VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

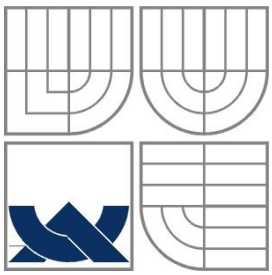
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BEZPEČNOST WEBOVÝCH SLUŽEB Z POHLEDU POSKYTOVATELE SLUŽBY

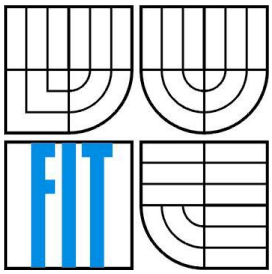
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE Stanislav Liška
AUTHOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

BEZPEČNOST WEBOVÝCH SLUŽEB Z POHLEDU POSKYTOVATELE SLUŽBY

NÁZEV BAKALÁŘSKÉ PRÁCE ANGLICKY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — Stanislav Liška
AUTHOR

VEDOUCÍ PRÁCE — Mgr. Marek Rychlý, Ph.D
SUPERVISOR

BRNO 2010

Abstrakt

Tato práce si dává za úkol poskytnou čtenářům pohled na možnosti zabezpečení webových služeb a jejich autentizaci. Druhá kapitola obecně vysvětluje webové služby a třetí kapitola bezpečností protokoly spolu s autentizací. Ve čtvrté kapitole jsou popisovány možnosti zabezpečení webových služeb v prostředí Java Enterprise Edition. Pátá kapitola je věnována návrhu webové služby z pohledu poskytovatele služby. Šestá kapitola popisuje implementaci zabezpečené služby a sedmá testování služby s možností dalšího rozšíření.

Abstract

This work makes the task of providing readers with insight into the possibilities of Web services security and authentication. The second chapter explains the general Web services and the third charter deals with security protocols along with authentication. The charter possibility of securing Web services in Java Enterprise Edition are described in the fourth chapter. The fifth chapter is devoted to Web services from the perspective of the service provider. The sixth chapter describes the implementation of secure services and seventh charter deals with testing services with the possibility of further extension.

Klíčová slova

Webové služby, bezpečnost, WS-Security, SOAP, http-auth Basic,SSL, SAML, REL, certifikát, autentizace, X.509, šifrování, XML, popisovač nasazení, WSIT

Keywords

Web service, security, WS-Security, SOAP, http-auth Basic, SSL, SAML, REL, certificate, authentication, X.509 , cipher, XML, deployment deskriptor, WSIT

Citace

Liška Stanislav:Bezpečnost webových služeb z pohledu poskytovatele služby, bakalářská práce, Brno, FIT VUT v Brně, 2010

Bezpečnost webových služeb z pohledu poskytovatele služby

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Mgr. Marka Rychlého.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Stanislav Liška
18.5.2010

Poděkování

Chtěl bych moc poděkovat vedoucímu bakalářské práce Mrg. Marku Rychlému za velkou trpělivost a odbornou podporu při psaní této práce. Dále děkuju mé přítelkyni a přátelům za podporu.

Obsah

1 Úvod

Jednou ze základních vlastností informačního světa byla a je komunikace. Již od počátku se vyvíjela zároveň s výkonem počítačů a děje se tak neustále. Fenoménem posledních letech se stal internet. Ten se začal značně používat ke komerčním účelům a stejně jako se postupně rozšiřovali jeho možnosti, tak vznikaly požadavky na jednoduché a bezpečné prostředí komunikace pro tyto účely. Komunikace po síti je zajištěna pomocí různých protokolů a mechanismů, které byly často vyvinuty pro různé hardwarové platformy, operační systémy, speciální aplikace, aj. Časem vznikl požadavek na sjednocení a standardizaci ohledně poskytování služeb uživatelům prostřednictvím komunikace po síti a výsledkem byly webové služby. Ty umožňují komunikaci poskytovatele služby s jejím spotřebitelem v různých prostředích.

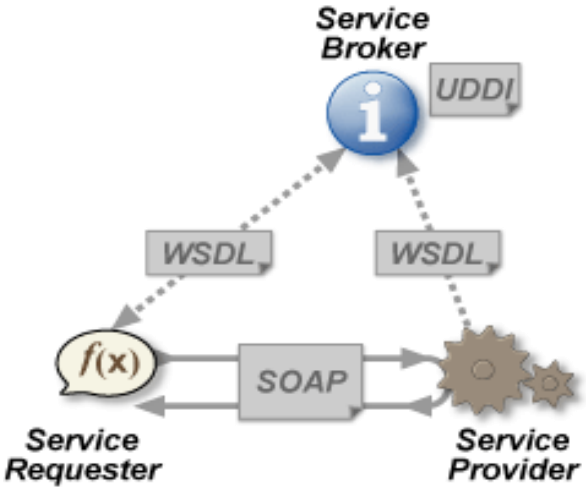
Webové služby se začaly těšit oblibě a postupně zasahovaly do dalších odvětvích, kde ale bylo kromě jejich předosti zapotřebí ještě jedné vlastnosti a tím byla jejich bezpečnost. Tuto otázku začalo řešit více firem a to vytvářelo zpočátku problém, neboť každý prosazoval svoje proprietální řešení místo cesty jednotné standardizace. Nakonec se ale tři firmy, jmenovitě IBM, Microsoft a VeriSign dohodli na spolupráci na vývoji standardizovaného řešení bezpečnosti webových služeb. Jejich první specifikace s názvem Web Service Security nebyla schválena žádnou institucí poskytující standardy, neboť připomínala více firemní standard těchto společností, ale postupem času se o tuto specifikaci začalo zajímat a využívat čím dál více firem.

Proto nakonec došlo 19. Dubna 2004, na druhý pokus, ke standardizaci této specifikace institutem OASIS a vznikl standard WS-Security 1.0, který v sobě zahrnuje dalších několik standardů. Prvním z nich je SOAP Message, dále řada token profiles, specifikace zajišťující autentizaci mezi spotřebitelem a poskytovatelem služby, specifikace definující bezpečnostní politiku systému. Struktura většiny těchto standardů je postaven na formátu XML.

2 Webové služby

Nejprve si řekneme, co přesně webová služba představuje. Definic vyjadřujících webové služby je více a my si řekneme následující od institutu W3C(převzato z [4]): „Webová služba je softwarový systém navržený pro podporu stroj – stroj interakcí po síti. Má rozhraní popsané v strojově zpracovatelném formátu (konkrétně WSDL). Ostatní systémy interagují s webovou službou způsobem předepsaným její definicí zprávou SOAP, typicky přepravených pomocí HTTP s XML serializací ve spojení s jinými webovými standardy."

Jednoduše řečeno, webová služba je běžící aplikace na serveru, která je dostupná pro vzdáleného klienta, tzv. spotřebitele služby, typicky pomocí protokolu http a formátu XML. Architektura webových služeb je postavena na protokolech SOAP, WSDL a UDDI v syntaxi jazyka XML, díky kterým jsou snadno strojově zpracovatelná. V komunikaci webových služeb jsou v interakci 3 účastníci a to jsou poskytovatel služby, registr služby a spotřebitel služby.



Obrázek 1: Architektura webových služeb (převzato z [3])

Poskytovatel služby(Service Provider) je subjekt poskytující webovou službu hardwarovým či softwarovým agentem. Zajišťuje provoz služby. Zveřejňuje definici služby pomocí protokolů UDDI a WSDL.

Registr služby(Service Broker) představuje databázi obsahující informace o webových službách a jejich poskytovatelích. V těchto registrech uživatelé mohou vyhledávat poskytovatele a informace o poskytovaných službách.

Spotřebitel služby(service requestor) představuje toho, kdo chce využívat službu a její metody. Může to být aplikace běžící ve webovém prohlížeči, která metody webové služby volá, ale i aplikace které nemají vlastní rozhraní a to nahrazuje právě prostřednictvím webových služeb.

2.1 Protokol SOAP

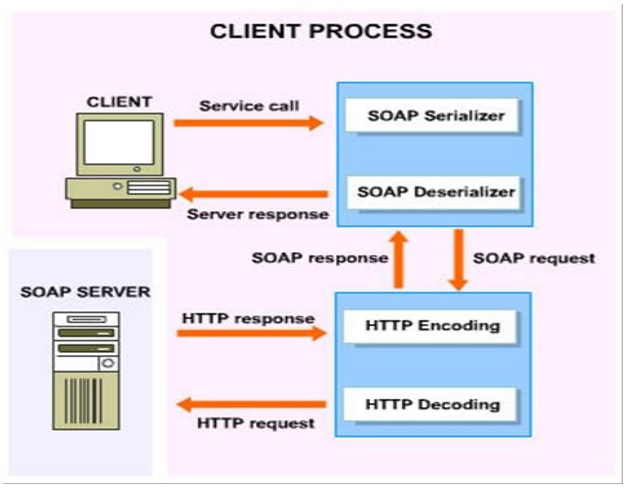
Je základní komunikační protokol webových služeb ve formátu XML a byl navržen v roce 1998 společností Microsoft. Zpočátku nebyl přijat z nadšením, ale postupem času začaly na vývoji spolupracovat firmy jako IBM, Developmentor a další. Vznikla verze specifikace SOAP 1.1 a již byla standardizována institutem W3C v roce 2000 jako protokol pro výměnu informací v různých softwarových a hardwarových prostředí. Došlo k dalšímu rozšíření o transportní protokoly jako jsou FTP, SMTP atd. Poslední verzí protokolu SOAP je verze SOAP 1.2.

Protokol SOAP tvoří základní vrstvu technologie webových služeb. Je to bezstavový protokol nezávislý na protokolu komunikace a implementaci. Definuje strukturu zprávy, tzn. obálku kolem hlavičky a těla.Následující texty vychází ze zdroje [12], [26] a [6].

2.1.1 Komunikace protokolu SOAP

Protokol SOAP využívá nejčastěji pro komunikaci protokol HTTP, případně zabezpečený HTTPS. Z toho plyne, že model komunikace je založen na systému požadavku a odpovědi.

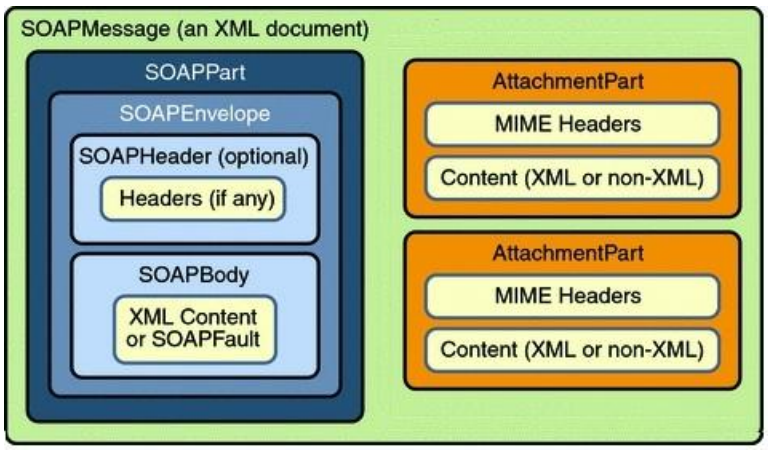
Princip komunikace protokolu SOAP je následující: klient pošle SOAP zprávu na server přes HTTP protokol. Zpráva je ve formátu XML a obsahuje popis činnosti protokolu SOAP, které se vyhodnotí na straně serveru Server následně pošle odpověď, či chybovou zprávu, klientovi ve stejném formátu.



