dostupná (element <service> a <port> ve verzi 1.1. resp. <endpoint> ve verzi 1.2).

(Hlava, 2011).

Struktura WSDL 1.1

„Nejdůležitější elementy dokumentu jsou uvedeny na obr. 3.11. Kompletní popis lze nalézt v příloze A4 specifikace W3C (W3C, 2001).

```
<definitions>
        <types> ... </types>
        <message name="..."> ... </message>
        <portType name="..."> ... </portType>
        <binding name="..."> ... </binding>
        <service name="..."> ... </service>
</definitions>
```

Obrázek 4.7 Struktura WSDL 1.1

První element <types> definuje datové typy použité ve webové službě. Definují se zpravidla pomocí XML Schema, ačkoli lze použít i jiné technologie: DTD, Relax NG i technologie, jež umí popisovat dokumenty, které nejsou ve formátu XML.

Element <message> popisuje jednosměrné zprávy, jež si klient a poskytovatel služby vyměňují. Lze jej přirovnat k volání funkcí v běžných programovacích jazycích. Každá zpráva může obsahovat více částí pomocí tagu <part>, ve kterém lze dodefinovat i typ předávaných dat. Element <part> tedy odpovídá parametrům funkce.

Nejdůležitějším elementem sloužícím pro popis nabízené funkcionality je <portType>. Popisuje množinu operací, které webová služba dokáže provádět a zprávy, jež k tomu využívá. Je možné jej přirovnat ke knihovně funkcí nebo třídě ze standardních programovacích jazyků. Operace jsou ve WSDL 1.1 buď párové (*požadavek – odpověď*) nebo nepárové (jednosměrné, *one-way*). Obecně lze definovat směr libovolně, ale z pohledu poskytovatele služby nemá odchozí požadavek příliš velký smysl, pokud poskytovatel nemá informaci, kam požadavek směřovat. (Možnost, jak dodat informaci o směru nabízí např. rozšíření WS-Addressing.)

Konkrétní implementace přenosu dat pro každý jednotlivý port (portType) se popisuje elementem <binding>. Obsahuje potřebné informace o protokolu, kterým jsou zprávy přenášeny. Tento tag obsahuje dva atributy: *name* je jednoznačný identifikátor a *type* nabývá hodnoty atributu name elementu portu, k němuž se vztahuje. Specifikace WSDL 1.1 nepředepisuje konkrétní prostředek, kterým má být služba přenesena, ale naopak nabízí tři možné, nikoliv jediné způsoby: SOAP, HTTP a MIME včetně příkladů (Hlava, 2011).

Struktura WSDL 2.0

„Strukturu WSDL dokumentu popisuje specifikace konzorcia W3C rozdělená do dvou dokumentů. První popisuje vlastní jazyk WSDL 2.0 a druhý ukazuje jeho možná rozšíření. Ve specifikaci WSDL 2.0 je narozdíl od verze 1.1 použitý obecnější popis pomocí XML Infosetu. Specifikace definuje webovou službu jako soubor komponent, jež popisují jednotlivé její části. Komponenty jsou kolekce datových typů a vlastností reprezentující konkrétní části. Základní struktura WSDL dokumentu serializovaného do XML je znázorněna na obr. 3.12. Kořenový element <description> obsahuje povinně element definující datové typy <types> a nepovinně jeden nebo více elementů popisujících

vlastnosti služby <interface>, <binding> a <service>. Výsledný WSDL dokument lze validovat podle XSD Schema.“ (Hlava, 2011)

```
<description>
        <types> ... </types>
        <interface name="..."> ... </interface>
        <binding name="..."> ... </binding>
        <service name="..." interface="..."> ...
</service>
</description>
```

Obrázek 4.8 Struktura WSDL 2.0 (W3C, 2007)

4.4 Autentifikace a autorizace

Autentifikace[11] je základní úloha bezpečnosti informačních systému. Jedná se o proces, který zabezpečuje spolehlivou identifikaci uživatele vůči informačnímu systému. Při práci s počítačovým systémem ji lze nejjednodušeji realizovat pomocí znalosti hesla, další možností je identifikace pomocí vlastnictví např. tokenu, certifikátu apod., popřípadě kombinace obojího vlastnictví i znalosti hesla (Paseka, 2012).

Autorizace je ověření přístupu autentizovaného uživatele k chráněným prostředkům a v systémem IBM Notes a Domino bývá ošetřena právě seznamem oprávnění příslušející k serveru (nastavené na *security* záložce server dokumentu), oprávnění ke každé jednotlivé aplikaci (konfigurované v ACL) a oprávnění k jednotlivým dokumentům v aplikaci (nastavitelné pomocí Readers a Authors polí).

4.5 LTPA

Lightweight Third Party Authentication – jedná se o technologii autentifikace, kterou vyvinula firma IBM primárně pro své systémy IBM WebSphere a IBM Domino servery, aby umožnila webovým uživatelům přecházet mezi jednotlivými servery dané infrastruktury bez nutnosti opakované autentifikace – tzv. funkce jednoho přihlášení.

[11] Podle Ústavu pro jazyk česky je možné užívat termíny autentizace i autentifikace. Viz http://prirucka.ujc.cas.cz/?slovo=autentifikace&Hledej=Hledej.

Prostředí obsahující více serverů, které umožňuje uživateli přístup po jednom přihlášení se označují jako SSO (Single Sign-On). (IBM, 2013a)

LTPA protokol využívá pro předávání autentizačních údajů tzv. tokenů, tedy zašifrovaných řetězců, které obsahují hierarchické (unikátní) jméno uživatele, datum a čas přihlášení, exspirace, doménu a další údaje. Pro zvýšení bezpečnosti se nové verzi LTPA tokenu, která je kompatibilní se systémem IBM Connections a dostupná na IBM Domino serverech od verze 8.5 obsahuje také digitální podpis serveru, který token vydal. Před touto verzí byla struktura používaného tokenu podstatně jednodušší a poměrně snadno implementovatelná z externích systémů.

Tokeny jsou ve webových aplikacích ukládány v cookie webového prohlížeče a jako takové jsou odesílány na cílový server v rámci HTTP požadavku. Cílový server rozpozná přítomnost cookie a pokud ověří platnost tokenu, pak automaticky autentizuje i webového uživatele a daný HTTP požadavek provede.

4.6 Base64, Hash, Digitální podpis

4.6.1 Base64

Jedná se o formu zápisu dat pomocí omezeného výčtu znaků vhodnou k ukládání a přenosu binárních dat použitelnou v různých prostředích a různými systémy (Josefsson, 2006). Existující aplikace a informační systémy pracují s daty v nejrůznějších formátech poplatných požadavkům doby a účelu, ke kterému vznikaly. Neumějí například pracovat s více bytovými znaky nebo nemají podporu kódování znaků jiných národních abeced apod. Z důvodu interoperability je pak vhodné data těmto aplikacím předávat v upravené podobě.

Jedna z nejznámějších implementací Base64 je ve standardu MIME, který rozšiřuje základní možnosti internetové pošty, například umožňuje vkládání příloh a digitálního podpisu.

