VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

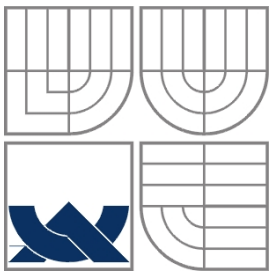
VYUŽITÍ WEBOVÝCH SLUŽEB VE SPRÁVĚ POČÍTAČOVÉ SÍTĚ

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

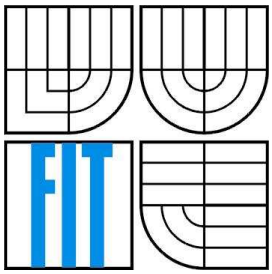
AUTOR PRÁCE
AUTHOR

MICHAL VANĚK

BRNO 2008



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

VYUŽITÍ WEBOVÝCH SLUŽEB VE SPRÁVĚ POČÍTAČOVÉ SÍTĚ

USING WEB SERVICES IN COMPUTER NETWORK ADMINISTRATION

BAKALÁŘSKÁ PRÁCE
BACHELOR´S THESIS

AUTOR PRÁCE
AUTHOR

MICHAL VANĚK

VEDOUCÍ PRÁCE
SUPERVISOR

ING. PETR WEISS

BRNO 2008

Abstrakt

Práce pojednává o možném využití webových služeb ve správě počítačové sítě, zvažuje výhody a nevýhody využití webových služeb v této oblasti. Součástí této práce je vytvoření systému pro centralizovanou správu počítačů a to za využití webových služeb. Soustředí se na problematiku bezpečnosti.

Klíčová slova

Webová služba, správa počítačové sítě, SOAP, XML, SSL, WS-Security, bezpečnost sítě

Abstract

This bachelor’s thesis is discusing using web services in computer network administration, comparing its benefits and problems of this solution. Creation of system for centralised computer network administration is part of this thesis. This thesis is focused on security.

Keywords

Web service, computer network administration, SOAP, XML, SSL, WS-Security, network security

Citace

Vaněk Michal: Využití webových služeb ve správě počítačové sítě. Brno, 2008, bakalářská práce, FIT VUT v Brně.

Využití webových služeb ve správě počítačové sítě

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Petra Weisse. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Michal Vaněk
31.7.2008

Obsah

Úvod

Motivace

O webových službách se hodně mluví ve spojitosti s inovací a zefektivněním mezipodnikové organizace. Často se webové služby totiž využívají při řešení integrace různých aplikací na různých platformách. Tento problém a nutnost jeho řešení je způsobena nárůstem důležitosti internetové komunikace a potřeby spolupráce aplikací mezi firmami.

Stejný problém integrace řeší mnoho různých oblastí a značný význam má i ve správě serverů. A jeho podstata narůstá s velikostí spravované infrastruktury. Nejvíce tento problém řeší firmy přebírající správu informačního systému jiné společnosti a pro efektivní využití vlastních prostředků vyžadují integraci monitorovacích a administrativních prostředků nad celou informační infrastrukturou zákazníka. Zde se často nejen setkávají s různými platformami, aplikacemi, ale i s mnoha specifickými požadavky zákazníka.

Bez ohledu na výše uvedený příklad je snaha o využití na platformě nezávislých prostředků i ve správě počítačové sítě dostatečně opodstatněná. Umožňuje spojit různé technologie, využívat nadstavby, přenášet výhodná řešení, atd.

Rozsah této práce

Účelem této práce je zvážit vhodnost využití webových služeb v oblasti správy počítačové sítě a demonstrovat toto využití vytvořením aplikace pro centralizovanou správu. První kapitola popisuje základy technologie webových služeb. Obsahem druhé kapitoly je návrh konkrétní aplikace pro správu počítačové sítě za využití webových služeb a popis její implementace. Třetí kapitola se věnuje podrobněji bezpečnostní problematice webových služeb, která je pro jejich reálné využití ve správě počítačové sítě podstatná.

1 Webové služby

Tato kapitola se věnuje teoretickým základům problematiky webových služeb a zároveň se snaží o lehkou úvahu nad výhodami a nevýhodami při jejich využití ve správě.

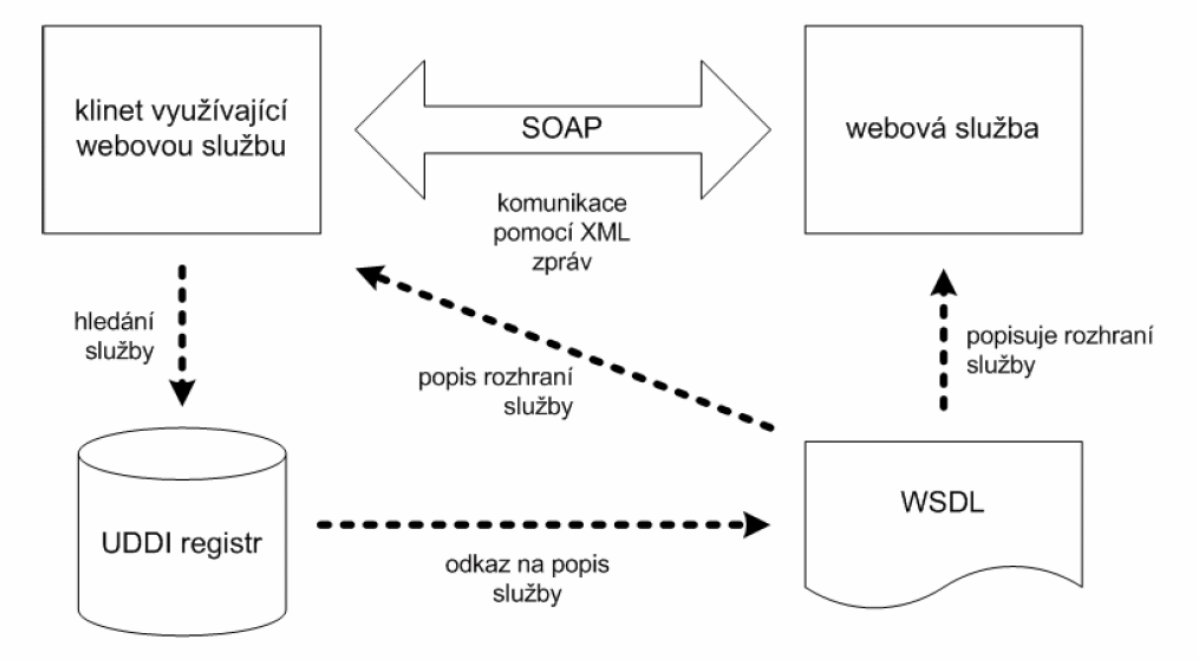
Webové služby jsou technologií pro vzdálené volání funkcí (podobné jako RPC, CORBA, RMI). Tato mladší technologie je v mnoha směrech méně dořešená než ostatní zmíněné, má však několik zásadních výhod. Mezinárodní konsorcium W3C (World Wide Web Consortium) definuje webovou službu takto [1]:

„Webová služba je softwarový systém zkonstruovaný k podpoře interakce stroji přes sít. Má rozhraní popsané ve strojově zpracovatelném formátu (specificky WSDL). Ostatní systémy interagují s webovou službou způsobem předepsaným jejím popisem za pomoci SOAP zpráv, typicky dopravovaných použitím HTTP s XML serializací v součinnosti s ostatními webovými standardy.“

Webová služba (dále jen WS) je spojení několika technologií. Značná část všech výhod i problémů spojených s webovými službami se odvíjí od technologií na nichž jsou webové služby vystavěny.

Technologii webových služeb tvoří:

- XML – jazyk pro popis a zápis dat
- SOAP – protokol pro vzdálené volání procedur, popis zprávy
- WSDL – jazyk pro popis služeb
- UDDI/WSIL – mechanismy pro registraci a vyhledávání služeb