Obrázek 2: Model komunikace protokolu SOAP (převzato z [6])

2.1.2 Zprávy protokolu SOAP

Všechny SOAP zprávy musí splňovat definovanou strukturu ve formátu XML.Základem je kořenový element Envelope (obálka) a uvnitř tohoto elementu se nachází povinný element Body(tělo).Ten může obsahovat volitelný element Header (hlavička), pomocí kterého přenášíme pomocné informace, jako je např. určení digitálního podpisu, uživatele, atd..Dále může zpráva obsahovat přílohy.



Obrázek 3: Struktura SOAP zprávy s přílohou (převzato z [12])

2.1.3 SOAP požadavek a odpověď

V této podkapitole si ukážeme, jak vypadá struktura odesílaných a přijímaných zpráv. Začneme nejprve SOAP zprávou obsahující požadavkem. Jak již bylo zmíněno, základní struktura obsahuje povinný element <Envelope>, uvnitř se nachází povinný element <Body>. V něm je požadavek GetStockPrice s parametrem Stockname.

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPrice>
    <m:StockName>IBM</m:StockName>
  </m:GetStockPrice>
</soap:Body>

</soap:Envelope>
```

Obrázek 4: Ukázka zprávy SOAP požadavku (převzato z [5])

Dále si uvedeme, jak by vypadala SOAP zpráva obsahující odpověď na předchozí požadavek. Strukturu tvoří opět element <Envelope> a <Body>, ve kterém se nachází odpověď GetStockPriceResponse s parametrem Price a hodnotou 34.5.

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Body xmlns:m="http://www.example.org/stock">
  <m:GetStockPriceResponse>
    <m:Price>34.5</m:Price>
  </m:GetStockPriceResponse>
</soap:Body>

</soap:Envelope>
```

Obrázek 5: Ukázka zprávy SOAP odpovědi (převzato z [5])

Volání služby může kromě odpovědi vrátit i chybové hlášení. To se projeví vložením elementu <fault> do elementu <Body>. V elementu <fault> můžou být další elementy popisující chybu. Jsou to elementy:

- <faultcode>
- <faultstring>
- <faultactor>
- <detail>

Kódy chyb jsou:

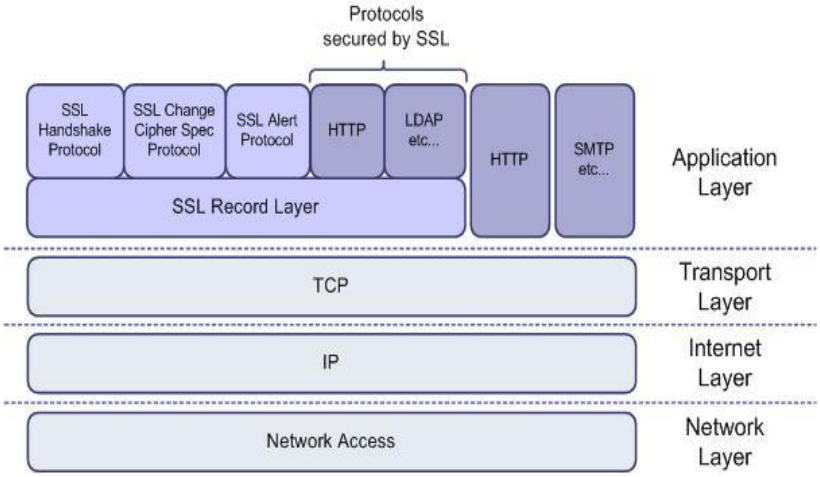
- VersionMismatch - nalezen neplatný jmenný prostor pro element SOAP obálky.
- MustUnderstand - pro následující dceřinný element elementu Header, s nastavením atributu mustUnderstand na jedna, příjímací strana zcela neporozomněla obsahu elementu.
- Client – zpráva má špatný formát nebo obsahuje nekorektní informace.
- Server – problém na straně serveru.

3 Zabezpečení webových služeb

Cílem této kapitole je seznámit čtenáře s obecnými technologiemi a možnostmi zabezpečení webových služeb. Samotný bezpečnostní protokol, není sám o sobě, nijak odlišný od ostatních komunikačních protokolů, ale má v sobě rozšíření pro zabezpečení a ochranu dat při přenosu.
Nejprve si uvedeme dva nejpoužívanější protokoly zabezpečení, jako je protokol SSL a rozšíření SOAP protokolu v podobě SOAP Message security. V dalších podkapitolách se budeme zabývat možností autentizace uživatele webových služeb a vzájemných porovnáním výhod a nevýhod. Bezpečností protokoly

3.1.1 Protokol SSL

Zkratka SSL znamená Secure Socket Layer a je to protokol pracující mezi transportní a aplikační vrstvou, viz Obrázek 6, text je inspirován [7] .

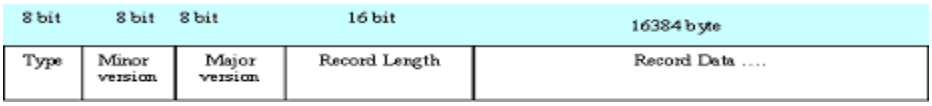


Obrázek 6: SSL (převzato z [7])

Byl vyvinut firmou Netscape a zajišťuje bezpečnou komunikaci a autentizaci mezi klientem a serverem. Každá strana má svoji dvojici šifrovacích klíčů označovaných jako soukromý a veřejný klíč. Princip je následující, např. pokud klient posílá zprávu serveru, použije jeho veřejný klíč k zašifrování a tím je zajištěno, že tuto zprávu rozšifruje pouze párový soukromý klíč serveru.
Protokol je hojně využívaný pro zabezpečení komunikace po internetu pomocí protokolu HTTPS, který je kombinací protokolu HTTP a SSL. Dále si popíšeme jednotlivé části protokolu, text bude čerpat ze zdrojů [8] a [19].

3.1.1.1 Struktura protokolu SSL

Protokol SSL není jeden protokol, ale skládá se s více podpůrných protokolů. Prvním z nich je SSL Record protokol. Ten zajišťuje zabalení dat protokolů vyšší vrstvy a části protokolu SSL. Hlavička protokolu je uvedena na obrázku 7.



Obrázek 7: SSL Record Protokol(převzato z [8])

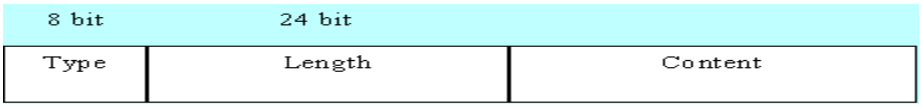
Hlavička je dlouhá 5 bytů a obsahuje následující pole:

- Type (8 bit) – určuje datový typ a protokol vyšší vrstvy, který má data zpracovat. Možné typy jsou:
  - change_cipher_spec - změna specifikace šifrování
  - alert - výstraha
  - handshake
  - application_data - aplikační data
- Version (16 bit) – obsahuje informace o verzi SSL protokolu (major i minor).
- Length (16 bit) – obsahuje informaci o délce datového pole.
- SSL Record Data – obsahují přenášená data.

Data se musí před odesláním příjemci upravit do určité podoby a následně zašifrovat. Tato změna dat se popisuje ve 4 fázích:

- Fragmentace - dělí data do SSL rekordů a maximální délce 16 MB nebo menších.
- komprimace (volitelně) - dochází ke komprimaci dohodnutým protokolem.
- Aplikace MAC (ověřovací kód zprávy) – dojde k připojení ověřovací informace MAC, což je dohodnutá hashovaní funkce. Může to být algoritmus MD5 s délkou 128 bitů nebo SHA s délkou 160 bitů.
- Šifrování – Proces šifrování. Je na výběr s více šifrovacích algoritmů. Máme dvě hlavní kategorie, proudové a blokové šifrování.

Druhým pomocným protokolem je SSL Handshake. Občas je nazýván jako key-exchange protokol. Protokol slouží pro domluvení zabezpečené komunikace mezi oběma účastníky. Definuje formát, jakým budou posílána data. Zajišťuje autentizaci serveru klientem pomocí certifikátů a volitelně i naopak. Rovněž ustanovuje výběr šifrovaní mezi účastníky a metody šifrování s veřejným klíčem. Také se stará o ustanovení zabezpečeného spojení SSL.

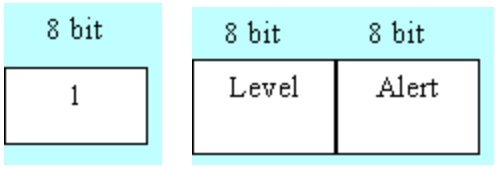


Obrázek 8: SSL Handshake Protokol(převzato z [8])

Hlavička protokolu je na Obrázku 8. Obsahuje tyto pole:

- Type - obsahuje typ zprávy SSL handshake
- Length - určuje délku zprávy (v bytech)
- Content - obsahuje parametry přidané ke zprávě

Třetí podpůrný protokol se jmenuje SSL Change Cipher. Má délku jeden byte, viz Obrázek 9. Jeho úkolem je zajistit účastníkům ukončení používání vyčkávacích algoritmů a přejít do provozního stavu, tzn. začnou platit předem definované šifrovací a ověřovací algoritmy.