V případě Base64 je základnou 64 vybraných ASCII znaků (tzv. „tisknutelné“ ASCII znaky) a jeden pomocný znak „=“, který slouží k dokončení výsledného řetězce v případě, že počet bytů vstupních dat nebyl dělitelný třemi.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/
```

Výpis 1 – základna ASCII znaků formátu Base64

Kódování do Base64 formátu probíhá tak, že binární data na vstupu jsou rozdělena do skupin po 24 bitech (3 * 8 bitů = 3 byty). Každá skupina je pak zpracována zvlášť, 24 bitů je rozděleno po 6 bitech a každé šestici odpovídá právě jeden znak ze základny ($2^6 = 64$). Převedeme-li 6-místné binární číslo do desítkové soustavy udává nám výsledek pořadí znaku v základně, viz Výpis 1, počítáno od nuly. Každá skupina je tímto způsobem převedena na čtveřici znaků ze základny. Drobná komplikace nastává, pokud poslední skupina není 24 bitová. Pak zbylý počet bitů zprava zarovnán nulami, tak abychom dostali 6-místné binární číslo. Pokud je takto doplňovaná skupina pouze 8-bitová, přidávají se nakonec dva speciální znaky ==, pokud je 16-ti bitová, přidává se jeden speciální znak = a je-li úplná, nepřidává se nic.

Např. v kódování UTF-8 je zápis české zkratky ČZU binárně uložen takto:

Tabulka 2 – Binární reprezentace znaků Č, Z, U v UTF-8 kódování.

Znak	**UTF-8 (hex.)**	**UTF-8 (binárně)**	**Unicode**
Č	C4 8C	11000100 10001100	U+010C
Z	5A	01011010	U+005A
U	55	01010101	U+0055

Znak Č je v tomto případě kódován dvěma byty. Těchto celkem 32 bitů zapíšeme za sebe a rozdělíme po 24 bitech:

11000100 10001100 01011010 01010101

Vzhledem k tomu, že poslední skupina je pouze 8-mi bitová, budeme k výslednému řetězci znaků ze základny přidávat dva speciální znaky ==.

Přeskládáme do šestic, poslední neúplnou šestici doplníme zprava čtyřmi nulami:

110001 001000 110001 011010 010101 010000

Převedeme do decimálních čísel pro určení pořadí jednotlivých znaků v základně:

49. znak = „x“, 8. = „I“, 49. = „x“, 26. = „a“, 21. = „V“ a 16. znak = „Q“.

Výsledný převod slova *ČZU* do base64 je tedy: *xIxaVQ*==.

Pokud byla zkratka ČZU na vstupu binárně uložena pomocí jiného kódování např. Windows-1252, pak by base64 obraz díky jiné binární reprezentaci vyšel zcela jinak a to WIU=. U textových dat je tedy velmi důležité vědět, které výchozí kódování bylo použito.

K Base64 kódování existují i další alternativy, které ovšem nejsou tolik rozšířené. Base32 redukuje základnu na 32 znaků. Motivem bylo vypuštění některých znaků, například „1“ (číslo jedna) a „l“ (písmeno malé el) nebo „0“ (nula) a „O“ (písmeno velké O), které mohou opticky splývat, zvláště při použití některých hůře čitelných fontů.

4.6.2 SHA-1

Jedná se matematickou funkci, která převádí vstupní data na 160 bitovou hodnotu (FIPS, Information Technology Laboratory, 2012). Typicky se vyjadřuje pomocí 40 znaků dlouhého hexadecimálního čísla. Výsledné hodnota tedy vyjadřuje otisk vstupních dat a většinou se označuje slovem *Hash* (česky haš). Funkční předpis je sestrojen tak, aby i malá změna vstupních dat vedla k výrazně odlišnému haši. Porovnáním relativně krátkých otisku pak můžeme usuzovat zda i vstupní data byla stejná či odlišná. Vzhledem ke konečné délce haše však vždy půjde nalézt různá vstupní data, která budou dávat shodný haš.

Standard algoritmu SHA-1 byl vytvořen Národním institutem standardů a technologií (NIST[12], USA) pro Národní bezpečnostní agenturu (NSA) v roce 1995 na základě méně bezpečného algoritmu SHA-0. Od roku 2005 je i tento algoritmus považován z hlediska bezpečnosti za nedostatečný a doporučuje se využití modernějších algoritmů SHA-2 resp. od roku 2010 SHA-3. Vzhledem k tomu, že hašovací algoritmus SHA-1 je široce rozšířen v mnoha aplikacích, je a k méně bezpečnostně náchylným účelům bude i v budoucnu nadále hojně využíván.

Přesnému vysvětlení algoritmu by vyžadovalo poměrně dobrou znalost modulární aritmetiky sahající nad rámec této práce, bude popsána pouze základní myšlenka:

Vstupní data se zprava doplní bitovou jedničkou a dále 0-512 nulami, tak aby výsledná délka vstupu byla dávala po celočíselném dělení 512 zbytek 448. Nakonec se přidá délka originální délka vstupních dat zapsaná pomocí 64 bitů. Tím vznikne pole bitů o délce dělitelné 512. Toto pole lze bezezbytku rozdělit do bloků po 512 bitech. Každý z bloků se pak pomocí předepsaných logických operací dále zpracovává a vrací pět 32-bitových mezivýsledků. Těchto pět mezivýsledků se využije pro zpracování dalšího bloku

[12] National Institute of Standards and Technology

a po zpracování posledního 512 bitového bloku se těchto pět mezivýsledku vypíše za sebe a tím vznikne požadovaný 160 bitů dlouhý výstup.

Př. SHA-1 otisk zkratky „ČZU“ je v hexadecimálním zápisu roven následujícímu řetězci:

```
Sha1("ČZU") = "AD28F362 A112E63A 1135A285 E08B82B9 EB988CE0"
```

Výpis 2 – SHA-1 otisk řetězce „ČZU“

Stejně jako při výpočtu Base64 závisí na tom, jak jsou vstupní data kódována. V ukázkovém případě byla kódována v UTF-8.

4.6.3 Digitální podpis

Ne vždy je nutné utajit a šifrovat celý dokument. Někdy je potřeba, aby dokument zůstal přístupný, ale byl nezpochybnitelný jeho autor viz s.152 (IBM, 1997). Digitální nebo elektronický podpis se používá k ověření původu dokumentů. Z tohoto pohledu lze chápat jako matematický ekvivalent ručně psaného podpisu. Algoritmus digitálního podpisu funguje tak, že se nejprve vypočítá otisk dokumentu (hash). Tento haš je pak zašifrován pomocí privátního klíče odesílatele - autora. Příjemce dokumentu nejprve spočte znovu otisk původního dokumentu a pak pomocí veřejného klíče autora dešifruje podpis a získá původní haš, porovnáním obou otisků usuzuje na pravost. Zde je ukryt zádrhel, protože ačkoli se nám oba otisky rovnají, nikde není zaručeno, že nám veřejný klíč byl poskytnut opravdu daným autorem a nikoli podstrčen podvodníkem. Pravost veřejného klíče je ověřována pomocí tzv. certifikačních autorit, které zaručí, že daný veřejný klíč byl konkrétnímu uživateli vydán po řádné identifikaci. Stejnou úvahou bychom zpochybnili i samotnou certifikační autoritu, proto jsou zavedeny tzv. kořenové certifikáty, tj. seznam ověřených certifikátů, které se již dále nezpochybňují. Seznam kořenových certifikátů je součástí operačního systému popř. jiné aplikace, takže je nutné dbát i na bezpečnost vlastního počítače.

Algoritmy pro generování digitálních podpisů jsou podobně jako SHA hašovací algoritmy spravovány americkým NIST. Mezi povolené algoritmy patří základní tři:

- DSA (The Digital Signature algorithm),
- RSA (Rivest-Shamir-Adleman algorithm)
- ECDSA (The Elliptic Curve Digital Signature algorithm).

Podrobnější popis fungování digitálních podpisů je nad rámec této práce a je popsaný ve standardu FIPS PUB 186-4 (Information Technology Laboratory, 2013).

4.6.4 SPNEGO

Jedná se o zkrácený název protokolu firmy Microsoft známého jako *Simple and Protected GSS-API Negotiation Mechanism*. Používá se při sdílení přihlašovacích údajů uživatele operačního systému Windows s webovými servery. Tento protokol definuje způsob, kterým jsou z uživatelské stanice důvěryhodně (ověřitelně) odeslány autentizační údaje webovému server. Pokud cílový server má implementovanou podporu tohoto protokolu může uživatele autentizovat aniž by vyžadoval přihlášení pomocí hesla nebo certifikátu. (IBM, Understanding SPNEGO, 2012)

Tento mechanizmus bohužel na IBM Domino server na rozdíl od WebSphere Application serveru není podporován.

5 Single Sign-On

Single Sing-On (SSO) je mechanizmus, kterým je webovým uživatelům umožněno přecházet mezi různými webovými aplikacemi na jednom nebo fyzicky více serverech bez nutnosti opakovaného zadání hesla. Čím vyšší je počet různých portálových a cloud systémů v organizaci, tím silnější je ze strany uživatelů tlak na implementaci SSO.

5.1 Popis řešeného problému

Uživatel se autentizuje svým jménem a heslem do hlavní webové aplikace (Portál), která není postavena na IBM technologiích, tedy nelze využít nativní podpory SSO. V Portálu jsou integrovány odkazy na aplikace, které jsou umístěny na IBM serverech, příkladem může být například webový mailový klient iNotes na serveru Domino nebo wiki web umístěný na IBM Connections serveru. Pokud uživatel klikne na odkaz je přesměrován na daný server, který ho okamžitě vyzve k autentikaci.

V této práci jsem se zabýval problémem, jak uživatele přihlášeného v hlavní webové aplikaci automaticky autentizovat i vůči systémům IBM s využitím jejich nativní podpory jednotného přihlášením mezi systémy IBM.

5.1.1 Popis nativního řešení IBM

Funkce jednoho přihlášení je založená na technologii LTPA (viz kapitola 4.5), tedy na použití šifrovaného tokenu, kterým je uživatel schopen prokázat svou identitu na všech serverech podporujících LTPA ve stejné doméně.

Aby byl LTPA token přijat jednotlivými servery bez ohledu na to, který server byl použit pro přihlášení a tento token vygeneroval, je nutné nejprve na všechny servery distribuovat stejný klíč, kterým se token generuje a také nastavit stejnou metodu generování klíče. LTPA token předávaný pomocí cookie je pak rozpoznán každým takto nakonfigurovaným serverem a uživatel je automaticky autentizován.

Základní předpoklady úspěšného použití SSO:

- Všechny servery musí být ve stejné doméně. (vychází z platnosti cookie)
- Všechny servery musí mít povolné SSO ve stejné verzi a musí používat stejnou adresářovou službu pro autentikaci uživatelů.

- Čas na jednotlivých serverech musí být synchronizován, kvůli konzistenci doby exspirace generovaného tokenu.
- Webový prohlížeč uživatele musí mít povoleno použití cookies.

5.1.2 Propojení IBM Domino a IBM Connections

Domino server neumí exportovat LTPA klíče ve tvaru, který je možné importovat do WebSphere serveru. Proto je potřeba začít s konfigurací SSO nejprve na WebSphere serveru, vyexportovat LTPA klíč a naimportovat ho na server Domino.

Pokud je potřeba propojit více WebSphere serverů je navíc potřeba označit jeden z nich jako tzv. *Master*, na něm vygenerovat LTPA klíč. Pokud chceme spojit servery různých verzí je nutné kvůli kompatibilitě jako master server vybrat ten nejstarší. (IBM, 2013a)

Vygenerovaný klíč je pak potřeba importovat do ostatních WebSphere serverů a také do serveru IBM Domino.

Nejprve je ale nutné nakonfigurovat jednotnou adresářovou službu pro všechny účastnící se servery. K tomu využijeme vlastnosti serveru Domino publikovat svou adresní knihu pomocí protokolu LDAP.

Adresní kniha – nastavení LDAP

IBM Domino server nativně podporuje LDAP protokol (Tuttle, Steven; Godbole, Kedar; McCarthy, Grant;, 2004). Navíc umožňuje skládání více adresních knih z různých zdrojů proto je Domino server vhodný kandidát pro centrálního poskytovatele adresářové služby v celé infrastruktuře.

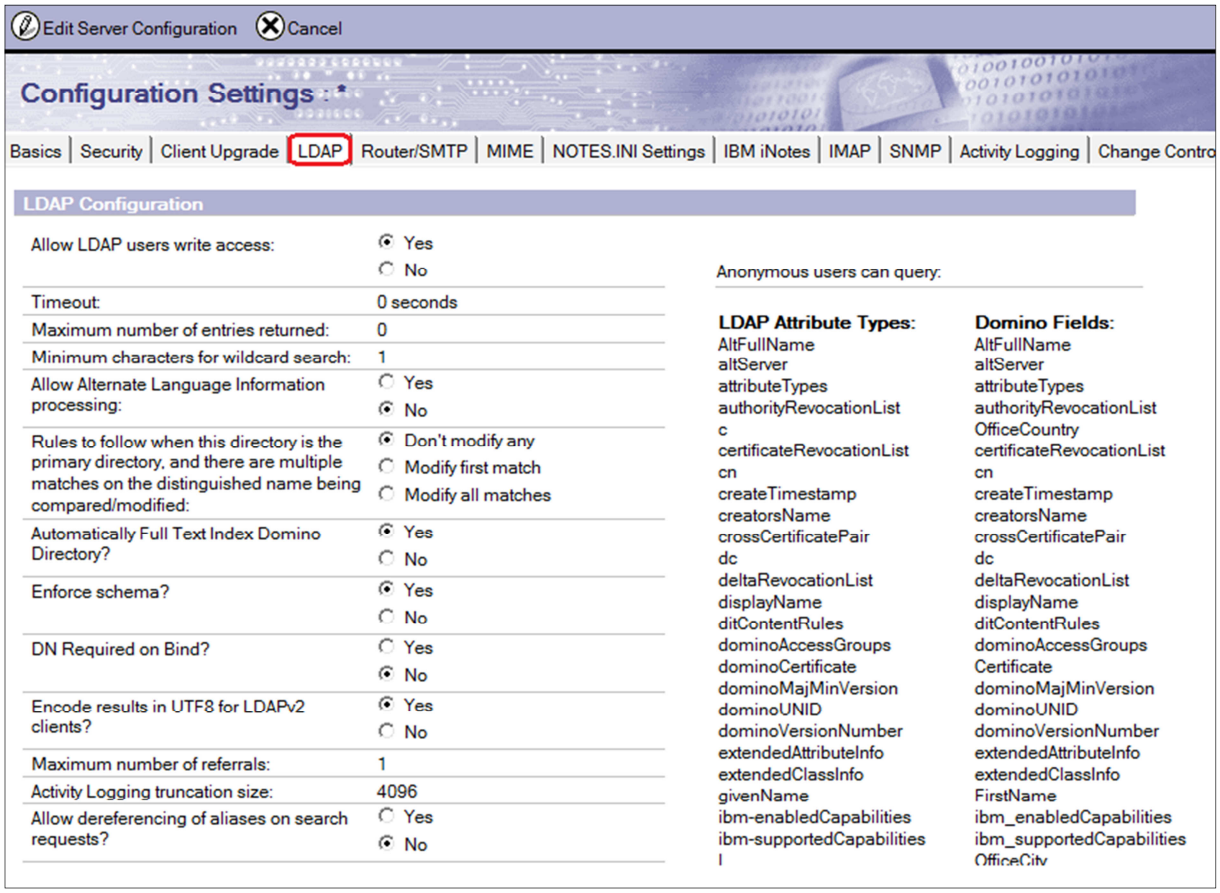
Využitím adresářové služby Domino server je externí systém schopen (kromě jiného) ověřovat existenci uživatelského jména, jednoznačnosti e-mailové adresy i autentizovat uživatele po zadání hesla. Kromě toho je v naplánovaných intervalech synchronizovat adresní knihu z Domino serveru a tím např. na IBM Connections serveru automaticky zakládat uživatelské profily pro nové uživatele.

Podobně jako LDAP na Domino serveru je možné vytvořit adresářovou službu i na WebSphere serveru s IBM Connections. Popřípadě lze využít dalších nástrojů např. IBM Tivoli Directory Intergrator k vytvoření externího LDAP serveru. Pokud však měla firma

nasazený systém IBM Domino dříve než WebSphere, tak je přirozené a nejsnazší využití právě Domino serveru[13] a stejně tomu tak bylo i při řešení SSO v rámci této práce.

Spuštění LDAP na Domino serveru je velice triviální, stačí spustit příslušnou službu (LDAP task). LDAP začne být okamžitě k dispozici ve svém výchozím nastavení (, které ve většině případů zcela postačuje).

Pokud je nutné výchozí nastavení jakkoli upravit, je potřeba vytvořit konfigurační dokument, viz Obrázek 5.1. Zde lze definovat, které položky jsou dostupné pro anonymní dotazy, nastavit logování, indexování a mnoho dalšího. Změna portu, na kterém je LDAP dostupný je možné měnit v nastavení TCP/IP protokolu na příslušném server dokumentu (IBM, 2011).



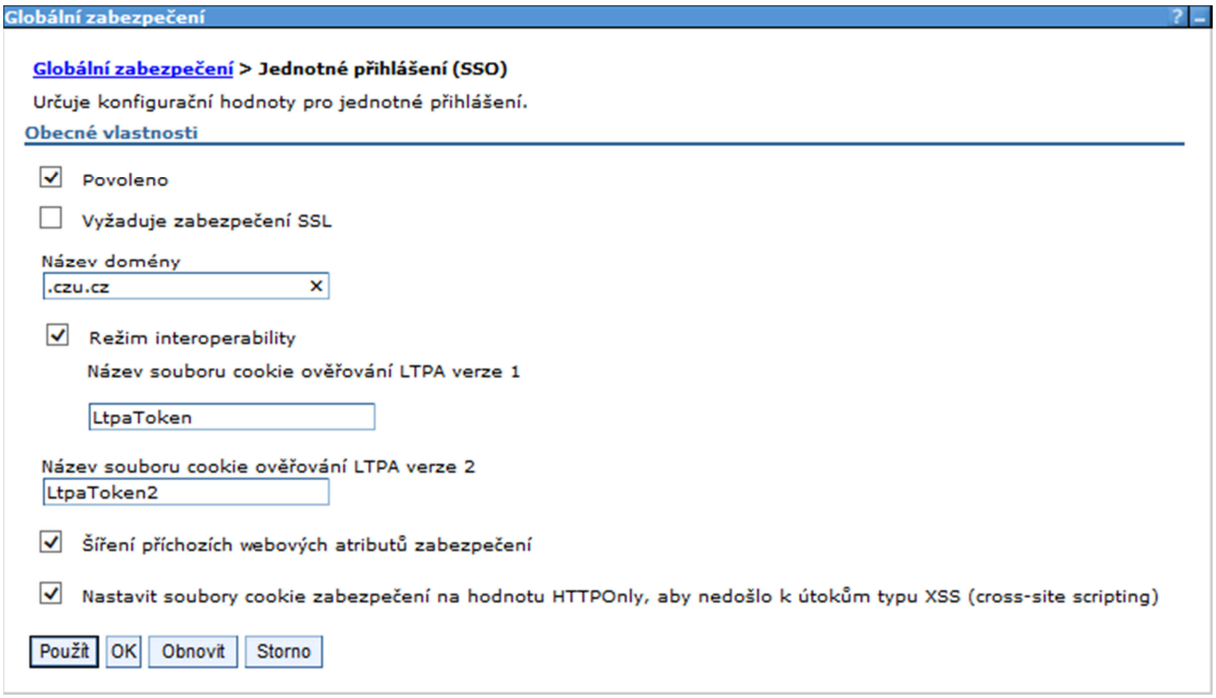
Obrázek 5.1 - Konfigurace LDAP na Domino serveru

[13] Ve všech implementacích, se kterými se autor dosud setkal, byl LDAP nakonfigurovaný na serveru Domino.

Konfigurace IBM Connections SSO pro Domino servery

Kompletní konfigurace funkce jednoho přihlášení je poměrně složitá a mezi jednotlivými verzemi se liší. V této kapitole proto představíme pouze nejdůležitější body. Kompletní návod pro jednotlivé verze je k dispozici v dokumentaci tohoto produktu (IBM, 2013a)[14].

- Nastavení LDAP – jednotného adresáře společného pro všechny servery. (záložka Security - Global Security - Federated Repositories)
- Povolení SSO a konfigurace doménového jména (záložka Security - Global Security – Autentication mechanism and expiration). Zde je důležité zapnout tzv. Interoperability mode, viz Obrázek 5.2.
- Export LTPA klíče (Security - Global Security – Autentication – LTPA). Při exportu je potřeba zadat heslo, které chrání exportovaný klíč.



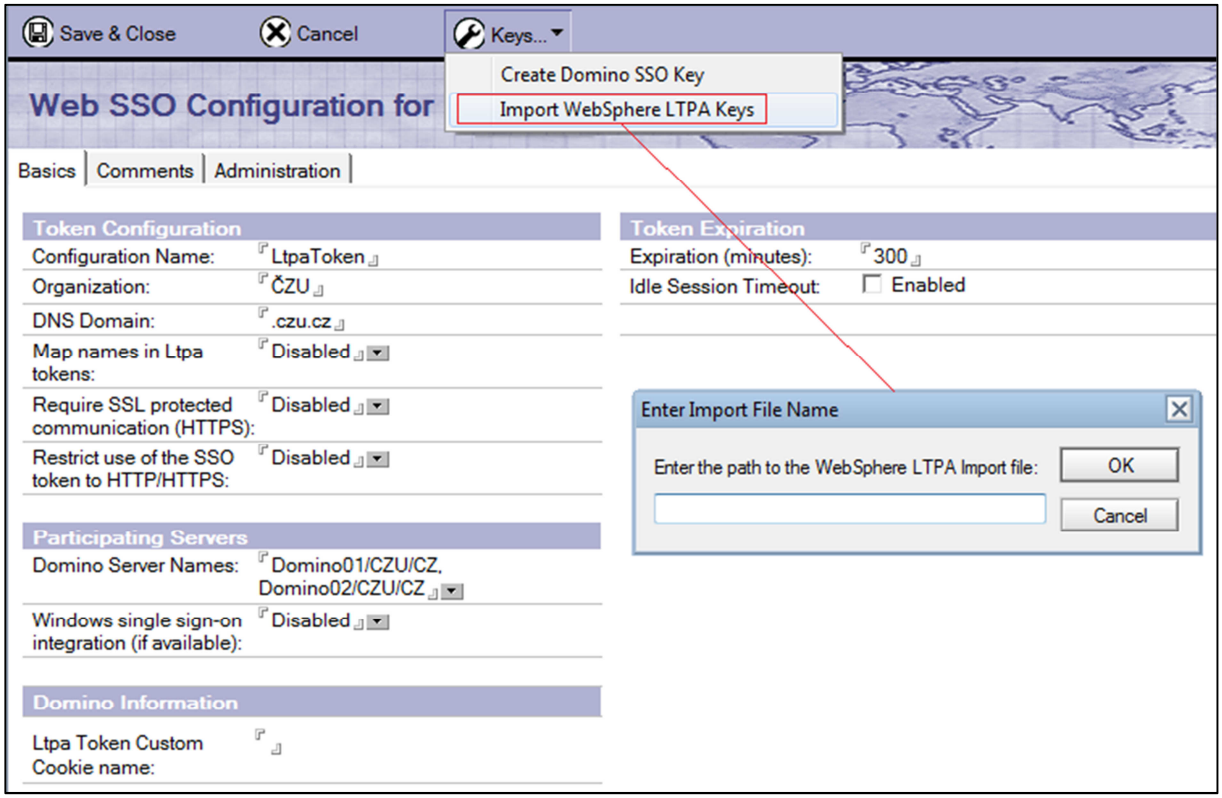
Obrázek 5.2 - IBM Connections, konfigurace SSO

Pokud již nepotřebujeme SSO povolit na dalších WebSphere serverech, je dalším krokem nastavení SSO konfigurace na serveru Domino.

[14] Product documentation - IBM Connection 4.5 Documentation, (aktuální verze 9, 2013)

Konfigurace SSO na Domino serveru

- Nejprve je potřeba připravit *Web Server Configuration* dokument v serverové adresní knize, ve kterém se definuje mezi kterými servery a jakým způsobem bude povolena funkce jednoho přihlášení.
- Dalším krokem je import připraveného LTPA klíče z Websphere serveru, viz Obrázek 5.3.