Obrázek 1.0: Webové služby: Vztahy základních technologií

1.1 XML (eXtensible Markup Language)

1.1.1 Základy XML

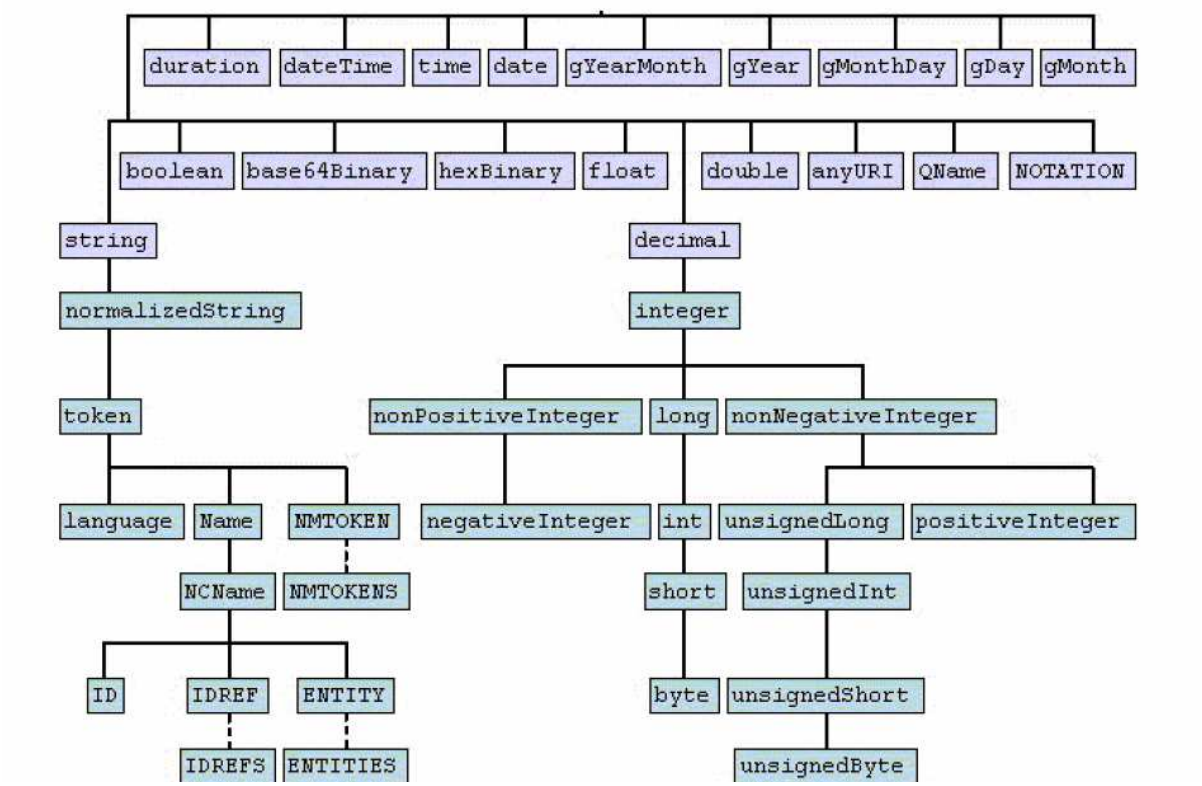
XML [2] [3] je jednoduchý značkovací jazyk, dokument v XML je stromová struktura s právě jedním kořenem, uzly stromu jsou tagy, listy mohou být atributy, tagy a texty. Atributy typu ID umožňují odkazy na tagy. Takto lze vyjádřit obecný orientovaný graf.

XML užívá popisu dat pomocí jmen. XML Namespace je norma pro definici příslušnosti jmen, tak zabraňuje kolizím stejných jmen pro různé věci. Podrobněji XML Namespace určuje **`qualified names`** pro **`tagy`** a **`atributy`**, kde jméno tvoří **`prefix`** a **`local part`** oddělené dvojtečkou. Prefixy mapujeme na URI (Uniform Ressource Identifier). A URI určuje Namespace.

1.1.2 Datové typy XML

Podstatná je také norma XML Schéma. Slouží pro definici datových struktur a datových typů v XML. Kromě základních typů dokáže popsat i složené objekty nebo typy.

- jednoduché typy - základní typy (string, byte, integer, long, double, boolean,…), pole, variantní, intervaly
- složené typy



Obrázek 1.1: XML: Datové typy (jednoduché)

1.1.3 XML a navazující technologie

Kromě toho je nad XML vystaveno mnoho dalších jazyků. Velká výhoda XML spočívá v jeho rozšířenosti. Důvodem je jeho univerzálnost, která umožňuje přenositelnost dat.

1.2 SOAP (Simple Object Access Protocol)

XML je pouze jazykem pro popis a zápis dat. Pro komunikaci je ale potřeba definovat celou zprávu. K tomu slouží protokol SOAP.

1.2.1 SOAP

SOAP [2] je protokol pro posílání XML zpráv – definuje formát XML zprávy. Jedná se o peer-to-peer komunikaci (komunikace mezi dvěma aplikacemi). Zpráva je „pouze“ jednosměrný přenos dat. Ale kombinováním těchto zpráv můžeme vytvářet různé základní komunikační scénáře. Nejčastěji se však setkáváme s modelem požadavek/odpověď, protože SOAP byl vytvářen jako náhrada vzdáleného volání procedur (RPC).

S tímto modelem je spojen i protokol HTTP (HyperText Transfer Protocol), který umožňuje přenos SOAP zpráv. Přesto je vhodné mít při vytváření návrhu aplikací na paměti možnost využití

SOAPu i v jiném komunikačním modelu, tak i v kombinaci s jiným přenosovým protokolem než je HTTP (například SMTP, RSS, Atom).

1.2.2 SOAP a HTTP protokol

Nejčastěji se pro přenos SOAP zprávy používá protokol HTTP a to pro některé vlastnosti spojené s HTTP protokolem [2]:

1. HTTP má širokou podporu v různých aplikacích
2. Je možné využít webového serveru pro umístění webové služby
3. Lze využít portu 80 (nezabezpečeného) nebo portu 443 (zabezpečeného)

SOAP požadavek se zasílá v těle HTTP požadavky, užívá se metody POST (přenos dat je umožněn v těle požadavku). V hlavičce je nutné uvést SOAPAction.

Prakticky je to pak webový server, který zajistí čekání na SOAP zprávu a její předání webové službě, ta ji zpracuje a webový server následně zajistí zaslání odpovědi zpět k žadateli. Pro adresování konkrétní služby se buď uvádí URI v hlavičce HTTP žádosti nebo je služba přímo adresátem požadavku. Hlavičky lze také využít pro údaje sloužící na filtrování na Firewallech, pokud to podporují. Příklad HTTP-SOAP požadavku a odpovědi viz [1.2.4].

1.2.3 Podrobnosti SOAP zprávy

Samotná zpráva SOAPu je jednoduchý XML dokument, který má kořenový dokument Envelope (obálka). V této obálce jsou pak uzavřeny dva elementy – Header (hlavička) a Body (tělo) [2].

Důležité je tělo zprávy, hlavička je nepovinná. Lze ji však využít pro přenos pomocných informací pro zpracování zprávy. Například při filtrování a přepískání SOAP zpráv tyto informace získají časté opodstatnění. Pokud je zpráva často přeposílaná je vhodné určit adresáta přímo v hlavičce. Podobně například atribut hlavičky `mustUnderstand` říká, že příjemce zprávy musí „rozumět“ obsahu hlavičky, jinak je pro něj zpracovávání těla zprávy nepodstatné. Těchto vlastností hlavičky se dá rozumně využít.

Tělo je nejpodstatnější částí zprávy, přenáší se v něm informace identifikující službu a její parametry v případě žádosti, a návratové hodnoty (výsledná data) v případě odpovědi. Vzhledem k podstatě dat ve formátu XML lze přenášet různě strukturovaná data. Pro identifikaci jednotlivých částí XML zprávy používá SOAP jmenné prostory. Obálka, hlavička a tělo zprávy patří do jmenného prostoru `http://schemas.xmlsoap.org/soap/envelope`

1.2.4 Příklady SOAP zprávy

Ukázky převzaty z [2].

SOAP požadavek zaslaný přes HTTP:

```
POST /StockQuote HTTP/1.1
Content-Type: text/xml; charset=utf-8
Content-Length: nnn
SOAPAction: ""
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">

<SOAP-ENV:Body>
<m:GetLastTradePrice xmlns:m="urn:x-example:services:StockQuote">
<symbol>MOT</symbol>
</m:GetLastTradePrice>|
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP odpověď zaslána přes HTTP:

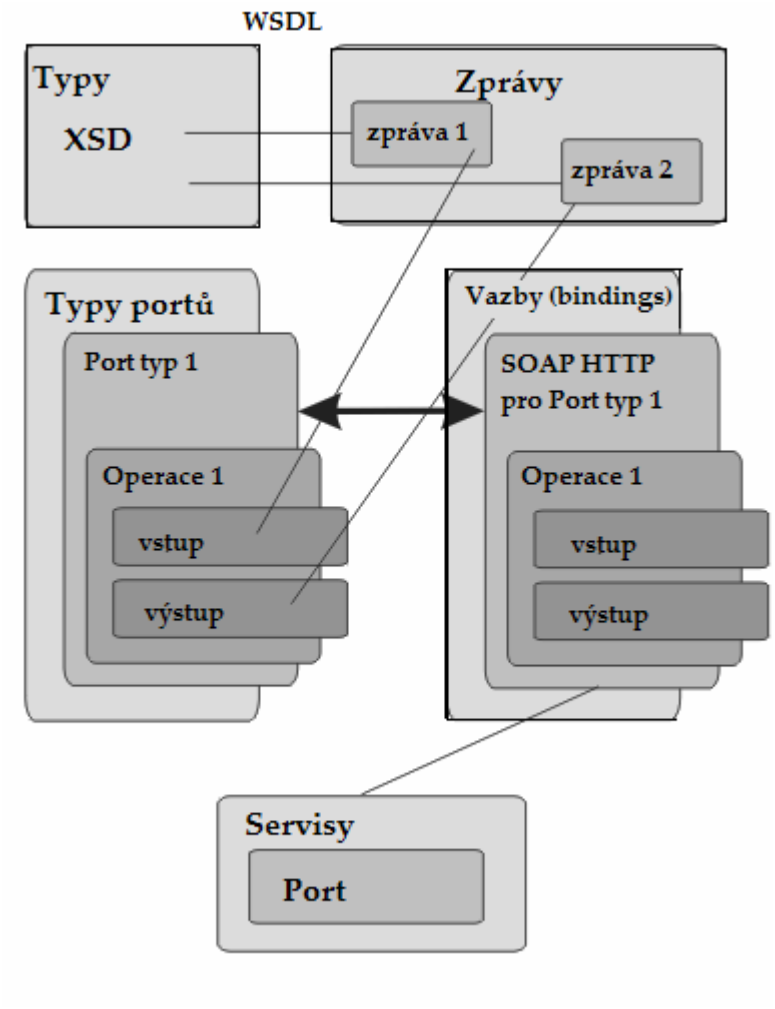
```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: nnn
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
<SOAP-ENV:Body>
<m:GetLastTradePriceResponse xmlns:m="urn:x-example:services:StockQuote">
<Price>14.5</Price>
</m:GetLastTradePriceResponse>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

1.3 WSDL

Jazyk WSDL slouží k popisu síťových služeb. Popisuje je jako množinu koncových bodů zpracovávajících zprávu. Velký význam má u webových služeb, protože umožňuje popsat rozhraní služby. A to tak dostatečně abstraktně, že je danou službu možné využít několika způsoby.