Obrázek 9: SSL Change Cipher Protokol a SSL Alert Protokol(převzato z [8])

Poslední používaný protokol je SSL alert. Jeho úkolem je předávat informace o chybách objevujících se během spojení. Výstrahy mají dvě úrovně a to fatální a varovné. Po fatální výstraze se spojení ukončí. U varovné pokračuje, ale session ID je označeno jako neplatné. Hlavička je na Obrázku 4 a obsahuje tyto pole:

- Level - indikuje fatální nebo varovnou výstrahu
- Alert - indikuje specifickou výstrahu

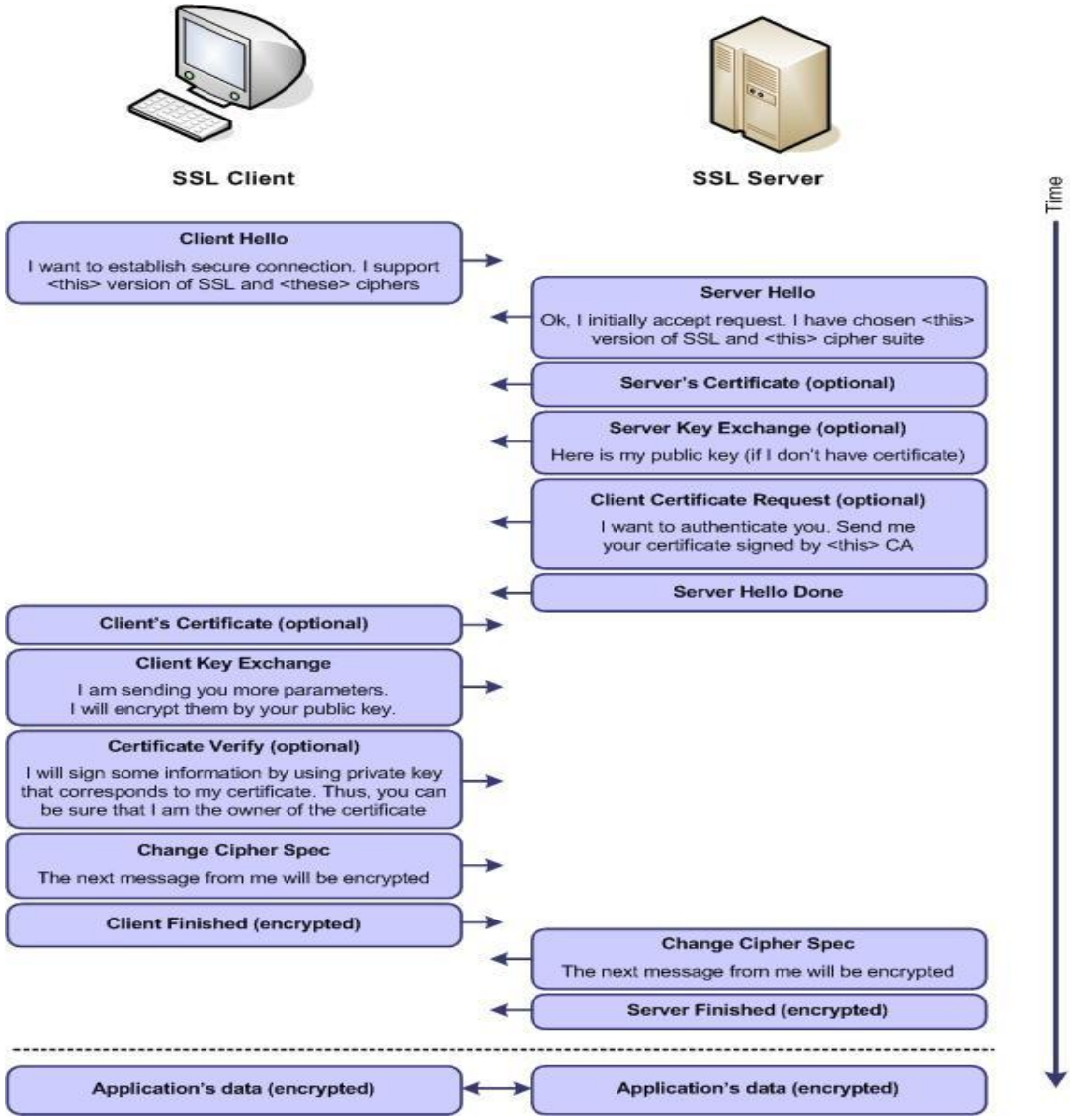
3.1.1.2 Princip komunikace SSL

Nyní si popíšeme podrobněji princip komunikace. Nejprve se musí ustanovit spojení pomocí protokolu SSL Handshake. Další kroky budou následující(převzato z [27]):

1. Klient zasílá serveru požadavek na SSL spojení a s ním další různé doplňující informace, jako je např. verze SSL, šifrování, atd.
2. Server pošle zpátky klientovi odpověď na jeho požadavek, který obsahuje stejný typ informací a také certifikát serveru. Server může požadovat i autorizaci od klienta a pak zasílá požadavek o klientský certifikát.
3. Klient si ověří podle přijatého certifikátu od serveru jeho autentičnost. Zároveň obsahuje certifikát také veřejný klíč serveru.
4. Následně klient vygeneruje pomocí obdržených informací základ šifrovacího klíče pro komunikaci a zašifruje ho veřejným klíčem serveru a pošle mu ho.

5. Server pomocí soukromého klíče rozšifruje zprávu od klienta a z vygenerovaného základu od klienta . Z tohoto základu vygenerují server a klient hlavní šifrovací klíč komunikace.
6. Klient a server si vzájemně potvrdí, že nyní bude jejich komunikace šifrovaná tímto klíčem. Fáze handshake tímto končí.
7. Nyní je již ustaveno zabezpečené spojení šifrované vygenerovaným šifrovacím klíčem.
8. Aplikace můžou komunikovat tímto spojením.

Tyto kroky můžeme vidět i na Obrázku 10 .



Obrázek 10: SSL komunikace klient-server (převzato z [7]).

3.1.2 Specifikace WS-Security

Tento návrh vznikl spoluprácí společností IBM, Microsoft a Verisng a je tuto specifikaci se stará standardizační skupina OASIS. Samotný SOAP protokol otázku bezpečnosti neřešil a nechával prostor pro ostatní protokoly. Webové služby dosáhli velké oblíbenosti a jak se začali využívat i v oblastech, kde byla bezpečnost klíčová, došlo k vytvoření specifikace WS-Security. Díky tomu je zajištěna kvalitní ochrana pomocí integrity zpráv, možností utajení a autentizace každé zprávy.
Dále je umožněno kombinování SOAP zpráv s dalšími bezpečnostními prvky, s tzv. security token. Typ těchto tokenů není předem určen. Specifikace WS-Security nezajišťuje sama o sobě bezpečnostní řešení, ale podobně jako je SOAP protokol základním kamenem přenosu webových služeb, tak je WS-Security základem pro jiné protokoly a technologie zajišťující bezpečnost.Text vychází ze zdroje [9].

3.1.2.1 Elementy WS-Security

Nové mechanismy zabezpečení jsou realizovány definováním dalších elementů SOAP zpráv. Tím hlavním je element <wsse:Security>, který poskytuje možnost připojení bezpečnostních informací určených pro příjemce. Příjemcem může být konečný adresát, ale i prostředník. Tento element je obsažen v elementu <Header>.Struktura SOAP zprávy může vypadat např. takto:

```
<S11:Envelope>
    <S11:Header>
            ...
        <wsse:Security S11:actor="..." S11:mustUnderstand="...">
            ...
        </wsse:Security>
            ...
    </S11:Header>
    ...
</S11:Envelope>
```

Obrázek 11: Struktura SOAP zprávy (převzato z [9]).

Element <wsse:Security> může obsahovat další podelementy, které si vysvětlíme a ukážeme na příkladech. Prvním z nich je element <wsse:UsernameToken>, který obsahuje uživatelské jméno. Je-li element prázdný nebo neplatný, dojde k chybě.

```
<S11:Envelope xmlns:S11="..." xmlns:wsse="...">
    <S11:Header>
            ...
        <wsse:Security>
            <wsse:UsernameToken>
                <wsse:Username>Zoe</wsse:Username>
            </wsse:UsernameToken>
        </wsse:Security>
            ...
    </S11:Header>
    ...
</S11:Envelope>
```

Obrázek 12: Struktura SOAP zprávy s UsernameToken (převzato z [9]).

Dalším možným elementem je <wsse:BinarySecurityToken>. Slouží k uložení autentizačních informací, které jsou v jiném formátu než je XML, např. certifikát X.509. Použité šifrování specifikuje atribut EncodingType. Další atribut ValueType definuje, o jaký bezpečnostní token se jedná. Struktura může vypadat takto:

```
<wsse:BinarySecurityToken wsu:Id=...
                          EncodingType=...
                          ValueType=.../>
```

Obrázek 13: Struktura SOAP zprávy s BinarySecurityToken (převzato z [9]).

Element <wsse:SecurityTokenReference> je mechanimus k získání identifikačních dat, které se nenacházejí přímo ve zprávách, ale někde jinde. Používá se při použití digitálních podpisu a šifrování. Poskytuje další podelementy, jako:

- <wsse:Reference>
- <wsse:KeyIdentifier>
- <ds:KeyName>
- <wsse:Embedded>

Struktura může vypadat takto:

```
<wsse:SecurityTokenReference wsu:Id="...">
    ...
</wsse:SecurityTokenReference>
```

Obrázek 14: Struktura SOAP zprávy s SecurityTokenReference (převzato z [9]).

Element <ds:Signature> slouží k ověření autentizace a integrity zprávy. Připojuje podpis, jenž odpovídá specifikaci XML Signature, ke zprávě. Poskytuje další podelementy:

- <SignedInfo>
- <SignatureValue>
- <KeyInfo>

Element < xenc:ReferenceList> slouží k šifrování libovolné části bloku těla a hlavičky zprávy SOAP. Vytváří seznam šifrovaných částí obsažených v element <xenc:EncryptedData>.

Element <xenc:EncryptedKey> nese informaci o použitém šifrovacím klíči. Uvnitř je další podelement <xenc:ReferenceList>, který představuje seznam zašifrovaných částí.

Element <wsu:Timestamp> slouží k vyjádření času vytvoření nebo expirece SOAP zprávy.

3.1.3 Porovnání protokolu SSL a specifikace WS-Security

Nyní se shrneme výhody a nevýhody těchto dvou řešení z hlediska zabezpečení webových služeb. Protokol SSL nám může poskytnou důvěrnost, integritu a standardní možnosti šifrování, digitálních podpisů a certifikátů.

WS-Security poskytuje důvěryhodnost a integritu pomocí XML šifrování a XML podpisů a podporuje celou řadu autentizačních mechanismů, které byli zmíněny v předchozích kapitolách.

Dá se říct, že co se týče možností, jsou na tom jak SSL tak metody WS-Security podobně. Rozdíl vzniká při použití v konkrétních situací. SSL pracuje s HTTP protokolem a pokud tento protokol webová služba nepoužívá, nelze ho použít. Druhou možnou nevýhodou je fakt, že SSL zavádí point-to-point spojení a přes každé místo, kde je potřeba číst data, musí být data rozšifrována.

S pohledu realizace nabízí SSL jednodušší, přizpůsobitelné a snadno konfigurovatelné prostředí než pomocí WS-Security, přesto velmi bezpečné. Co se týče rychlosti zpracování, je na tom SSL díky menší složitosti mnohem lépe než WS-Security. Pokud ale potřebujeme například zabezpečit to, že pouze určití uživatelé mají přístup ke službám použijeme spíše WS-Security standard.

3.2 Autentizace

Autentizace představuje proces ověření uživatele. Je to jeden z bezpečnostních prvků ochrany mnoha služeb, včetně webových, a zabraňuje falšování indenty. Máme čtyři možné druhy autentizace:

- Podle toho co uživatel zná, např. jméno, heslo, pin.
- Podle toho co uživatel vlastní, např. klíče, smart cart.
- Podle toho co uživatel je, např. biometrika.
- Podle toho co uživatel zná, např. odpověď na otázku.