- Na Server dokumentu každého serveru, který bude dostupný pomocí SSO, na záložce Internet Protocols – Domino Web Engine, je nutné zapnout tento typ autentizace a zadat jednoznačný identifikátor konfiguračního dokumentu.
- Posledním krokem je restart webového serveru Domina (restart http task).



Obrázek 5.3 - Konfigurace SSO na Domino serveru

5.1.3 Generování tokenu

O vygenerování LtpaTokenu se stará systém, do kterého se uživatel přihlašuje jako první. Uživatel se autentizuje vůči tomuto systému standardním přihlášením pomocí uživatelského jména a hesla (případně jiným způsobem – identifikační karta, certifikátem).

Po úspěšné autentizaci vygeneruje server podle svého nastavení token a ten předá zpět klientské aplikaci. Klientská aplikace disponující tímto tokenem již příště místo uživatelského jména a hesla předá tento token a nemusí uživatele obtěžovat opakovaným logováním.

Pokud se tedy uživatel příště připojí k serveru a předá mu token, server reverzním postupem token ověří, zjistí jméno uživatel, zda existuje v adresní knize, ověří haš a podpis a také datum vytvoření a exspiraci. Pokud vše odpovídá, je uživatel autentizován a pokračuje ve své práci.

Typy LTPA tokenů

Obecně token obsahuje zašifrované informace o uživateli, čas exspirace a podpis vydavatele. V průběhu času vzniklo několik typů tokenů, které se liší svým obsahem a úrovní zabezpečení. (IBM, 2009)

Domino LTPA - je původní verze využívaná již na Domino serverech verze 5.5. Tento token obsahuje hlavičku konstantní délky, která určuje verzi tokenu, dále čas vytvoření a exspirace tokenu, uživatelské jméno a autentifikační kód (*Message authentication code*), Autentifikační kód je vytvořen jako SHA-1 otisk hlavičky, obou času a uživatelského jména a LTPA klíče. Takto zkonstruovaný token je nakonec zakódován pomocí Base64.

WebSphere Version 1 (LTPA1) – Tento typ byl používán na WebSphere aplikačních serverech až do verze 6.1. LTPA1 token obsahuje čas exspirace, uživatelské jméno (resp. LDAP Distinguished name, viz kapitola 4.1) a digitální podpis. Podpis může navíc obsahovat další uživatelská data. Podpis je vytvářen pomocí SHA-1 hašovací funkce a RSA šifrování.

WebSphere Version 2 (LTPA2) – Výchozí typ pro WebSphere servery od verze 6.1. Struktura tohoto tokenu je podobná verzi LTPA1, navíc může obsahovat další údaje o autentizovaném uživateli (údaje o vydavateli tokenu, exspirace, SPNEGO status a další).

Vzhledem k tomu, že WebSphere server nepodporuje Domino tokeny, je potřeba pro úplnou funkčnost SSO využít jednoho z WebSphere tokenů.

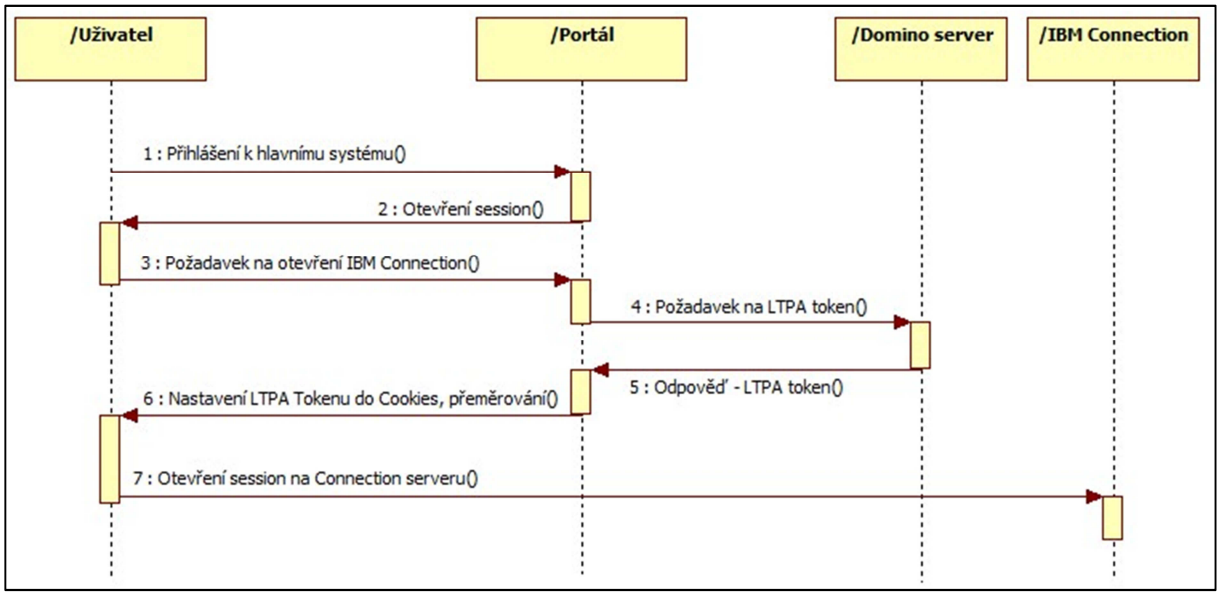
5.2 Implementace podpory SSO na externím systému

Vzhledem k tomu, že výše popsaný SSO mechanizmus je podporován pouze na IBM infrastruktuře, nastává problém v okamžiku, kdy stejný postup chceme použít pro jiný systém. V naší situaci, kdy je externí webová aplikace tzv. primární (tj. uživatelé se do ní

logují jako první), je v ní nutné implementovat mechanizmus, který uživatelům dokáže vygenerovat LTPA token ve tvaru, který je podporován systémy IBM[15].

Řešení navržené v rámci této práce spočívá ve vytvoření webové služby umístěné na Domino serveru, která se stará o vygenerování platného tokenu na základě požadavku externího systému, viz kapitola 5.2.1.

Při otevření odkazu aplikace systému IBM Domino nebo aplikace IBM Connections proběhne na pozadí komunikace primární aplikace s Domino serverem, pomocí webové služby je vyžádán LTPA token, který je ve formě cookie předán webovému prohlížeči uživatele a následně je uživatel přesměrován na požadovaný server. Pokud IBM server ověří platnost tokenu umožní uživateli přístup bez nutnosti opětovného přihlášení. Jednoduchý příklad průběhu komunikace je zobrazen na sekvenčním diagramu, viz Obrázek 5.4.



Obrázek 5.4 - Zprostředkování autentizace externím systémem, zdroj Autor

Řešení vyžaduje, aby na serveru poskytujícím webovou službu byl spuštěn webový server, což není vlastně žádným omezením, protože právě web server zprostředkovává webový přístup k požadovaným aplikacím, nicméně má jeden omezující důsledek a to, aby byl Domino server přístupný i anonymním uživatelům. Důvodem tohoto nastavení je problém s autentizací webové služby při použití HTTPS v kombinaci s povoleným SSO.

[15] V Systémech postavených na WebSphere serveru (IBM Connections) lze využít i jiné SSO algoritmy např. SPNEGO. Na Domino serveru lze využít pouze technologie LTPA.

Povolením anonymního přístupu k serveru vystavujeme všechny aplikace publikované na tomto serveru hrozbě neoprávněného přístupu. Je potřeba nastavit ACL všech kritických aplikací, tak aby do nich anonymní uživatel neměl žádný přístup.

Další problém nastává, pokud externí systém neumí spolupracovat s LDAP adresářem a tudíž nemá k dispozici shodnou adresní knihu. Autentifikace pomocí LTAP tokenu funguje na Domino serverech výhradně, jen když je token vytvářen z primárního hierarchického jména[16].

K vyřešení druhého problému je nutný předpoklad, aby externí systém znal hierarchické jméno autentizovaného uživatele, což se v řešeném případě ukázalo jako poměrně náročné a to hlavně díky tomu, že uživatelé mohli vznikat obou systémech IBM i externím nezávisle na sobě. Navíc v tomto konkrétním případě není žádoucí, aby všem uživatelům externího webového portálů byl umožněn přístup na aplikace umístěné na Domino serveru. Dále popisované řešení není obecně použitelné a bude se lišit při každé další implementaci SSO, je uvedeno pouze pro úplnost řešení.

Synchronizace adresní knihy externího systému

Spolehnout se na správné vyplnění hierarchického jména uživatelem při prvním pokusu o připojení se ukázalo jako nepřípustné z hlediska ochrany dat. Nakonec se přistoupilo ke kompromisnímu řešení, které spočívalo v importu hierarchických jmen z adresáře Domino serveru do adresáře externí webové aplikace a k implementaci pomocné webové služby. Vstupními parametry webové služby jsou unikátní internetová adresa, jméno uživatele, další osobní údaje uživatele, časové razítko a bezpečnostní otisk. Webová služba pak vrací:

- Hierarchické jméno uživatele zaregistrovaného do systému IBM Domino pomocí vstupních parametrů
- Hierarchické jméno již existujícího uživatele (shodu vyhodnocuje dle jména a unikátní e-mailové adresy)

---

[16] Na Domino serveru je možné mít hierarchické jméno v aliasech vícekrát. Například variantu bez diakritiky, pokud je ovšem použit k ověření uživatele LDAP je nutné používat pouze primární hierarchické jméno, ze kterého se vytváří distinguished name.

- Chybovou hlášku, že uživatel definovaný v požadavku nemohl být vytvořen, protože se nepodařilo prokázat unikátnost záznamu.

Vzhledem k tomu, že unikátní jméno nemůže být uživateli v externím systému měněno, je zajištěna ochrana vůči úmyslnému zadání cizí adresy za účelem přístupu k datům umístěných na Domino serveru.

Součástí kompromisního řešení je i dohoda, že všichni uživatelé využívající oba systémy se budou primárně registrovat v externím systému a všechny případně další administrátorské úkony (přejmenování uživatele, smazání uživatele apod.) budou řešeny manuálně správcem Domino serveru.

5.2.1 Generování tokenu pomocí webové služby

Na rozdíl od struktury Domino LTPA tokenu nebyla přesná struktura WebSphere tokenů oficiálně zveřejněna a navíc může být slovy IBM „kdykoli změněna“. Proto není možné generovat validní LTPA token vlastními prostředky přímo externím systémem.[17]

Navržené řešení spočívá v tom, že si externí systém pomocí webové služby vyžádá validní token přímo z Domino serveru. Domino server jako poskytovatel této webové služby využije svých interních funkcí, znalosti certifikátu, uživatelského jména a nastavení SSO pro vygenerování platného tokenu.

Webová služba má pouze jeden vstupní parametr typu *String*, který v zakódované podobně obsahuje datum vytvoření, datum exspirace, uživatelské jméno a otisk vytvořený hašováním těchto informací spojených se symetrickým klíčem. Webová služba na serveru nejprve dekomponuje jednotlivé parametry ze vstupní proměnné, pak ověří platnost požadavku, existenci uživatelského jména a vlastním výpočtem otisku zkontroluje zda souhlasí symetrický klíč.

Následuje generování LTPA tokenu pomocí volání funkcí z vnitřní knihovny. Pokud ověření i generování tokenu proběhne úspěšně vrací webová služba vygenerovaný token, v opačném případě vrací chybovou hlášku.

---

[17] Pokud neuvažujeme nelegální možnost reverzního programování zdrojového kódu IBM Domino serveru nebo přilinkování zdrojových knihoven, které se starají o generování tokenů, k aplikačnímu serveru externího systému.

Vývojáři externí aplikace musí ovšem počítat také s variantou, že webová služba nevrátí v požadovaném čase žádnou hodnotu kvůli nedostupnosti cílového Domino serveru, nebo že místo očekávané odpovědi přijde HTTP Response se stavovým řádkem *500 - Internal Server Error*.

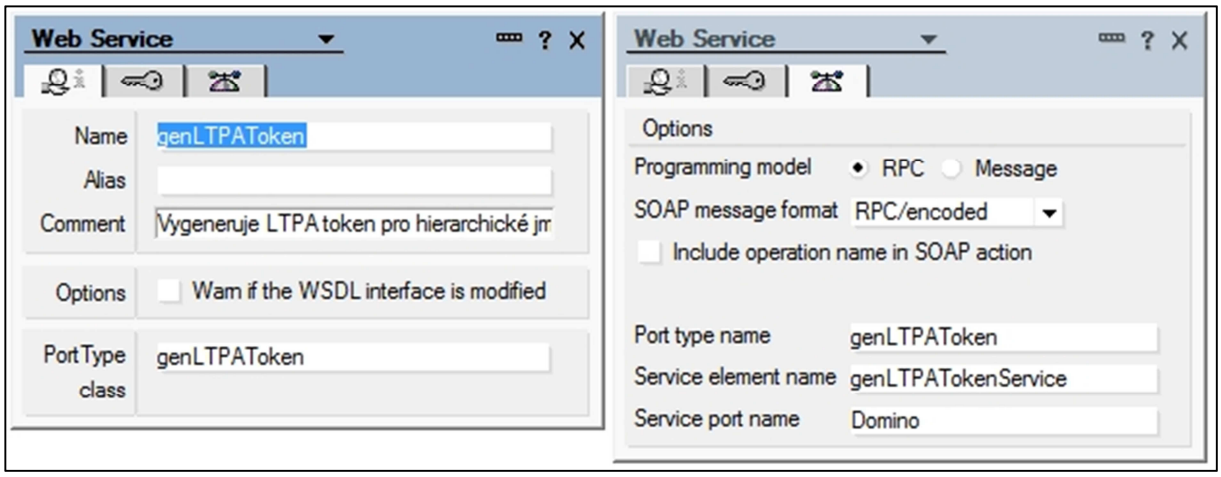
Samotný kód poskytovatele webové služby je omezen na volání funkce z *LotusScript* knihovny. Nejdůležitější funkce této knihovny jsou popsány v této kapitole, celá knihovna je uvedena v příloze 8.1.

