VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

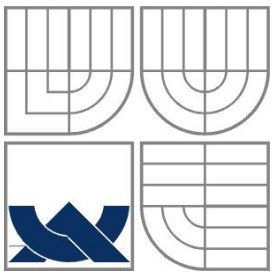
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BEZPEČNOST WEBOVÝCH SLUŽEB Z POHLEDU SPOTŘEBITELE SLUŽBY

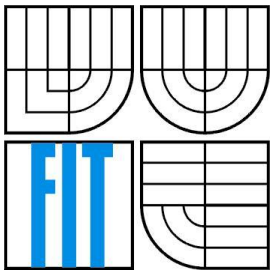
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE PETR NOVOTNÝ
AUTHOR

BRNO 2009



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

BEZPEČNOST WEBOVÝCH SLUŽEB Z POHLEDU SPOTŘEBITELE SLUŽBY

NÁZEV BAKALÁŘSKÉ PRÁCE ANGLICKY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PETR NOVOTNÝ

VEDOUCÍ PRÁCE
SUPERVISOR

Mgr. MAREK RYCHLÝ

BRNO 2009

Abstrakt

Úkolem této práce bylo seznámení čtenáře s různými možnosti webového zabezpečení a její autentizace. Další část práce bylo vytvoření klientské aplikace komunikující se zadanou webovou službou. Klientská část by měla zároveň sloužit pro vývojáře webových služeb kontrolující zvolené webové zabezpečení.

Abstract

The objective of this project was to introduce to a reader different possibilities of a web security and it's authentication. Another part of this project was to create a client application communicates with the web service. The client part should at the some time serve the web services developer for checking chosen web securities.

Klíčová slova

Webová služba, WS, WS-Security, SOAP, Simple Object Access Protocol, SSL, HTTP-Auth, SOAP Message Security, Username, X.509, SAML, REL Token Profile, bezpečnost, webová bezpečnost

Keywords

Web service, WS, WS-Security, SOAP, Simple Object Access Protocol, SSL, HTTP-Auth, SOAP Message Security, Username, X.509, SAML, REL Token Profile, security, web security

Citace

Petr Novotný, Bezpečnost webových služeb z pohledu spotřebitele služby, bakalářská práce, Brno, FIT VUT v Brně, 2009

Bezpečnost webových služeb z pohledu spotřebitele 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.

.....
Petr Novotný
12.5.2009

Poděkování

Rád bych na tomto místě poděkoval vedoucímu bakalářské práce Mgr. Marku Rychlému za odbornou pomoc, vedení a trpělivost při tvorbě této práce, dále pak mé rodině za jazykovou korekturu.

© Petr Novotný, 2009
Tato práce vznikla jako školní dílo na Vysokém učení technickém v Brně, Fakultě informačních technologií. Práce je chráněna autorským zákonem a její užití bez udělení oprávnění autorem je nezákonné, s výjimkou zákonem definovaných případů..

Obsah

1 Úvod

Webové služby v posledních letech, díky stálému růstu uživatelů internetu, zaznamenaly velký rozvoj. Jejich cílem je zajistit vzájemnou a bezproblémovou komunikaci mezi systémy, kde nezáleží na hardwarové platformě, operačním systému, druhu aplikace nebo jinak odlišném prostředí. Všechna tato různá prostředí může mezi sebou žadatel služby a zprostředkovatel služby překlenout za pomocí webových služeb.

S postupem růstu internetu, a tím i rozšiřování webových služeb do dalších odvětví, se tyto protokoly začaly používat i v oblastech, kde je kladen velký důraz na bezpečnost např.: pro vládní informace, bankovnictví a elektronické obchodování. To bylo v počátcích problém, protože webové služby standardně neposkytovaly zabezpečení. Kvůli tomuto problému začaly firmy vytvářet různé vlastní specifikace a standardy. Tento přístup řešení problémů s bezpečností ovšem odporoval základní myšlence webových služeb. To se nelíbilo velkým společnostem a za účelem standardizace se dohodly na vzájemné spolupráci firmy IBM, Microsoft a VeriSign. Vytvořily první verzi standardu Web Service Security, která ovšem zatím nebyla přijata žádnou institucí vydávající standardy, a tím se jednalo spíše o firemní standard pro spolupracující firmy. Velmi brzy tuto specifikaci začaly používat i ostatní firmy a to vedlo k její standardizaci 19. dubna 2004 jako WS-Security 1.0 organizací OASIS. Tato organizace je s W3C standardizační institucí v oblasti webových služeb.

Specifikace WS-Security se skládá z několika standardů. Vedle hlavního standardu SOAP Message Security je jeho součástí řada token profiles určující způsob práce a formát těchto tokenů, specifikace určující autentizaci mezi žadatelem a zprostředkovatelem služby, specifikace určující bezpečnostní politiku systému. Většina standardů WS-Security je postavena na možnosti šifrování XML dokumentů a vytváření digitálních podpisů.

2 Webové služby

Pod názvem webová služba si můžeme představit jakoukoliv běžící aplikaci na serveru dostupnou pro vzdáleného klienta typicky pomocí protokolu HTTP ve formátu XML. To ovšem není úplně přesné, tak si řekněme, jak se definuje pojem webové služby W3C. Webová služba je softwarový systém navržený pro komunikaci mezi dvěma počítači přes síť. Služba má rozhraní popsané ve strojovém formátu (konkrétně WSDL). Ostatní systémy komunikují s webovou službou způsobem předepsaným v popisu pomocí zpráv SOAP, typicky přes HTTP protokol ve formátu XML a ve spojení s dalšími webovými standardy.

Mezi výhody webových služeb a taky důvody, proč byly vyvinuty, patří jejich nezávislost na hardwarové platformě, operačním systému, druhu aplikace nebo jinak odlišném prostředí. Webová služba je pouze rozhraní pro proces, který je spuštěn na vzdáleném počítači, který si žadatel vyžádal. Mezi další výhody webových služeb patří, že jsou standardizovány institucí W3C a tyto standardy jsou otevřeny.

Tato kapitola nám poskytne seznámení, jak v dnešní době zabezpečit tyto webové služby proti neoprávněnému použití a získání chráněných dat, které jsou přenášeny pomocí těchto služeb. V první části se zaměříme především na popis protokolu SOAP a různé způsoby zabezpečení. A v druhé části se podíváme na možné způsoby autentizace.

3 Bezpečnostní protokoly

Bezpečnostní protokol můžeme chápat jako sekvenci operací vedených k zabezpečení nebo k autentizaci dat. Tyto protokoly se používají s transportními protokoly pro ochranu dat mezi komunikujícími stranami. V této kapitole se seznámíme s protokoly SOAP, SSL a HTTP-Auth, které jsou jedny z nejvíce používaných.

3.1 Protokol SOAP

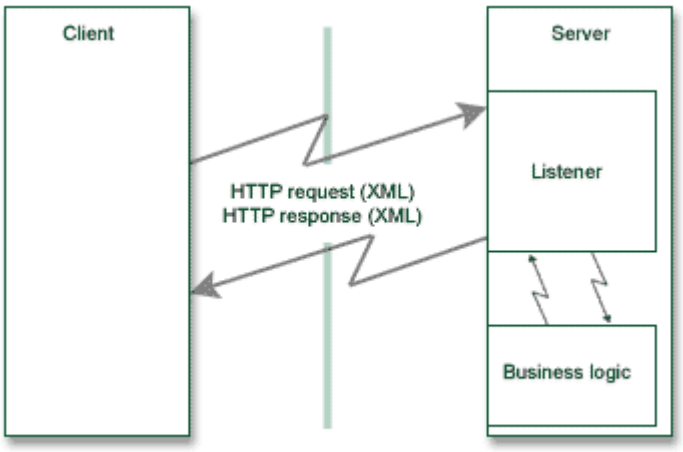
Zkratka SOAP pochází ze slovního spojení Simple Object Access Protocol (jednoduchý přístup k objektům). V dnešní době jsou aplikace vytvářeny tak, aby mohly mezi sebou komunikovat za pomocí internetu. Aplikace mezi sebou většinou komunikovaly za pomocí vzdáleného volání procedur (Remove procedure calls, RPC). Toto řešení má z pohledu dneška svoje velké nevýhody. RPC může mít problém s kompatibilitou systému tak i s webovou bezpečností. Tyto problémy nám řeší protokol SOAP, který byl prvotně zaměřen na vzdálené volání procedur přenášených pomocí transportního protokolu HTTP.
Protokol HTTP byl vybrán díky faktu, že si dobře rozumí s dnešní infrastrukturou internetu. Nemá problémy se většinou dostat přes proxy servery ani přes firewally, protože na většině dnešních počítačů je tento protokol povolen. Protokol samozřejmě funguje i na protokolu HTTPS, který nám probíhající komunikaci šifruje. Protokol HTTPS se skládá ze dvou protokolů se zmíněným HTTP a s protokolem SSL, o kterém si něco bližšího řekneme v sekci o SSL.
Protokol SOAP je založený na XML struktuře a umožňuje připojení klientské aplikace na vzdálenou službu a volání její metody. Služby jsou volány na principu požadavek-odpověď.

3.1.1 Vznik SOAP

Protokol SOAP byl navržen společností Microsoft. Když roku 1998 ohlásila společnost vydání první verze, dostalo se jí dost skeptických reakcí od společností Sun, IBM. Tyto reakce se postupem času uklidnily a společnosti Developmentor, IBM, Lotus and Userland začaly spolupracovat na vývoji protokolu. Toto seskupení navrhlo specifikaci SOAP 1.1 a ta byla přijata jako standard konzorciem W3C v květnu roku 2000 jako protokol pro výměnu informací v různých hardwarových i softwarových prostředích. Tato specifikace byla rozšířena o další transportní protokoly jako SMTP, FTP a další, které dokáží přenášet text. V červnu roku 2003 byla publikována verze SOAP 1.2, která je aktuální dodnes.

3.1.2 Jak SOAP pracuje

Protože SOAP nejčastěji používá protokol HTTP, budeme jeho princip komunikace demonstrovat na něm. Jak už bylo řešeno, protokol standardně používá komunikaci HTTP požadavek-odpověď. Jak vypadá model požadavek a odpověď můžeme vidět na: Obrázek 1. Klient zabalí SOAP zprávu a pošle ji přes HTTP na server. Zprávy jsou XML dokumentem obsahujícím instrukce protokolu SOAP. Tím se zajistilo, že odesílatel i příjemce budou zprávu moci snadno rozparsovat a přečíst jejich jména metod a parametrů nezávislých na použitém typu klienta nebo serveru. Server posílá odpověď, která obsahuje požadovanou odpověď nebo chybovou zprávu, zpět klientovi. Odeslanou zprávu klient rozparsuje a vyhodnotí.