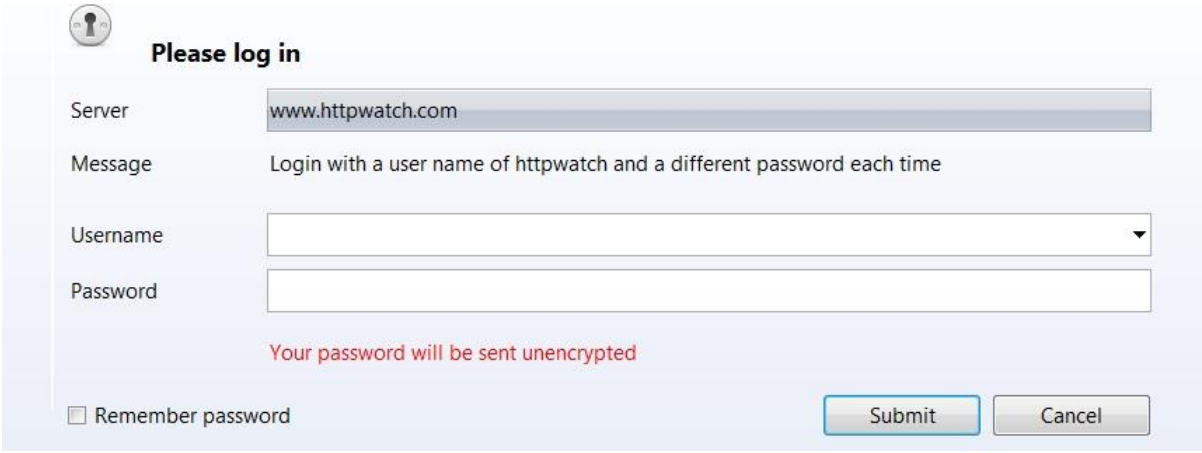
V následujících podkapitolách se seznámíme z možnostmi autentizace webových služeb a na konci provedeme vzájemné porovnání.

3.2.1 Http-auth

Metoda http-auth představuje rozšíření standardního http protokolu o autentizaci část. Existují dvě varianty, Basic Authentication a Digest Acces Authentication, které si následně vysvětlíme. Další možností je formulářová autentizace, využívající dalších bezpečnostních protokolů, jako např. SSL. Následující text bude čerpat ze zdrojů [10], [20], [22] a [23].

3.2.1.1 Basic Authentication

Basic Authentication slouží jako základní ověření přístupu uživatele. Pokud klient vstoupí na takto zabezpečenou stránku, zobrazí se mu výzva k zadání jména a hesla, viz Obrázek 15.



Obrázek 15: Ukázka Basic Authentication (převzato z [10])

Tento mechanismus je založen na využití stavového kódu 401 http protokolu a WWW-Authenticate hlavičce v odpovědi. Tuto hlavičku zašle server klientovi. Klient následně vyplní požadované údaje a pošle serveru, ten pak udělí přístup klientovi.

Samotné údaje jsou před odesláním upraveny. Mezi jménem a heslem je umístěna dvojtečka a jsou zakódovány pomocí formátu Base64. Při další výměně požadavků si webové prohlížeče tyto údaje pamatují a doplňují je většinou automaticky.

Jestliže nejsou zadány údaje správně, bude přístup uživateli odepřen ze stavovou chybou 401.Status kód a WWW –Authenticate hlavička v odpovědi bude vypadat takto:

```
HTTP/1.1 401 Access Denied
WWW-Authenticate: Basic realm="My Server"
Content-Length: 0
```

Obrázek 16: Odpověď s odepření přístupu (převzato z [10])

3.2.1.2 Digest Acces Authentication

Tato metoda je založena na podobném principu, jako Basic. Poskytuje větší míru bezpečnosti, neboť jméno a heslo není přenášeno v otevřeném tvaru na rozdíl od Basic Authentication. Navíc server posílá klientovi náhodně generovaný řetězec, který je zašifrován společně se jménem a heslem pomocí algoritmu MD5.

Pokud klient přistoupí na server s touto autentizací, zobrazí se mu přihlašovací formulář. Po jeho vyplnění je odeslán zpět na server pro ověření údajů. Tento formulář vypadá stejně, jako u předchozí metody, viz. Obrázek 15.

3.2.2 Certifikát X.509

Certifikáty slouží pro ověření identity vlastníků šifrovacích klíčů. Jeden z nejrozšířenějších formátů se stal standard X.509, jehož předchůdcem byl standard X.500. Byl vydán v roce 1988 a popsán v RFC 3280.

Certifikáty nachází uplatnění i při komunikaci pomocí protokolu SSL. Certifikáty jednoznačně spojují šifrovací klíče s jejich vlastníkem a umožňují dostatečně ověřit, komu patří.

Poslední verzí standardu X.509 je verze číslo 3. Obecně standart X.509 předpokládá přísnou hierarchii certifikačních autorit, ale v poslední verzi zavádí podporu dalších topologií certifikačních autorit.Jeho struktura je následující(převzato z [13]):

- Číslo verze (version)
- Sériové číslo (serialNumber)
- ID podpisového algoritmu (signature)
- Vydavatel (issuer)
- Platnost (validity)
    - Od (notBefore)
    - Do (notAfter)
- Subjekt (subject)
- Informace o veřejném klíči subjektu (subjectPublicKeyInfo)
    - Algoritmus veřejného klíče subjektu (algorithm)
    - Veřejný klíč subjektu (subjectPublicKey)
- Jednoznačná identifikace vydavatele (issuerUniqueID)
- Jednoznačná identifikace subjektu (subjectUniqueID)
- Rozšíření certifikátu (extensions)
- ID podpisového algoritmu certifikátu (signatureAlgorithm)
- Digitální podpis certifikátu (signatureValue)

Certifikáty jsou vydávány certifikačními autoritami(CA), které potvrzují vlastnictví veřejného klíče uvedenému vlastníkovi certifikátu. Základní předpokladem je, že CA jsou důvěryhodné.

Vlastník může být zadán více způsoby, jako je jméno, email, dns záznam, atd. Certifikační autority vydávají i své vlastní kořenové certifikáty. Ty poskytují důvěryhodnost v rámci dalších firem. Certifikáty se nachází i v rámci webových prohlížečů, které obsahují certifikáty známých společností.

V případě odcizení či zneužití lze certifikát zneplatnit nahlášením certifikátu příslušné certifikační autoritě. Tato činnost se nazývá revokace. Každá CA vydává a pravidelně aktualizuje seznam revokovaných certifikátů, které podepsala. Pokud uživatel ověřuje platnost nějakého certifikátu, musí zkontrolovat, zda se nenachází v těchto seznamech.

3.2.2.1 Vytvoření certifikátu

Šíření a vytváření certifikátů probíhá zcela volně. Vytvoření certifikátu probíhá tak, že nejprve uživatel vygeneruje soukromý a veřejný klíč. Veřejný klíč předá certifikační autoritě a ta ověří totožnost uživatele a následně vydá certifikát. Ten potvrzuje vazbu veřejného klíče s identifikátorem vlastníka klíče, ale nikoliv s konkrétní osobou. Přesto tato vazba v řadech případu dostačuje k autentizaci. Dále je nutné ještě dodat, že samotný certifikát neslouží k autentizaci, k tomu slouží znalost odpovídajícího soukromého klíče a schopnost rozšifrování zpráv.Tento text čerpá z [14].

```
Data:
    Version: 1 (0x0)
    Serial Number: 7829 (0x1e95)
    Signature Algorithm: md5WithRSAEncryption
    Issuer: C=ZA, ST=Western Cape, L=Cape Town, O=Thawte Consulting cc,
            OU=Certification Services Division,
            CN=Thawte Server CA/emailAddress=server-certs@thawte.com
    Validity
        Not Before: Jul  9 16:04:02 1998 GMT
        Not After : Jul  9 16:04:02 1999 GMT
    Subject: C=US, ST=Maryland, L=Pasadena, O=Brent Baccala,
             OU=FreeSoft, CN=www.freesoft.org/emailAddress=baccala@freesoft.org
    Subject Public Key Info:
        Public Key Algorithm: rsaEncryption
        RSA Public Key: (1024 bit)
            Modulus (1024 bit):
                00:b4:31:98:0a:c4:bc:62:c1:88:aa:dc:b0:c8:bb:
                33:35:19:d5:0c:64:b9:3d:41:b2:96:fc:f3:31:e1:
                66:36:d0:8e:56:12:44:ba:75:eb:e8:1c:9c:5b:66:
                70:33:52:14:c9:ec:4f:91:51:70:39:de:53:85:17:
                16:94:6e:ee:f4:d5:6f:d5:ca:b3:47:5e:1b:0c:7b:
                c5:cc:2b:6b:c1:90:c3:16:31:0d:bf:7a:c7:47:77:
                8f:a0:21:c7:4c:d0:16:65:00:c1:0f:d7:b8:80:e3:
                d2:75:6b:c1:ea:9e:5c:5c:ea:7d:c1:a1:10:bc:b8:
                e8:35:1c:9e:27:52:7e:41:8f
            Exponent: 65537 (0x10001)
Signature Algorithm: md5WithRSAEncryption
    93:5f:8f:5f:c5:af:bf:0a:ab:a5:6d:fb:24:5f:b6:59:5d:9d:
    92:2e:4a:1b:8b:ac:7d:99:17:5d:cd:19:f6:ad:ef:63:2f:92:
    ab:2f:4b:cf:0a:13:90:ee:2c:0e:43:03:be:f6:ea:8e:9c:67:
    d0:a2:40:03:f7:ef:6a:15:09:79:a9:46:ed:b7:16:1b:41:72:
    0d:19:aa:ad:dd:9a:df:ab:97:50:65:f5:5e:85:a6:ef:19:d1:
    5a:de:9d:ea:63:cd:cb:cc:6d:5d:01:85:b5:6d:c8:f3:d9:f7:
    8f:0e:fc:ba:1f:34:e9:96:6e:6c:cf:f2:ef:9b:bf:de:b5:22:
    68:9f
```

Obrázek 17: Ukázka certifikátu (převzato z [14])

Tento příklad dekódovaného certifikátu pro www.freeesoft.org byl vytvořen pomocí SLL a vydán firmou Thawte. Obsahuje řadu detailů o vlastníkovy, veřejný RSA klíč,podpis zašifrovaný MD5 a šifrování soukromým RSA klíčem Thawte.

3.2.3 SAML

Zkratka SAML znamená Security Assertion Markup Language. Je to standard organizace OASIS vydaný v roce 2002 a slouží pro výměnu autentizacích a autorizačních údajů mezi doménami, tzn. mezi poskytovateli identity a služeb. Následující text čerpá z [16]. SAML je rámec založený na XML formátu, který obsluhuje autentizaci uživatelů, jejich oprávnění a informace o atributech. Umožňuje entitám vytvářet tvrzení s ohledem na jejich identitu, atributy a oprávnění a zasílat jiným entitám, jako jsou obchodní partneři nebo jiné aplikace.

