Curso de Mestrado em Sistemas de Dados
e Processamento Analítico
Segurança e Privacidade em Sistemas de Armazenamento e Transporte de Dados

Segurança em Web Services

Carla Aliete Santos

Índice

Índice de Figuras

1. Introdução

O que é um Web Service

O Web Services é uma tecnologia de chamada remota de objectos, que permite uma infra-estrutura para criação de aplicações distribuídas (web ou não). Com os Web Service é possível a criação de pequenos módulos de código reutilizáveis e disponibilizados para construção de aplicações, utilizando protocolos Web como meio de transporte e comunicação. Desta forma consegue-se um alto grau de abstração em relação a linguagens de programação e plataformas de hardware / software.

Quando usar?

Os Web Services podem ser usados em vário cenário, por exemplo, quendo se pretende:

- ✓ Integrar sistemas heterogêneos dentro da empresa
- ✓ Integrar sistemas remotos através da Internet
- ✓ Integrar diferentes plataformas de hardware / software / SO
- ✓ Fornecer serviços a terceiros
- ✓ Web Services são identificados por uma URI(*Unique Resource Identifier*), e são descritos e definidos usando XML.
- ✓ Um dos motivos que tornam os Web Services atractivos é o facto deste modelo ser baseado em tecnologias standards, em particular XML e HTTP. Os Web Services são usados para disponibilizar serviços interactivos na WEB, podendo ser acedidos por outras aplicações.

Os Web Services são são uma tecnologia Microsoft, e o seu objectivo é que sejam o mais independentes possível, dai que sejam compostos por linguagens e protocolos abertos e largamente adotados pelo mercado:

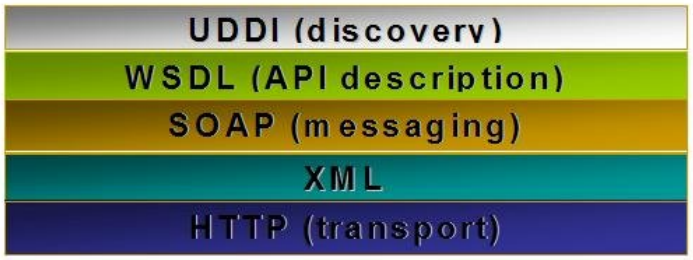


Figura 1 – Camada de Protocolos

A Figura 2 apresenta os vários protocolos e em que fase da troca de mensagens estão envolvidos. O que acontece em relação à arquitectura .Net da Microsoft, é esta suporta todos estes protocolos e linguagens.

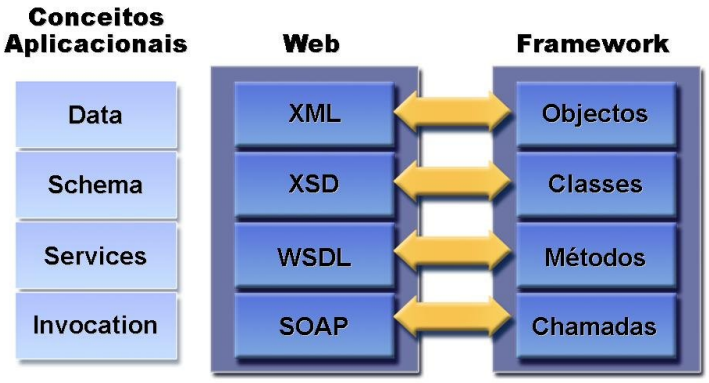


Figura 2- Mapeamento Web Services <-> .NET

A .NET Framework fornece um mapeamento bi-direcional entre cada componente da Framework e cada um dos Protocolo.

2. WS – Solução

O que há de errado no SSL?

Na Figura 3 apresenta-se um cenário da utilização do HTTPS. O que verificamos é que a segurança só é garantida em determinado caminho. Desta forma não podemos confiar numa segurança apenas ponto-a-ponto, ainda mais se existirem muitos pontos na rota, torna-se um desafio manter as credenciais.
O HTTPS tem como desvantagem ser dependente do protocolo http, e além disso cifrar a mensagem toda, tornando-se assim excessivo e caro na utilização de recursos.

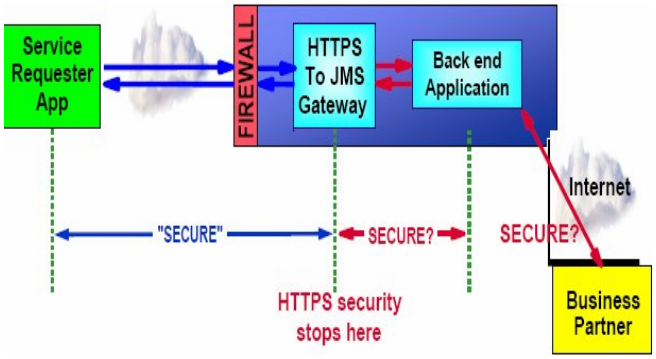


Figura 3 - Cenário utilizando o HTTPS

A solução encontrada é que as credenciais não dependam do canal e transporte, mas sejam integradas na própria mensagem. Desta forma não importa quantas rotas o caminho tenha, a mensagem irá levar consigo a sua própria informação de segurança.
A Figura 4 mostra um cenário em que a troca de mensagens é feita através do protocolo SOAP, que será abordado mais à frente.