Obrázek 1 – model SOAP komunikace

3.1.3 Popis zprávy SOAP

Každá zpráva SOAP (SOAP message) musí splňovat důležitá pravidla. Zpráva musí být ve formátu XML a musí obsahovat kořenový element Envelope (obálka). V tomto elementu je potom uzavřen povinný element Body (tělo zprávy) a může obsahovat nepovinný element Header (hlavička zprávy). Element Header se používá pro přenos pomocných informací. Může být použit například pro identifikaci uživatele, pro určení digitálního podpisu ve službách, které potřebují ověření klienta, nebo jiného bezpečnostního protokolu. Element Envelope nám definuje XML dokument jako SOAP zprávu. Obsahuje povinný namespace xmlns:soap a atribut encodingStyle, který se používá k definici datového typu. SOAP zprávy nemají defalutní kódování. Příklad struktury SOAP zprávy můžete vidět na: Obrázek 2.

```
<?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:Header>
...
</soap:Header>
<soap:Body>
...
  <soap:Fault>
  ...
  </soap:Fault>
</soap:Body>
</soap:Envelope>
```

Obrázek 2 - příklad struktury SOAP zprávy

3.1.4 SOAP požadavek

Volání SOAP metody požadavku je vidět na: Obrázek 3. Vidíme tady jeden povinný element Envelope, ve kterém se nachází jeden povinný element Body. Jak jsme si mohli všimnout zpráva obsahuje i nepovinný element Header. Elemet Header nám tady slouží pro uživatelskou autentizaci.

Element Body nám nabízí požadavek od klienta. Jedná se o element calcPluc se dvěma subelementami number1 a number2, které mají hodnotu int.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Header>
    <AuthSoapHd xmlns="http://localhost:1846/SoapHeaderAuth/">
      <strUserName>string</strUserName>
      <strPassword>string</strPassword>
    </AuthSoapHd>
  </soap:Header>
  <soap:Body>
    <calcPlus xmlns="http://localhost:1846/SoapHeaderAuth/">
      <number1>int</number1>
      <number2>int</number2>
    </calcPlus>
  </soap:Body>
</soap:Envelope>
```

Obrázek 3 - příklad SOAP požadavku

3.1.5 SOAP odpověď

Jak je vidět na: Obrázek 4 jsou stejně jako v SOAP požadavku elementy Envelope a Body. Element Body obsahuje nový subelement calcPlusResponse, ve kterém se nachází subelement calcPlusResult, který obsahuje námi požadovanou odpověď od serveru.

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
  <soap:Body>
    <calcPlusResponse xmlns="http://localhost:1846/SoapHeaderAuth/">
      <calcPlusResult>int</calcPlusResult>
    </calcPlusResponse>
  </soap:Body>
</soap:Envelope>
```

Obrázek 4 - příklad SOAP odpovědi

Jestliže volání služby způsobilo chybu, objeví se v elementu Body element Fault. Tento element může obsahovat elementy faultcode, faultstring, faultactor a volitelný <detail>.

- faultcode - kód pro identifikaci chyby
- faultstring - srozumitelný popis chyby
- faultactor - informace kdo způsobil chybu
- detail - další informace o chybě

Tento element se vyskytuje pouze v SOAP odpovědi. Příklad zprávy s oznámením chyby můžeme vidět na: Obrázek 5.

```
<soap:Envelope xmlns:soap='http://schemas.xmlsoap.org/soap/envelope/'
soap:encodingStyle='http://schemas.xmlsoap.org/soap/encoding/'>
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Client.InvalidRequest</faultcode>
      <faultstring>
        Neplatný požadavek: Operace není podporována
      </faultstring>
      <faultactor>
        http://marting.develop.com/soap/calcxslt.asp
      </faultactor>
      <detail>
        <m:ChybaMetody xmlns:m='uuid:361C5CDE-FC66-4B17-A2C1-
EB221DEFFD66'>
          <pozadavek>Divide</pozadavek >
          <duvod>Operace není podporována</duvod >
        </m:MethodError>
      </detail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

Obrázek 5 - příklad SOAP odpovědi s elementem Fault

3.1.6 SOAP message security

Protože aplikace začaly potřebovat bezpečnou výměnu SOAP zpráv, začala se vytvářet za tímto účelem rozšiřující specifikace SOAP zprávy. Tato specifikace byla vytvářena s dostatečnou flexibilitou a rozšířitelností bezpečnostních protokolů do SOAP zpráv. Bylo požadováno, aby se mohly používat různé formáty tokenů a různé kontroly důvěryhodnosti domény. Stejně tak byly požadovány neomezující podmínky na použité algoritmy pro digitální podpisy a různé typy šifrování. Dále byl kladen důraz na dosažení bezpečnosti typu end-to-end přenášených. Snadné zabezpečení a rozšíření SOAP zpráv stanovuje specifikace SOAP message security místo pro uložení bezpečnostních informací pro příjemce. Tímto místem je nový element <Security> umístěný do elementu <Header> v SOAP zprávě. Pro tento přidaný element se vztahují stejná pravidla jako pro ostatní hlavičkové elementy. Je možnost obsahovat atributy typu mustUnderstand a actor.

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

Obrázek 6 - Ukázka atributů a elementů v SOAP Message Security (převzato z [15])

3.1.6.1 Usernames

Tento element <wsse:UsernameToken> obsahuje pouze uživatelské jméno a heslo, a pokud je element nevyplněný nebo neplatný, mél by způsobit chybu.

```
<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 7 - ukázka elemetu Username (převzato z [15])

3.1.6.2 Binary security token

Toto zabezpečení je reprezentováno elementem <wsse:BinarySecurityToken>. Jedná se o binární bezpečnostní formáty jako X.509 certifikát vyžadující speciální kódování. Element definuje dva atributy. Atribut ValueType, který definuje o jaký bezpečnostní token se jedná. A atribut EncodingType, který nám definuje, jak je bezpečnostní token šifrován.

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

Obrázek 8 - ukázka elementu BinarySecurityToken (převzato z [15])

3.1.6.3 SecurityTokenReference

Digitální podpisy a šifrovací operace vyžadují speciální klíč. Tento element <wsse:SecurityTokenReference> poskytuje flexibilní mechanismus pro reference bezpečnostních tokenů a dalších klíčových elementů. Element v sobě poskytuje spoustu dalších elementů, jako je <wsse:Reference> element poskytující přímou referenci na bezpečnostní token za pomocí URI adresy. Element <wsse:KeyIdentifier> se používá, když je potřeba jedinečná identifikace bezpečnostního tokenu jako pro hash z významného elementu z bezpečnostního tokenu. Nebo element <wsse:Embedded>, který slouží jako ukazovatel na token umístěný na jiným místě.

```
<wsse:SecurityTokenReference wsu:Id="...", wsse11:TokenType="...",
wsse:Usage="...", wsse:Usage="...">
  ...
</wsse:SecurityTokenReference>
```

Obrázek 9 - ukázka elementu SecurityTokenReference (převzato z [15])

3.1.6.4 Podpisy

Nebo-li také Signatures se používají pro potřebu přijemce se ujistit o autentizaci a integrity zprávy. SOAP Message Security využívá pro tyto účely standard XML Signature a přebírá pravidla pro vytváření a ověřování těchto podpisů, přebírá definované elementy a používané algoritmy. Element <Signature> je základní element podpisů a obsahuje veškeré informace k nim. Může obsahovat

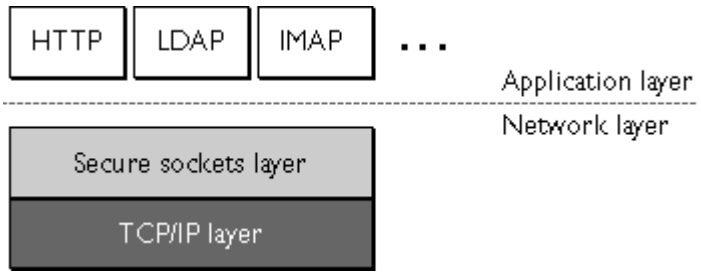
elementy <SignedInfo>, která obsahuje data vytvořená z hashovací funkce a otisků. Element <SignatureValue> obsahující pouze hodnotu podpisu elementu <SignedInfo> zakódované v base64. Dále může obsahovat element <KeyInfo> obsahující informace o klíči potřebné k ověření podpisu.

```
<?xml version="1.0" encoding="utf-8"?>
<S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
xmlns:ds="...">
  <S11:Header>
    <wsse:Security>
      <wsse:BinarySecurityToken ValueType="...#X509v3"
EncodingType="...#Base64Binary" wsu:Id="X509Token">
      MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
      </wsse:BinarySecurityToken>
      <ds:Signature>
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
          <ds:SignatureMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
          <ds:Reference URI="#myBody">
            <ds:Transforms>
              <ds:Transform Algorithm=
"http://www.w3.org/2001/10/xml-exc-c14n#"/>
            </ds:Transforms>
            <ds:DigestMethod Algorithm=
"http://www.w3.org/2000/09/xmldsig#sha1"/>
            <ds:DigestValue>EULddytSo1...</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>
          BL8jdfToEb1l/vXcMZNNjPOV...
        </ds:SignatureValue>
        <ds:KeyInfo>
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="#X509Token"/>
          </wsse:SecurityTokenReference>
        </ds:KeyInfo>
      </ds:Signature>
    </wsse:Security>
  </S11:Header>
  <S11:Body wsu:Id="myBody">
    <tru:StockSymbol xmlns:tru="http://www.fabrikam123.com/payloads">
      QQQ
    </tru:StockSymbol>
  </S11:Body>
</S11:Envelope>
```

Obrázek 10 - ukázka podpisu v SOAP Message Security (převzato z [15])

3.2 Protokol SSL

SSL protokol celým názvem Secure Sockets Layer je protokol pro zabezpečení transportní vrstvy. Jak vidíme: Obrázek 11 pozice protokolu v TCP/IP modelu je mezi vrstvou síťovou a aplikační, která poskytuje zabezpečenou komunikaci za pomocí asynchronního šifrování a s možnou autentizací serveru i klienta. Tento protokol se nejčastěji používá pro zabezpečenou komunikaci na internetu pomocí HTTPS, který je kombinací protokolu HTTP a SSL.
Protokol byl vyvinut společností Netscape za účelem bezpečné komunikace a autentizace mezi klientem a serverem, kdy každá z komunikujících stran má dvojici šifrovacích klíčů tzv. veřejný a soukromý klíč. Veřejný klíč předáme klientovi, s kterým chceme komunikovat. Klient použije klíč k zašifrování zprávy a my budeme mít zajištěno, že ji rozšifruje jen vlastník soukromého klíče.