```
Class genLTPAToken
     Function getToken(UserDominoToken As String) As String
           Dim result As String
           result = GEN_LTPA_TOKEN(UserDominoToken)
           getToken = result
     End Function
End Class
```

Výpis 3 - Webové služba genLtpaToken

Obrázek 5.5 zobrazuje nastavení webové služby v IBM Domino Designerovi. Kromě pojmenování služby, jména třídy portu a portu služby je potřeba na Security záložce nastavit účet, pod kterým bude služba spouštěna. V tomto případě je služba nastavena s právy serveru, je ovšem možné použít libovolný účet, který bude mít nastaven dostatečná práva a povolené použití vyhrazených operací (allow restricted operations). Vyhrazené operace jsou potřeba kvůli volání funkcí interních knihoven, které pomocí C API[18] rozhraní Domino serveru generují konečný LTPA token (viz str. 71).

[18] C API – Toolkit, který vytváří aplikační rozhraní k Notes/Domino systému využitelné programy psanými v jazyce C. (IBM, Lotus C/C++ API Toolkits for Lotus Notes and Domino, 2012b)



Obrázek 5.5 - Nastavení webové služby genLtpaToken

Nastavení webové služby se automaticky promítá do generovaného popisu webové služby – WSDL (Výpis 4), které vývojové prostředí IBM Domino Designer nabízí od verze 7.0. Pomocí tohoto popisu lze ve většině programovacích prostředích vygenerovat klientskou část webové služby[19].

[19] Tedy pouze její definici. Programátoři se musí postarat o implementaci vstupního parametru a o interpretaci a využití vrácené hodnoty.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions targetNamespace="urn:DefaultNamespace"
 xmlns="http://schemas.xmlsoap.org/wsdl/"
 xmlns:apachesoap="http://xml.apache.org/xml-soap"
 xmlns:impl="urn:DefaultNamespace" xmlns:intf="urn:DefaultNamespace"
 xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
 xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema">

 <message name="GETTOKENRequest">
   <part name="USERDOMINOTOKEN" type="xsd:string"/>
 </message>
 <message name="GETTOKENResponse">
     <part name="GETTOKENReturn" type="xsd:string"/>
 </message>

 <portType name="genLTPAToken">
   <operation name="GETTOKEN" parameterOrder="USERDOMINOTOKEN">
     <input message="impl:GETTOKENRequest" name="GETTOKENRequest"/>
     <output message="impl:GETTOKENResponse" name="GETTOKENResponse"/>
   </operation>
 </portType>

 <binding name="DominoSoapBinding" type="impl:genLTPAToken">
   <wsdlsoap:binding style="rpc"
    transport="http://schemas.xmlsoap.org/soap/http"/>
   <operation name="GETTOKEN">
     <wsdlsoap:operation soapAction=""/>
     <input name="GETTOKENRequest">
       <wsdlsoap:body namespace="urn:DefaultNamespace" use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     </input>
     <output name="GETTOKENResponse">
       <wsdlsoap:body namespace="urn:DefaultNamespace" use="encoded"
        encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
     </output>
   </operation>
 </binding>

 <service name="genLTPATokenService">
   <port binding="impl:DominoSoapBinding" name="Domino">
     <wsdlsoap:address location="http://localhost"/>
   </port>
 </service>
</definitions>
```