WSDL definice rozhraní je XML dokument. Nejpodstatnější následující elementy [9]:

- types – definice datových struktur/typů. Nejčastěji XML schémata.
- message – definice formátu zpráv pomocí dříve definovaných datových typů
- portType – sdružení několika operací
- binding – navázání určitého portu (portType) na protokol a formát zprávy (message)
- port – koncový bod služby (síťová adresa a binding)
- service – sdružení několika koncových bodů (portů) do jedné služby



Obrázek 1.3: Vztahy WSDL

WSDL popisuje službu srozumitelně i pro člověka. Proto je možné postupovat při návrhu webové služby prvně vytvořením WSDL dokumentu [8], který ji popisuje, a následně z něj odvodit třídy a kód. Jedná se o contract-based přístup. Ten přináší lepší udržovatelnost, rozšiřitelnost a znuvupoužitelnost navrženého systému. Opačný postup se nazývá contract-last.

K definici typů dat ve zprávách se používá XML Schéma (XSD).

1.4 Nástroje pro tvorbu WS

Nástrojů a balíčků pro tvorbu webových služeb je mnoho, existují pro různé programovací jazyky: C++, Python, Perl, Java, Visual Basic,…

Zmíním tu podrobněji několik z nich, mají různou vhodnost pro náš projekt. Ale výběr z více možností poskytuje přizpůsobitelnost dle potřeb uživatele. Většina z nich nabízí možnost vygenerování všech potřebných souborů z WSDL a tím i usnadňují přenositelnost našeho projektu.

gSOAP má nejlepší optimalizaci, služba je pak rychlejší a také je tento nástroj zdarma
generátor zdrojových kódů pro C/ C++
program wsdl2h z WSDL popisu služby vygeneruje speciální .h soubor
program soapcpp2 z .h vygeneruje stub v C nebo C++, a WSDL popis služby
podporuje WS-Security

Apache Axis z rodiny Apache, Java, zdarma
knihovny pro komunikaci
nástroj WSDL2Java
nástroj Java2WSDL
servletová aplikace pro umístění serverové části služby
umožňuje sestavit SOAP volání dynamicky, ale je mnohem pomalejší než gSOAP
podporuje WS-Security

MS ADO .NET jeho součástí je podpora tvorby webových služeb
podporuje WS-Security (součást Web Services Enhancements)

1.5 Shrnutí poznatků

Mezi výhody užití webových služeb pro komunikaci aplikací patří jejich nezávislost na platformě, snadná přenositelnost a rozšiřitelnost při vhodném návrhu, bezpečnostní standardy a možnost využití HTTP jako transportního protokolu.

Otázce bezpečnosti a vlivu na použitelnost v našem projektu se věnuje 3. a 4. kapitola.

2 Návrh aplikace

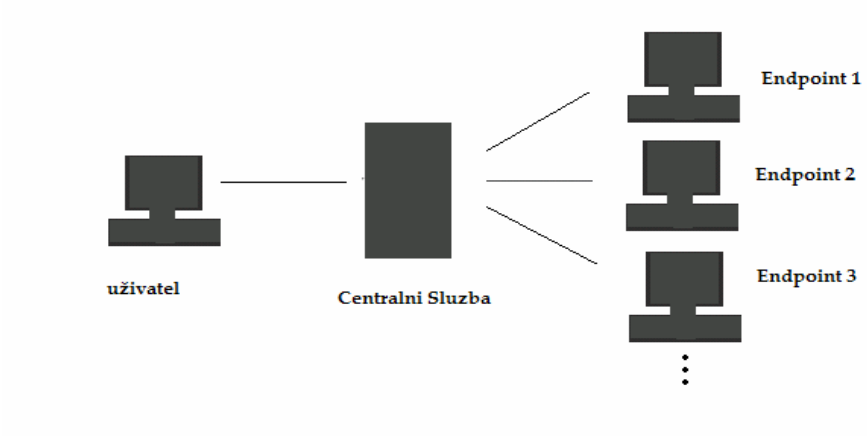
Součástí této práce je navržení a implementace aplikace pro centralizovanou správu počítačů v síti. Správa počítačů v síti zahrnuje nespočet menších služeb a různých požadavků, které se časem mění. Základem pro jejich centralizace je vzdálený přístup. Centralizací samotnou pak míníme možnost přistupovat ke všem počítačům přes jednu službu nebo aplikaci.

Návrh lze rozdělit na základ, který bude umožňovat vzdálený přístup a služby, které tento vzdálený přístup budou využívat. Ty budou pak vytvářeny dle konkrétních potřeb uživatele. Bude se jednat tedy o neustálý vývoj založený na komponentách, protože takto navržený systém umožní samostatný a nezávislý vývoj těchto komponent. Klíčová bude integrace, kde webové služby hrají stěžejní roli jako rozhraní pro komunikaci mezi těmito komponentami.

2.1 Základní komponenty

2.1.1 Rozložení a vztahy komponent

Z pohledu umístění webových služeb lze navrhnout základní rozdělení na služby umístěné na jednotlivých počítačích a centrálním serveru, který bude poskytovat přístup ostatním klientům (službám a uživatelům). Takto:

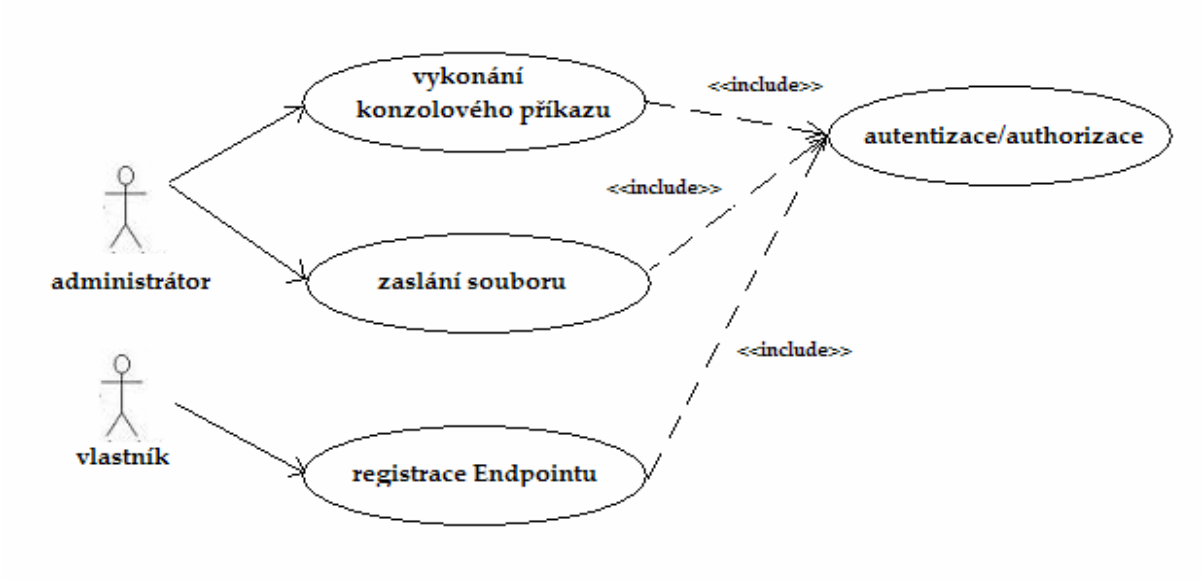


Obrázek 2.1: Vztahy komponent

Služba umístěná na jednotlivých počítačích musí zajistit vzdálený přístup centralizované službě. Měli by být schopny manipulovat se spravovaným počítačem. Budeme tuto službu nazývat endpoint.

Centrální služba bude přijímat požadavky a směrovat je na určené endpointy. Pro určení je bude muset být schopna adresovat. Nejvhodnější řešení je možnost vést si záznamy o endpointech, tedy umožňovat jejich registraci. Další funkce centrální služby se odvíjí od endpointů, měla by poskytovat přístup k adresovaným endpointům, což prakticky znamená adresovat požadavky a odpovědi mezi klientem a požadovaným endpointem.