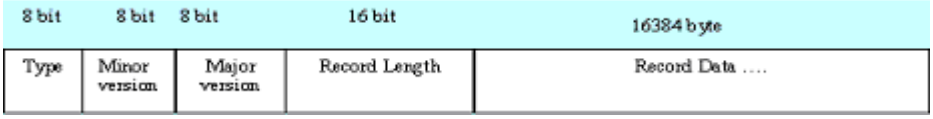
Obrázek 11 - umístění SSL vrstvy (převzato z [28])

3.2.1 Popis protokolu SSL

Protokol SSL se skládá z více takzvaných subprotokolů. Tyto hlavní subprotokoly jsou SSL Record protokol a SSL Handshake protokol. SSL Record protokol má na starosti zabalení dat protokolů z vyšší vrstvy a části protokolu SSL. Protokol SSL Handshake se stará o navázání bezpečné komunikace mezi klientem a serverem. Pro zprávu komunikace ovšem k protokolu SSL Handshake přibývají ještě protokoly SSL Change Cipher Spec a SSL Alert.

3.2.2 Protokol SSL Record

Přenášená data jsou balena do objektu nazývaného record, který obsahuje hlavičku a data. Jak vidíme na: Obrázek 12 hlavička je dlouhá 5bitů. Obsahuje pole Type určující datový typ a protokol vyšší vrstvy, který se má zpracovat. Pole Major a Minor Version obsahují informace o SSL verzi protokolu. Pole Lenght obsahuje informace o velikosti datového pole a pole Record Date obsahují přenášená data.



Obrázek 12 - Hlavička SSL Record protokolu (převzato z [10])

Než se požadovaná data mohou poslat k příjemci, je potřeba je poupravit do specifické podoby a zašifrovat. Tyto úpravy můžeme rozdělit do 4 fází. První fáze je fragmentace. Data se dělí do

textových SSL recordů o maximální délce 16MB nebo menších. Výsledná operace se nazývá SSLPlaintext. Ve druhé fázi jsou data komprimována dohodnutým komprimačním protokolem a výsledná operace se nazývá SSLCompressed. Komprimovaná data nesmí způsobit žádnou ztrátu dat a nesmí překročit maximální délku 16MB. Ve třetí fázi je k SSLCompressed připojena ověřovací informace MAC. Informace MAC je domluvená hashovací funkce. Protokol SSL podporuje dva hashovací algoritmy. Algoritmus MD5 s délkou výsledku 128bitů a SHA s délkou výsledku 160bitů. Poslední fáze, kterou SSL record prochází, než je odeslán, je proces šifrování. SSL protokol podporuje velké množství šifrovacích algoritmů, ale můžeme je rozdělit do dvou kategorií - proudové algoritmy a blokové algoritmy. Proudové algoritmy mají tu výhodu, že nepotřebují doplňovat velikost jako blokové na násobek bloku, který nesmí ovšem překročit 16MB, a mohou být předány ve formátu, v jakém je předala předchozí fáze. Těchto algoritmů je hodně. Ať už se jedná o algoritmy pro výměnu klíčů, jako jsou RSA, Diffie-Hellman, DSA a Fortezza, nebo pro pozdější komunikace RC2, RC4, IDEA, DES a 3DES. Několik algoritmů a jejich popisy můžete vidět v: Tabulka 1. Algoritmy jsou poskládány od nejsilnějšího podporovaného šifrování k nejslabšímu.

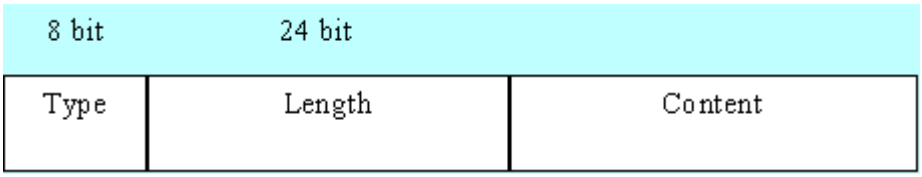
Šifra	Kodování	typ	Auten.	Přibližně klíčů	Verze protokolu	poznámka
3DES	168bit	bloková	SHA-1	$3.7 * 10^{50}$	SSL 2, SSL3	nasazení banky a velmi citlivá data
RC4, RC2	128bit	proudová	MD5	$3.4 * 10^{38}$	SSL 2, SSL3	nasazení obchodní a vládní data, RC2 pomalejší než RC4
DES	56bit	bloková	SHA-1	$7.2 * 10^{16}$	SSL 2, SSL3	silný jako RC4, SSL 2 používá k autentizaci MD5
RC4, RC2	40bit	proudová	MD5	$1.1 * 10^{12}$	SSL 2, SSL3	nejrychlejší, RC2 pomalejší než RC4
Bez šifry			MD5		SSL 3	strany se nedohodly na šifrování, nebezpečný

Tabulka 1 - Nejpoužívanější algoritmy pro šifrování SSL zpráv (převzato z [28])

3.2.3 Protokol SSL Handshake

Tento protokol se občas nazývá také key-exchange protokol, a jak už bylo řečeno, jeho význam spočívá na domluvení bezpečné komunikace mezi oběma účastníky. Mohli bychom říci, že protokol nám definuje, jestli budou data poslána a jakým způsobem. Protokol se stará o ověření serveru klientem za pomocí vydaných certifikátů a zároveň server může ověřovat klienta stejným způsobem. Stará se o vyjednání společného šifrovacího algoritmu, který podporují oba účastníci. O použití metody šifrování s veřejným klíčem pro výměnu šifrovacích parametrů. A nakonec ustavení SSL spojení mezi sebou.

Strukturu protokolu, který se skládá ze tří částí, můžete vidět na: Obrázek 13. Skládá se z pole Type, které nám určuje typ SSL Handshake zprávy. Parametr Lenght nám určuje délku zprávy v bitech a s parametrem Content, informuje o parametrech přidaných ke zprávě.



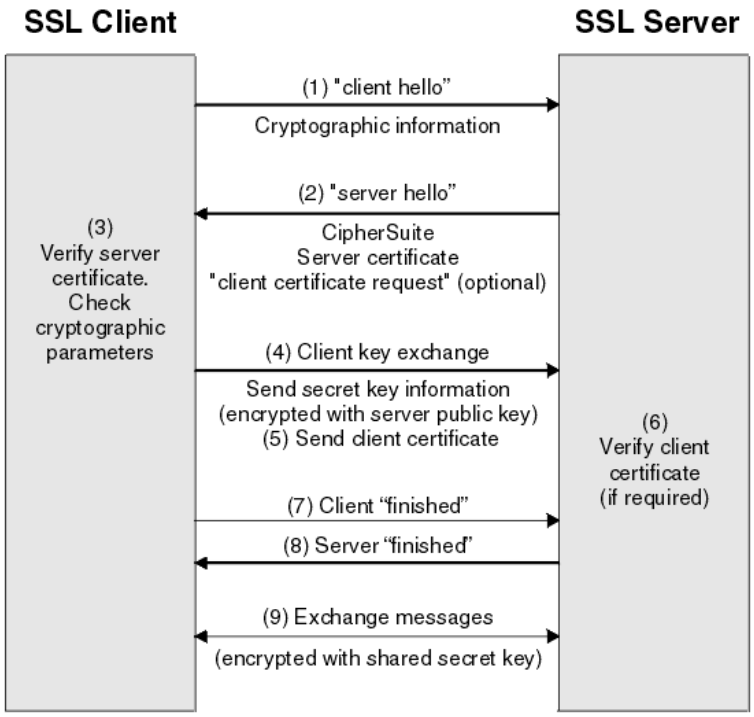
Obrázek 13 - Hlavička SSL Handshake protokolu (převzato z [10])

3.2.4 Protokol SSL Change Cipher Spec a SSL Alert

Tyto protokoly jsou používány v poslední části při domlouvání bezpečné komunikace mezi klientem a serverem.
SSL Change Cipher Spec protokol nám zajišťuje, že klient a server ukončí používání algoritmu výměnou klíčů a začnou používat domluvené šifrovací algoritmy. Protokol má délku jeden bite.
SSL Alert předává informace o chybách v průběhu celého spojení. Chyby mohou být dvou typů - varovné a fatální. Když se objeví fatální chyba, spojení bude okamžitě ukončeno. Protokol má délku dvou bitů.

3.2.5 SSL komunikace

O SSL spojení se stará protokol SSL Handshake a probíhá následujícím způsobem, jak je vidět na: Obrázek 14. Klient pošle serveru požadavek o SSL spojení spolu s různými informacemi jako verzi SSL protokolu, nastavení šifrování, náhodně vygenerovaná data a ostatní věci, které server potřebuje. Server na oplátku pošle podobná data ke klientovi a připojí k tomu svůj certifikát. Pokud server požaduje, aby se klient také autorizoval, pošle i žádost o klientský certifikát. Klient zpracuje poslaná data od serveru a pokusí se o jeho autorizaci. Pokud server nemůže být autorizován, klient dostane varování a informace, že autorizace a šifrování nemusí být ustanoveny. Pokud je server úspěšně autorizován, tak se pokračuje v komunikaci o SSL spojení. Klient zpracuje všechna poslaná data z protokolu SSL Handshake, ve kterém zjistil typ šifrování a další věci. Údaje jsou zpracovány, vytvoří se takzvaný premaster secret, který se zakóduje veřejným klíčem získaným ze serverového certifikátu. Pokud server požadoval ověření klienta, připojí k premaster secret svůj certifikát a odešle tyto informace na server. Když server požadoval ověření klienta, pokusí se ho autorizovat. Jestliže klient nemůže být autorizován, je tato relace ukončena. Když byl klient autorizován nebo autorizace nebyla zapotřebí, server použije svůj primární klíč k dešifrování premaster secret a vytvoří master secret. Server i klient použijí master secret k vytvoření session key, které jsou symetrické klíče k šifrování a dešifrování informací a integrity přenášených dat. Dále si pošlou zprávu o informaci, že zprávy budou šifrovány relačním klíčem, a o ukončení části domlouvání relace pro SSL spojení. Pro další SSL relace bude použit domluvený klíč.



Obrázek 14 - SSL komunikace mezi klientem a serverem (převzato z [29])

3.3 HTTP-Auth

HTTP autentizace byla navržena za účelem identifikace klienta. Jedná se o rozšíření internetového protokolu http, původně určeného na výměnu hypertextových dokumentů po internetu. V současné době obsahuje dvě možnosti autentizace - Basic Authentication a Digest Access Authentication. Pokud se pokusíme vstoupit na stránku chráněnou HTTP autentizací, zobrazí nám prohlížeč systémové okno pro zadání jména a hesla.