Výpis 4 - WSDL genLtpaToken

WSDL je tedy nutné poskytnout vývojářům externí webové aplikace, tak aby mohli na vhodné místo vložit volání webové služby a pak předat výsledný token do cookies uživatele. V řešeném případě bylo volání webové služby vloženo do odkazů na aplikace systémů IBM.

Volání webové služby je z hlediska bezpečnosti riskantní a je potřeba ji dostatečně zabezpečit. Domino servery (nastavené na použití SSO) neumějí v použité verzi 8.5.3

poskytovat přes HTTPS autentizovanou webovou službu, takové volání bohužel v používané verzi vede k chybové hlášce.

Proto se autor rozhodl ponechat webovou službu přístupnou pro anonymní uživatele a bezpečnost zajistit implementací tří bezpečnostních opatření:

- Použití symetrického šifrování obsahu vstupního parametru.
- Omezení dostupnosti webové služby pouze z konkrétních IP adres
- Omezení stáří požadavku na administrátorem nastavený počet minut

Použití druhých dvou opatření zabezpečení této služby samozřejmě obecně nijak nezvyšuje, protože případný útočník dokáže simulovat odchozí e-mailovou adresu i čas odeslání požadavku. Nicméně je autor přesvědčen, že sníží pravděpodobnost uhádnutí šifrovacího klíče. Navíc je tento algoritmus založen na původním algoritmu Domino LTPA tokenu (viz kapitola 5.1.3), tedy alespoň nedošlo ke snížení bezpečnosti ve srovnáním s nativním SSO mechanizmem.

Zpracování požadavku webové služby je rozděleno do následujících kroků.

1. Kontrola IP adresy žadatele
2. Dekompozice vstupního parametru: Získání data vytvoření, data exspirace a hierarchického jména
3. Vlastní generování LTPA tokenu
4. Vrácení hodnoty nebo chyby

Kontrola IP adresy žadatele

Prvním krokem je ověření, že IP adresa žadatele je uvedena v předvolbách aplikace. Stejně jako standardní webový agent i webová služba má na Domino serveru při svém spuštění k dispozici CGI proměnné[20], které jsou dostupné z kontextového dokumentu. Kontextový dokument vzniká v paměti Domino serveru při zpracování HTTP požadavku a zaniká (pokud není uložen) při ukončení zpracování a odeslání odpovědi. Při zpracování požadavku je s ním možné pracovat jako s každým jiným dokumentem (viz Výpis 5). Hodnota systémového pole *Remote_Addr* kontextového dokumentu udává IP adresu žadatele. IP adresa je porovnána se seznamem IP adres uloženým v profilovém dokumentu

[20] Tabulka všech dostupných CGI proměnných dostupných na Domino serveru je k dispozici v nápovědě (IBM, 2011).

aplikace. Za správné nastavení adres v profilovém dokumentu zodpovídá administrátor systému.

```
'Zjistíme IP adresu žadatele z CGI proměnných
Set cgiDoc = session.DocumentContext
If cgiDoc Is Nothing Then
   remoteIPAddress = "N/A"
Else
   remoteIPAddress = cgiDoc.Remote_Addr(0)
End If
```

Výpis 5 - Načtení CGI proměnné Remote_Addr

Dekompozice vstupního parametru webové služby

Vstupní parametr obsahuje hlavičku, hierarchické jméno uživatele, čas vytvoření, čas exspirace a 20 bytový SHA-1 otisk vytvořený pomocí symetrického klíče (známého obou stranám komunikace). Získání těchto parametrů je implementováno ve funkci parse_token (viz příloha 8.1) . Kontrola vstupního tokenu je rozdělena do následujících kroků:

1. Nejprve je pomocí knihovny Base64 převeden vstupní parametr na pole bytů, ze kterého je načtena hlavička, čas vytvoření, čas exspirace, uživatelské jméno a 20 bytový otisk.
2. Opět pomocí Base64 je dekódován symetrický klíč, jehož Base64 podoba je uložena v předvolbách aplikace. Tento klíč musí být shodný s klíčem pomocí kterého byl zkonstruován otisk vstupního parametru na externím serveru.
3. Pole bytů z prvního kroku (bez posledních 20 bytů) je doplněno polem bytů získaným v dekódováním klíče v kroku 2 a pro takto zkonstruované pole je spočten otisk pomocí SHA-1 algoritmu. Výsledný 20-bytový haš je porovnán s posledními 20 byty získanými v prvním kroku. Pokud haš souhlasí, je vstupní token považován za validní a algoritmus pokračuje.
4. Ověření času vytvoření požadavku je provedeno porovnáním času získaného v prvním kroku a aktuálního systémového času na serveru Domino. Protože synchronizace času mezi servery není vždy dokonalá, je zde ponechána několika minutová rezerva a pokud je rozdíl času větší, skončí algoritmus s chybovou hláškou „Požadavek není aktuální“.
5. Pokud je haš validní a souhlasí i datum vytvoření je předáno hierarchické jméno uživatele další funkci, která zajistí vygenerování tokenu.

Generování tokenu pomocí Notes C API

Interní algoritmus je postaven na IBM Notes/Domino C API a je zkompilován do systémové knihovny - DLL souboru[21], který je potřeba umístit do programového adresáře Domino serveru. Použití DLL souboru je nutné uvést v deklaraci LotusScript knihovny.

```
Declare Private Function CreateLtpaToken Lib "LtpaToken.dll"
  ( ByVal userName As LMBCS String, ByVal ltpaToken As String,
    ByVal errorMessage As String) As Long

Declare Private Function UserName Lib "LtpaToken.dll"
  ( ByVal userName As LMBCS String, ByVal ltpaToken As String,
    ByVal errorMessage As String) As Long
```

Výpis 6 - Deklarace Notes API knihovny

Funkce CreateLtpaToken využívá interních funkcí C-API, které si načtou konfigurační dokument SSO (viz str. 61), Z konfiguračního dokumentu jsou načteny všechny další potřebné parametry pro vygenerování WebSphere LTPA2 tokenu tj. název domény, LTPA klíč, čas exspirace atd. a pro dané uživatelské jméno se vygeneruje výsledný token.

Výsledný token je převeden do Base64 a vrácen webovou službou externímu systému, který se postará o jeho předání přihlášenému uživateli.

[21] Dynamic link library (Domino systém s webovou službou je umístěn na Windows 2003 serveru).

6 Shrnutí

IBM Notes a Domino jsou robustním systémem, který dovoluje vytvářet aplikace splňující maximální požadavky na bezpečnost aplikací a dokumentů. Systém je dokonce schopen zamezit přístupu administrátorů k uživatelským datům, aniž by jim ubral možnosti konfigurace a úpravy designu aplikace. Vysoká míra bezpečnosti je jedním z důvodů, proč systém IBM Notes a Domino stále přežívá v obrovské konkurenci jiných aplikací navzdory některým černým odhadům na jeho budoucnost. Hlavním cílem teoretické části této práce bylo vysvětlení bezpečnostního modelu systému IBM Domino a popsání základních technologií potřebných pro vypracování praktické části.

Bezpečí uživatelských dat v aplikacích IBM je závislá jednak na správném nastavení úrovně uživatelských přístupů k Domino serveru, nastavení zabezpečení do jednotlivých aplikací a jednak na úrovni implementace bezpečnostních mechanizmů vývojáři jednotlivých aplikací. V této práci bylo na příkladu aplikace „databáze hesel“ ukázáno, jak lze využít nedbalého nastavení uživatelských práv, a vytvořením kategorizovaného uživatelského pohledu získat heslo i z dokumentů, které uživatel v databází nemá právo vůbec vidět.

V praktické části této práce bylo cílem navrhnout bezpečnou metodu funkce jednoho přihlášení mezi externím webovým systémem a systémy postavenými na IBM platformě. Podmínkou řešení bylo, že uživatelé se nejprve autentizují do externího webového portálu, ze kterého se pak mohou přepínat do jednotlivých aplikací umístěných přímo na tomto serveru nebo na jednom z IBM serverů. Při přechodu na jiný server je nutné autentizovat uživatele vůči tomuto systému.