2.1.2 Uživatelé - Usecase Diagram



Obrázek 2.1: Usecase diagram

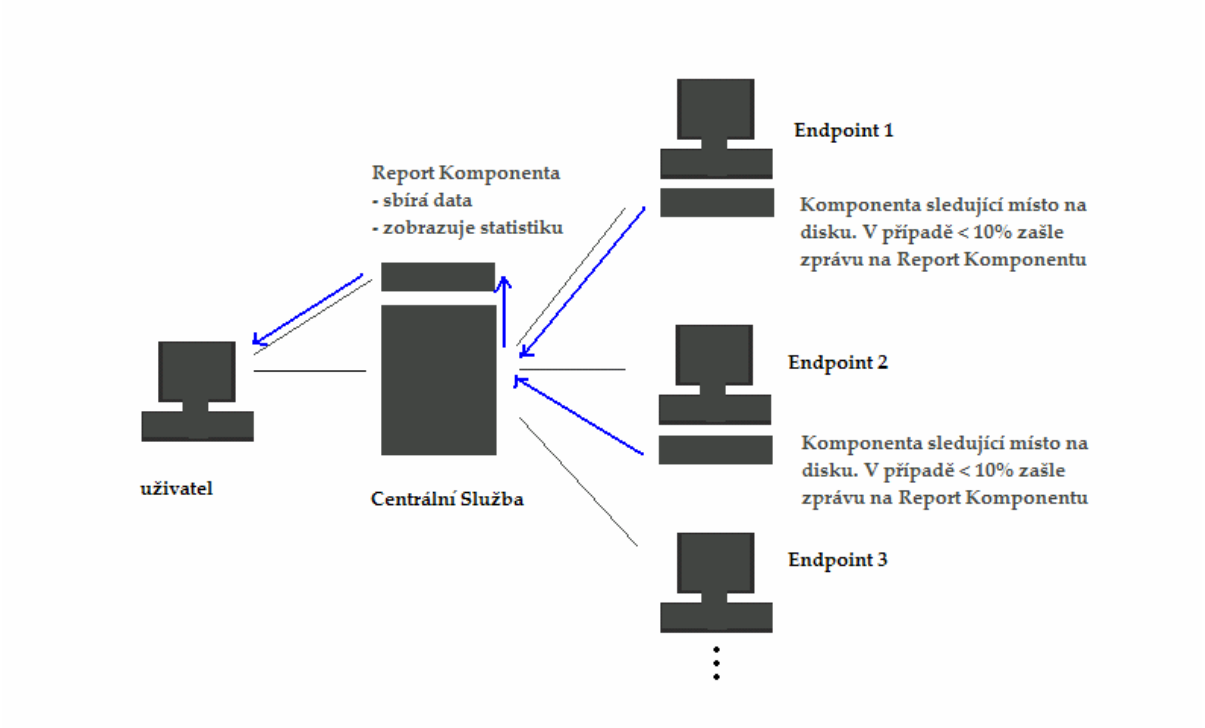
Popis Usecase diagramu:

1. Uživatel v roli administrátora (spravovaných počítačů) má možnost pomocí aplikace vykonávat vzdálené příkazy na počítačích.
2. Uživatel v roli administrátora má možnost zasílání souborů na vzdálené počítače.
3. Vlastník počítačů umožňuje jejich přidání do systému, tedy jejich registraci jako endpointů.
4. Každá akce vyžaduje ověření identity a oprávnění ji provádět.

Autentizace, autorizace, bezpečnost komunikace je rozebrána ve 3. kapitole. Konkrétnímu návrhu řešení bezpečnosti pro naši aplikaci se věnuje pak 4. kapitola.

2.2 Ostatní komponenty (rozšíření)

Možných ostatních rozšiřujících komponent je nespočet. Můžou plnit pravidelné automatické úkony, reagovat na události. Rozšířené komponenty můžou být i na straně endpointu. Ten například pak může zasílat zprávy o událostech na počítači. Princip těchto rozšíření vystihuje následující obrázek:



Obrázek 2.2: Ukázka rozšíření aplikace

Pro demonstraci a otestování našeho základu navrhneme jednu ukázkovou komponentu. Tato komponenta bude využívat centrální služby.

Bude se jednat o aplikaci, která bude v pravidelných intervalech zjišťovat dostupnost počítače (endpointu).

- bude se jednat o klienta centrální služby
- endpoint bude zadán parametry

Několik dalších rozšíření bude založeno na základě zkušenosti z bezpečnostního návrhu.

3 Bezpečnost

3.1 Požadavky na bezpečnost

Účelem této kapitoly je nejen navrhnutí řešení bezpečnosti pro náš projekt, ale i celková studie této problematiky v rámci zabezpečení webových služeb.

První podstatnou otázkou při vytváření plánu pro zabezpečení [15] je, co zabezpečujeme a proti čemu (jaké útoky můžeme očekávat). Naším cílem bude zabezpečení sítě s důrazem na webové služby a možnosti navázat na již existující bezpečnostní politiku sítě (spojení autentizace webových služeb s Active Directory, Kerberos, atd.)

Základní požadavky na zabezpečení sítí vychází z bezpečnostních problémů, které se běžně vyskytují. Bezpečnostní služby komunikační sítě, které je zajišťují jsou také popsány v mezinárodním standardu ISO 7498-2 *ISO/OSI Security Architecture*. Lze je rozdělit do těchto skupin: autentizace, řízení přístupu (autorizace), zajištění důvěrnosti, integrity, nepopíratelnosti a neodmítnutelnosti.

Autentizace je proces ověření identity subjektu (uživatele) počítačové sítě. Většinou se používá kombinace uživatelského jména a hesla (Username Token) nebo digitálního certifikátu, jakým je například X.509 nebo Kerberos.

Autorizace je procesem validace oprávnění již ověřeného subjektu využívat požadované zdroje. Zdrojem je například i samotná webová služba, pak tedy ověření oprávnění uživatele ji volat. Autorizace není mnohdy součástí síťových protokolů, ale samotných aplikací. Je však potřeba aby služba byla schopna identifikovat žadatele.

Integrita dat znamená, že zpráva nebyla pozměněna během přenosu neoprávněným subjektem. Digitální podpis pomáhá určit zda byla zpráva pozměněna. Digitální podpis pracuje na principu vygenerování řetězce nebo hash kódu z originální zprávy, který je víceméně jedinečný a jakákoliv malá změna ve zprávě se na něm projeví. Klíč použitý při jeho vygenerování je znám pouze oprávněným stranám a je neodvoditelný ze zprávy.

Důvěrnost znamená, že pouze oprávněný subjekt má přístup k obsahu zprávy. K tomuto účelu obsah zprávy musí být šifrován. Takto její obsah není čitelný stranou nevlastnící klíč potřebný k jejímu dešifrování.

Zajištění ***neodmítnutelnosti*** je ochrana proti neoprávněnému odepření přístupu ke zdrojům (službám i datům).

3.2 Technologie

3.2.1 Kryptografie a certifikáty

Služby pro autentizaci a zajištění důvěrnosti využívají šifrování, a tedy kryptografii. Kryptografie je transformace otevřeného textu na šifrovaný. Popíšeme proto krátce její princip a navazující technologie.

Existují dva principy kryptografie:

Symetrická

šifrování E_k (M) = C

dešifrování D_k (C) = M

Asymetrická

Klíč pro šifrování je jiný než pro dešifrování.

Dešifrovací klíč se nedá odvodit ze šifrovacího.

šifrování E_{pub} (M) = C

dešifrování D_{sec} (C) = M

Problém symetrické kryptografie je výměna klíčů. U asymetrické je potřeba ověření pravosti veřejného klíče pomocí certifikátu.

Certifikát X.509 je nejpoužívanější typ certifikátu. Každá z komunikujících stran může díky certifikátům ověřit identitu té druhé. Certifikáty totiž jednoznačně spojují šifrovací klíče s jejich vlastníky a dovolují v dostatečné míře ověřit, komu patří. Podle normy X.509 musí certifikát obsahovat číslo použité verze, sériové číslo, dobu platnosti, informace o vlastníkovi a řadu dalších podrobností, které uživateli umožní, aby si ověřil jeho pravost a platnost.

Kerberos je síťový autentizační protokol umožňující komukoli komunikujícímu v nezabezpečené síti prokázat bezpečně svoji identitu někomu dalšímu. Kerberos zabraňuje odposlechnutí nebo zopakování takovéto komunikace a zaručuje integritu dat. Kerberos je postavený na symetrické kryptografii a potřebuje proto důvěryhodnou třetí stranu. Kerberos poskytuje již ověřenému uživateli metodu snadnějšího ověřování na dalších systémech pomocí Kerberos lístku.

Přesný průběh je tento:

1. Klient ověří svou identitu u Key Distribution Center (KDC) a obdrží lístek TGT
2. Klient použije TGT lístek pro přístup do Tiket Granting Service (TGS)
3. Klient vyžádá lístek (TS) pro požadovaný zdroj. A TGS mu ho poskytne.
4. Klient se prokáže tímto lístkem u zdroje (služby) a podle uvedených práv ho může využívat.