3.3.1 Basic Authentication

Tato metoda je nejstarší a nejjednodušší metodou HTTP autentizace a funguje následovně. Klient se pokusí vstoupit na zabezpečenou stránku, ale server mu odpoví chybovým kódem 401 Access Denied a k odpovědi je přidána hlavička WWW-Authenticate: Basic Realm="Název aplikace". Klient si vyžádá jméno a heslo a pokus opakuje, ale do požadavku je přidána hlavička Authorization: Basic username:password. Potřebné uživatelské jméno a heslo pro vstup na stránku, které je odděleno dvojtečkou a zakódováno ve formátu Base64. Server ověří uživatelské jméno a heslo, a jestli je správné, pustí klienta do chráněné stránky. Pokud jméno a heslo není správné, je znovu požadováno. Tyto ověřovací údaje jsou poslány znovu vždy při každém novém požadavku na server. Údaje jsou smazány až při zavření komunikace. Definice schématu pro basic authentication je následující [9]:

```
challenge   = "Basic" realm
credentials = "Basic" basic-credentials
```

REALM: Řetězec, podle kterého uživatel pozná, co má vyplnit za login a heslo. Příklad hodnoty reg-user@fit.vutbr.cz
credentials: zakódované uživatelské jméno a heslo v base64
Pro příklad: pokud bude uživatelské jméno Aladdin a heslo open sesame bude base64 vypadat následovně QWxhZGRpbjpvcGVuIHNlc2FtZQ==

3.3.2 Digest Access Authentication

Princip této metody je stejný jako metoda Basic Authentication Scheme, ale heslo se v tomto případě přenáší v zašifrovaném tvaru za pomocí MD5. Protože se jedná o složitější autentizaci než byla basic, definujeme si jeho schéma [9].

```
challenge = "Digest" digest-challenge
digest-challenge = 1#( realm | [ domain ] | nonce |
[ opaque ] |[ stale ] | [ algorithm ]|
[ qop-options ] | [auth-param] )
domain = "domain" "=" <"> URI ( 1*SP URI ) <">
URI = absoluteURI | abs_path
nonce = "nonce" "=" nonce-value
nonce-value = quoted-string
opaque = "opaque" "=" quoted-string
stale = "stale" "=" ( "true"|"false" )
algorithm = "algorithm" "=" ( "MD5" | "MD5-sess" | token )
qop-options = "qop" "=" <"> 1#qop-value <">
qop-value = "auth" | "auth-int" | token
```

REALM: Řetězec, podle kterého uživatel pozná, co má vyplnit za login a heslo. Příklad hodnoty reg-user@fit.vutbr.cz
DOMAIN: Seznam URI definující chráněnou oblast. Když je vynechána, chráněná oblast je celý server.
NONCE: Specifický řetězec vygenerovaný při návratové hodnotě 401. Řetězec by měl být v hexadacimálním nebo base64 tvaru.
OPAQUE: Řetězec specifikovaný serverem, který by měl být navrácen klientem nezměněn. Řetězec by měl být v hexadacimálním nebo base64 tvaru.
STALE: Může nabývat hodnoty TRUE nebo FALSE. Když se rovná hodnotě FALSE nebo hodnota není, musí se poslat login a heslo.
ALGORITHM: Řetězec ukazující algoritmus a kontrolní součet použitý pro vytvoření DIGESTU. Když není algoritmus zadán, měl by být použitý algoritmus MD5.
QOP-OPTIONS: Položka volitelná. Přítomna pro zpětnou kompatibilitu. Může nabývat hodnoty Auth jako autentizaci, Auth-int jako autentizace s integritou. Pokud je hodnota neznámá, je ignorována.
AUTH-PARAM: Položka pro budoucí rozšíření. Všechny hodnoty jsou ignorovány.

3.3.2.1 Authorization Request Header

Od klienta je očekáváno, že odpoví na žádost o autentizaci posláním autorizace, která je definována takto [9].

```
credentials = "Digest" digest-response
digest-response = 1#( username | realm | nonce |
digest-uri | response | [ algorithm ] |
[cnonce] | [opaque] | [message-qop] |
[nonce-count] | [auth-param] )
username = "username" "=" username-value
username-value = quoted-string
digest-uri = "uri" "=" digest-uri-value
digest-uri-value = request-uri ; dle specifikace HTTP/1.1
message-qop = "qop" "=" qop-value
cnonce = "cnonce" "=" cnonce-value
cnonce-value = nonce-value
nonce-count = "nc" "=" nc-value
nc-value = 8LHEX
response = "response" "=" request-digest
request-digest = <"> 32LHEX <">
LHEX = "0" | "1" | "2" | "3" |
"4" | "5" | "6" | "7" |
"8" | "9" | "a" | "b" |
"c" | "d" | "e" | "f"
```

OPAQUE a ALGORITHM: hodnoty dostal od serveru v response hlavičce
USERNAME: login pro specifikovaný REALM
DIGEST-URI: URI z Request-URI v Request-Line, duplikováno, protože Proxy mohou měnit Request-Line.
QOP: Volitelná hodnota kvůli zpětné kompatibilitě je stejná jako v WWWAuthenticate hlavičce. Pokud je v ní přítomna.
CNONCE: Musí být specifikována, pokud server zaslal QOP. Pokud server QOP neposlal, nesmí být specifikována. Hodnota se nazývá CNONCE-VALUE a je vytvořena jako OPAQUE. Dodává vzájemnou autentizaci a integritu dat.

NONCE-COUNT: NC-VALUE je hexidecimální čítač počtu poslaných požadavků. První poslaný požadavek bude mít hodnotu "nc=00000001". Pokud stejná hodnota dorazí vícekrát, jedná se o požadavek poslaný znovu.
AUTH-PARAM: Položka pro budoucí rozšíření.

Jakýkoliv nerozpoznatelný údaj musí být ignorován. Pokud chceme poslat požadavek se špatným údajem nebo se špatnou hodnotou údaje, je ze serveru poslána chybová hodnota 400 Bad Request.

3.3.2.2 REQUEST-DIGEST:

Pokud je tato hodnota REQUEST-DIGEST [9] chybná, měla by být zaznamenána serverem kvůli možnosti útoku na uhádnutí hesla. Jestliže hodnota QOP je rovna auth nebo auth-int, je REQUEST-DIGEST rovna:

```
request-digest  = <"> < KD ( H(A1),     unq(nonce-value)
                                          ":" nc-value
                                          ":" unq(cnonce-value)
                                          ":" unq(qop-value)
                                          ":" H(A2)
                                    ) <">
```

Pokud qop není přítomna, je tato konstrukce kvůli kompatibilitě s RFC 2069, a hodnota REQUEST-DIGEST rovna:

```
request-digest  =
                 <"> < KD ( H(A1), unq(nonce-value) ":" H(A2) ) >
   <">
```

3.3.2.3 Hodnoty A1 a A2

Když se hodnota algoritmu rovná MD5 nebo není specifikována, pak je hodnota A1 rovna:

```
A1       = unq(username-value) ":" unq(realm-value) ":" passwd

   kde

      passwd   = < user's password >
```

Když se hodnota algoritmu rovná MD5-sess, pak je hodnota A1 počítána jen při prvním vytváření request klientem podle WWW-Authenticate Challenge od serveru. Potom je A1 rovna:

```
  A1       = H( unq(username-value) ":" unq(realm-value)
                    ":" passwd )
                    ":" unq(nonce-value) ":" unq(cnonce-value)
```

Když se hodnota qop rovná auth nebo není specifikována, pak je hodnota A2 rovna:

```
A2       = Method ":" digest-uri-value
```

Když se hodnota qop rovná auth-int, pak je hodnota A2 rovna:

```
A2       = Method ":" digest-uri-value ":" H(entity-body)
```

3.4 Závěr

Každý z výše popsaných protokolů má svoje výhody a nevýhody. Pro větší bezpečnost je lepší používat více protokolů společně. V následující tabulce je uvedeno základní srovnání těchto protokolů.

Protokol	Autentizace	Šifrování dat	Integrita dat
SOAP	Ano	Ne	Ano
SSL	Ano	Ano	Ano
HTTP-Auth	Ano	Ne	Ne

4 Autentizace

Autentizace [25] je proces pro ověření identity klienta. Autentizace patří k bezpečnostním opatřením a k zajišťování bezpečné ochrany nejen na internetu, aby se neidentifikovaný člověk nedostal k informacím, které mu nenáleží. Existují čtyři možné způsoby autentizace klienta k systému.

- K systému se klient autorizuje podle toho co zná např. login a heslo.
- K systému se klient autorizuje podle toho co má např. privátní klíč
- K systému se klient autorizuje podle biometriky např. otisk prstu
- K systému se klient autorizuje podle toho, co může prokázat např. zná odpověď na náhodně vygenerovanou otázku

Tyto metody autentizace je možné mezi sebou kombinovat, ale nesmíme zapomenout, že žádná z těchto metod není dokonalá. Obecně platí, když použijeme spolehlivější metodu, tím obtížněji se používá. To by ovšem klientům nebylo příjemné, a proto musíme volit takový způsob autentizace, který je spolehlivý a moc nezatěžuje klienta.

4.1 Username

Jedná se o základní autentizační metodu. Každý uživatel musí mít vlastní účet nebo být ve skupině uživatelů s právy pro přihlašování. Účty jsou většinou identifikovány uživatelským nebo skupinovým jménem a většinou heslem, které chrání před neoprávněným přístupem. Abychom se k systému mohli přihlásit, musíme znát uživatelské jméno a heslo. Tyto údaje představují nejjednodušší způsob autentizace, jak zabezpečit data, která sdílí mezi sebou klient a počítač.

4.1.1 HTTP Autentizace

Jedná se o autorizaci za pomocí protokolu HTTP, za pomocí odezvy 401 Unauthorized a následné hlavičce WWW-Authenticate, kde sdělí server klientovi, že od něho očekává poslání přihlašovacích informací. Tyto informace si aplikace, většinou webový prohlížeč, pamatuje a automaticky je zasílá na server s každým dalším požadavkem. Příklad, jak může HTTP autentizace vypadat, je na: Obrázek 15.
Tato autentizace v sobě nese několik nevýhod. Je nemožné donutit prohlížeč, aby se odhlásil a přestal používat přihlašovací informace. Další nevýhoda je přenášení přihlašovacích informací při každém požadavku na server a je většinou méně pohodlná.