Přenos autentizačních informací bez nutnosti opakovaného přihlašování mezi systémy je IBM je nativně řešen pomocí tzv. LTPA tokenů, které v šifrované podobě obsahují informace o přihlášeném uživateli a pomocí nich je garantován i přístup k jinému serveru se stejným nastavením a LTPA klíči.

Navržený mechanizmus využívá právě technologie LTPA tokenů, které si hlavní aplikace vyžádá pomocí webové služby z IBM Domino serveru při pokusu o otevření aplikace z jiného serveru. Token vygenerovaný a vrácený webovou službou je pak předán webovému prohlížeči uživatele a uživatel je následně přesměrován do zvolené aplikace.

Implementace webové služby včetně šifrování přenášeného tokenu mezi jednotlivými systémy je hlavní součástí praktické částí této práce.

Při vlastní implementaci bylo nutné řešit další dílčí technické problémy s kódováním předávaného tokenu, českou diakritikou, synchronizací času mezi servery a také vyřešit dílčí synchronizaci adresních knih pro skupinu uživatelů přistupujících z hlavní portálové aplikace k serverům IBM.

7 Seznam použité literatury

BAEHR, R., A. Guirard, M. Holte, T. Hampel, A. Jain a S. Shaikh. IBM. *IBM Lotus Domino Development Best Practices* [online]. 2012. vyd. 2012 [cit. 2013-07-06]. IBM Redbooks. TIPS0859. Dostupné z: http://www.redbooks.ibm.com/

BENZ, Brian, Rocky Oliver a [překlad Milan DANĚK]. *Mistrovství v Lotus Notes a Domino 6: [od základů k pokročilým technikám, podpora JSP, CSS, DHTML, rozšíření jazyka a možnosti integrace, testování, sdílení a vzdálené ladění aplikací]*. Vyd. 1. Brno: CP Books, 2005. ISBN 978-802-5107-508.

BERGLAND, John, Pascal David, Muhammad Ali Sabir, Jérôme & el al. *Redbooks Wiki: Building Domino Web Applications using Domino 8.5.1* [online]. [cit. 2013-05-20]. IBM Redbooks. TIPS0769. Dostupné z: http://www.redbooks.ibm.com/

DAHM, Frederic, Paul Ryan, Richard Schwartz, Amy Smith, Dieter Stalder. *Security considerations in Lotus Notes and Domino 7 making great security easier to implement* [online]. 1st ed. Norwood Mass: Books24x7.com, 2006 [cit. 2013-05-26]. ISBN 978-073-8497-341. Dostupné z: http://www.redbooks.ibm.com/

CERAMI, Ethan. *Web services essentials*. 1st ed. Sebastopol, CA: O'Reilly, c2002, xiii, 288 p. ISBN 05-960-0224-6.

FIELDING, R., J. Gettys J. Mogul & at al. Hypertext Transfer Protocol -- HTTP/1.1. In: NETWORK WORKING GROUP. *Request for Comments* [online]. June 1999 [cit. 2013-07-20]. RFC. RFC 2616. Dostupné z http://tools.ietf.org/pdf/rfc2616.pdf

FIPS PUB 180-4. *Secure Hash Standard (SHS)*. Gaithersburg: National Institute of Standards and Technology, 2012. Dostupné z: http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf

FIPS PUB 186-4. *Digital Signature Standard (DSS)*. Gaithersburg: National Institute of Standards and Technology, 2013. Dostupné z: http://cryptome.org/2013/07/NIST.FIPS.186-4.pdf

HLAVA, Vít. *Webové služby na zvolené platformě*. Praha, 2011. Bakalářská práce. Česká zemědělská univerzita. Vedoucí práce Ing. Miloš Ulman.

CHEN, Whei-Jen, Enio Rubens Basso, Gabriella Davis, Adam Hamilton & at al. IBM. *Installing and Deploying IBM Connections* [online]. 2013. [cit. 20133-03-27]. IBM Redbooks. TIPS0994. Dostupné z: http://www.redbooks.ibm.com/

IBM. *Lightweight Third Party Authentication* [online]. 2005. vyd. WebSphere Information Center. [cit. 2013-09-09]. Dostupné z: http://publib.boulder.ibm.com/infocenter/wsdoc400/v6r0/index.jsp?topic=/com.ibm.webspher e.iseries.doc/info/ae/ae/csec_ltpa.html

IBM. *Understanding LTPA* [online]. 2009. vyd. WebSphere Information Center. [cit. 2013-10-21] Dostupné z: ftp://public.dhe.ibm.com/software/integration/datapower/library/prod_docs/Misc/UnderstandingLTPA-v1.pdf

IBM. *Domino Administrator Help* [online]. IBM Lotus Domino and Notes Information Center, 2011 [cit. 2013-10-20]. Dostupné z: http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=%2Fcom.ibm.help.domino.admin.doc%2FDOC%2FH_DOMINO_LDAP_SCHEMA_DATABASE_STEPS.html

IBM. *Domino Designer Help* [online]. IBM Lotus Domino and Notes Information Center, 2011 [cit. 2013-10-22]. Dostupné z:

http://publib.boulder.ibm.com/infocenter/domhelp/v8r0/index.jsp?topic=%2Fcom.ibm.help.domino.admin.doc%2Fsuper_welcome3_DominoDesigner.html

IBM. *Understanding SPNEGO* [online]. WebSphere Information Center, 2012 [cit. 2013-11-02]. Dostupné z: http://pic.dhe.ibm.com/infocenter/wsdatap/v5r0m0/index.jsp?topic=%2Fcom.ibm.dp.xb.doc%2Funderstandingspnego.htm

IBM. *IBM Connections* [online]. 2013 [cit. 2013-07-12]. Dostupné z: http://www-03.ibm.com/software/products/cz/cs/conn/

IBM. *IBM Connections 4.5 Documentation* [online]. IBM Connection Wiki, 2013a [cit. 2013-10-20]. Dostupné z: http://www-10.lotus.com/ldd/lcwiki.nsf/xpDocViewer.xsp?lookupName=IBM+Connections+4.5+Documentation#action=openDocument&content=catcontent&ct=prodDoc

IBM. *Lotus C/C++ API Toolkits for Lotus Notes and Domino* [online]. IBM developerWorks, 2012b [cit. 2013-10-22]. Dostupné z: https://www.ibm.com/developerworks/lotus/documentation/capi/

JOSEFSSON, S. *The Base16, Base32, and Base64 Data Encodings.* In: NETWORK WORKING GROUP. *Request for Comments* [online]. 2006 [cit. 2013-08-17]. RFC. RFC 4648. Dostupné z: http://www.ietf.org/rfc/rfc4648.txt

KRISTOL, D. a L. Montulli. *HTTP State Management Mechanism*. In: NETWORK WORKING GROUP. *Request for Comments* [online]. 2000 [cit. 2013-07-27]. RFC. RFC 2965. Dostupné z: http://www.ietf.org/rfc/rfc2965.txt

MACGREGOR, Rob a et aL. IBM. *The Domino defense: security in Lotus Notes and the Internet*. S.l.: IBM Corp, 1997. IBM Redbooks. ISBN 978-073-8403-724. Dostupné z: http://www.redbooks.ibm.com/

PASEKA, J. *Kryptografie* [online]. 2012 [cit. 2013-10-08], Masarikova universita Brno: Dostupné z: https://is.muni.cz/el/1431/jaro2012/M0170/um/um/Finalkrypto2012.pdf

SERMERSHEIM, E. J. *Lightweight Directory Access Protocol (LDAP): The Protocol*. In: NETWORK WORKING GROUP. *Request for Comments* [online]. 2006 [cit. 2013-08-06]. RFC. RFC 4511. Dostupné z: http://www.ietf.org/rfc/rfc4511.txt

TUTLE, Steven; Kedar Godbole, Grant McCarthy. *Using LDAP for Directory Integration* [online]. 2004. [cit 2013-08-12]. IBM Redbooks. SG24-6163-01. Dostupné z: http://www.redbooks.ibm.com/

W3C. *XML Schema for WSDL 2.0. (Latest version).* [on-line]. 2007. [cit. 2011-01-16]. Dostupné z : http://www.w3.org/2007/06/wsdl/wsdl20.xsd

W3C. *Web Services Description Language (WSDL) 1.1: W3C Note* [on-line]. 2001. [cit. 2011-01-16]. Dostupné z : http://www.w3.org/TR/2001/NOTE-wsdl-20010315

WEERAWARANA, C., F. Cerbera & et al. *Web Services Platform Architecture.* New Jersey: Prentice Hall PTR. 2005. 456 p. ISBN 0-13-148874-0.