3.2.2 Secure socket Layer (SSL)

SSL [14] je protokol, který poskytuje zabezpečení komunikace šifrováním a autentizací komunikujících stran. Tvoří bezpečnostní vrstvu mezi transportní (TCP/IP) a aplikační (HTTP) vrstvou.

Na SSL postavený protokol HTTPS slouží pro bezpečnou komunikaci mezi webovým klientem a serverem. Po vytvoření SSL spojení (*session*) je komunikace mezi serverem a klientem šifrovaná. Ustavení SSL spojení funguje na principu asymetrické šifry, kdy každá z komunikujících stran má dvojici šifrovacích klíčů - veřejný a soukromý.

Ustavení SSL spojení (*SSL handshake*) probíhá následovně:

1. Klient pošle serveru požadavek na SSL spojení, spolu s různými doplňujícími informacemi (verze SSL, nastavení šifrování atd.).
2. Server pošle klientovi odpověď na jeho požadavek, která obsahuje stejný typ informací a hlavně certifikát serveru.
3. Podle přijatého certifikátu si klient ověří autentičnost serveru. Certifikát také obsahuje veřejný klíč serveru.
4. Na základě dosud obdržených informací vygeneruje klient základ šifrovacího klíče, kterým se bude šifrovat následná komunikace. Ten zašifruje veřejným klíčem serveru a pošle mu ho.
5. Server použije svůj soukromý klíč k rozšifrování základu šifrovacího klíče. Z tohoto základu vygenerují jak server, tak klient hlavní šifrovací klíč.
6. Klient a server si navzájem potvrdí, že od teď bude jejich komunikace šifrovaná tímto klíčem. Fáze handshake tímto končí.
7. Je ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.
8. Aplikace od teď dál komunikují přes šifrované spojení.

Během první fáze ustanovení bezpečného spojení si klient a server dohodnou kryptografické algoritmy, které budou použity. V dnešní implementaci jsou následující volby:

- pro výměnu klíčů: RSA, Diffie-Hellman, DSA nebo Fortezza;
- pro symetrickou šifru: RC2, RC4, IDEA, DES, 3DES nebo AES;
- pro jednocestné hašovací funkce: MD5 nebo SHA.

3.2.3 WS-Security

Standard pro zabezpečení komunikace webových služeb (SOAP zpráv) byl vytvořen za spolupráce IBM, Microsftu a VeriSign roku 2002.

WS-Security popisuje rozšíření pro komunikaci SOAP zprávami, aby bylo možné zajistit ochranu integrity zprávy, důvěrnost jejího obsahu a jednoduchou autentizaci. Na těchto základních mechanismech může být vystavěno velké množství bezpečnostních modelů.

WS-Security umožňuje implementovat tyto bezpečnostní funkce [7]:

Autentizace – WSS umožňuje výměnu identifikačních údajů (uživatelské jméno a heslo) jejich vložením do security tokenu přímo v hlavičce SOAP zprávy.

Digitální podpis – ke zprávě je připojen kryptografický podpis, který identifikuje odesílatele. Příjemce může ověřit identitu odesílatele i integritu zprávy. V případě porušené integrity je vyvolána SOAP výjimka.

Šifrování – umožňuje šifrovat SOAP zprávu pro zajištění důvěrnosti. Je možno využít různé šifrovací algoritmy.

Cílem WSS specifikace je poskytnout mechanismy pro zajištění bezpečné výměny zpráv. Nejedná se ale o vytvoření nových bezpečnostních principů, ale o vytvoření standardu pro začlenění již známých a ověřených bezpečnostních principů do SOAP zpráv a webových služeb samotných.

WSS je nezávislé na platformě i na použitém transportním protokolu. Bezpečnostní informace jsou vkládány přímo dovnitř obálky SOAP zprávy. Při zpracování zprávy (žádosti) webovou službou jsou prvně prověřeny bezpečnostní informace, až po té provede požadované akce. V případě, že zpráva nesplní požadované ověření, webová služba vrátí SOAP hlášení o chybě klientovi.

3.2.4 Popis mechanismů bezpečnosti za použití WS-Security

WS-Security stanovuje místo pro vkládání bezpečnostně orientovaných informací. Tímto místem je „bezpečnostní hlavička“ tvořená elementem <Security>. Podíváme se na její schéma [7]:

```
<xs:element name="Security">
    <xs:complexType>
        <xs:sequence>
        <xs:any processContents="lax"
            minOccurs="0" maxOccurs="unbounded">
        </xs:any>
        </xs:sequence>
        <xs:anyAttribute processContents="lax"/>
    </xs:complexType>
</xs:element>
```

Jak lze vidět ve schématu, bezpečnostní hlavička umožňuje vložit jakýkoliv element nebo atribut. Toto umožňuje tuto hlavičku přizpůsobit různým požadavkům bezpečnostního mechanismu, který aplikace vyžaduje. SOAP zpráva je schopna nést více různých bezpečnostních hlaviček. Každá však musí mít unikátní identifikaci role.

3.2.4.1 Časová razítka

V mnoha případech je potřeba nést ve zprávě informaci o době platnosti bezpečnostní hlavičky a čas jejího vytvoření. K tomuto účelu definuje WS-Security element `<Timestamp>` s podelementy `<Expires>` a `<Created>`. Specifikace uvádí, že se tento element smí použít v každé bezpečnostní hlavičce pouze jednou.

3.2.4.2 Autentizace

WS-Security umožňuje použít mnoho cest pro identifikaci uživatele (vycházíme-li z možnosti libovolného obsahu bezpečnostní hlavičky). Je možné používat různé tokeny.

Z hlediska WS-Security je token kolekcí prohlášení provedených určitou entitou. Autentizace pak je ověřením zda jsou tato prohlášení platná pro danou entitu.

WS-Security popisuje elementy pro přenos těchto tokenů:

- Token uživatelských jmen nesoucí uživatelské jméno a heslo - UsernameToken
- Binární bezpečnostní tokeny pro přenos binárních informací (lístek Kerberos, X.509 certifikát) - BinarySecurityToken.

Jako ‚Security Token Servis“ může sloužit Kerberos, PKI nebo jiná služba pro ověření platnosti tohoto prohlášení (tokenu). Tento servis by neměl být webovou službou. Může to být například Kerberos Ticket Granting Servis komunikující přes Kerberos protokol či jednoduché ověření v databázi.

Typickými příklady tokenů jsou:

- Uživatelské jméno a heslo
- PKI skrze X.509 certifikáty
- Kerberos

Podrobně rozebereme všechny tyto příklady tokenů.. Rozebereme jak jsou vloženy do mechanismu SOAP komunikace a z toho vyplívající možnosti útoku a co je skutečně potřeba zajistit pro požadovanou bezpečnost.

Uživatelské jméno a heslo

Jeden z nejběžnějších způsobů identifikace je kombinace identifikace uživatele (uživatelského jména) a společného tajemství (hesla). Tato technika je použita v HTTP Basic a Digest Authentication. WS-Security definuje UsernameToken element pro přenos těchto informací.

Schéma UsernameToken [7]:

```
<xs:element name="UsernameToken">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="Username"/>
            <xs:element ref="Password" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="Id" type="xs:ID"/>
        <xs:anyAttribute namespace="##other"/>
    </xs:complexType>
</xs:element>
```

Toto schéma odkazuje na dva typy: `Username` a `Password`. Tyto dva typy jsou v zásadě řetězce a jejich speciální atributy. `Password` má atribut Type, který určuje zda se jedná o prostý text nebo o hashované heslo. Je zřejmé, že druhý typ poskytuje větší bezpečnost proti zneužití odchyceného hesla. Podrobně popíšeme celý princip.

Hashované heslo lze vygenerovat pouze z hesla. Ale to by nezabránilo útoku přehráním (reply attack). Proto je potřeba hashované heslo vygenerovat za kombinace nonce (hodnoty pro určení unikátnosti), času vytvoření a samotného hesla. Server pak bude ukládat informaci o hodnotách nonce, aby zaručil jejich unikátnost a zabránil tak útoku. Se znalostí času vytvoření a času platnosti se

omezí platnost zpráv a tak i doba potřebná pro udržovaní jejich hodnoty nonce. Můžeme využít wsu:Timestamp, do kterého zaznačíme čas platnosti. Nesmíme zapomenout tuto informaci digitálně podepsat, aby šlo určit, zda s ní nebylo manipulováno.

X.509 Certifikáty

Jiná možnost je pro identifikaci uživatele poslat se správou X.509 certifikát. Tímto přesně server určí uživatele. Pomocí PKI jsou identity uživatelů mapovány s certifikáty. I v tomto případě nedokážeme určit útok přehráním. Pro tento účel je potřeba postupovat obdobně jako v předchozím případě.

WSS vkládá informace o certifikátu do BinarySecurityTokenu. Samotný certifikát je zaslán jako base64 data.

BinarySecurityToken schéma:

```
<xs:element name="BinarySecurityToken">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="Id" type="xs:ID" />
                <xs:attribute name="ValueType" type="xs:QName" />
                <xs:attribute name="EncodingType" type="xs:QName" />
                <xs:anyAttribute namespace="##other"
                    processContents="strict" />
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
```

3.2.4.3 Digitální podpis

Digitální podpis se používá pro možnost rozpoznat, zda nebyla zpráva neoprávněně modifikována, nebo případně i pro autentizaci autora podpisu. Za použití veřejného klíče pak zajišťuje i nepopiratelnost podpisu.