Obrázek 15 – Přihlašovací okno HTTP autentizace

4.1.2 Formulářová autentizace

Jedná se o častěji používanou autentizaci. Webová stránka nebo aplikace zašle v případě potřeby autentizace klientovi webovou stránku s přihlašovacím formulářem. Tento formulář je většinou naprogramovaný v php nebo v asp. Po jeho vyplnění a odeslání na server, jsou údaje ověřeny. Po úspěšném přihlášení je vytvořena session. Tuto session klient posílá při každém požadavku. Na rozdíl od HTTP autentizace už není posíláno uživatelské jméno a heslo. Přihlašovací informace by měly být odesílány metodou POST, která se neukládá do klientovi historie ani do logů při cestě. Posílání přihlašovacích informací by mělo být chráněné. Chráněny mohou být už zmiňovaným SSL protokolem nebo metodou challange / response.
Challenge / response slouží pro autentizaci nezabezpečeného komunikačního kanálu bez nutnosti posílání hesla. Tato metoda pracuje takto. Se stránkou webového přihlašovacího formuláře server

pošle klientovi i náhodně vygenerovaný řetězec. Když uživatel vyplní přihlašovací údaje a dá odeslat, data se začnou zpracovávat. Pomocí skriptu prohlížeč seskupí náhodně vygenerovaný řetězec od serveru a zadané heslo uživatele. Tento vzniklý řetězec je zašifrován za pomocí hashovací funkce jako je MD5 a teprve tento řetězec je odesílán na server. Server si pamatuje vygenerovaný řetězec a také zná uživatelovo heslo, tak si vytvoří řetězec stejným postupem jako prohlížeč na straně klienta. Tyto vzniklé řetězce mezi sebou porovná a tím ověří, zda uživatel zadal platné heslo a jestli může být přihlášen. Tato autentizace by měla být zvolena pro své výhody jednoduchého odhlašování, bezpečného přenosu dat a možnosti použití SSL protokolu na rozdíl od HTTP autentizace

4.2 Certifikát X.509

V současné době je standard X.509 nejběžnější formát certifikátu. Předchůdcem tohoto formátu byl standard X.500. Standard X.509 byl poprvé vydán roku 1988. Dnešní verze standardu je X.509v3. Specifikuje formát certifikátů a způsob ověření podpisů certifikačních autorit. Certifikát jednoznačně identifikuje vlastníka certifikátu a rozlišuje podřízené a nadřízené certifikační autority. Vlastník může být zadán jednoznačným jménem nebo alternativním označením jako název společnosti. V České republice vydává kvalifikovaný certifikát I.CA, která vydává certifikáty jednotlivým právnickým a fyzickým osobám a Postsignum RootCA, která vydá certifikát Postsignum QCA. Tento certifikát jde teprve k právnickým a fyzickým osobám. Struktura certifikátu vypadá následovně [21]:

```
Version: verze certifikátu
SerialNumber: sériové číslo certifikátu
Signature: označení algoritmu (ID)
Issuer: vydavatel
Validity: platnost
notBefore: nepoužívat před datem
notAfter: nepoužívat po datu
Subject: vlastník veřejného klíče
SubjectPublicKeyInfo: informace o veřejném klíči vlastníka
Public Key Algorithm: algoritmus pro veřejný klíč
SubjectPublicKey: veřejný klíč (data)
IssuerUniqueId: unikátní identifikátor vydavatele (volitelný), X509v2 nebo
X509v3
SubjectUnique Id: unikátní identifikátor vlastníka (volitelný), X509v2
nebo X509v3
extensions: rozšíření (volitelné), X509v3
SignatureAlgorithm: algoritmus pro certifikát (elektronické podpis)
SignatureValue: certifikát(elektronický podpis)
```

Jak je vidět z tabulky ve verzi 3 přibylo pole extensions, díky kterému se může do certifikátů přidat vlastní rozšíření a řídit akceptaci těchto rozšíření. Rozšíření definováno jako kritické musí být v aplikaci pracující s certifikátem implementováno. Když nejsou, certifikát musí být odmítnut. Některá z významných rozšíření si uvedeme.

Key Usage
Toto rozšíření může omezit použití certifikátu pro konkrétní účely. Jedná se o rozšíření definováné jako kritické. Je tvořeno bitovým polem, ve kterém můžeme nastavit různou kombinací bitů, a tím dosáhnout různých vlastností [13].

```
digitalSignature – umožňuje pouze podpis
nonRepudiation – umožňuje pouze ověření podpisu
keyEncipherment – pro šifrování klíčů
dataEncipherment – pro šifrování ostatních dat
keyAgreement – klíč používán pro další dohody
keyCertSign – umožňuje klientovi podepisovat další certifikáty, většinou
CA bit nepovoluje
cRLSign – veřejný klíč umožnuje ověřit pravost zneplatněných certifikátů
encipherOnly - nedefinovaný
decipherOnly - nedefinovaný
```

Subject Alternative Name
Umožňuje přidávat ke jménu osoby různé další informace jako například emailovou adresu, IP adresu.

Certification Policies
Umožňuje se odkázat na politiku certifikační autority. Pomocí toho může nastavit vhodné používání certifikátu.

4.2.1 Vytvoření certifikátu

Certifikát se vytvoří žádostí o certifikát a vygenerováním jeho soukromým a veřejným klíčem. Žádost certifikátu má strukturu obsahující identifikační údaje vlastníka, jeho veřejný klíč a je podepsána soukromým klíčem. Žádost o certifikát si vlastník generuje sám za pomocí nějaké aplikace. S takto vygenerovaným souborem vlastník navštíví registrační autoritu.

```
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 16 - příklad certifikátu X.509 (převzato z [21])

Z různých důvodů může být certifikát i zneplatněn. Ať už se jedná o prolomení certifikátu nebo změnu vlastníka certifikátu. Možnost, jak certifikát zneplatnit, je umístění jeho seriového čísla do seznamu zneplatněných certifikátů CRL – Certificate Revocation List.

4.2.2 Hodnocení certifikátu

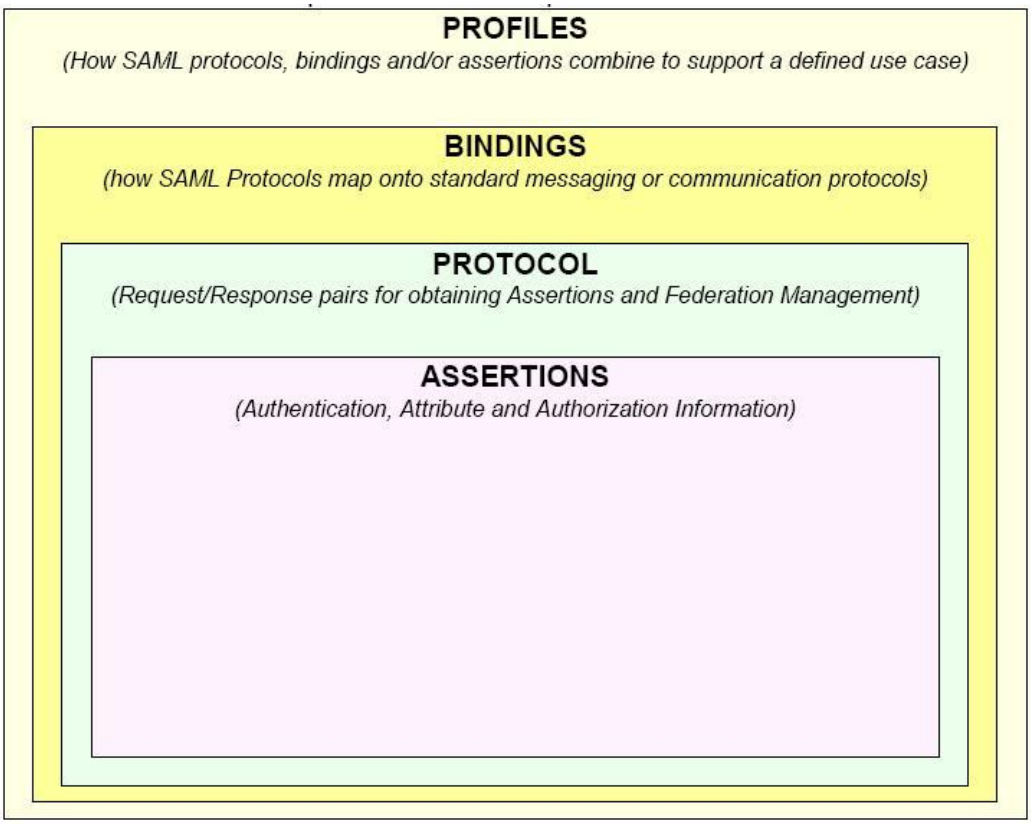
Při využití certifikátu X.509 si musíme uvědomit, že jeho úkolem je svázání identity uživatele s jeho veřejným klíčem k zajištění identifikace a autentizace. Ostatní aktéři při vytváření a ověřování elektronického podpisu věří certifikační autoritě, která má odkaz v certifikátu. Můžeme tedy říci, že certifikát funguje jako identifikátor, se kterým je možné ověřit svou identitu na dálku. Musíme si uvědomit, že certifikát může být při nesprávném používání odcizen nebo se útočník může dostat mezi komunikaci uživatele a serveru.

4.3 SAML

Standard SAML byl vyvinut společností OASIS v listopadu roku 2002. V dnešní době je aktuální verze SAML 2.0. SAML je framework založený na formátu XML. Obsahuje autentizaci a oprávnění uživatelů a informace o atributech. Umožňuje jejich entitám vytvářet tvrzení s ohledem na jejich atributy, oprávnění a identitu. Mohou tyto tvrzení zasílat jiným entitám jako jiné aplikace a podobně. Tento standard je flexibilní a rozšiřitelný a můžeme ho kombinovat s jinými standardy. Největší výhody standardu SAML jsou nezávislost na platformě, slabá vazba mezi správci entit, zlepšení podmínek pro koncové uživatele, jako jednotné přihlášování na různé služby, snížení nákladů poskytovatelů služeb, díky jednotnému přihlašování.
SAML lze využít v mnoha situacích např.: jednotné přihlášení na web. Uživatel se autentizuje vůči jedné webové stránce a potom má přístup k dalším stránkám bez nutného přihlašování. Autorizace je na základě atributů. Metoda se autentizuje vůči jedné stránce, která předává informace dalším stránkam, jak byl uživatel autentizován. Zabezpečení webových služeb je využíváno v SOAP zprávách za účelem zabezpečení přenosu informací a autentizací mezi uživatelem a poskytovatelem. Doporučení je se SAML používat i jinou bezpečnostní techniku jako SSL nebo šifrování XML podpisů a zpráv.

4.3.1 SAML komponenty

SAML je tvořen vazbami mezi tvrzením, protokoly, vazbami a profily, jak je vidět na: Obrázek 17



Obrázek 17 - vazba mezi SAML komponentami (převzato z [19])

4.3.1.1 Tvrzení

Tvrzení obsahuje informace dodávající jedno nebo více sdělení vytvořené SAML autoritou. Tato autorita může vytvořit tyto tvrzení:

- Autentizace – subjekt byl autentizován určitými prostředky a v určitém čase.
- Atribut – subjekt je přiřazený k danému atributu.
- Autorizační informace – Povolí nebo zamítne požadavky subjektu.