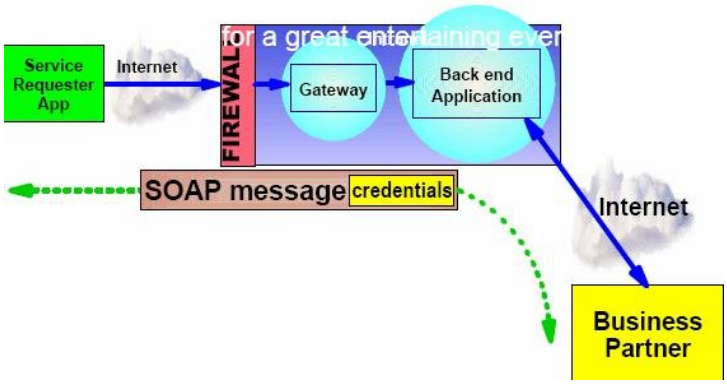


Figura 4 – Cenário onde é usado o protocolo SOAP

A troca de mensagens utiliza o Protocolo HTTP como camada de transporte, uma vez que as chamadas SOAP não são mais do que chamadas http GET ou POST, e utiliza a porta 80, permitida pelos firewalls. A formatação destas mensagens são em XML

Tanto a Figura 5 como a 6 apresentam as entidades envolvidas na chamada a um web Service e como é feita a troca de mensagens entre eles.

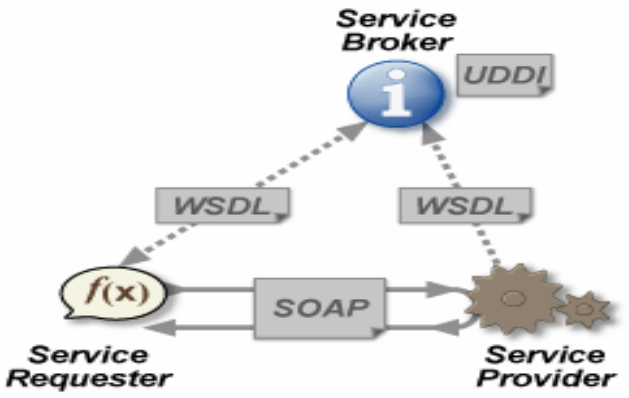


Figura 5 – Entidades envolvidas na chamada a um Web Service

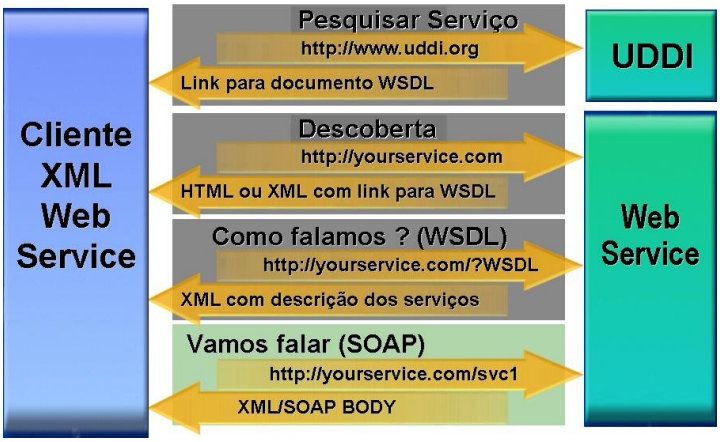


Figura 6 – Mensagens trocadas na invocação de um Web Service

WSDL

Quando alguém cria um Web Service este necessita de ser publicado. Como foi dito, a sua utilização é independente da linguagem em que foi escrito, por isso este deve ser descrito de forma a que qualquer plataforma entenda. O WSDL (Wed Service Description Language) define um sistema para a descrição de serviços, baseado em XML. Este XML Schema descreve o formato dos métodos a serem chamados, parâmetros a serem passados, operações, esquemas de codificação, etc

O seu principal objectivo é descrever as interfaces apresentadas e apontar a localização dos seus Web Services, disponíveis num local da rede, o qual permite que o cliente aceda de maneira confiável. Por ser um documento XML, a sua leitura torna-se fácil e acessível.

UDI

A questão que se coloca agora é como encontrar os Web Services? O *Universal Description, Discovery, and Integration (UDDI)* é uma especificação técnica que tem como objectivo descrever, descobrir e integrar Web Services. Segundo a Organization for the Advancement of Structured Information Standards, é um elemento central do grupo de padrões que compõe a pilha de componentes dos serviços web.
No momento que construímos um Web Service, necessitamos que os serviços sejam acedidos de algum lugar da Internet por um cliente. Uma das maneiras é fazer com que a aplicação cliente conheça a URI do serviço.

Gunzer define UDDI como um padrão desenvolvido para fornecer um directório de busca para os negócios e seus serviços. Tem como objectivo ser um mediador do serviço, permitindo que os clientes requisitantes encontrem um fornecedor do serviço apropriado.

SOAP

É um dos principais elementos dos Web Services, apesar de não ser necessário o conhecimento do seu funcionamento para se criar e consumir um Web Service. Porém, o entendimento geral do protocolo é útil para se lidar com eventuais situações de erros e problemas com a interoperabilidade entre plataformas no uso de Web Services.

- ✓ Service Oriented Architecture Protocol: no caso geral, uma mensagem SOAP representa a informação necessária para invocar um serviço ou analisar o resultado de uma chamada e contém informações específicas da definição da interface do serviço.

- ✓ Service Object Access Protocol: representação opcional do SOAP RPC, a mensagem SOAP representa um método de invocação de um objecto remoto, e a serialização da lista de argumentos do método que precisam ser movidos do ambiente local para o ambiente remoto.

SOAP e RPC

Os RPCs são chamadas remotas de procedimento, que permitem fazer chamadas locais a métodos de objectos (ou serviços) remotos, aceder aos serviços de um objecto localizado num outro ponto da rede. Cada chamada ou requisição exige uma resposta.
O SOAP foi desenhado para encapsular e transportar chamadas de RPC.