WS-Security využívá standardu XML Signature [11] [13]. WS-Security pouze definuje jakým způsobem by měl být standard XML Signature použit (například pořadí vkládání elementů podpisu do hlavičky).

XML Signature neurčuje, které algoritmy a technologie digitálního podpisu by měly být použity, přesto požaduje, aby použité hashovaní funkce byly bezkolizní a jednosměrné (nešlo z daného otisku

zpětně vypočítat zprávu). Takovýto asymetrický algoritmus podepisuje pouze hodnotu otisku, nikoli celou zprávu. Pro ověřování se procedura zopakuje a výsledky porovnají.

Jednou z výhod použití XML Signature je možnost lépe pracovat s jednotlivými elementy XML dokumentu. Je možné podepisovat více různých elementů jedním podpisem. Pro tento účel definuje XML Signature element <`SignedInfo`>, který slouží pro vytvoření jednoznačné reprezentace podpisovaných elementů a dokumentů. Toho je dosaženo tak, že se prvně vloží otisk všech těchto elementů a dokumentů do elementu <`SignedInfo`> a ten se pak podepíše klasickým digitálním podpisem.

Element <`SignedInfo`> nese ještě další důležité informace. První informací je použitý algoritmus digitálního podpisu a druhou pak kanonizační algoritmus. Kanonizační algoritmus je podstatný pro serializaci podepisovaného XML stromu, tedy jeho převod na posloupnost bytů. Tím se zabrání aby s ním bylo nadále pracováno jako s XML stromem. Nad podepisovaným stromem je možné provádět mnoho dalších transformací, ty jsou zapsány v elementu <`transform`> ve stejném pořadí v jakém jsou provedeny.

O transformacích toho řekneme více, protože mohou přinášet problémy a narušovat základní princip digitálního podpisu. Může se jednat například o komprese a dekomprese nebo o vynechání některých elementů a mnohé další. Problém nastává například v případě vypouštění elementů, které pak mohou být modifikovány aniž by to bylo pak v hodnotě digitálního podpisu poznat. Nebo naopak mohou být v podpisu původně nepožadované elementy.

Je vhodné rozumět použitým algoritmům a transformacím a analyzovat možné chyby, které mohou nastat při jejich nevhodném použití a tím pádem i možnost útoků. XML Signature přímo uvádí některé známé problémy, na které je potřeba dávat pozor.

3.2.4.4 Šifrování

Pro důvěrnost zprávy je potřeba její obsah šifrovat. WS-Security zde plně používá specifikace XML Encryption [12] [13]. Lze zvolit jak symetrické tak asymetrické šifrování. Symetrické šifrování vyžaduje sdílené tajemství (klíč).

XML-Encryption umožňuje šifrovat buď celý dokument nebo obsah elementu, který může být šifrován buď jako posloupnost elementů nebo jako znaková sada.

XML-Encryption definuje element jak šifrovaná data, tak i pro šifrované klíče. Což umožňuje například za použití asymetrického šifrování komunikace poslat symetrický klíč.

Další elementy definované XML-Encryption nesou informace o dohodnuté metodě, hodnotě klíče, metodě pro šifrování a dešifrování. Element <`SecuritytokenReference`> umožňuje odkazovat na bezpečnostní token v bezpečnostní hlavičce. Ten poskytuje informace o použitém klíči.

3.2.4.5 Shrnutí

Specifikace WS-Security popisuje jak implementovat různé technologie umožňující identifikovat odesílatele zprávy, digitálně zprávu podepsat, a šifrovat její obsah. Protože všechny potřebné bezpečnostní informace jsou neseny uvnitř zprávy, je tato metoda zcela nezávislá na transportní vrstvě. Z toho vyplývají také některé důvody pro využití WS-Security namísto SSL.

3.2.5 Srovnání SSL a WS-Security

Bezpečnost na transportní vrstvě znamená ochranu dat zabezpečením samotného komunikačního kanálu (transportní vrstvě). Nejběžnějším případem je HTTPS kanál, který zabezpečuje spojení mezi prohlížečem a webovým serverem. HTTPS je postaveno na SSL protokolu. Použití HTTPS je nenáročné, kdy a proč tedy použít WS-Security ?

1. ***bezpečnost pouze při přenosu (wire protection)***
   V případě SSL je zpráva chráněná pouze při přenosu. Když zpráva dorazí k cíli, může být uložena jako prostý text (nezařídí-li aplikace jinak). Pokud není aplikace správně zajištěna, vystavuje se obsah zprávy neoprávněnému přístupu.

2. ***šifrování na transportní úrovni, point-to-point bezpečnost***
   V některých případech je potřeba šifrovat jen pouze některé elementy zprávy. Šifrování celé zprávy (na transportní úrovni) pak může být zbytečné. Ale větší nedostatek se projeví, potřebujeme-li některé elementy nechat zpracovat uzly na cestě, ale není žádoucí, aby tyto uzly měli kryptografické klíče ke všem elementům zprávy.

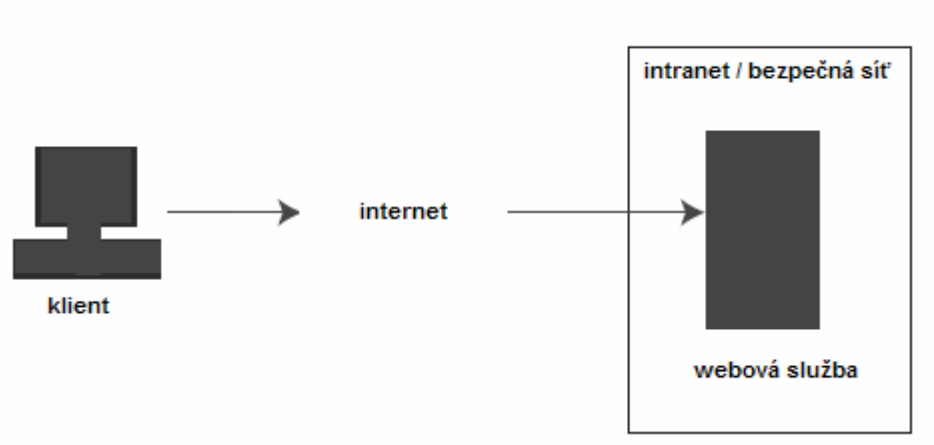
3.3 Modely

3.3.1 Běžné bezpečnostní scénáře

Při hledání vhodného řešení pro zajištění bezpečné komunikace naší aplikace je vhodné si projít běžné scénáře [5], v kterých je potřeba řešit bezpečnost komunikace různými způsoby. Z těchto poznatků pak budeme vycházet v našem projektu.

3.3.1.1 Veřejná služba

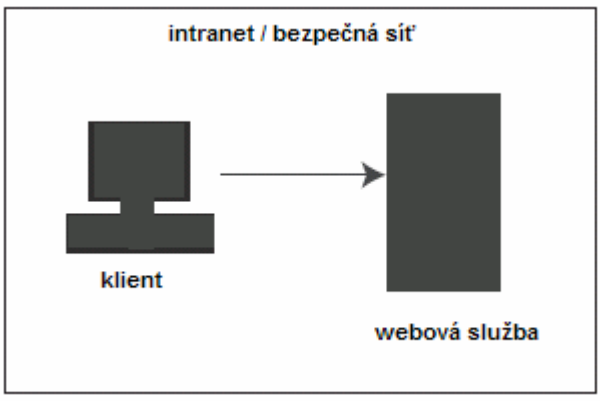
Webová služba je přístupná přes internet. V tomto scénáři se nejčastěji uživatel identifikuje (autentizuje) zasláním uživatelského jména a hesla na webovou službu. Ta je pak ověří v lokálním uložišti identit. Vzhledem k tomu, že se zpráva šíří po internetu a nepředpokládá se, že bude potřeba do zprávy nahlédnout během její cesty, je nevhodnější využít HTTPS.



Obrázek 3.3: Veřejná služba

3.3.1.2 Soukromá služba (intranet)

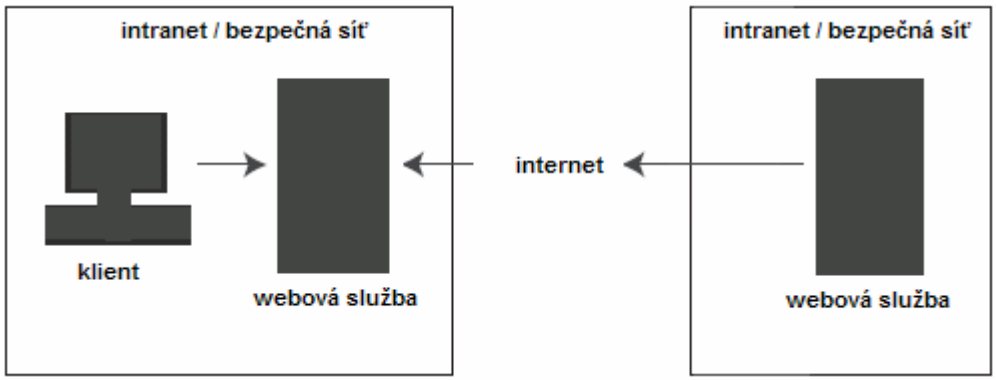
Webová služba je přistupná pouze v intranetu organizace. V tomto scénáři zpráva cestuje vždy vnitř organizace. Ta nejčastěji správu bezpečnosti v síti zajišťuje pomocí komplexní správy identit a přístupů (například Active Directory). Většinou je podporován protokol Kerberos, který poskytuje autentizační, šifrovací, podpisové mechanismy pro zajištění bezpečné komunikace.