```
<?xml version="1.0" encoding="UTF-8"?>
<saml:Assertion xmlns:saml="urn:oasis:names:tc:SAML:2.0:assertion"
Version="2.0"
IssueInstant="2005-01-31T12:00:00Z">
  <saml:Issuer>
    www.acompany.com
  </saml:Issuer>
  <saml:Subject>
    <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress">
j.doe@company.com
    </saml:NameID>
  </saml:Subject>
  <saml:Conditions NotBefore="2005-01-31T12:00:00Z"
NotOnOrAfter="2005-01-31T12:00:00Z">
  </saml:Conditions>
  <saml:AuthnStatement
AuthnInstant="2005-01-31T12:00:00Z" SessionIndex="67775277772">
    <saml:AuthnContext>
      <saml:AuthnContextClassRef>
urn:oasis:names:tc:SAML:2.0:ac:classes:PasswordProtectedTransport
      </saml:AuthnContextClassRef>
    </saml:AuthnContext>
  </saml:AuthnStatement>
</saml:Assertion>
```

Obrázek 18 - SAML tvrzení (převzato z [19])

4.3.1.2 Protokoly

SAML má několik protokolů, díky nim může poskytovatel služby provádět následující operace:

- Požadavek na registraci jmenného prostoru
- Požadavek na SAML autoritu na jedno nebo více tvrzení
- Požadavek na IP autentizaci uživatele a vyhodnocení následného tvrzení

4.3.1.3 Vazby

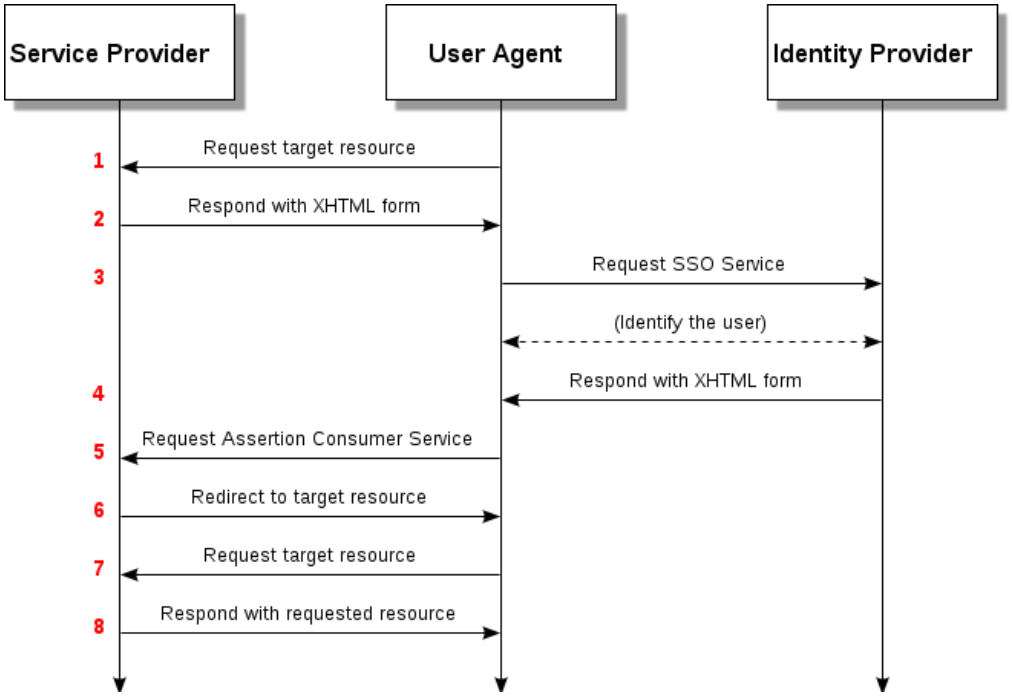
Zabývájí se přechodem ze SAML request/response do standardních komunikačních protokolů. SAML SOAP vazba definuje, jak má být použitá zpráva SAML v SOAP zprávách.

4.3.1.4 Profily

Definují omezení a možné rozšíření SAMLu v definovaných aplikací.

4.3.2 SAML použití

Nejčastěji SAML komunikuje metodou jednotného přihlášení na internetu - Web Browser Single Sign-On (SSO). Uživatel ovládající user agent, nejčastěji to bývá webový prohlížeč, posílá požadavek na service provider, který je chráněn pomocí SAMLu. Pokud user agent je už autorizován, je již vytvořena bezpečná komunikace a service provider vrací požadovaná data. Jinak si Service provideri požádá o autentizaci uživatele a pošle mu dokument pro autentizaci. Po vyplnění autorizačních parametrů jsou tyto data zašifrována base64 a poslána na identity provider. Identity provider klientovi posílá odpověď na autorizaci zabalenou taky přes base64. User agent tento zabalený token přepošle k service provider, ten pošle odpověď a vytvoří bezpečnou komunikaci. User agent je přesměrován na cílový zdroj. User agent se opět zeptá na cílový zdroj service provider a od vytvoření bezpečné komunikace service provider vrací požadavky na user agent.



Obrázek 19 - SAML web browser SSO komunikace (převzato z [17])

4.4 REL Token Profile

REL je zkratka pro Right Expression Language a toto zabezpečení se používá v SOAP zprávách, které se používají pro kódování tvrzení a pro práva jejich použití. Jeho zpracování v SOAP zprávách se neliší od ostatních token formátů jak například SOAP Message Security. Tokeny se připojují standardně do hlavičky SOAP zpráv do elementu <wsse:Security>

4.4.1 Autentizace

REL umožnuje více možností autentizace. Tento profil zabezpečení vyžaduje, aby zprávy od odesílatele a příjemce podporovaly požadavek na <r:keyHolder> od zmocnitelů. Doporučuje se, aby XML podpis byl použítý pro vytvoření vztahu mezi zprávou odesílatele a jeho potvrzením. To je zvlášť důležité, když komunikace probíhá přes nechráněnou komunikaci.

4.4.1.1 Požadavek <r:keyHolder> odesílatel

Zpráva se musí nacházet uvnitř <wsse:Security> SOAP hlavičce elementu <r:license> obsahující nejméně jeden element <r:grant> obsahující element <r:keyHolder> identifikující klíč pro potvrzení zprávy. Jestli že zpráva obsahuje více elementů <r:grant> a v něm je element <r:keyHolder>, musí být elementy <r:keyHolder> stejné. Aby příjemce přijmul zprávu musí se odesílatel prokázat platným klíčem. Odesílatel se možná prokázal platným klíčem uvnitř zprávy, která obsahuje element <ds:Signature> uvnitř elemntu <wsse:Security>. Licence obsahující nejméně jeden element <r:grant> by měly obsahovat element <r:issuer> s elementem <ds:Signature> identifikující vydavatele licence a chrání integritu klíče.

4.4.1.2 Požadavek <r:keyHolder> příjemce

Když příjemce zjistí, že odesílatel se prokázal platným klíčem uvedeným v elementu <r:keyHolder> požadavky nalezené v licencích možná náleži odesilateli. Pokud je jedna z těchto požadavků shodná a když podmínky požadavků jsou schváleny, pak nějaké elementy ve zprávě jejíž integrita je chráněna klíčem lze považovat za tvůrce této identifikace.

```
<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>
</S:Envelope>
```

Obrázek 20 - ukázka bezpečnostního tokenu s <r:keyHolder> příjemce (převzato z [20])

4.4.2 Důvěryhodnost

Popisuje jak licence může být použita k ochraně utajené SOAP zprávy. SOAP Message Security nám ovšem nepřikazuje, jak utajení musí být vytvořeno. REL nám umožnuje použití různých typů utajení. To ovšem vyžaduje, aby zpráva odesílatele a příjemce podporovala důvěrnost element <r:keyHolder> zmocnitelů. Doporučuje se použít XML šifrování pro zabezpečení utajení. To je zvlášť důležité, když komunikace probíhá přes nechráněnou komunikaci.

4.4.2.1 Požadavek <r:keyHolder> odesílatel

Zpráva se musí nacházet uvnitř <wsse:Security> SOAP hlavičce elementu <r:license> obsahující nejméně jeden element <r:grant> obsahující element <r:keyHolder> identifikující klíč pro potvrzení

zprávy. Jestliže zpráva obsahuje více elementů <r:grant> a v něm je element <r:keyHolder>, musí být elementy <r:keyHolder> stejné. Aby přijímač vědel, kdy dešifrovat data nebo klíč, odesílatel musí označit místo šifrované ve zprávě. Odesílatel možná označí toto místo elementem <xenc:EncryptedData> nebo <xenc:EncryptedKey> zahrnující s následním elementem <xenc:ReferenceList> nebo <xenc:EncryptedKey> uvnitř elementu <wsse:Security>. <xenc:ReferenceList> nebo <xenc:EncryptedKey> elementy vytvořené za tímto účelem musí být v souladu s pravidly stanovenými WS-Security.

4.4.2.2 Požadavek <r:keyHolder> odesílatel

Pokud příjemce má znalost dešifrovacího klíče specifikovaný v <r:keyHolder>, potom možná dešifruje data nebo klíč.