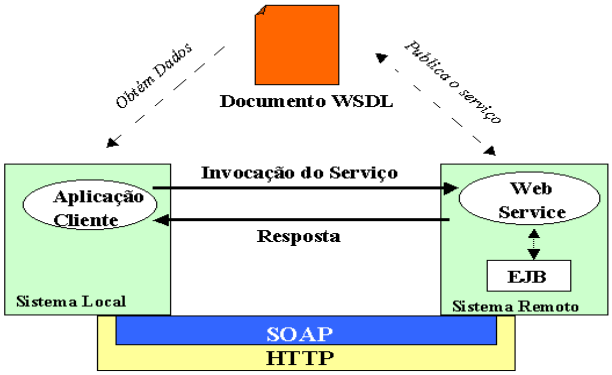


Figura 7 – Invocação de um Web Service

Em outras palavras, SOAP possibilita que dois processos comuniquem entre si, independentemente do hardware e da plataforma onde estão a correr

Ao utilizar XML para codificar mensagens o SOAP nos dá alguns benefícios:

- ✓ XML pode ser facilmente lido pelos utilizadores, portanto, mais fácil de entender e eliminar erros.
- ✓ XML parsers (analistas) e tecnologias correlatas são mundialmente disponíveis.
- ✓ XML é um padrão aberto.
- ✓ XML inclui várias tecnologias que podem fortalecer o SOAP.
- ✓ Simplificação da especificação, diferente de outros protocolos binários como COM, DCOM e CORBA.

Principais vantagens da utilização do protocolo SOAP, em relação a outros sistemas de comunicação:

- ✓ Pode atravessar *firewalls* com facilidade.

- ✓ Os dados do SOAP são estruturados usando XML. Portanto, as mensagens podem ser compreendidas por quase todas as plataformas de *hardware*, sistemas operacionais e linguagens de programação.
- ✓ Pode ser usado, potencialmente, em combinação com vários protocolos de transporte de dados, como HTTP, SMTP e FTP.
- ✓ O SOAP faz o mapeamento satisfatoriamente para o padrão de solicitação / resposta HTTP.
- ✓ Pode ser usado tanto de forma anónima como com autenticação básica (username/password).

Principais desvantagens:

- ✓ Falta de interoperabilidade entre ferramentas de desenvolvimento do SOAP. Embora o SOAP tenha amplo suporte, ainda existem problemas de incompatibilidades entre diferentes implementações do SOAP.
- ✓ Mecanismos de **Segurança** Imaturos. O SOAP não define mecanismo para criptografia do conteúdo de uma mensagem SOAP, o que evitaria que outros tivessem acesso ao conteúdo da mensagem.
- ✓ Não existe garantia quanto à entrega da mensagem. Quando uma mensagem estiver sendo transferida, se o sistema falhar, ele não saberá como reenviar a mensagem.
- ✓ Um cliente SOAP não pode enviar uma solicitação a vários servidores, sem enviar a solicitação a todos os servidores.

O facto das aplicações permitirem que o SOAP seja usado com o HTTP permite transpor barreiras como *firewalls* com facilidade, permitindo que os softwares que aceitem SOAP estejam disponíveis internamente e externamente na rede. Esta característica pode ser vista como vantagem e também como desvantagem, já que pode causar um sério problema de segurança, onde as aplicações do SOAP seriam acessíveis por partes não autorizadas.
Resumindo, SOAP é um protocolo leve, o que faz ele ter poucos recursos e de fácil entendimento. Mas, por outro lado, nos faz ter uma preocupação maior com relação a segurança e transporte das mensagens.

3. Segurança: Soluções

Quando falamos de segurança há alguns pontos fundamentais a ter em conta: Autenticação, Autorização, Confidenciabilidade e a Integridade. Para cada para um deles é possível implementar soluções:

Autenticação: Quem enviou esta mensagem?

- Credenciais
  - Login/Password
  - Certificado Digital

Autorização: O que esta pessoa pode fazer?

- Usar Roles para definir privilégios

Confidencialidade: Quem pode ler esta mensagem?

- Encriptação
- Partilha de chaves secretas ou pares de chaves públicas/privadas para cifrar e descifrar

Integridade: Alguém alterou esta mensagem?

Assinatura Digital usada para comparar a mensagem enviada e recebida

WS- Especificações*

As especificações dos WS surgem como um meio para tornar standard várias partes dos Web Services, nomeadamente: Segurança, Políticas, Anexos, Confiança, Descoberta, etc.

A parte que interessa a este estudo é a que se refere à Segunça. Para tal existe o WS-Security que suporta, integra e unifica vários modelos, mecanismos e tecnologias de segurança em uso no mercado, permitindo que vários sistemas possam inter-operar em plataformas e linguagens neutras. Estas especificações de segurança definem um conjunto de padrões para extensões SOAP (Simple Object Access Protocol) ou para cabeçalhos de mensagens, utilizados para oferecer maior integridade e confidencialidade às aplicações de Web Services.. O WS-Security oferece os mecanismos-padrão de segurança necessários para realizar o intercâmbio seguro de mensagens certificadas, e define como usar encriptação XML e Assinatura XML no SOAP para tornar segura a troca de mensagens, como uma alternativa ou extensão do uso do protocolo HTTPS para tornar seguro o canal num ambiente de Web Services.