Obrázek 3.3: Soukromá služba (intranet)

3.3.1.3 Mezi-podniková služba

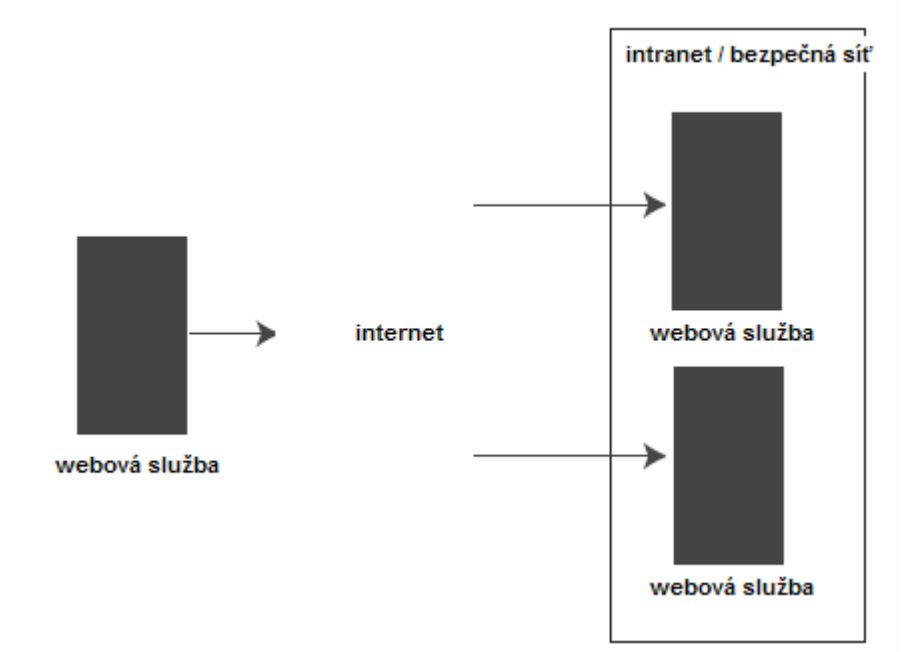
Jedná se jak o komunikaci vnitř sítě jedné organizace, tak i mezi dvěma různými organizacemi. Proto tento scénář vyžaduje spojení dvou řešení. Vnitřní komunikace může být stále zajištěna pomocí protokolu Kerberos. Pro zajištění komunikace mezi organizacemi je nejvhodnější použít X.509 protokol.



Obrázek 3.3: Mezi-podniková služba

3.3.1.4 Více veřejných služeb jedné organizace

Tento scénář má prezentovat situaci, kdy je vhodné umožnit „Single Sign-On‘, tedy ověření identity je vyžadováno jen při prvním volání, pro další komunikaci s webovými službami je ustanoven bezpečný komunikační kanál. To vede ke zvýšení výkonu, protože se sníží počet opakovaného ověřování identity v databázi hesel.



Obrázek 3.3: Více veřejných služeb

3.3.2 Bezpečnostní strategie

Z běžných scénářů vychází pak několik možných strategií využití zmíněných technologií. V úvahu je brán standard WS-Security tak i SSL. Názvy následujících strategií jsou převzaty z odpovídajících autentizačních módů ve WSE 3.0 (Web Service Enhancements) pro platformu ASP.NET [5]. Jejich vyjmenování nemá za účel vyjmenovat všechny možné strategie, které můžeme použít, ale pouze vytvořit si jisté vodítko.

3.3.2.1 UsernameOverTransportSecurity (SSL)

Tato strategie předpokládá vytvoření bezpečného komunikačního kanálu na transportní vrstvě, například pomocí SSL. Klient pak zašle Username Token na server pro prokázání své identity. Server bude odpovědný za ověření tohoto tokenu nesoucího uživatelské jméno a heslo. Po úspěšném ověření identity ověří i práva využít vyžádaných zdrojů.

V tomto scénáři klient a server využívají sdíleného tajemství pro autentizaci. Server může ověřit heslo, které obdržel od klienta například v lokální databázi, adresáři, Active Directory.

3.3.2.2 UsernameForCertficateSecurity

Zde se také využívá uživatelského jména a hesla pro identifikaci uživatele, ale obsah zprávy je šifrován pomocí X.509 certifikátu na úrovni zprávy, ne na transportní vrstvě. Hlavním rozdílem je, že

v prvním případě musí existovat bezpečný komunikační kanál, jakým je například HTTPS. V druhém případě klient používá veřejnou část klíče náležící serveru pro zašifrování zprávy. Server ji pak pomocí svého soukromého klíče dešifruje.

V tomto případě bude zpráva zaslána přes HTTP protokol (nezabezpečený), ale obsah zprávy je stále zabezpečen proti čtení neoprávněnou stranou.

3.3.2.3 AnonymousForCertificateSecurity

V tomto případě server nepotřebuje určit identitu uživatele webové služby. Znovu klient využívá veřejného klíče serveru pro zašifrování zprávy.

V případě potřeby identifikace mohou být identifikační údaje zaslány vnitř těla zprávy, ne však jako Username Token (v bezpečnostní hlavičce).

3.3.2.4 MutualCertificate

Klient a server využívají certifikáty pro vzájemné dokazování identity a zabezpečení zpráv. Odesílatel (klient i server) zašifruje podpis pomocí svého soukromého klíče a příjemce jej rozšifruje pomocí náležitého veřejného klíče.

3.3.2.5 KerberosSecurity

Tato strategie je vhodná pro intranet, kde zpráva cestuje v rámci jedné nebo více domén používající Kerberos. Jednou velkou výhodou je, že tohle řešení snižuje počet ověřování identifikačních údajů a tím poskytuje lepší výkon. Pro další ověřování identity odesilatele zprávy slouží klíč přiřazený Kerberos tiketem po prvním ověření identity.

4 Bezpečnost navrhované aplikace

Při návrhu řešení bezpečnostního scénáře naší aplikace pro správu v síti musíme brát ohled na:

Spolupráce systémů a různé bezpečnostní politiky
Existuje velké množství šifrovacích mechanismů a velké množství platforem s různými bezpečnostními modely. Servis a jeho uživatel musí být schopni použít stejného standardu pro ustanovení úspěšné komunikace.

Klient i server musí být schopni vytvořit vzájemnou důvěru. Každý systém, který používá službu, nebo ji provozuje může být součástí různých bezpečnostních modelů (Kerberos, Username, atd.) Stále však obě strany budou muset být schopny se autentizovat a autorizovat svá oprávnění.

Proto zvolíme nezávislý systém ověřování identit našich komponent. Ty bude ověřovat centrální služba, a pouze centrální služba bude důvěryhodná pro ostatní komponenty (nebude možné je přímo kontaktovat).

Bezpečnost zprávy
Zpráva může projít přes nedůvěryhodné síťové prvky. Proto integrita, důvěrnost zprávy musí být zajištěna. Proto všechna komunikace bude šifrována.

4.1.1 Popis návrhu bezpečnostního scénáře

Všechna oprávněná komunikace bude procházet přes centrální službu. Ta bude umožňovat ověření identity pomocí uživatelského jména a hesla nebo pomocí digitálního podpisu. Centrální služba bude poskytovat svůj veřejný klíč pro šifrování zprávy.

4.1.1.1 Komunikace mezi Endpointem a Centrální službou

Rozdílná komunikace proběhne mezi zaregistrovaným a nezaregistrovaným endpointem. Jediná možná komunikace s nezaregistrovaným je jeho registrace.

Registrovaný Endpoint
Nejvýhodnějším řešením je vzájemná autentizace digitálním podpisem.

Registrace Endpointu

Neregistrovaný Endpoint není ověřený, tudíž centrální služba nemá jeho veřejný klíč mezi důvěrnými. Proto je vhodné registraci provést za pomoci uživatelského jména a hesla uživatele oprávněného k registraci endpointů. Žádost o registraci je šifrována pomocí veřejného klíče Centrální služby. Při registraci si Centrální služba uloží veřejný klíč endpointu mezi důvěryhodné.