```
<S:Envelope xmlns:S="..." xmlns:ds="...">
  <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="...">SOME COMPANY</sx:commonName>
        </r:grant>
        <r:issuer>
          <ds:Signature>...</ds:Signature>
        </r:issuer>
      </r:license>
      <xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
        <KeyInfo xmlns="http://www.w3.org/2000/09/xmldsig#">
          <wsse:SecurityTokenReference>
            <wsse:Reference URI="urn:foo:SecurityToken:ef375268"/>
          </wsse:SecurityTokenReference>
        </KeyInfo>
        <xenc:CipherData>
          <xenc:CipherValue>dNYS...fQ=</xenc:CipherValue>
        </xenc:CipherData>
        <xenc:ReferenceList>
          <xenc:DataReference URI="#enc"/>
        </xenc:ReferenceList>
      </xenc:EncryptedKey>
    </wsse:Security>
  </S:Header>
  <S:Body wsu:Id="body"
xmlns:wsu="http://schemas.xmlsoap.org/ws/2003/06/utility">
    <xenc:EncryptedData Id="enc"
Type="http://www.w3.org/2001/04/xmlenc#Content"
xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
      <xenc:EncryptionMethod
Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-cbc"/>
      <xenc:CipherData>
        <xenc:CipherValue>d2s...GQ=</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </S:Body>
</S:Envelope>
```

Obrázek 21 - ukázka bezpečnostního tokenu s <r:keyHolder> odesílatel (převzato z [20])

4.4.3 Hodnocení

Tato autentizace má v sobě některá protiopatření, která zmařují některé hrozby útoku. Hlášení o změně a odposlouchávání můžeme řešit pomocí integrity a důvěrnosti popsaných v WS-Security. Některé útoky můžeme řešit pomocí timestamps a caching. Toto řešení v sobě obsahuje licence a tudíš si musíme dát pozor, aby tyto licence nebyly padělány nebo zfalšovány.

5 Implementace

V této kapitole se seznámíme s požadavky na výslednou desktopovou aplikaci komunikující s webovými službami.

5.1 Neformální specifikace

Základní cíl je vytvoření aplikace pracující se zabezpečenými webovými službami, které by sloužily pro ověření dostupnosti a zabezpečení webové služby a její funkčnost. Aplikace má být navržena pro ověření a komunikaci s webovými službami a kvůli tomu velká část funkčnosti aplikace bude orientována na parsování WSDL dokumentu a komunikaci mezi aplikací a webovou službou. Dále klient musí umět rozpoznat, jestli je potřeba se k využití služby autorizovat, jestli je komunikace chráněna certifikátem nebo zpráva obsahuje další zabezpečení. K aplikaci je nutné vytvořit ukázkovou službu, na které se předvedou možnosti aplikace a její funkčnost.

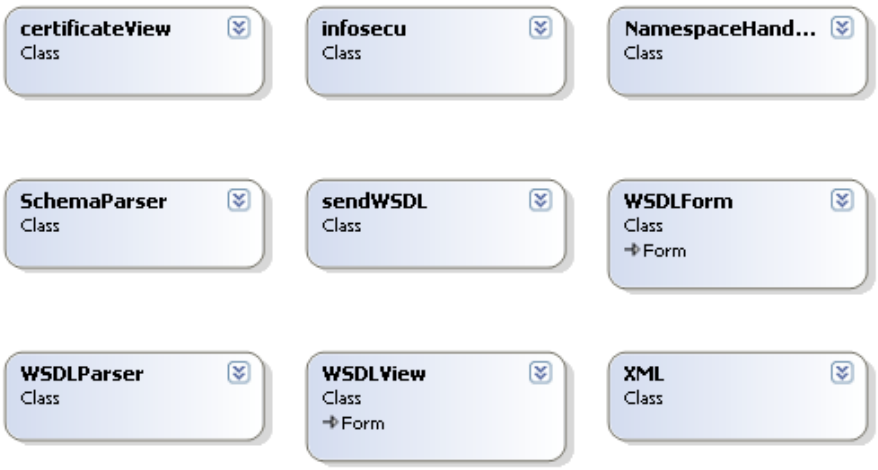
5.2 Klientská aplikace

Klientská aplikace může být naprogramována jako desktopová aplikace, přes kterou bude možné komunikovat se zabezpečenými webovými službami. Uživatel si zažádá o konkrétní webovou službu a bude se k ní chtít připojit. Po připojení k zadané službě si aplikace automaticky stáhne WSDL soubor, který rozparsuje a vyhodnotí způsoby zabezpečení webové služby. Uživateli bude nabídnut výběr metody, kterou má webová služba k dispozici a pracuje s ní. Po zvolení metody si uživatel doplní potřebná data do předem definované zprávy nebo si tuto zprávu může sám napsat. Takto vytvořenou metodu si bude moci uživatel uložit pro další zpracování. Nakonec zprávu odešle k vyhodnocení na server.

6 Implementace

6.1 C# .NET

Programovací jazyk pro vytvoření celé práce byl použit C#. Je to objektově orientovaný jazyk vytvořený společností Microsoft. Pro bezproblémový chod aplikace napsané v tomto jazyce je nezbytné mít nainstalovanou platformu NET v minimální verzi 2.0.
Při tvorbě ukázkové služby byly využity výhody Microsoft platformy NET. Jedna z jejich vlastností je automatické generování SOAP zpráv a WSDL popisu na základě vytvořených webových služeb. Jako výhodu a pro kontrolu webových služeb nabízí webový přehled pro všechny webové metody obsahující webové služby. Jako nevýhoda je možné podotknout nutnost operačního systému Windows a serveru pro platformu ASP.NET.



Obrázek 22 - Třídy programu

6.2 WSDL parser

Dříve než je možné poslat požadavek webové služby na server, musíme zjistit, jaké metody webová služba nabízí, jestli jsou SOAP zprávy zabezpečeny, a v jakém formátu jsou posílány zpět na server. Po vyplnění požadované adresy, loginu a hesla, pokud jsou nutné, a takto vytvořený dotaz pomocí tlačítka sign in odešleme na server. Pokud je služba dostupná a požadované údaje jsou správně vyplněné, server nám stáhne WSDL soubor ke zpracování. Pokud nastane chyba objeví se příslušná chybová hláška.
O parsování se stará třída z jmeného prostoru WSDL.Parser. WSDL je tady vytvořeno pro informaci, že se jedná o dokument WSDL. Tato třída byla převzata z projektu wsdl and schema parser[1] a následně poupravena. Po stažení WSDL dokumentu je vytvořen objekt třídy WSDLParser, který rozparsuje celý dokument do stromu pro další použití a načtení dostupných metod a základních informací o webové službě. Po zvolení určené metody se nám doplní informace o vybrané metodě.

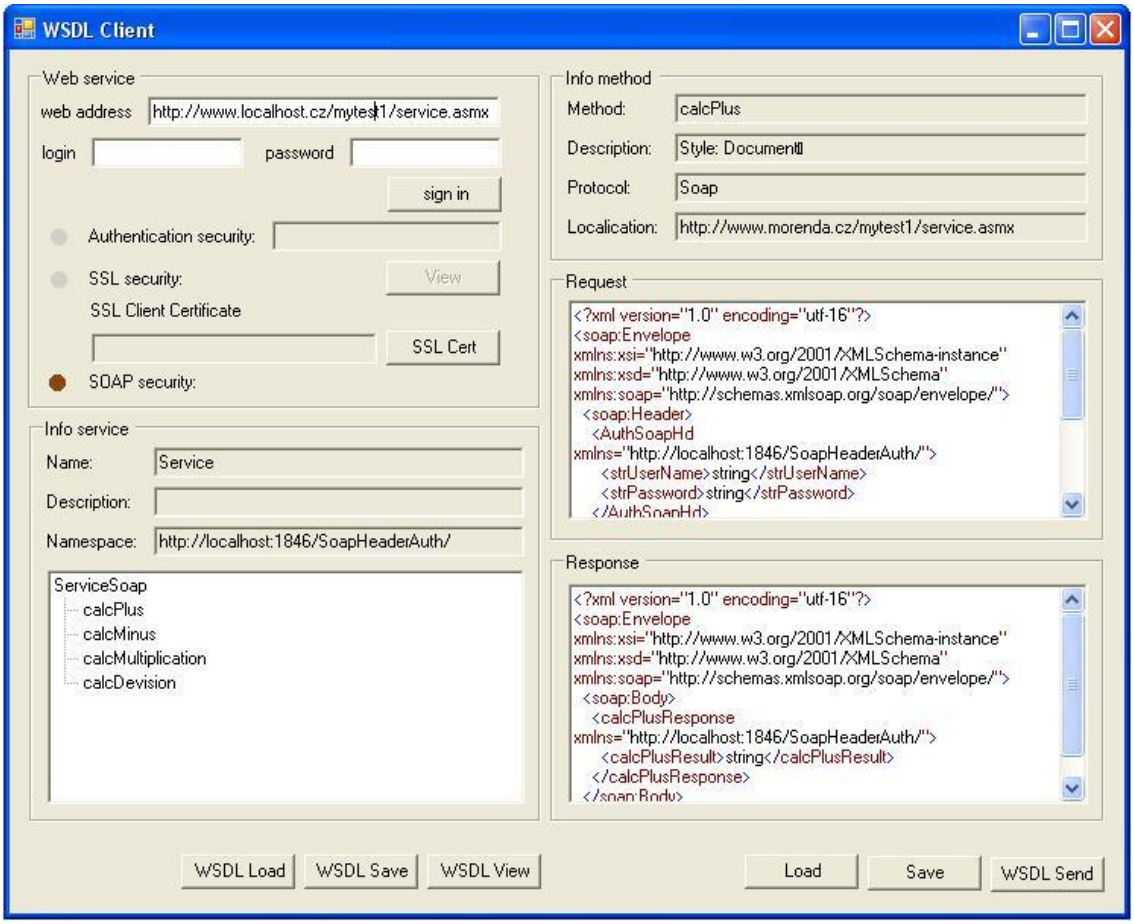
[1] http://www.codeproject.com/KB/cpp/wsdlparser.aspx

6.3 Komunikace se serverem

Pravá část aplikace nabízí komunikaci webové metody se serverem. Umožnuje díky tlačítku WSDL Send posílat zprávu na server a poté přijmout následnou odpověď. Tato část je vždycky změněna pro určitou metodu webové služby jejími specifickými informacemi. Najdeme zde informace o metodě, jako je její název, popis, přes který protokol pracuje, její lokalizaci a náhledy, jak má vypadat požadavek posílaný na server a náhled odpovědi od serveru. Pokud je požadavek nesprávný nebo špatně napsaný, server nám zprávu ohodnotí a napíše chybovou hlášku, kterou aplikace vyhodnotí a vypíše chybu. O posílání požadavků na server se stará třída sendWSDL.

6.4 Bezpečnost

V kolonce web service zadáváme webovou adresu a možný login a heslo. Dále se tady vyhodnocuje úroveň zabezpečení. Máme zde kolonku Authentication security, která nám vyhodnocuje, zda je potřeba se ke službě před stažením WSDL dokumentu přihlásit. Vyhodnocuje se tu, jaká HTTP-Auth je použita a metoda se vypíše. Dále tady máme kolonku SSL security, která nám vyhodnocuje SSL připojení. Pokud je na webu dostupný certifikát, je stažený a je vytvořena bezpečná komunikace přes SSL. Pokud je server neplatný, tak se na webovou službu z bezpečnostních důvodů nemůžeme připojit. Certifikát si můžeme prohlédnout za pomocí tlačítka View. Máme tady kolonku pro nahrání klientského certifikátu. Bohužel tato možnost není úplně funkční z důvodu nepodaření konfigurace



Obrázek 23 - Klientká aplikace

webového serveru. Nakonec se tady nachází položka SOAP security. Ta nám určuje, jestli je zabezpečení součástí SOAP zprávy. Toto zabezpečení vidíme v tabulce Request(požadavek) a pokud je potřeba, tak je doplníme.

6.5 WSDL saver

Aplikace umožnuje jednoduchý náhled na stažený celý WSDL dokument. Tento náhled slouží pro studijní účely WSDL dokumentu. Tento dokument je možné stáhnout na lokální disk za účelem rychlejšího nalezení WSDL zprávy. Tyto soubory s příponou wsdl je možné rychle načíst a začít používat.
Aplikace umožnuje taky předpřipravit si jednotlivé metody pro rychlejší použití. Metoda, která bude uložena, je vidět v kolonce Request. Tuto zprávu je možné uložit v předpřipraveném stavu od webové služby nebo si ji můžete předpřipravit do požadované formy pro rychlejší načítání častějších dotazů. Při uložení jednotlivé metody si program uloží i další nezbytné věci jako adresa webové služby nebo přihlašovací údaje. Všechny tyto WSDL dokumenty a jejich metody se načítají v přehledném a barevně rozlišeném XML formátu pro lepší uživatelskou přehlednost.

7 Testování a ukázka aplikace

Pro testování jsem zvolil tři testy:

- Test s www autentizací
- Test spojení pomocí SSL.
- Test zabezpečení v SOAP zprávě.

7.1 Test s www autentizací