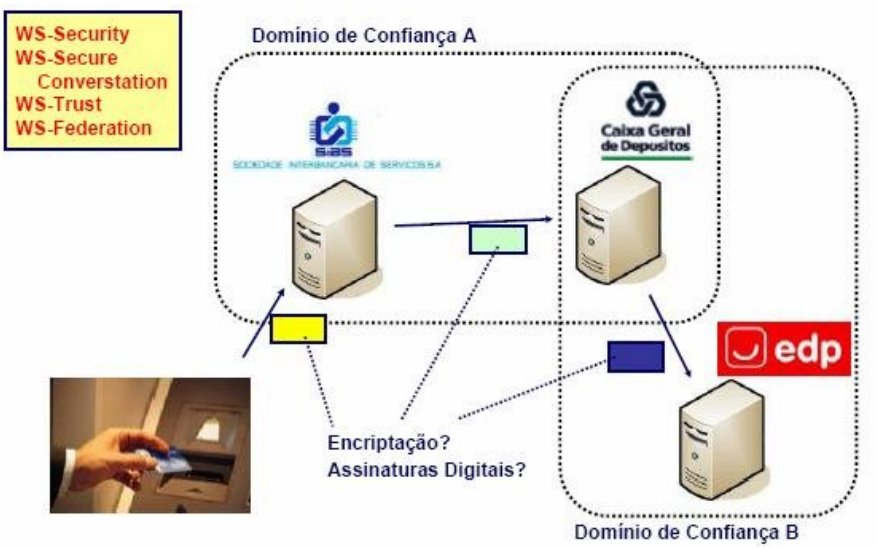


Figura 8 - Exemplo de um cenário com domínios de confiança

WS-Security e o SOAP (Simple Object Access Protocol)

Tal como já foi referido, as credenciais de segurança estão contidas na mensagem SOAP. O WS-Security define o elemento <Security> element, que permite que as security extensions sejam colocadas no <soapenv:header> da mensagem SOAP. Estas podem incluir:

- ✓ Username/password
- ✓ Encryption details
- ✓ XML Signature
- ✓ x.509 certificate
- ✓ Kerberos ticket
- ✓ XrML
- ✓ SAML

Uma mensagem SOAP é composta pelos principais elementos apresentados na figura 10.

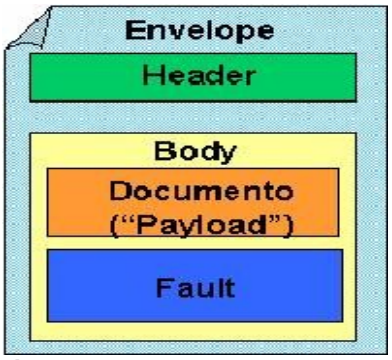


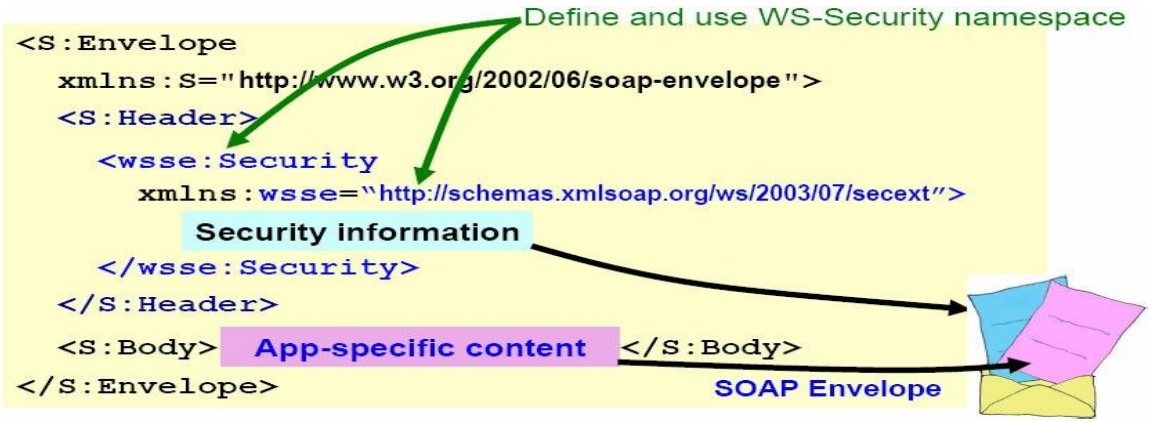
Figura 9 – Mensagem SOAP

Envelope: Toda mensagem SOAP deve conter. É o elemento raiz do documento XML. O *Envelope* pode conter declarações de namespaces e também atributos adicionais como o que define o "encoding style" que indica como os dados são representados no documento XML.

Header: É um cabeçalho opcional. Ele carrega informações adicionais, como por exemplo, se a mensagem deve ser processada por um determinado nó intermediário

Body: Este elemento é obrigatório e contém o payload, ou a informação a ser transportada para o seu destino final. O elemento *Body* pode conter um elemento opcional *Fault*, usado para carregar mensagens de status e erros retornadas pelos "nós" ao processarem a mensagem.

De seguida podemos ver um exemplo do código de um envelope SOAP, onde é especificado no Header a definição e a utilização do **WS-Security namespace**.



Estes protocolos são tipicamente implementados no headerSOAP!

3.3.1 Security Tokens para o elemento <Security>

O modelo base de serviços seguros foi proposto inicialmente pela IBM e Microsoft e faz actualmente parte da norma OASIS. O modelo descreve a forma como se processam as invocações que se pretendem seguras. Cada serviço tem uma política que define as condições de acesso e os *tokens* de segurança exigidos. Por exemplo, um serviço pode exigir que uma mensagem que recebe seja cifrada e que contenha um *token* com utilizador e senha. Se a mensagem chegar sem provas suficientes, o serviço pode ignorar ou rejeitar a mensagem.
A figura 10 ilustra o modelo de serviços seguros. As setas representam comunicação entre extremidades de serviços.