SAML představuje flexibilní a rozšiřitelný protokol, který se může kombinovat s dalšími standardy. Největší klady spočívají v těchto bodech:

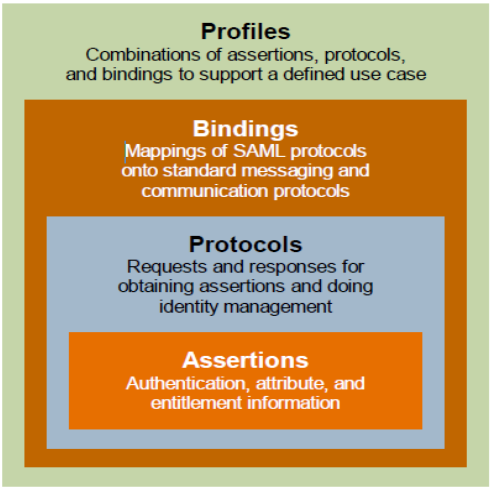
1. Nezávislost na platformě
2. Slabá vazba mezi správci entit
3. Zlepšení podmínek koncového uživatele
4. Snížení administrativních nákladů poskytovatelů služeb

Z prvního bodu plyne, že SAML odděluje bezpečnostní prvky od platformy a tím dosahuje větší nezávislosti na aplikační logice. Tato vlastnost je velmi důležitá v SOA systémech. Díky slabé vazbě nemusí být prováděna synchronizace dat mezi jednotlivými správci. Z třetího bodu plyne podpora technologií single sign-on, která je využívána v systémech, do kterých se přihlašuje uživatel jen jednou a tyto údaje se dál předávají mezi systémy, aniž by o tom uživatel věděl. Zároveň se tím snižují náklady provozovatelů těchto služeb.

3.2.3.1 Struktura SAML

SAML je tvořen s více pohledů, které si nyní stručně uvedeme.

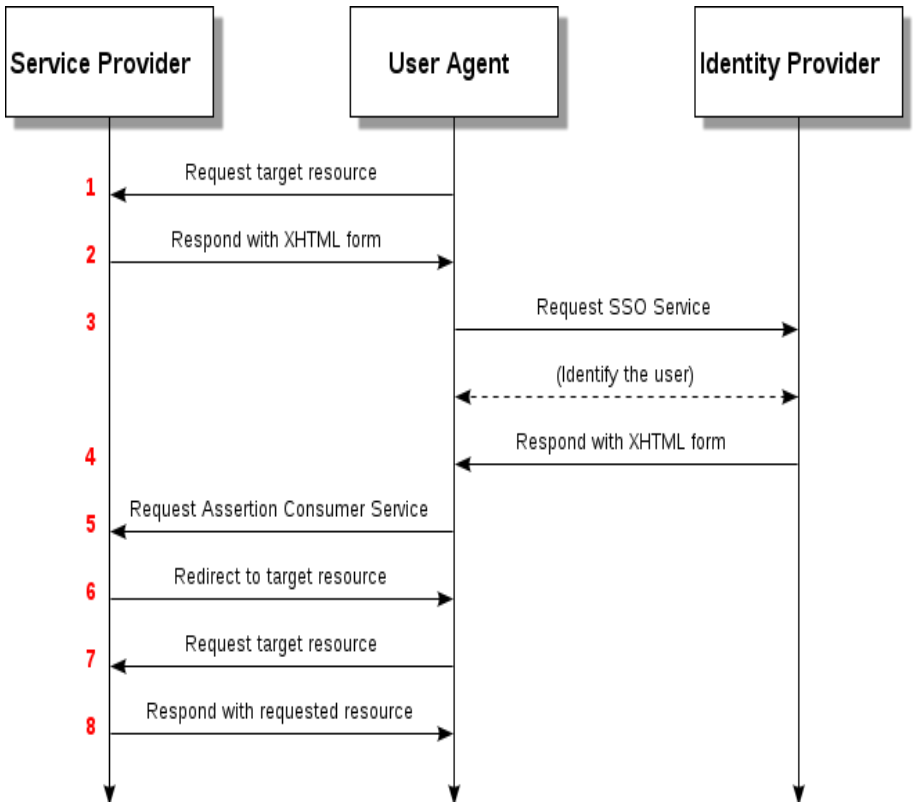
- Profily - definují možnosti omezení a rozšíření SAML v určitých aplikacích.
- Vazby - reprezentují přechody ze stavů SAML request-response do standardních komunikačních protokolů.
- Protokoly – je definováno více typů a umožňují poskytovateli služeb tyto operace.
    - Požadavek na IP autentizaci na vrácení tvrzení.
    - Požadavek na registraci jmenného prostoru.
    - Požadavek SAML autority na jedno a více potvrzení.
- Tvrzení představuje množinu informací, která dodává jedno nebo více sdělení vytvořené SAML autoritou. Rozlišujeme tyto druhy tvrzení:
    - Autentizace - daný subjekt byl v určitém čase určitými prostředky autentizován.
    - Atribut - určuje přiřazení subjektu k atributu.
    - Autorizační informace – reprezentuje povolení nebo zamítnutí požadavku subjektu na zdroj.



Obrázek 18: Vztah mezi pohledy (převzato z [15])

3.2.3.2 Případ využití SAML

SAML může být využit v zabezpečení webových služeb v podobě bezpečného přenosu informací o identitě poskytovatele a spotřebitele služby. Uvedeme si příklad primárního použití SAML nazývaného Sign-ON(SSO). Uživatele představuje webový prohlížeč(User agent) a posílá požadavek na webový zdroj zabezpečený SAML(Service provider). Ten pošle prostřednictvím uživatelského agenta žádost o autentizaci poskytovateli identity SAML. Tok protokolu je znázorněn na následujícím obrázku 19.(Text čerpá z [17])



Obrázek 19: Vztah mezi pohledy (převzato z [17])

3.2.4 REL Token profile

Right Expression Languague Token Profile je mechanismus používaný v SOAP zprávách pro kódování tvrzení a práva jejich použití. Je definován obdobně jako již zmíněné druhy bezpečnostních tokenů v podkapitole o SOAP Message Security. Definuje nové rozšiřující podelementy pro autentizaci a důvěryhodnost základního elementu <wss:security> použitém v hlavičkách zpráv. Následující text bude čerpat z [28].

3.2.4.1 Autentizace

Tento mechanismus definuje několik druhů potvrzení. Je vyžadována oboustranná podpora, od odesílatele a příjemce, požadavku potvrzení <r:keyHolder>. Dále se doporučuje použití XML podpisu pro ustanovení spojení mezi odesílatelem a tvrzením, zvláště pokud není komunikace chráněna. Následně si uvedeme body popisující <r:keyHolder> způsob stanovení korespondence mezi odesílatelem zprávy SOAP a tvrzením v rámci licence.

Odesílatel zprávy musí mít v záhlaví <wsse:Security> element <r:license> obsahující alespoň jeden element <r:grant> obsahující <r:keyHolder>, který identifikuje klíč použítý k potvrzení požadavků. Pokud zpráva odesílatele obsahuje element <r:grant> vícekrát, potom se musí jednotlivé elementy <r:keyHolder> rovnat. Aby mohl příjemce potvrdit přijetí zprávy, musí se odesílatel prokázat znalostí potvrzovacího klíče. Toho může odesílatel dosáhnout potvrzovacím klíčem uvnitř zprávy, která obsahuje element <ds:Signature> uvnitř elementu <wsse:Security> obsažený v hlavičce zprávy.

Element <ds:Signature> vytvořený pro tento účel musí odpovídat kanonizaci a token zahrnující pravidla ve specifikaci WS-Security.

Licence, které obsahují alespoň jedno <r:grant> s podelementem <r:keyHolder> by měla obsahovat element <r:issuer> s elementem <ds:Signature>, který identifikuje vydavatele licence a chrání integritu potvrzení klíče stanoveného podle licence vydavatele.

Za předpokladu, že odesílatel prokázal znalost klíče uvedeného v <r:keyHolder>, pak požadavky(nalezené v licenci), týkající se tohoto klíče, náleží odesílateli. Je-li jedno z těchto tvrzení identita a pokud jsou podmínky tohoto tvrzení splněny, pak elementy ze zprávy, jejíž integrita je chráněna potvrzovacím klíčem, mohou být považovány za autorizované touto identitou.

Následující obrázek 20 ilustruje použití licence bezpečnostní tokenu s <r:keyHolder> zmocnitelem může být použita s <ds:Signature> k prokázání, že John Doe požaduje zprávu o stavu zásob o FOO.

```
<S:Envelope xmlns:S="...">

  <S:Header>
    <wsse:Security xmlns:wsse="...">

      <r:license xmlns:r="..."
licenseId="urn:foo:SecurityToken:ef375268">
        <r:grant>
          <r:keyHolder>
            <r:info>
              <ds:KeyValue>...</ds:KeyValue>
            </r:info>
          </r:keyHolder>
          <r:possessProperty/>
          <sx:commonName xmlns:sx="...">John Doe</sx:commonName>
        </r:grant>
        <r:issuer>
          <ds:Signature>...</ds:Signature>
        </r:issuer>
      </r:license>

      <ds:Signature>
        <ds:SignedInfo>
          ...
          <ds:Reference URI="#MsgBody">
            <ds:DigestMethod
              Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"
            />
            <ds:DigestValue>...</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>...</ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference
              URI="urn:foo:SecurityToken:ef375268"
              ValueType="http://docs.oasis-open.org/wss/oasis-wss-rel-
token-profile-1.0.pdf#license"
            />
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>

    </wsse:Security>
  </S:Header>

  <S:Body wsu:Id="MsgBody" xmlns:wsu="...">
    <ReportRequest>
      <TickerSymbol>FOO</TickerSymbol>
    </ReportRequest>
  </S:Body>
```

Obrázek 20: REL token profile příklad (převzato z [28])

3.2.4.2 Důvěryhodnost

V této podkapitole si popíšeme, jak může licence slouží k ochraně důvěrnosti zprávy SOAP ve WS-Security. WS-Security nedefinuje, jakým způsobem se musí provádět utajení zpráv. Stejně tak i REL umožňuje více typů důvěrnosti, ale stejně jako u autentizace, vyžaduje, aby zpráva odesílatele a příjemce podporovali k utajení elementy <r:keyHolder> a jejich principy. Opět se doporučuje použití XML šifrování komunikace k zajištění důvěrnosti. To se doporučuje obzvláště, pokud probíhá posílaní zpráv SOAP přes nechráněnou komunikaci.

Odesílatel zprávy musí mít v záhlaví <wsse:Security> element <r:license> obsahující alespoň jeden element <r:grant> obsahující <r:keyHolder>, který identifikuje klíč použitý k potvrzení požadavků.

Pokud zpráva odesílatele obsahuje element <r:grant> vícekrát, potom se musí jednotlivé elementy <r:keyHolder> rovnat.