Pro tento test jsem si vytvořil jednoduchou webovou službu a zabezpečil jsem si ji pomocí HTTP-Auth. Po vyplnění potřebných doplňujících informací, jako uživatelské jméno a heslo, jsem zkusil stáhnout WSDL. Přihlašování na webovou službu bylo úspěšné a stáhnutí WSDL dokumentu se podařilo a opětovné posílání zprávy pro vyhodnocení metody bylo také úspěšné. Pro navázaní spojení bylo nezbytné mít správně vyplněné uživatelské jméno a heslo po celou dobu komunikace. Jinak aplikace upozorňovala na nesprávné uživatelské jméno nebo heslo.

7.2 Test spojení pomocí SSL

K tomuto testu jsem použil pro kontrolu komunikace na síti program Wireshark. Připojil jsem se na veřejnou službu za pomocí SSL spojení a stáhnul jsem si WSDL dokument. Opět vše proběhlo naprosto v pořádku a podařilo se dokument stáhnout a získat odpověď z webové metody. Pro náhled, jestli komunikace byla šifrovaná, jsem se podíval do programu Wireshark, který po celou dobu komunikace odchytával pakety. Po shlédnutí získaných informací jsem zjistil, že komunikace byla šifrovaná a tudíž vše proběhlo podle předpokladů.

7.3 Test zabezpečení v SOAP zprávě

Tento test jsem si zvolil i pro implementaci ukázkové služby. Službu jsem implementoval pomocí technologie ASP.NET. Zabezpečení služby není v různých ohledech dostačující, ale pro studijní účely a pro otestování aplikace bylo vyhovující. Služba byla navrhnuta, aby hlavička služby v sobě nesla autentizační údaje. Bez těchto informací metoda webové služby nemohla být vykonána. Po načtení webové služby a vyplnění požadovaných údajů, vše proběhlo podle předpokladů a dostal jsem od serveru požadovanou odpověď. Bez vyplnění těchto autentizačních informací mě server vrátil zprávu o špatném požadavku.

8 Závěr

V této práci jsme se seznámili s technologií webových služeb a jejich zabezpečení. Uvedli jsme si jednotlivé protokoly sloužící pro zabezpečení webové služby a různé druhy autentizace. Všechny tyto protokoly mají svoje výhody a nevýhody, takže nemůže jednoznačně určit, který používat, a který nepoužívat. Rozhodnutí by záleželo na konkrétní situaci. Měla by se používat vždy nejnovější verze protokolu a také používat více protokolů zároveň. U autentizace by se také měly používat nejnovější verze a uvědomit si kam se autentizujeme.

Aplikace byla navrhována pro vývojáře webových služeb, pro kontrolu dostupnosti zabezpečení a správného návrhu webové služby. Konečná podoba umožňuje ukládání WSDL dokumentů nebo jejich metod pro jejich pochopení. Jedním z omezení aplikace je použití jednoho vlákna pro zobrazené okno. Další omezení aplikace je v načítání WSDL dokumentu jak ze serveru tak i z lokálního počítače. Problém byl i v možnosti testování bezpečnosti webových služeb. Většina zde uvedených bezpečnostních služeb není volně dostupná, proto byl problém s jejich otestováním.

Podívejme se na možnosti rozšíření aplikace. Jedním z rozšíření by mohlo být výkonější WSDL parser, které umožňuje parsovat dokumenty paralerně nebo použití více vláken. Doplnění způsobů zabezpečení pro webové služby nebo na implementovat funkci pro vyhledávání metod.

Tato práce mě objasnila způsob používání webových služeb a jejich velkou budoucnost pro komunikaci mezi aplikacemi a jejich možnosti zabezpečení.

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] Microsoft. MSDN : Visual C# [online]. 2008 [cit. 2008-04-20]. Dostupný z WWW: <http://msdn2.microsoft.com/en-us/library/kx37x362.aspx>

[4] Wikipedia. SOAP [online]. 12. April 2009 [cit. 2009-04-14]. Dostupný z WWW: <http://en.wikipedia.org/wiki/SOAP>

[5] W3school. SOAP Tutorial [online]. 1999-2009 [cit. 2009-04-13]. Dostupný z WWW: <http://www.w3schools.com/soap/default.asp>

[6] W3C. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) [online]. 27. April 2007 [cit. 2009-04-10]. Dostupný z WWW: <http://www.w3.org/TR/soap12-part1/>

[7] Wikipedia. Basic access authentication [online]. 2001-2009 [cit. 2009-04-20]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Basic_access_authentication>

[8] Wikipedia. Digest access authentication [online]. 2001-2009 [cit. 2009-04-20]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Digest_access_authentication>

[9] FRANKS, J., et al. HTTP Authentication: Basic and Digest Access Authentication [online]. 1. June 1999 [cit. 2009-04-25]. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2617.txt>

[10] ODVÁRKA, Petr . SSL protokol . SSL protokol a jeho akcelerace [online]. 2002 [cit. 2009-05-01]. Dostupný z WWW: <http://www.svetsiti.cz/view_list.asp?rubrika=Tutorialy&temaID=171>

[11] W3C. Transport Layer Security [online]. 2001-2009 [cit. 2009-04-06]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Transport_Layer_Security>

[12] W3C. Secure Sockets Layer [online]. 2001-2009 [cit. 2009-04-10]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Secure_Sockets_Layer>

[13] HOUSLEY, R., et al. Internet X.509 Public Key Infrastructure Certificate and CRL Profile [online]. 1. January 1999 [cit. 2009-04-23]. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2459.txt>

[14] FIELDING, R., et al. Hypertext Transfer Protocol -- HTTP/1.1 [online]. 1. June 1999 [cit. 2009-04-22]. Dostupný z WWW: <http://www.ietf.org/rfc/rfc2616.txt>

[15] NADALIN, Anthony , et al. *Web Services Security: SOAP Message Security 1.1 (WS-Security 2004)* [online]. 1. February 2006 [cit. 2009-05-10]. Dostupný z WWW: <http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf>

[16] První certifikační autorita, a.s.. *Teorie a principy* [online]. 2000-2008 [cit. 2009-04-13]. Dostupný z WWW: <http://www.ica.cz/home_cs/?acc=teorie_a_principy>

[17] Wikipedia. Security Assertion Markup Language [online]. 2001-2009 [cit. 2009-05-03]. Dostupný z WWW: <http://en.wikipedia.org/wiki/SAML>.

[18] OASIS. SAML V2.0 Executive Overview [online]. 12. April 2005 [cit. 2009-04-28]. Dostupný z WWW: <http://www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf>

[19] OASIS. Security Assertion Markup Language (SAML) 2.0 Technical Overview [online]. 20. February 2005 [cit. 2009-04-25]. Dostupný z WWW: <http://www.oasis-open.org/committees/download.php/11511/sstc-saml-tech-overview-2.0-draft-03.pdf>

[20] 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>

[21] Wikipedia. X.509 [online]. 2001-2009 [cit. 2009-04-16]. Dostupný z WWW: <http://en.wikipedia.org/wiki/X.509>

[22] FERENC, Jakub. Bezpečnost ve službově orientované architektuře [online]. 2008 [cit. 2009-04-01]. Dostupný z WWW: <http://is.muni.cz/th/98993/fi_m/dp.pdf>.

[23] Wikipedia. X.509 [online]. 2001-2009 [cit. 2009-04-16]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/X.509>

[24] Wikipedia. Autentizace [online]. 2001-2009 [cit. 2009-04-12]. Dostupný z WWW: <http://cs.wikipedia.org/wiki/Autentizace>

[25] Wikipedia. Autentizace [online]. 2001-2009 [cit. 2009-04-12]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Autentizace>

[26] OČENÁŠEK, Pavel. IBS - Bezpečnost a počítačové sítě [online]. 2008- [cit. 2009-04-10]. Dostupný z WWW: <https://wis.fit.vutbr.cz/FIT/st/course-files-st.php/course/IBS-IT/lectures>

[27] TICHÝ, Jan. Autentizace a správa uživatelů [online]. 2004-2009 [cit. 2009-04-16]. Dostupný z WWW: <http://www.jantichy.cz/diplomka/pozadavky/autentizace>

[28] Sun Microsystems. Introduction to SSL [online]. 10. September 1998 [cit. 2009-03-10]. Dostupný z WWW: <http://docs.sun.com/source/816-6156-10/contents.htm>

[29] IBM. Secure Sockets Layer (SSL) concepts [online]. 1995 [cit. 2009-04-14]. Dostupný z WWW: <http://publib.boulder.ibm.com/infocenter/wmqv6/v6r0/index.jsp?topic=/com.ibm.mq.csqzas.doc/sy10660_.htm>

[30] Rosenberg, Jothy. Securing web services with WS-security :demystifying WS-security, WS-policy, SAML, XML signature, and XML encryption / Indianapolis: SAMS, 2004. 378 s. ISBN 0-672-32651-5

[31] Troelsen, Andrew. C# a .NET 2.0 profesionálně / Vyd. 1. Brno : Zoner Press, 2006. 1197 s. : il. ISBN 80-86815-42-0

Seznam příloh

Příloha 1. CD s webovou službou a klientskou aplikací