8 Přílohy

8.1 Lotusscript knihovna pro generování LTPA tokenu

```
%REM
      Library TokenLib
      Created Jul 22, 2013 by Vit Hlava
%END REM
Option Public
Option Declare

Use "RegistrationLib"  'Includes NabLib, MailLib
Use "libBase64"
Use "SHA1"

Declare Private Function CreateLtpaToken Lib "LtpaToken.dll" ( _
ByVal userName As LMBCS String, ByVal ltpaToken As String, _
ByVal errorMessage As String) As Long

Declare Private Function UserName Lib "LtpaToken.dll" ( _
ByVal userName As LMBCS String, ByVal ltpaToken As String, _
ByVal errorMessage As String) As Long

Const ERROR_01 = "01 Error - Neplatný hash vstupního tokenu."
Const ERROR_02 = "02 Error - Požadavek není aktuální."
Const ERROR_03 = "03 Error - Požadavek zaslán z neautorizované adresy."
Const ERROR_04 = "04 Error - Nepodařilo se vygenerovat LTPA token."

Class DominoToken

	hierName As String
	created_DT As NotesDateTime
	expiration_DT As NotesDateTime
	ltpa_Token As String

	Sub New()
		hierName = ""
		Set created_DT = New NotesDateTime("1.1.2000")
		Set expiration_DT = New NotesDateTime("1.1.2000")
	End Sub

	Function parse_Token(token As String, sha_key As String) As Boolean

		Dim decodedToken As String
		Dim decodedKey As String
		Dim base64 As New CBase64()
		Dim sha As New NCT_SHA1_PROVIDER()

		Dim dateStr As String
		Dim expdateStr As String
		Dim partStr As String
		Dim strToHash As String
		Dim hash As String
		Dim len_DecodedToken As Integer
		Dim len_DecodedKey As Integer
		Dim len_StrToHash As Integer
		Const LEN_HASH = 20
		Const LEN_VERSION = 4
		Const LEN_DATE = 8

		Dim user_HierName As String
		Dim len_userName As String
		Dim parsed_HierName As String
```

```
	Dim i As Integer
	Dim x As String
	Dim y As String
	Dim ok As Boolean

	Dim created_Str As String
	Dim expiration_Str As String
	Dim date_str As String


	parse_token = False

	'Base64 decode both params
	decodedToken = base64.decode_Token_UTF8(token, parsed_HierName)
	len_DecodedToken = Len(decodedToken)

	decodedKey = base64.decodeBase64(sha_key)
	'decodedKey = base64.decodeStringToString(sha_key)
	len_decodedKey = Len(decodedKey)

	len_StrToHash = len_DecodedToken - LEN_HASH
	If len_StrToHash <= (LEN_VERSION + LEN_DATE + LEN_DATE) Then
		Exit function
	End If

	'Get token part without digest and hash token & key with sha-1 alg
	partStr = Mid$(decodedToken, 1, len_StrToHash)
	strToHash = partStr & decodedKey

	'hash = sha.sha1("", strToHash)  'vrati 40 Hex znaku
	hash = sha.sha1WS(strToHash) ' vrati 20 byte charu

	'Compare token digest part with generated hash?
	ok = true
	For i = 1 To 20
		x = Mid$(decodedToken, len_StrToHash + i, 1)
		y = Mid$(hash, i, 1)
		If x <> y Then ok = false
	Next

	If not OK Then
		Print "[SSO Token] - SHA-1 hash does not fit."
		Exit Function
	End If

	'Get dates form token and compare to now (+- 5 mins)
	date_str = Mid$(decodedToken, LEN_VERSION + 1,  8)
	Set Me.created_DT = Me.get_DTFromBytes(date_str)

	Date_str = Mid$(decodedToken, LEN_VERSION + LEN_DATE + 1, 8)
	Set Me.expiration_DT = Me.get_DTFromBytes(date_str)


	'Get user name
	Me.hierName = parsed_HierName
	parse_Token = True

End Function

%REM
Function generate_Token
Description: Pomocí DLL class
%END REM
Function generate_Token(uName As String) As String

	Dim decode As String
	Dim token As String
	Dim errMsg As String
```

```
	Dim state As Long
	Dim encode As String
	Dim formula As String
	Dim result As Variant

	On Error GoTo ERROR_HANDLE
	Me.generate_Token = "Error"

	'Transformace kódování
	formula = |@URLEncode("Domino";"| + uName + |")|
	result = Evaluate(formula)
	encode = result(0)
	'Print "URL Encode: " + encode
	formula = |@URLDecode("Domino";"| + encode + |")|
	result = Evaluate(formula)
	decode = result(0)

	token = String(2048, " ")		'vysledek pak zkratime
	errMsg = String(2048, " ")
	state = CreateLtpaToken(decode, token, errMsg)


	'2048 bytu je potřeba zkrátit na správnou délku
	Dim i As Integer
	Dim char As String
	Dim max_left As Integer
	i = 0
	max_left = Len(token)
	Do Until i >= Len(token)
		i = i + 1
		char = Mid(token, i, 1)
		If Asc(char) = 0 Then
			max_left = i - 1
			Exit Do
		End If
	Loop

	Me.ltpa_Token = Left(token, max_left)
	Me.generate_Token = "OK"
EXIT_FUNCTION:
	Exit Function
ERROR_HANDLE:
	Print "Error " + CStr(Err) + " - " + Error + " on line " + _
	CStr(Erl) + " in function generate_Token."
	Me.generate_Token = "Error"
	Resume EXIT_FUNCTION
End Function

Private Function get_DTFromBytes(ahsayDate As String) As NotesDateTime

	Dim dt As New NotesDateTime("1/1/1970 00:00:01 GMT+00:00")
	Dim Value As Currency
	Dim Days As Currency
	Dim Hours As Currency
	Dim Seconds As Currency
	Dim hlpNCT As New NCT_SHA1_PROVIDER

	On Error GoTo ErrorOut

	Value = CCur(hlpNCT.hextodec(ahsayDate))

	Days = Value \ 86400				'number of full days
	Hours = (Value Mod 86400) \ 3600		'number of full hours
	Seconds = (Value Mod 86400) Mod 3600	'number of seconds

	dt.Adjustday(Days)
	dt.Adjusthour(Hours)
	dt.Adjustsecond(Seconds)
```

```
		Set get_DTFromBytes = New NotesDateTime(dt.Gmttime)
		Delete dt
		Exit Function

	ErrorOut:
		Error Err, (CStr(Err) &" : " & Error$ & " on line: " & CStr(Erl) & "
		in " & " function BytesToDate ")
		Exit Function
	End Function


	Property Get get_HierName As string
		get_HierName = Me.hierName
	End Property

	Property Get get_Created_DT As NotesDateTime
		Set get_Created_DT = Me.created_DT
	End Property

	Property Get get_Expiration_DT As NotesDateTime
		Set get_Expiration_DT = Me.expiration_DT
	End Property

	Property Get get_LTPAToken As String
		get_LTPAToken = Me.ltpa_Token
	End Property
End Class

Sub Initialize

End Sub

Sub Terminate

End Sub

Function get_URLVariable(cgi_Doc As NotesDocument, value As String) As String

	Dim content As String

	If cgi_Doc Is Nothing Then
		Dim session As New NotesSession
		Set cgi_Doc = session.DocumentContext
	End If

	If cgi_Doc Is Nothing Then
		get_URLVariable = ""
	Else
		content = cgi_Doc.getItemValue("Query_String_Decoded")(0)
		content = StrRight(content, value + "=")
		content = StrLeft(content + "&", "&")
		get_URLVariable = content
	End If

	MsgBox "URL Variable " + value + ": "  + get_URLVariable
End Function

%REM
	Function GEN_LTPA_TOKEN
	Description: Funkce volaná webovou službou, vrací vygenerovaný Token
%END REM
Function GEN_LTPA_TOKEN(dLtpaToken As String) As string

	Dim dToken As new DominoToken
	Dim sha_key As String
	Dim token As String
	Dim now_DT As New NotesDateTime(Now)
```

```
	Dim profile_Doc As NotesDocument
	Dim remote_IP As String
	Dim max_MinuteDelay As integer

	GEN_LTPA_TOKEN = ERROR_04

	' 1. kontrola IP addresy žadatele.
	If not is_Authorized_IP(profile_Doc, remote_IP) Then
		GEN_LTPA_TOKEN = ERROR_03
		Exit Function
	End If

	'2. Get SHA key
	sha_key = get_ProfileFieldStr("LTPA_SHA_Key")

	'3. Decompose the domino Token
	if Not dToken.parse_Token(dLtpaToken, Sha_key) Then
		GEN_LTPA_TOKEN = ERROR_01
		Exit Function
	End If

	'Tolerance nepřesnosti serverových časů - maximální povolený rozdíl 7 min.
	If IsNumeric(profile_Doc.getItemValue("timeStampDelay")(0)) Then
		max_MinuteDelay = CInt(profile_Doc.getItemValue("timeStampDelay")(0))
	Else
		max_MinuteDelay = 7
	End If

	If Not abs(now_DT.Timedifferencedouble(dToken.get_Created_DT)) <= _
	(max_MinuteDelay * 60)  Then
		'nejedná se o aktuální datum
		GEN_LTPA_TOKEN = ERROR_02
		Exit function
	End If

	'5. generate LTPA (type 2) token
	If dToken.generate_Token(dToken.get_HierName) = "OK" Then
		GEN_LTPA_TOKEN = dToken.get_LTPAToken
	Else
		GEN_LTPA_TOKEN = ERROR_04
		Exit Function
	End If

	'zalogování požadavku o token
	Dim session As New NotesSession
	Dim db As NotesDatabase
	Dim doc As NotesDocument

	Set db = session.Currentdatabase
	Set doc = db.createdocument
	Call doc.Replaceitemvalue("Form", "TokenRequest")
	Call doc.Replaceitemvalue("Remote_Addr", remote_IP)

	Call doc.Replaceitemvalue("EntryDominoToken", dLtpaToken)
	Call doc.Replaceitemvalue("Parse_Creation", dToken.get_Created_DT.Localtime)
	Call doc.Replaceitemvalue("Parse_Expiration", _
	dToken.get_Expiration_DT.Localtime)
	Call doc.Replaceitemvalue("Parse_UserName", dToken.get_HierName)
	Call doc.Replaceitemvalue("LtpaToken", dToken.get_LTPAToken)
	Call doc.Save(True, False)

End Function
```