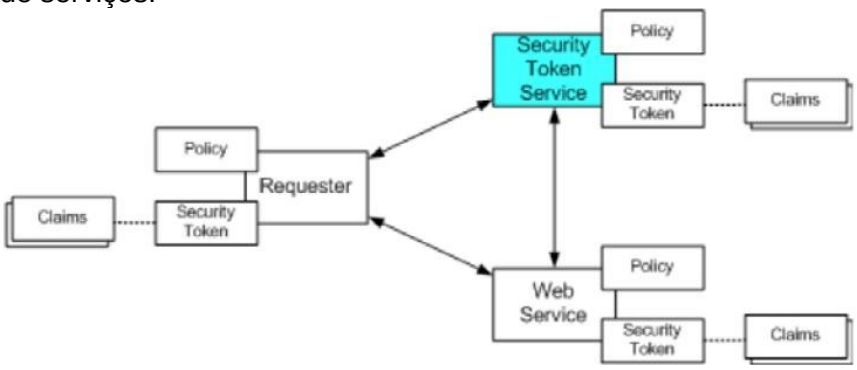


Figura 10 - Modelo de base para Web Services seguros.

Um *token* é, literalmente, um símbolo. Para os Web Services um Token é uma reivindicação qualquer que geralmente representa a identificação do utilizador. Trata-se de uma declaração feita por uma entidade, como nome, identidade, chave grupo, privilégio, capacidade, etc. O "username" é um exemplo de um unsigned security token.

Com o WSE é possível adicionar estes tokens à mensagem SOAP e verificar, em qualquer um dos lados, a validade dele.

O exemplo abaixo gera um token e adiciona à classe proxy do Web Service. Antes é preciso importar o namespace necessário:

(C#)

```
using Microsoft.Web.Services2.Security.Tokens;

UsernameToken token = new UsernameToken( "admin", "admin",
PasswordOption.SendPlainText );
Service1.RequestSoapContext.Security.Tokens.Add(token);
Service1.RequestSoapContext.Security.Elements.Add(new
MessageSignature(token));
```

Um Signed Security Token é criptograficament assinado por uma autoridade específica, como um X.509 certificate ou um Kerberos ticket. Este elemento pode ser usado para providenciar um user name no <wsse:Security>, para Basic Authentication.

```
<S:Envelope
   xmlns:S="http://www.w3.org/2002/06/soap-envelope">
   <S:Header>
     <wsse:Security
        xmlns:wsse="http://schemas.xmlsoap.org/ws/2003/07/secext">
        <wsse:UsernameToken wsu:ID="myToken">
           <wsse:Username>tigerplaid</wsse:Username>
           <wsse:Password>passw0rd</wsse:Password>
        </wsse:UsernameToken>                    Security Info
     </wsse:Security>
   </S:Header>
   <S:Body>
        App-specific content
   </S:Body>
</S:Envelope>
```

Figura 11 - < wsse:Security>, para Basic Authentication

Os Signed security tokens, como o Kerberos ticket ou x.509 certificate, são de conteúdo binário. Devem ser codificados para serem incluidos no wsse:Security container

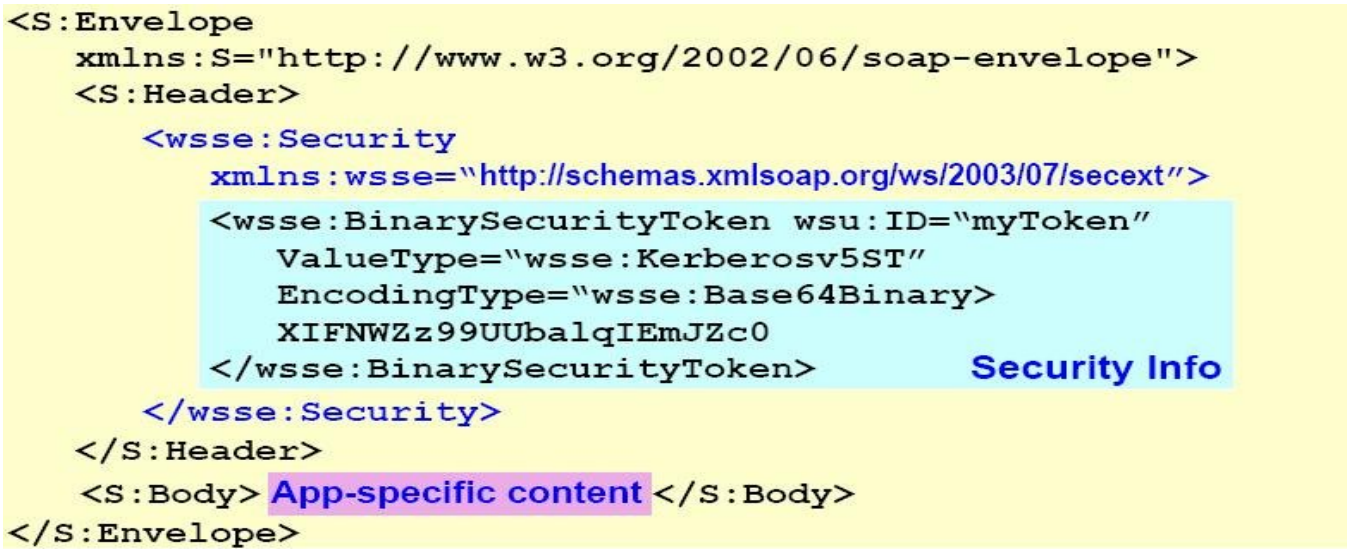


Figura 12 - O WS-Security elemento <BinarySecurityToken>

3.3.2 Encriptação XML

O standard de encriptação XML define a forma de cifrar todo um documento XML ou apenas partes. A informação cifrada é substituída por um único elemento <EncryptedData>. Pode-se cifrar diferentes partes do mesmo documento com diferentes chaves, ou então, pode-se cifrar todo o documento, um único elemento, ou apenas o texto de um elemento

