UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE ESPECIALIZAÇÃO EM DESENVOLVIMENTO, SEGURANÇA E QUALIDADE NA INTERNET

ADAIR JOSÉ WATHIER

Segurança em Web Services com WS-Security

Trabalho de Conclusão apresentado como requisito parcial para a obtenção do grau de Especialista

Prof. Dr. Cláudio F. R. Geyer
Orientador

Profa. Dra. Mara Abel
Coordenadora do Curso

Porto Alegre, dezembro de 2005

AGRADECIMENTOS

Agradeço inicialmente a Deus, pela vida.

Agradeço especialmente minha esposa Cibele, pela compreensão, companheirismo e estímulo para a realização do curso.

Além disso, muito obrigado a UFRGS pela oportunidade, aos professores, monitores do curso e colegas, que juntos concretizamos o curso MBA-Internet IV - 2004/2005.

Obrigado a todos.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

3DES	Triple – Data Encription Standard
AES	Advanced Encryption Standard
B2B	Business-to-business
DTD	Document Type Definition
ebXML	e-business XML
FTP	File Transfer Protocol
HTTP	Hiper Text Transfer Protocol
IDE	Integrated Development Enviroment
IETF	Internet Engineering Task Force
IPSec	Internet Protocol Security
MIME	Multipurpose Internet Mail Extension
OASIS	Organization for the Advancement of Structured Information Standards
RPC	Remote Procedure Call
SAML	Security Assertion Markup Language
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Security Socket Layer
SSO	Single Sign-On
TCP	Transmission Control Protocol
TSL	Transport Security Layer
UDDI	Universal Description Discovery and Integration
UDP	User Datagram Protocol
UFRGS	Universidade Federal do Rio Grande do Sul
URI	Uniform Resource Identifier
URL	Uniform Resource Locator

VPN	Virtual Private Network
WAN	Wide Area Network
WSDL	Web Service Description Language
WSS	Web Serices Security
WS-Security	Web Serices-Security
XML	eXtensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Uma nova tecnologia da Internet, chamada de serviços Web (*Web Services*), tem se destacado nos últimos tempos no meio computacional como uma revolução em termos de interoperabilidade de sistemas heterogêneos. Esta tecnologia tem sido considerada como a evolução da arquitetura de *middleware* tradicionais e ganhou o apoio de grandes produtores de *software* do mercado que a tratam como um novo paradigma no desenvolvimento de sistemas.

Por sua infra-estrutura pública, sujeita a ataques, questões de segurança na Internet são importantes. A segurança pode ser implementada em vários níveis de um determinado serviço. Os níveis, como de rede, transporte e aplicação, têm características próprias que devem ser analisadas com muito cuidado para se adequarem aos serviços Web sem criar impactos negativos que inviabilizam sua construção.

Pela grande complexidade na arquitetura de uma aplicação que usa serviços Web em relação a uma aplicação que simplesmente troca informações entre dois pontos, a segurança no nível de aplicação torna-se a mais viável nestes casos. Ela permite que a segurança esteja presente na própria mensagem, transportada de um ponto a outro até chegar ao destinatário final.

Contudo, já existem padrões promissores para a segurança no nível de aplicação, em especial o padrão WS-Security, baseado em assinaturas digitais e criptografia de documentos XML. A especificação mais recente deste padrão, definida pelo grupo OASIS (Organization for the Advancement of Structured Information Standards), descreve avanços na mensagem SOAP para garantir integridade e confidencialidade. O modelo especificado pode ser usado para acomodar uma variedade de modelos de segurança e tecnologias de criptografia.

Neste contexto, o presente trabalho contempla conceitos dos principais componentes e mecanismos de segurança envolvidos com os serviços Web. Apresenta um estudo dos principais padrões de segurança disponíveis no nível de aplicação para serviços Web, realizando uma análise desses padrões de segurança com principais ameaças e ataques que cada padrão propõe combater, além de implementar um estudo de caso onde é avaliado o uso do padrão de segurança WS-Security nos serviços Web.