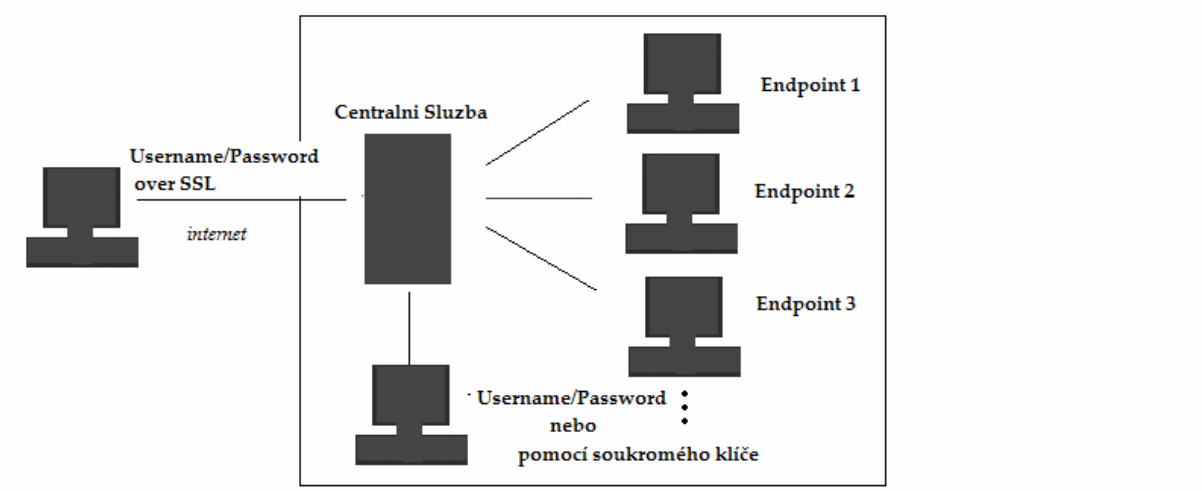
4.1.1.2 Komunikace mezi centrální službou a ostatními komponenty (uživatelem)

Na internetu dostupná služba

SSL pro bezpečnou komunikaci a uživatelské jméno a heslo.

V intranetu dostupná služba

Autentizace uživatelským jménem a heslem nebo digitálním podpisem.



Obrázek 4.1: Návrh bezpečnostního scénáře

4.2 Testování bezpečnosti

V rámci testování jsem provedl pár základních testů funkcionality bezpečnostních mechanismů vytvořené aplikace. Tyto testy ale nejsou plnohodnotným ověřením bezpečnosti aplikace, jako spíše pouhé prvotní prověření implementace. Jednotlivé provedené testy:

1. Test důvěrnosti

Popis: Za pomoci TCP/IP monitoru odchytíme SOAP zprávy mezi službami a analyzujeme.

Výsledek: Splňuje naše požadavky.

2. Test integrity dat

Popis: Odchycenou zprávu upravíme v různých částech zprávy.

Výsledek: Webová služba rozpozná zásah do zprávy dle očekávání.

3. Test útoku opakováním zprávy

Popis: Odchycenou zprávu znovu pošleme na službu.

Výsledek: Zpráva je zamítnuta.

Existují však nástroje pro automatické formální ověřování bezpečnosti. Jedním z nich, jehož primárním určením je verifikace právě bezpečnostních protokolů WS-Security, je nástroj TulaFale. Jeho základem je pi-kalkul, tedy formální metoda pro popis komunikujících procesů a analýzu jejich vlastností.

Za pomoci popisu a ukázek uvedených v diplomové práci Jana Urbana [13] jsem doladil bezpečnost aplikace k stanoveným předpokladům. Samotné jádro aplikace je vcelku jednoduché. Problém však může nastat v případě rozšiřování, při použití jiných technologií a algoritmů, transformací, složitějších zpráv, křížení digitálních podpisů (různých kolekcí elementů).

4.3 Bezpečnostní problémy

4.3.1 Chyba nebo útok oprávněného uživatele.

Tato aplikace nemůže zabránit všem možným chybám, ale některými způsoby může omezit pravděpodobnost. Jedním z nich je vytvoření evidence uživatelů, zapisování aktivity.

Další možná rozšíření:

a) Složitější autorizace s podporou na straně Endpointu. Omezení volné činnosti na počítači a vytvoření předdefinovaných skriptů včetně oprávnění je spouštět. Problémem tohoto řešení, je buď nutnost vytvoření centralizované databáze všech skriptů na všech endpointech a jejich oprávněných uživatelů, nebo samostatná správa těchto oprávnění přímo na jednotlivých endpointech.
b) Rozšiřující komponenta s definovanou akcí a oprávněním plně používat centrální služby, bude nabízet svou službu dalším uživatelům. Vyžaduje oddělenou správu identit a oprávnění.

4.3.2 Chyba komponent

Pří implementaci WS-Security je potřeba pamatovat na to, že WS-Security poskytuje možnost integrace bezpečnostních mechanismů, ale jejich správné použití je otázkou programátora.

Špatně vytvořený klient může například zaslat „prázdnou“ zašifrovanou zprávu pomocí XML-encrypt, která bude obsahovat pouze tagy. Zveřejněné WSDL takovouto zprávu popisuje. Takže útočník tím získá znalost zašifrované i nezašifrované zprávy (stejného obsahu).

Z těchto důvodů je vhodné používat v komunikaci pouze otestovaných komponent a neposkytovat popis služby (WSDL) veřejně. Každé rozšíření by mělo projít formálním ověřením a schválením.

Závěr

Cílem práce bylo zvážit vhodnost využití webových služeb v oblasti správy počítačů v síti. Zároveň toto použití webových služeb demonstrovat na návrhu a implementaci aplikace pro centralizovanou správu.

V úvodní teoretické části práce jsme se seznámili s technologiemi webových služeb. Účelem tohoto úvodu bylo získat přehled o webových službách a technologiích potřebných pro následující návrh a určit oblasti, které je potřeba důkladněji prozkoumat. Na základě těchto poznatků jsem oddělil problematiku bezpečnosti nejen z teorie, ale i samotného návrhu aplikace.

Problémem webových služeb není nemožnost implementovat mechanismy pro zajištění bezpečnosti komunikace. Pro tento účel existuje několik standardů, které umožňují zabezpečit webové služby za použití dnes běžných a ověřených principů. Přesto může problém nastat při jejich nesprávné implementaci chybou vývojáře těchto služeb, protože tyto standardy ponechávají značnou volnost. Ta je ale v jistém smyslu jejich hlavní výhodou, proto byl vývoj těchto standardů zdlouhavější.

Nejvíce jsem se v této práci věnoval specifikaci WS-Security a souvisejícím standardům. Ty považuji za dostatečné pro většinu implementací webových služeb. V případě správy počítačů se ale zvětšuje značně pravděpodobnost chyby při tvorbě webové služby i klientů. Nebezpečí vidím v rozsahu této oblasti a množství potřeby přizpůsobovat služby okolnostem a technologiím. Vzhledem k tomu doporučuji vytvářet aplikace v této oblasti pouze za větší znalosti mechanismů a dodržovat doporučení standardů, používat ověřené principy a provádět důslednou formální verifikaci.

Literatura

[1] Oasis reference model for service oriented architecture, committee draft 1.0, 7 February 2006. URL: www.oasis-open.org/committees/download.php/16587/wd-soa-rm-cd1ED.pdf.

[2] Jiří Kosek. *Inteligentní podpora navigace na WWW s využitím XML*. diplomová práce, Vysoká škola ekonomická v Praze, 2002.

[3] Kuba, M. *Web Services*. Zpravodaj ÚVT MU, 2003, issue 3, vol. 13, p. 9 - 14, ISSN 1212-0901.

[4] Jan Stoklasa. *Budoucnost xml webových služeb - bezpečnost*. URL: objekty.pef.czu.cz/2003/sbornik/Stoklasa2003.pdf

[5] Jeffrey Hasan, Mauricio Duran. *Expert Service-Oriented Architecture in C# 2005: Defining Web services development with ASP.NET and WSE 3.0,* Second Edition, Apress, 2006, ISBN 978-1-59059-701-9.

[6] Sam Thompson. *Implementing WS-Security*. IBM, May 2003. URL: http://www.ibm.com/developerworks/webservices/library/ws-security.html

[7] Scott Seely. *XML and Web Services Security: Understanding WS-Security*, Microsoft corporation, October 2002. URL: http://msdn.microsoft.com/en-us/library/ms977327.aspx

[8] Robert Novotný. *Java: Od WSDL k webovém službe*. 4. července 2008. URL: http://ics.upjs.sk/~novotnyr/wiki/Java/OdWSDLKWebovejSluzbe

[9] Cerami, E., *Web Services Essentials*, O'Reilly, 2002, ISBN 0-596-00224-6.

[10] Mactaggart, M., *Enabling XML security*, 2001. URL: http://www-128.ibm.com/developerworks/security/library/s-xmlsec.html

[11] XML Signature Syntax and Processing (second edition), W3C Recommendation, 10 června 2008. URL: http://www.w3.org/TR/xmldsig-core/

[12] XML Encryption Syntax and Processing, W3C Recommendation, 10 prosince 2002. URL: http:// www.w3.org/TR/xmlenc-core/

[13] Jan Urban. *Specifikace „Web Services Security“*. diplomová práce, Fakulta Informatiky Masarykovy Univerzity v Brně, 2006.

[14] *Secure Socket Layer*. Wikipedia. 10. června 2008. URL: http://cs.wikipedia.org/wiki/Secure_Sockets_Layer

[15] Kouřil, D., *Bezpečnost v distribuovaném prostředí.* Zpravodaj ÚVT MU. ISSN 1212-0901, 2005, roč. XV, č. 4, s. 2-6.

Seznam příloh