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
        <Name>John Smith<Name/>
        <CreditCard Limit='5,000' Currency='USD'>
                <Number>4019 2445 0277 5567</Number>
                <Issuer>Bank of the Internet</Issuer>
                <Expiration>04/02</Expiration>
        </CreditCard>
</PayBalanceDue >
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
        Type='http://www.isi.edu/in-notes/iana/assignments/media-     types/text/xml'>
<CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
</EncryptedData>
```

A identidade do elemento de pagamento está escondida no form de encriptação. Não é possível ver nem o tipo de transacção que é!

No exemplo seguinte, o número <Number> é encriptado, mantendo-se os restantes elementos visíveis.

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
        <Name>John Smith<Name/>
        <CreditCard Limit='5,000' Currency='USD'>
                <Number>4018 2445 0277 5567</Number>
```

```
            <Issuer>Bank of the Internet</Issuer>
            <Expiration>04/02</Expiration>
        </CreditCard>
</PayBalanceDue >
```

O número é substituido pelo EncryptedData element:

```
<PayBalanceDue xmlns='http://example.org/paymentv2'>
        <Name>John Smith<Name/>
        <CreditCard Limit='5,000' Currency='USD'>
            <Number>
            <EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'
            Type='http://www.isi.edu/in-notes/iana/assignments/media-
        types/text/xml'>
            <CipherData><CipherValue>A23B4C6</CipherValue></CipherData>
            </EncryptedData>
            </Number>
            <Issuer>Bank of the Internet</Issuer>
            <Expiration>04/02</Expiration>
        </CreditCard>
</PayBalanceDue >
```

A *XML-Signature* e *XML-Encryption* são duas normas que definem como assinar e cifrar documentos XML, respectivamente. Ambas podem ser aplicadas selectivamente a partes da mensagem ou a conteúdos externos referenciados nos documentos. A diferença mais notória entre ambas é que elemento de assinatura (Signature) *referencia* o que está a ser assinado enquanto que o elemento de dados cifrados (EncryptedData) *contém* o que está a ser cifrado. A *WS-Security* permite proteger a mensagem SOAP e transportar tokens de segurança, definindo o modelo base de segurança descrito anteriormente. A WS-Security permite as seguintes operações sobre a mensagem SOAP:

- Acrescentar data e identificação à mensagem;
- Enviar tokens de segurança no cabeçalho;
- Usar XML-Signature para assinar toda ou parte da mensagem e enviar a assinatura no cabeçalho;
- Usar XML-Encryption para cifrar toda ou parte da mensagem;
- Enviar chaves criptográficas ou referências no cabeçalho.

Estas operações são descritas pelos seguintes elementos:

O elemento **<EncryptedData>** substitui o conteúdo a ser cifrado, e este contém:

<EncryptionMethod> Algoritmo usado para cifrar os dados

- ✓ **<CipherData>**
  <CipherValue> contém os dados encriptados

<EncryptedKey> elemento do security header, contém:

- ✓ **<EncryptionMethod>** Algortitmo usado para cifrar a chave simétrica
- ✓ **<KeyInfo>** Identificado da chave usado para cifrar a a chave simétrica
- ✓ **<CipherData>**
  <CipherValue> Chave simétrica cifrada
- ✓ **<ReferenceList>** Lista de **<DataReference>**s encriptados com a chave

```
<S:Envelope>
  <S:Header>
    <wsse:Security S:mustUnderstand="1"
       xmlns:wsse="http://schemas.xmlsoap.org/ws/2002/07/secext">
      <wsse:BinarySecurityToken EncodingType="wsse:Base64Binary">
         MIIDQTCC4ZzO7tIgerPlaid1q ... [truncated]
      </wsse:BinarySecurityToken>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
      ....see XML Signature example for full content...
      </ds:Signature>
    </wsse:Security>
 </S:Header>

 <S:Body>
    <m:OrderAircraft quantity="1" type="777" config="Atlantic"
       xmlns:m="http://www.boeing.com/AircraftOrderSubmission"/>
 </S:Body>
<S:Envelope>
```

Figura 13- Exemplo: Assinatura XML (no SOAP)

4. .NET Web Services e WSE

Apesar de os Web Services serem independentes da plataforma, refere-se aqui a plataforma .NET por ter suporte para todos os protocolos referidos. O ASMX é a implementação de Web Services para o .NET FrameWork, e este suporta uma segurança básica para serviços simples, mas não implementa especificações WS-*. Para tal existe o WSE é que uma extensão do .NET FrameWork, e extende as funcionalidades dos Web Services, provendo suporte para diversas especificações WS-*.