Palavras-Chave: Segurança, Serviços Web , SOAP, WSDL, UDDI, Assinatura digital, Criptografia, XML, Padrão WS-Security, Internet.

Securing Web Services with WS-Security

ABSTRACT

A new technology of the Internet, call of Web Services, has if outstanding in the last times in the computing environment as a revolution in terms of interoperability of heterogeneous systems. This technology has been considered as the evolution of the architecture of traditional middleware and it won the support of big producing of software of the market that treat her as a new paradigm in the system development.

For his public infrastructure, it subjects to you attack, Internet security subjects are important. The security can be implemented in several levels of a certain service. The levels, as of network, transport and application, they have own characteristics that should be analyzed with very care for if they adapt to the Web Services without creating negative impacts that make unfeasible her construction.

For the great complexity in the architecture of an application that uses Web Services in relation to an application that simply exchanges information among two points, the security in the application level becomes the viable in these cases. Allows the security to be present in the own message, transported of a point the other to arrive to the final addressee.

However, already promising standards exist for the security in the application level, especially the WS-Security standard, based on digital signatures and encryption of documents XML. The most recent specification of this standard, defined for the group OASIS (Organization goes the Advancement of Structured Information Standards), it describes progresses in the message SOAP to guarantee integrity and confidentiality. The specified model can be used to accommodate a variety of models of security and encryption technologies.

In this context, the present work contemplates concepts of the main components and mechanisms of security involved with the Web services. It presents a study of the main available standards of security in the application level for Web services, accomplishing an analysis of those standards of security with main threats and attacks that each standard intends to combat, besides implementing a case study where the use of security's WS-Security standard it is evaluated in the Web services.

Keywords: Security, Web Services, SOAP, WSDL, UDDI, Digital Signature, Encryption, XML, WS-Security Standard, Internet.

1 INTRODUÇÃO

Os serviços Web começam a amadurecer, ganham novos padrões e já cumprem o que parecia impossível: a integração de plataformas a baixo custo. No entanto, a questão da segurança ainda é alvo de muito estudo e pesquisa.

Muitas aplicações precisam de algum nível de segurança, e segurança tem sido freqüentemente considerada um grande obstáculo na adoção de serviços Web. Problemas de segurança comuns a tecnologias de integração e de computação distribuída devem ser enfrentados.

No universo dos serviços Web, que vai muito além da simples transmissão de dados, segurança significa mais que inviolabilidade da informação. Constitui um conjunto de propriedades que inclui também identificação, autenticação, autorização, integridade, privacidade e não-repúdio. Estas são propriedades de segurança dos serviços Web, quando utilizadas com os respectivos padrões de segurança estabelecidos, garantem um nível de segurança desejado.

Motivado pelas vantagens dos serviços Web e a segurança exigida dos mesmos na construção de aplicações que usam esses serviços, tornou-se o principal objetivo deste trabalho estudar os mecanismos de segurança utilizados como padrões de segurança nos serviços Web, em especial o padrão WS-Security. O trabalho será finalizado com um estudo de caso, onde serão aplicadas algumas tecnologias de segurança estabelecidas pelos padrões vistos neste trabalho.

O estudo de caso procura explorar os conceitos estudados, através da implementação de um protótipo de serviço Web. Apesar da simplicidade do protótipo, atende os propósitos deste trabalho, com a demonstração do comportamento e funcionamento de um serviço web considerado seguro.

A estrutura do trabalho que segue, consiste no capítulo 2 que traz o referencial teórico dos assuntos relacionados a serviços Web e padrões de segurança na Internet. O capítulo 3 apresenta as principais ameaças que os serviços Web podem sofrer e as medidas de segurança para diminuir os riscos dessas ameaças. O capítulo 4 detalha um estudo de caso da implementação de um serviço Web seguro, no padrão da especificação de segurança WS-Security. Enfim, o capítulo 5, conclui o trabalho e apresenta algumas idéias e sugestões de trabalhos futuros.