Aby příjemce věděl, kde a jaká data nebo klíče dešifrovat, musí toto odesílatel uvést ve zprávě. Tuto činnost provádí tak, že umístí elementy <xenc:EncryptedData> nebo <xenc:EncryptedKey> na požadovaném místě zašifrování ve zprávě. Dále pak umístí výsledný element <xenc:ReferenceList> nebo <xenc:EncryptedKey> v elementu<wsse:Security> v hlavičce SOAP zprávy.

Elementy <xenc:ReferenceList> nebo <xenc:EncryptedKey> musí být definované v souladu s pravidly stanovenými ve specifikaci jádro WS-Security a tento profil specifikace.

Pokud příjemce zjistí, že má znalosti dešifrovací klíč, uvedeného v <r:keyHolder>, pak může dešifrovat související data nebo klíče. V případě dešifrování klíče, může rekurzivně dešifrovat data nebo klíče, tímto klíčem zašifrované.

3.2.5 Zhodnocení metod autentizace

Název	Výhody	Nevýhody
Basic autentizace	Snadná realizace	Otevřený přenos údajů, lehce napadnutelné
Digest autentizace	Snadná realizace, šifrovaný přenos	Stále napadnutelný přenos údajů, slabé šifrování
Certifikát X.509	Snadná realizace, bezpečný	Certifikáty lze zneužít
SAML	SSO,nezávislost na platformě	Možná náročnost realizace
REL	Využívá WS-Security techniky	Možná náročnost realizace

Tabulka 1 : Zhodnocení metod autentizace

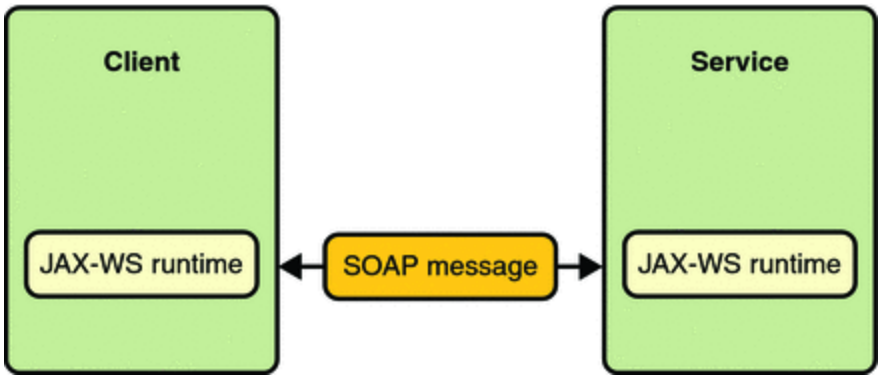
4 Webové služby v prostředí Java EE a jejich zabezpečení

Webové služby v programovacím jazyku Java se v dnešní době tvoří pomocí aplikačního rozhraní JAX-WS (aktuální verze je 2.2), který nahradil dřívější rozhraní JAX-RPC založené na vzdáleném volání procedur. Rozhraní JAX-WS je součástí platformy JAVA EE vyvinuté firmou Sun Microsystems. Toho rozhraní používá anotace pro vývoj služeb a tím se snaží zjednodušit vývoj webových služeb(koncových bodů) a jejich klientů.

Jak jsme si již zmínili, specifikace SOAP definuje strukturu zprávy, pravidla šifrování a konvence pro reprezentaci webové služby metodou vyzývání a odpovědi. Tyto volání jsou vysílány jako zprávy SOAP v podobě XML souborů přes protokol HTTP. Tvorba SOAP zpráv komplexní a náročná a proto se snaží JAX-WS vývojářům jejich tvorbu ulehčit.

Samotná tvorba služby na straně poskytovatele pak spočívá v definicích metod rozhraní zapsaných v jazyce Java. Na straně klienta jsou pak tyto metody volány pres proxy. Nemusí se proto nikde generovat samotné SOAP zprávy nebo dělat jejich rozbor. O převádění API volání a odpovědí na SOAP zprávy se stará právě runtime systém JAX-WS.

Díky tomuto systému u JAX-WS mají webové služby a jejich klienti velikou výhodu v podobě nezávislosti na platformě programovacího jazyka Java. Spotřebitel služby napsaný v jazyce Java a pomocí JAX-WS může přistupovat ke službě, která je na jiné platformě než je Java a naopak. To je možné díky implementaci JAX-WS tak, že využívá standardy W3C jako je http, SOAP a WSDL.

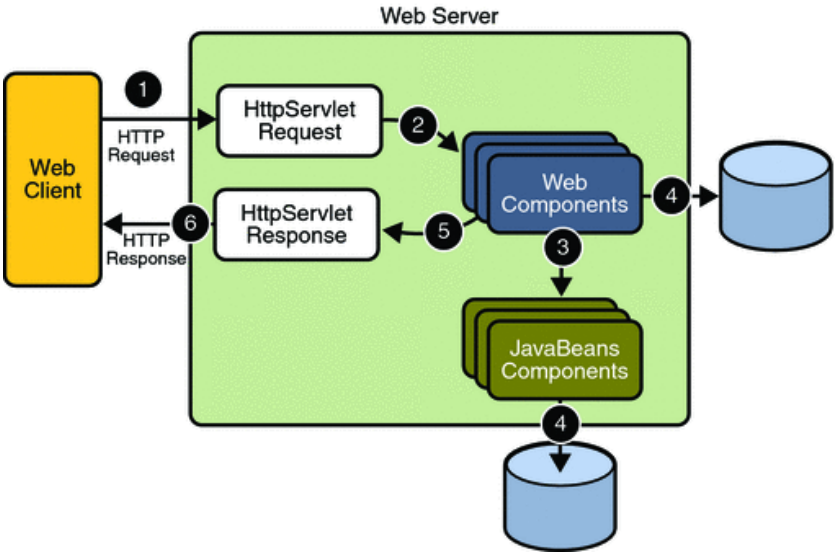


Obrázek 21: Princip komunikace pomoc JAX-WS(převzato z [12])

Referenční implementace JAX-WS je vyvíjena jako open source projekt a je součástí projektu GlassFish, který představuje open source Java EE aplikační server. Tato implementace se nazývá JAX-WS RI a je součástí distribuce Metro.(Text vychází ze zdroje [12])

4.1 Zabezpečení služeb v Java EE

Platforma Java EE nabízí mnoho webových komponent jako je Java Servlet, JSP stránka nebo koncové body webových služeb, které poskytují velké možnosti rozšíření webového serveru. Vzájemná interakce mezi webovou aplikací (službou) a klientem je ukázána na následujícím obrázku. Tyto webové komponenty jsou podporovány tzv. webovým kontejnerem, který představuje runtime službu prostředí a poskytuje další služby jako např. bezpečnost, souběžnost, atd. Tento a následující text vychází ze zdroje [12].



Obrázek 22: Zpracování požadavku v Java EE (převzato z [12])

V určité části zabezpečení webových aplikací může být konfigurováno, kdy má být aplikace instalována či vkládána do webového kontejneru. Anotace pro připojení bezpečnostních prvků jsou společně s popisovači nasazení (deployment descriptors) používány pro předání informací o zabezpečení a dalších aspektech žádostí. Zadaní těchto údajů pomáhá nastavit odpovídající bezpečnostní politiku pro aplikace. Zabezpečení aplikací v Java EE lze provádět 3 způsoby, které si následně popíšeme.

4.1.1 Deklarativní zabezpečení

Deklarativní zabezpečení můžeme provádět pomocí popisovačů nasazení(deployment deskriptorů) nebo také pomocí metadat anotací. Tyto anotace můžeme použít pro specifikování informací o zabezpečení uvnitř třídy souboru. Naproti tomu popisovač nasazení je XML soubor, který je mimo aplikaci a vyjadřuje žádost zabezpečení struktury, včetně bezpečnostních rolí, řízení přístupu a ověření požadavků. Když je aplikace nasazena(deployed), bezpečnostní informace specifikované anotací mohou být potlačeny popisovačem nasazení.

4.1.2 Programové zabezpečení

Programové zabezpečení je vestavěné do webových aplikací a používá se pro bezpečností rozhodnutí. Využívá se v případech, kdy nestačí deklarativní zabezpečení dostatečně vyjádřit bezpečnostní model aplikace, např. pokud bychom chtěli rozlišit přihlášení jen určitých spotřebitelů služby ke koncovému bodu služby v rámci stejného datového toku.
Ve specifikaci JAVE EE verze 6 je umožněna autentizace, přihlášení, odhlášení jako metody rozhraní HttpServletRequest. S přidáním těchto metod do specifikace Servletu, není požadován popisovač umístění pro webové aplikace, ale může být použit k další specifikaci bezpečnostních požadavků nad rámec základních hodnot.

4.1.3 Zabezpečení na úrovni zpráv

Zabezpečení na úrovni(WS-Security) zpráv pracuje s webovou službou a zahrnuje různé bezpečnostní prvky jako jsou digitální podpisy, šifrování uvnitř hlavičky v SOAP zprávě. Je realizováno na aplikační vrstvě a poskytuje bezpečnost na úrovni koncových bodů. Toto zabezpečení ale není zahrnuto jako součást Java EE 6.

4.1.3.1 Projekt Metro

Implementace specifikace WS-Security od firmy Sun's, která stojí i za jazykem Java, je součástí projektu Metro. Ten implementuje tuto specifikaci poskytováním interoperabilních zpráv obsahující integritu a důvěryhodnost.
Metro také poskytuje implementaci specifikace WS-TRUST pro vydávání, obnovování a ověřování bezpečnostních tokenů používaných ve WS-Security a tak zřizuje a ruší vzájemné vztahy. Tato část Metra se nazývá WSIT(Web Services Interoperability technologies). Tento subsystém Metra je implementací mnoha otevřených webových specifikací k podpoře podnikových funkcí, který kromě bezpečnosti přidává spolehlivé zasílání zpráv a operace na atomické úrovni.

5 Návrh a implementace ukázkové služby

Obsahem této kapitoly bude neformální specifikace ukázkové služby a jejího klienta, tzn. spotřebitele služby. Dále si popíšeme výslednou implementaci ukázkové služby.

5.1 Návrh poskytovatele služby

Cílem praktické části této práce je vytvořit ukázkovou službu, která bude následně zabezpečena různými bezpečnostními prvky. Tato služba bude mít jednu třídu s webovou metodou hello(), která po zavolání od spotřebitele služby vrátí spotřebiteli řetězec s pozdravem od webové služby.

5.2 Návrh spotřebitele služby

Spotřebitel služby bude jednoduchá terminálová Java aplikace, která s připojí ke koncovému bodu ukázkové služby a bude volat její jedinou metodu hello(). Předá jí jako parametr metody svůj řetězec obsahující jméno a následně získá zavoláním nový řetězec, který tiskne na výstup.