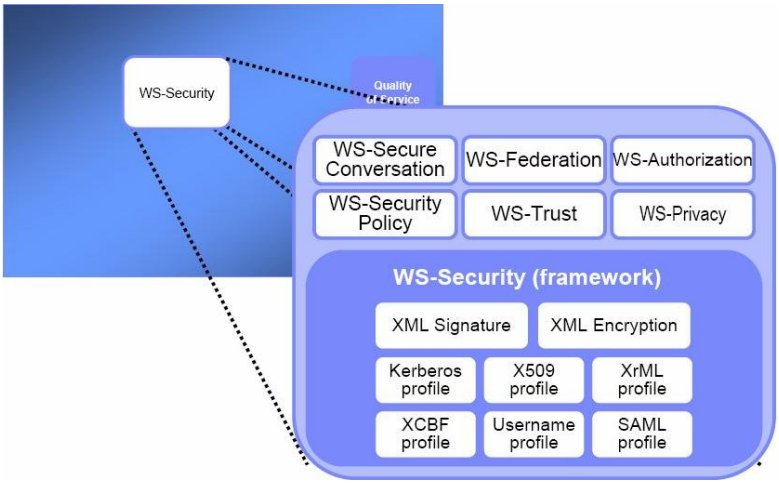


Figura 14 - WS-Security framework

A *WS-SecurityPolicy* é uma especialização da WS-Policy para definir políticas de segurança que descrevem a forma como as mensagens devem ser tornadas seguras. A política pode aplicar-se a mensagens individuais, a operações ou a toda a extremidade do serviço. As asserções WS-SecurityPolicy referem-se às funcionalidades de segurança de WS-Security, WS-Trust e WS-SecureConversation. Podem também referir segurança no transporte, como HTTPS.
Cada cenário é representado por um tipo de segurança, como apresentado na Figura 15.

Policy Assertion	Integrity	Confidentiality	Secure Conversation	Client Authentication	Service Authentication
UsernameForCertificateAssertion	✓	✓	✓	Username/password	X.509
MutualCertificate10Assertion	✓	✓	✓	X.509	X.509
MutualCertificate11Assertion	✓	✓	✓	X.509	X.509
AnonymousForCertificateAssertion	✓	✓	✓		X.509
UsernameOverTransportAssertion				Username/password	
KerberosAssertion	✓	✓	✓	Kerberos	

Figura 15– Policies Assertions

As asserções de *tokens de segurança* identificam quais são os tokens aceites e qual o processamento que lhes deve ser dado. Os tokens podem ser: HttpsToken, UsernameToken, X509Token, KerberosToken, SamlToken e RelToken. Por exemplo, o token HttpsToken indica o uso de HTTPS, podendo também especificar se o certificado cliente é obrigatório.
As *asserções* permitem expressar autenticação, atributos e autorização de agentes com identidade num domínio de segurança.

WS-Trust

A *WS-Trust* define um modelo de confiança com operações para adquirir, emitir, renovar e validar tokens de segurança e formas de criar novas relações de confiança através de serviços intermediários.

Define:

- ✓ O "security token service"
  - Uma autoridade de confiança para security tokens implementados como um Web service

- ✓ As mensagens SOAP são enviadas para este serviço para emissão de security token, validação e troca

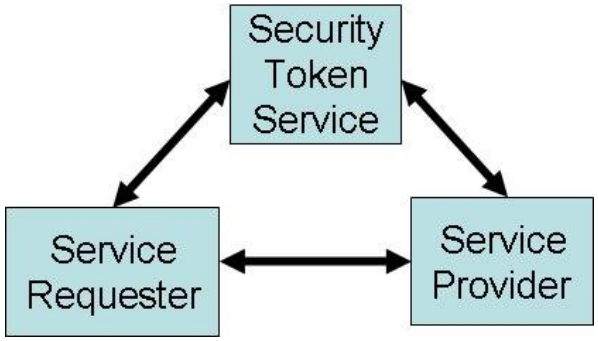


Figura 16 – WS-Trust

WS-Federation

A *WS-Federation* define federações de serviços para partilhar informação sobre identidade, atributos, autenticação e autorização entre diferentes domínios de confiança.

- ✓ Construido sobre o modelo WS-Trust
- ✓ Define mecanismos para troca de security tokens entre domínios de confiança
- ✓ Não requer a identidade local nos serviços destino
- ✓ Opcionalmente permite esconder a informação da identidade e de outros atributos.

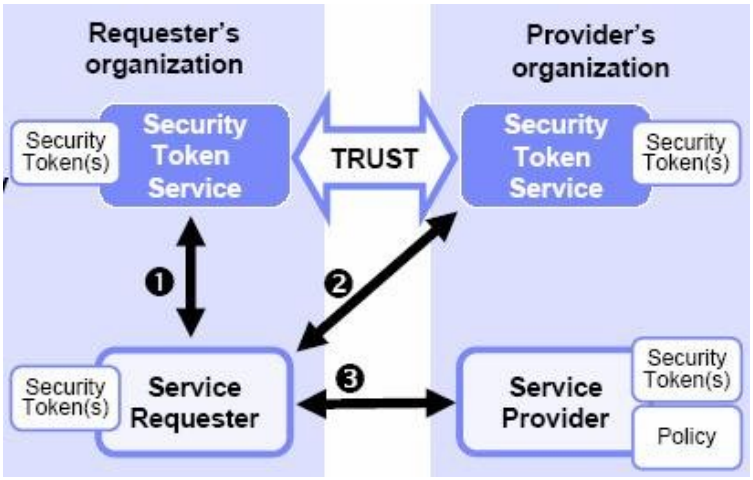


Figura 17 – WS-Federation

WS-Secure Conversation

A *WS-SecureConversation* permite que dois serviços estabeleçam uma sessão segura entre si para trocarem várias mensagens. Assim, é possível tirar partido de segredos partilhados para derivar chaves simétricas de sessão e ter segurança de forma mais eficiente e robusta.