5.3 Implementace služby

V této části si názorně vysvětlíme samotnou implementaci ukázkové služby. Nejprve začneme nezabezpečenou službou. Dále tuto službu zabezpečíme autentizační metodou http-auth Basic. Třetí služba bude mít zabezpečený přenos dat pomocí SSL protokolu. Jako poslední si uvedeme výslednou ukázkovou službu kombinující obě zabezpečení, tzn. služba, ke které spotřebitel přistupuje pomocí http-auth Basic autentizace a má zabezpečený přenos pomocí SSL.
Na obrázku 14 vidíme výsledný kód třídy implementující nezabezpečenou ukázkovou službu a nyní si ji podrobně popíšeme.

```
package servletws;

import javax.jws.WebMethod;
import javax.jws.WebService;

/**
 *
 * @author Stanislav Liška
 */
@WebService()
public class hello {
    private String message = new String("Web Services here, Hello, ");
    @WebMethod
    public String hello(String name) {
        return message + name + ".";
    }
}
```

Obrázek 23: Implementace ukázkové webové služby

Webová služby je realizována jako třída hello{} s anotací @WebService.Metoda této třídy je pojmenována string hello(String name) a má anotací @WebMethod a musí být public. Třída má privátní proměnnou string message, která reprezentuje řetězec použitý při volání metody spotřebitelem, jako pozdrav od služby.Samotná metoda má jeden parametr name typu string.
Tato implementace ukázkové služby je použita u všech následujících příkladech. Vývoj probíhal za pomocí integrovaného prostředí Netbeans.

5.4 Použité nástroje ke tvorbě služby

Pro tvorbu webových služeb v prostředí Java EE byl použit tento software :

- SUN JAVA JDK 6 update 20 – základní nástroje pro jazyk Java
- JAVA NETBEANS 6.8 – grafické vývojové prostředí
- Glassfish v3 – aplikační server

6 Implementace zabezpečení poskytovatele služby

Možnosti zabezpečení služeb jsme uvedli v úvodu této kapitoly. Pro tuto práci jsem si zvolil deklarativní zabezpečení pomocí modifikace popisovačů nasazení aplikace(deployment descriptors) a v ukázaných příkladech byly modifikovány dva. První z nich je aplikační, jmenuje se web.xml, a pro nezabezpečenou službu vypadá následovně:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
</web-app>
```

Obrázek 24: Web.xml deployment deskriptor nezabezpečené služby

Druhý je runtime popisovač se jmenuje sun-web.xml a vypadá takto:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD
Application Server 9.0 Servlet 2.5//EN"
"http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
<sun-web-app error-url="">
  <context-root>/WS</context-root>
</sun-web-app>
```

Obrázek 25: Sun-web.xml deployment deskriptor nezabezpečené služby

Jak jde vidět s obrázků, pro nezabezpečenou službu skoro žádné informace. Viditelná změna nastane až po implementaci zabezpečovacích prvků služby.

6.1 Implemetace webové služby s http-auth Basic

V této podkapitole si již ukážeme první implementaci zabezpečení webové služby a tím bude autentizace http-auth Basic. Jako první modifikaci můžeme použít anotaci @RolesAllowed, která umožňuje přístup ke službě pouze uživatelům patřící do definované skupiny. V našem případě je to skupina userBasic.

```
package servletws;


import javax.annotation.security.RolesAllowed;
import javax.jws.WebMethod;
import javax.jws.WebService;
/**
 *
 * @author Stanislav Liška
 */
@WebService
public class helloBasic {
    private String message = new String("Web Services with Basic-Auth " +
                                        "here, Hello, ");
    @WebMethod
    @RolesAllowed("userBasic")
    public String hello(String name) {
        return message + name + ".";
    }
}
```

Obrázek 26: Ukázka třídy webové služby s http-auth Basic

Hlavní modifikace pro zabezpečení služby spočívá s doplněním zabezpečovacích elementů do popisovačů nasazení web.xml a sun-web.xml. Následuje přehled přidaných elementů a jejich popis:

- <security-constraint> : představuje základní element pro definici zabezpečení.
- <display-name> : element definující zobrazené jméno služby.
- <web-resource-collection> : element definující zdroj, pro který platí zabezpečení.
- <web-resource-name> : popisný název pro webový zdroj.
- <url-pattern> : element sloužící pro určení jaké http požadavky budou zabezpečeny.
- <auth-constrain> : element pro určení jací uživatelé mají přístup ke službě.
- <role-name> : jméno role, která může přistupovat ke službě.
- <user-data-constraint> : element pro ustanovení chráněného přenosu.
- < Transport-guarantee> : element určující druh přenosu.
- <login-config> : element sloužící pro stanovení druhu přihlašovací metody.
- <auth-method> : element určující druh přihlašovací metody.
- <real-name> : element určující druh souboru s přihlašovacími údaji na serveru.

Po definování těchto elementů bude výsledný popisovač web.xml vypadat takto:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
    <security-constraint>
        <display-name>WS-BASIC</display-name>
        <web-resource-collection>
            <web-resource-name>Everything</web-resource-name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>userBasic</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>NONE</transport-guarantee>
        </user-data-constraint>
    </security-constraint>
    <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>file</realm-name>
    </login-config>
</web-app>
```

Obrázek 27: Web.xml deployment deskriptor zabezpečené služby http-Auth Basic

Poslední změnou pro fungování zabezpečení je přidaní těchto elementu do popisovače sun-web.xml a výsledek je na obrázku 28:

- <security-role-mapping> : element pro mapování role.
- <role-name> : element definující jméno role.
- <group-name> : element definující název skupiny.
- <context-root> : element definující zobrazené jméno kontextu.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD
Application Server 9.0 Servlet 2.5//EN"
"http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
<sun-web-app error-url="">
  <context-root>/WS-BASIC</context-root>
  <security-role-mapping>
    <role-name>userBasic</role-name>
    <group-name>Users</group-name>
  </security-role-mapping>
</sun-web-app>
```

Obrázek 28: Sun-web.xml deployment deskriptor zabezpečené služby http-Auth Basic

Po těchto úpravách se stává aplikace již zabezpečenou .Ještě patří uvést, že na aplikačním serveru je nutné mít nastavené jména, hesla uživatelů a jejich příslušnost ke skupinám pro správné fungování tohoto zabezpečení. Pro zdroj informací ohledně programování tohoto zabezpečení jsem použit zdroj [12].

6.2 Implemetace webové služby s SSL

Další možností zabezpečení webové služby bude použití SSL protokolu pro přenos dat. Ve třídě definující službu se žádná anotace, jak tomu bylo u http-Auth basic, přidávat nemusí. Provedeme změnu pouze v popisovači web.xml přidáním těchto elementů:

- <security-constraint> : představuje základní element pro definici zabezpečení.
- <display-name> : element definující zobrazené jméno služby.
- <web-resource-collection> : element definující zdroj, pro který platí zabezpečení.
- <web-resource-name> : popisný název pro webový zdroj.
- <url-pattern> : element sloužící pro určení jaké http požadavky budou zabezpečeny.
- <user-data-constraint> : element pro ustanovení chráněného přenosu.
- < Transport-guarantee> : element určující druh přenosu.

Výsledný upravený soubor popisovače vypadá následovně:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
    <security-constraint>
        <display-name>WS-SSL</display-name>
        <web-resource-collection>
            <web-resource-name>Everything</web-resource-name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
     <user-data-constraint>
       <transport-guarantee>CONFIDENTIAL</transport-guarantee>
     </user-data-constraint>
   </security-constraint>
</web-app>
```

Obrázek 29: Web.xml deployment deskriptor zabezpečené služby pomocí SSL

Samotné definování anotací je u zabezpečení pomocí SSL jednoduché. Složitost nastává z nutným nastavením cest k certifikátum aplikačním serveru. Při implementaci tohoto zabezpečení jsem narazil na řadu nepříjemností s nastavením správných hodnot proměnných určující tyto cesty v prostředí Netbeans. Pro zdroj informací ohledně programování tohoto zabezpečení jsem použit zdroj [24].

6.3 Implemetace webové služby s http-auth Basic a protokolem SSL

Poslední webová služba, kterou si uvedeme, využívá zabezpečení pomocí autentizace http-auth Basic a zároveň bezpečného přenosu SSL. Představuje zároveň ukázkovou službu a je k ní implementován klient, který bude popsán v další podkapitole.

Tato služba kombinuje dříve zmíněné zabezpečené služby a pro zabezpečení musíme opět měnit popisovače umístění a to tak, že popisovač web.xml obsahuje tyto elementy:

- <security-constraint> : představuje základní element pro definici zabezpečení.
- <display-name> : element definující zobrazené jméno služby.
- <web-resource-collection> : element definující zdroj, pro který platí zabezpečení.
- <web-resource-name> : popisný název pro webový zdroj.
- <url-pattern> : element sloužící pro určení jaké http požadavky budou zabezpečeny.
- <auth-constrain> : element pro určení jací uživatelé mají přístup ke službě.
- <role-name> : jméno role, která může přistupovat ke službě.
- <user-data-constraint> : element pro ustanovení chráněného přenosu.
- < Transport-guarantee> : element určující druh přenosu.
- <login-config> : element sloužící pro stanovení druhu přihlašovací metody.
- <auth-method> : element určující druh přihlašovací metody.
- <real-name> : element určující druh souboru s přihlašovacími údaji na serveru.

Hlavní změnou vůči popisovači umístění zabezpečení pomocí http-auth Basic je změna elementu < Transport-guarantee>, který má hodnotu CONFIDENTIAL a znamená zabezpečený přenos.

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
         http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
         version="3.0">
    <security-constraint>
        <display-name>WS-SSL-BASIC</display-name>
        <web-resource-collection>
            <web-resource-name>Everything</web-resource-name>
            <url-pattern>/*</url-pattern>
        </web-resource-collection>
        <auth-constraint>
            <role-name>userBasic</role-name>
        </auth-constraint>
     <user-data-constraint>
       <transport-guarantee>CONFIDENTIAL</transport-guarantee>
     </user-data-constraint>
    </security-constraint>
       <login-config>
        <auth-method>BASIC</auth-method>
        <realm-name>file</realm-name>
    </login-config>
</web-app>
```

Obrázek 30: Web.xml deployment deskriptor zabezpečené služby pomocí SSL a http-auth Basic

Nakonec upravíme ještě popisovač umístění sun-web.xml pomocí těchto elementů:

- <security-role-mapping> : element pro mapování role.
- <role-name> : element definující jméno role.
- <group-name> : element definující název skupiny.
- <context-root> : element definující zobrazené jméno kontextu.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sun-web-app PUBLIC "-//Sun Microsystems, Inc.//DTD
Application Server 9.0 Servlet 2.5//EN"
"http://www.sun.com/software/appserver/dtds/sun-web-app_2_5-0.dtd">
<sun-web-app error-url="">
  <context-root>/WS-SSL-BASIC</context-root>
  <security-role-mapping>
    <role-name>userBasic</role-name>
    <group-name>Users</group-name>
  </security-role-mapping>
</sun-web-app>
```

Obrázek 31: Web.xml deployment deskriptor zabezpečené služby pomocí SSL a http-auth Basic

6.4 Implementace spotřebitele služby zabezpečené pomocí http-auth Basic a SSL

Na závěr implementační části si ukážeme realizaci spotřebitele služby neboli klienta služby zabezpečené pomocí http-auth a SLL protokolu.Klient je tvořen hlavní třídou Main, která obsahuje třídu main. Klient služby vytvoří instanci třídy helloSerletService a proměnou typu HelloSslBasic reprezentující port služby. Pak zavolá metodu webové služby hello s parametrem typu string. Aby byla autentizace vůči aplikačnímu serveru úspěšná, vloží přihlašovací údaje do požadavku. Po zavolání metody výsledný řetězec tiskne na výstup.

```
package wssslbasicclient;

/**
 *
 * @author godlike
 */
import java.util.Map;
import javax.xml.ws.BindingProvider;
import servletws.HelloSslBasic;
import servletws.HelloSslBasicService;

public class Main {

    public static void main(String[] args) throws Exception {
        try {
            HelloSslBasicService helloServletService = new HelloSslBasicService();
            HelloSslBasic helloServletPort = helloServletService.getHelloSslBasicPort();

            Map<String, Object> reqContext = ((BindingProvider)helloServletPort).getRequestContex
            reqContext.put(BindingProvider.USERNAME_PROPERTY, "godlike");
            reqContext.put(BindingProvider.PASSWORD_PROPERTY, "aaa");
            String result = helloServletPort.hello("Godlike");

            System.out.println("Result = '" + result + "'");
        } catch (Exception e) {
            System.err.println(e.getMessage());
            throw e;
        }
    }
}
```

Obrázek 32: Ukázka implementace spotřebitele služby zabezpečené služby

7 Testování a rozšíření implementovaných metod zabezpečení

7.1 Testování

Nejprve proběhlo testování za pomocí aplikačního serveru Glassfish, který přímo umožňuje spuštění služby pomocí JSP stránky v prohlížeči a zobrazení pozdravu Hello World. Nezabezpečená služba zobrazila pozdrav Hello World okamžitě. Pokud byla webová služba zabezpečená pomocí http-auth, v prohlížeč požadoval zadání přihlašovacích údajů, které museli být zadány a souhlasit z údaji o uživatelích na aplikačním serveru. Služba zabezpečená pomocí SSL zobrazila pozdrav Hello World pomocí zabezpečeného protokolu https. Hlavní ukázková služba, která je kombinací předchozích zobrazila pozdrav po zadání příslušných přihlašovacích údajů rovněž pod protokolem https.
Dále v rámci testování byly pro každou webovou službu implementováni spotřebitelské aplikace na způsob ukázkového klienta služby. Tito terminálovi klienti zobrazovali na výstupu pozdrav od příslušné služby.

7.2 Možnosti rozšíření

Ukázková aplikace využívá rozšířených možností zabezpečení. Může ale být zabezpečena pokročilejšími způsoby, jako je oboustranná autentizace mezi aplikačním serverem a klientem. Místo http-auth by pro větší bezpečnost mohla využívat DIGEST autentizaci. Mohla by být rozšířena i o prvky zabezpečení ze standartu WS-Security prostřednictvím projektu Metro a jeho části WSIT.

8 Závěr

Obsahem této práce bylo seznámit čtenáře s technologiemi webových služeb a možnostem jejich zabezpečení. Byly uvedeny používané protokoly a rozebrány jednotlivé možnosti autentizace. Je potřeba si uvědomit, že pro zabezpečení služby je důležité použití více prvků, tzn. chránit jak samotný přístup k datům tak i samotný přenos dat. Z hlediska vývoje webové služby je důležité rozhodnutí, zda zvolit zaběhnutý SSL protokol spolu s některou metodou autentizace nebo zvolit implementaci za použití standardu WS-Security.

V rámci realizace ukázkové služby jsem zvolil metodu zabezpečení autentizace http-auth Basic a zabezpečeného přenosu SSL. Cílem bylo ukázat, jakým způsobem lze zabezpečit ukázkovou aplikaci. Nejprve za použití zabezpečení služby http-auth Basic, pak za SSL protokolu a nakonec za použití obou prvků zároveň. Dále jsme v poslední kapitole rozebrali testování webové služby a možnosti jejího rozšíření použitím jiných bezpečnostních prvků. Práce pro mě byla přínosná. Objasnil jsem si architekturu a obecné fungování webových služeb. Protokol SSL nebyl pro mě nový, ale standard WS-Security spolu s některými metodami autentizace jako SAML již ano.

Literatura

[1] Rábová, Z., Hanáček, P., Peringer, P., Přikryl, P., Křena, B.: Užitečné rady pro psaní odborného textu [online]. Brno: c1993-2008, aktualizováno 2008-11-01 [cit. 2008-11-28]. Dostupné na URL: <http://www.fit.vutbr.cz/info/statnice/psani_textu.html>

[2] Kolektiv autorů: *Pravidla českého pravopisu*. Academia, Praha, 1993. ISBN 80-200-0475-0

[3] Wikipedia. Webová služba [online]. 11.11. 2006 [cit. 2010-03-25]. Dostupný z WWW: < http://cs.wikipedia.org/wiki/Webová_služba>

[4] Jarchovský, David. Co jsou to webové služby [online]. 17.Juli.2009 [cit. 2010-04-20].Dostupný z WWW: < http://www.viralne.cz/2009/07/co-jsou-webove-sluzby/>

[5] W3school. SOAP Example [online]. 1999-2010 [cit. 2010-04-20]. Dostupný z WWW: < http://www.w3schools.com/soap/soap_example.asp>

[6] Sharat. Explain SOAP messaging model [online]. 24.April.2007 [cit. 2010-04-20].Dostupný z WWW: < http://sharat.wordpress.com/2007/04/24/explain-soap-messaging-model-2/>

[7] Maj, Artur. Apache 2 with SSL/TLS: Step-by-Step, Part 1 [online]. 18.January.2005 [cit. 2010-04-20].Dostupný z WWW: < https://www.securityfocus.com/print/infocus/1818>

[8] Odvárka, Petr. SSL protokol a jeho akcelerace [online]. 9.May.2002 [cit. 2010-04- 20]. Dostupný z WWW: <http://www.svetsiti.cz/view_list.asp?rubrika=Tutorialy&temaID=171>

[9] OASIS. Web Services SOAP Message Security 1.0 (WS-Security 2004) [online]. 1. March 2004 [cit. 2010-04-20]. Dostupný z WWW: <http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0.pdf>

[10] Simtec Limited. 10. HTTP Authentication [online]. 2002-2010 [cit. 2010-04-20]. Dostupný z WWW: <http://www.httpwatch.com/httpgallery/authentication/>

[11] SUN. *The JavaTM EE 5 Tutorial*. 2007: Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054, U.S.A. [online].2007 [cit. 2010-04-20]. Dostupný z WWW: <http://java.sun.com/javaee/5/docs/tutorial/doc/>

[12] SUN. *The JavaTM EE 6 Tutorial,Volume I*. December 2009: Sun Microsystems, 4150 Network Circle, Santa Clara, CA 95054, U.S.A. [online].2009 [cit. 2010-04-20]. Dostupný z WWW: <http://dlc.sun.com/pdf/820-7627/820-7627.pdf>

[13] Bitto, Ondřej. Ukryto pod rouškou X.509 [online]. 10.Juli.2005 [cit. 2010-04-20]. Dostupný z WWW: < http://www.lupa.cz/clanky/ukryto-pod-rouskou-x-509/>

[14] Wikipedia. X.509 [online]. 5.January. 2009 [cit. 2010-04-21]. Dostupný z WWW: < http://cs.wikipedia.org/wiki/X.509>

[15] OASIS. Security Assertion Markup Language (SAML) 2.0 Technical Overview [online]. 20. February 2005 [cit. 2010-04-20]. Dostupný z WWW: < http://www.oasis-open.org/committees/download.php/22553/sstc-saml-tech-overview-2%200-draft-13.pdf >

[16] FERENC, Jakub. Bezpečnost ve službově orientované architektuře [online]. 2008 [cit. 2010-04-21]. Dostupný z WWW: <http://is.muni.cz/th/98993/fi_m/dp.pdf>
[17] Wikipedia. Security Assertion Markup Language [online]. 10.September. 2004 [cit. 2010-04-21]. Dostupný z WWW: < http://en.wikipedia.org/wiki/Security_Assertion_Markup_Language>
[18] OASIS. SAML V2.0 Executive Overview [online]. 12. April 2005 [cit. 2010-04-21]. Dostupný z WWW: < http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf >
[19] Bitto, Ondřej. Skryté vnady protokolu SSL [online]. 1.Juli.2005 [cit. 2010-04-21]. Dostupný z WWW: < http://www.lupa.cz/clanky/skryte-vnady-protokolu-ssl/>
[20] Měcháček, Jaroslav. Autentizace na webu: ÚVT MU [online]. December 2009 [cit. 2010-04-21]. Dostupný z WWW: < http://www.ics.muni.cz/zpravodaj/articles/173.html>
[21] Kouřil, Daniel. Autentizace na webu: ÚVT MU [online]. April 2000 [cit. 2010-04-21]. Dostupný z WWW: < http://www.ics.muni.cz/zpravodaj/articles/173.html>
[22] Wikipedia. Basic access authentication [online]. 2001-2010 [cit. 2010-04-21]. Dostupný z WWW: http://en.wikipedia.org/wiki/Basic_access_authentication
[23] Wikipedia. Digest access authentication [online]. 2001-2010 [cit. 2010-04-21]. Dostupný z WWW: http://en.wikipedia.org/wiki/Digest_access_authentication
[24] Chan Wai Shing.Using JAX-WS-Based Web Services with SSL: Sun Microsystems, Inc. [online]. 2004-2005 [cit. 2010-05-10]. Dostupný z WWW: http://www.java-tips.org/java-ee-tips/java-api-for-xml-web-services/using-jax-ws-based-web-services-wit.html
[25] W3school. SOAP Tutorial [online]. 1999-2010 [cit. 2010-04-20]. Dostupný z WWW: < http://www.w3schools.com/soap/default.asp>
[26] W3C. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [online]. 27.April.2007 [cit. 2010-04-20]. Dostupný z WWW: < http://www.w3.org/TR/2007/REC-soap12-part1-20070427/>
[27] Wikipedia. Security Socket Layer [online]. 2007-2009 [cit. 2010-04-21]. Dostupný z WWW: < http://cs.wikipedia.org/wiki/Secure_Sockets_Layer>
[28] OASIS. Web Services Security Rights Expression Language (REL) Token Profile 1.1 [online]. 1. February 2006 [cit. 2009-05-11]. Dostupný z WWW: < http://www.oasis-open.org/committees/download.php/16687/oasis-wss-rel-token-profile-1.1.pdf >

Seznam příloh

Příloha 1. CD