Define os mecanismos para estabelecer e partilhar contextos de segurança e derivar chaves de sessão dos contextos de segurança. Existem 3 formas de estabelecer o contexto de segurança:

- ✓ Security context token criado pelo security token service
- ✓ Security context token criado por uma das partes da comunicação e propagado com a mensagem
- ✓ Security context token criado através de negociação

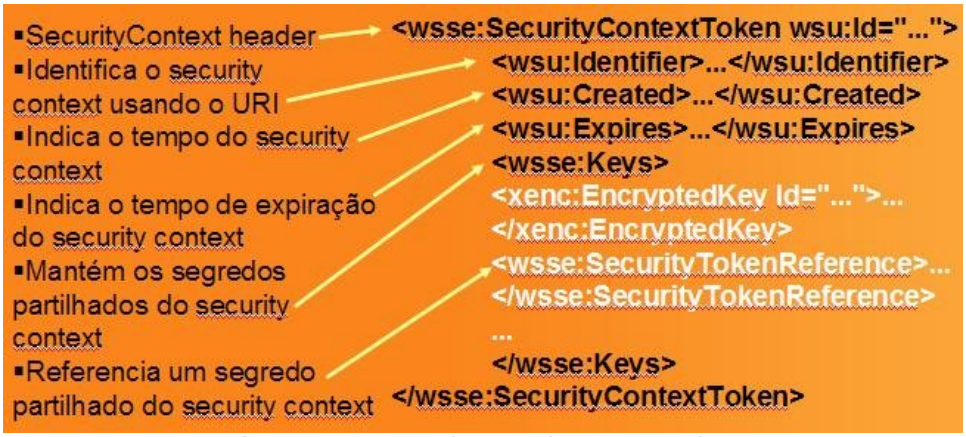


Figura 18 - Exemplo SecurityContextToken

WS-Privacy

O WS-Privacy define as políticas para a forma como o service provider irá usar a informação fornecida pelo service requester. É um modelo de como os utilizadores definem as suas preferências de privacidade, e como os os web services implementam as práticas de privacidade. As preferências são especificadas sob a forma de policy assertions no WS-Policy container.

WS-Authorization

O WS-Authorization decreve como as politicas de acesso ao Web Service são especificadas e geridas. Descreve com os pedidos podem ser especificados com security tokens, e como serão interpretados no final.

A Figura 19 apresenta algumas soluções existentes no mercado para a implementação de Web Services seguros, em várias plataformas.

Fornecedor	Implementação	Normas suportadas
Microsoft	**WSE 3**: Dot Net Framework 2.0, Visual Studio 2005, Web Services Enhancements 3.0	WS-Security: Username, X.509, Kerberos WS-Secure Conversation, WS-Trust SAML[2]
Apache	**WSS4J**: Apache Axis2, Rampart module of Web Services Security for Java (WSS4J)	WS-Security: Username, X.509 WS-Policy SAML
Sun Microsystems	**XWSS**: Java Web Services Developer Pack 2.0, XML and Web Services Security 2.0	WS-Security: Username, X.509 SAML

Figura 19 - Normas suportadas nas principais implementações de serviços seguros.

5 . Exemplo de um Web Service

calculadora.asmx

```
<!--Define que é um WebService, o tipo de linguagem e o nome da classe-->
<%@ WebService language="C#" class="Calculadora" %>

using System;
//Inclui as classes que são necessárias para um WebService
using System.Web.Services;
using System.Xml.Serialization;

//Define que o nome da classe
public class Calculadora
{
//O que torna um metodo de um classe público para ser invocado é a tag

//[WebMethod todas os métodos que tiverem esta tag serão públicos.
[WebMethod]
//Metodo que recebe dois inteiros os soma e retornar um tipo int
public int Add(int a, int b)
{
      return a + b;
}
catch
{
return 0;
} }}
```

Acede-se a um web service a partir de uma aplicação,criando uma web service proxy class. Depois usa-se a ferramenta de linha de comandos *Wsdl.exe* . A proxy class funciona como um intermediário entre o servidor de web service e o cliente de web service. A proxy class camufla toda a complexidade da chamada de um web service. O utilizador não precisa de saber SOAP, onde o web service reside e nem precisa de saber lidar com a mensagem de XML.

Conclusão

A implementação de mecanismos de seguranças em ambientes Web em que são partilhados serviço entre domínios de confiança é fundamental. As plataformas que implementam os Web Services já são dotadas de pacotes, como o WSE do .NET, que apresentam soluções práticas e funcionais para a maioria destes problemas. A configuração do WSE é auxiliada por wizards o que torna o processo para o utilizador bem mais fácil e transparente. As principais áreas de actuação do WSE incluem a possibilidade de envio de arquivos anexos, a inclusão de tokens e assinaturas digitais, e, por fim, a utilização de uma política de segurança. Para aplicações mais cruciais é aconselhável uma segurança mais robusta, utilizando as políticas que envolvem certificados X.509, Secure Conversations ou KerberosTokens.

Referências Bibliográficas

1. Kaler, C. & Nadalin, A., “Web Services Security Policy Language (WS-SecurityPolicy) Version 1.1”, Microsoft, IBM, VeriSign, RSA Security, 2005

2. Anthony Nadalin, C.K., “Web Services Security: SOAP Message Security 1.0 (WSSecurity 2004)”, OASIS, 2004

3. Hogg, J.; Smith, D.; Chong, F.; Taylor, D.; Wall, L. & Slater, P., “Web Service Security Scenarios, Patterns, and Implementation Guidance for Web Services Enhancements (WSE) 3.0”, Microsoft, 2005

4. IBM & Microsoft, “Security in a Web Services World: A Proposed Architecture and Roadmap Version 1.0”, IBM, Microsoft, 2002

5. Microsoft, “Microsoft Web Services Enhancements (WSE) 3.0 documentation”, 2005 http://msdn.microsoft.com/webservices/webservices/building/wse/default.aspx
40.