2 SEGURANÇA NOS SERVIÇOS WEB

Esta seção fornece as informações e conceitos subjacentes exigidos para se implementar serviços Web seguros com êxito. Antes de empregar serviços Web seguros, é necessário entender quais os problemas eles resolvem, a motivação por trás deles e a segurança necessária exigida de cada um deles. Isso deve garantir a aplicação dos serviços Web seguros em lugares apropriados dentro de um determinado aplicativo.

2.1 Introdução a Serviços Web

Os serviços Web, também conhecidos como Web Services, são vistos por muitos como a próxima onda da revolução da Internet. A visão é a de uma Web tão rica em funcionalidades como a atual é rica em informação. O desafio é expor essa funcionalidade de uma maneira consistente e útil.

Nos últimos tempos o termo Web Services tem chamado a atenção dos profissionais de informática, principalmente os mais voltados para *business-to-business* (B2B). O conceito foi criado, implementado e agora está começando a ser utilizado. As expectativas são grandes, altos investimentos, *frameworks*[1] poderosos, ganhos em produtividade, portabilidade e independência, para que os serviços Web formarão uma grande parte do desenvolvimento de aplicativos nos próximos anos (BOND, 2003).

Serviços Web foram criados para prover serviços comuns para aplicações B2B. Porém, não faz parte do conceito de serviços Web a criação de interfaces gráficas para os usuários, deixando esta parte para outros desenvolverem. É aceito afirmar que serviços Web disponibilizam serviços somente para desenvolvedores, ou que serviços Web nada mais são do que chamadas de métodos usando XML (PAMPLONA, 2004).

Atualmente as vantagens de serviços Web são atrativas para muitos serviços, oferecendo certos benefícios sobre outras tecnologias. Entre essas vantagens encontra-se a integração própria para B2B. Serviços Web oferecem desenvolvimento rápido e de baixo custo, fácil distribuição e localização na rede oferecendo flexibilidade e interoperabilidade.

No entanto, muitas aplicações precisam de algum nível de segurança, e segurança tem sido freqüentemente considerada um grande obstáculo na adoção de serviços Web.

Para implementar requisitos de segurança em aplicações para Web, geralmente são utilizados os seguintes mecanismos de proteção:

---

[1] Conjunto de funcionalidades comuns para um determinado domínio de aplicações.

a) **Autenticação** - que permite identificar se o usuário ou aplicação cliente é quem ele diz ser;
b) **Autorização** - que permite o estabelecimento de direitos diferentes a usuários diferentes;
c) **Privacidade -** que restringe a capacidade de leitura de mensagens por interceptação, com o uso de criptografia;
d) **Integridade** - que permite detectar se os dados foram alterados no percurso entre a origem e o destino na rede;
e) **Não repúdio (auditoria)** - com o registro dos eventos ocorridos no sistema, como transações efetuadas e acessos a recursos (STALLINGS, 2000).

Para serviços Web, é necessário que a implementação de requisitos de segurança mantenha a interoperabilidade, que é um dos maiores atrativos da tecnologia. Além disso, é desejável que as medidas de segurança não dificultem a implementação dos serviços Web e de seus clientes (COSTA, 2005).

A implementação de segurança em serviços Web pode ser tratada em diferentes níveis ou camadas que são: camada de rede, camada de transporte e camada de aplicação, com os mais variados padrões para cada camada.

Na camada de rede, por exemplo, existe o IPSec (Internet Protocol Security) que tenta tornar o canal Internet seguro por si só, o que é feito através da proteção à própria rede. As redes que implementam essa segurança são chamadas de VPNs (Virtual Private Networks) que possibilitam garantir a privacidade e a autenticação dos dados, além da autenticação do usuário, em redes públicas (HENDRICKS, 2002).

No nível de transporte, para garantir confidencialidade e integridade dos dados, serviços Web podem utilizar o HTTPS, a extensão do protocolo HTTP que utiliza Secure Socket Layer (SSL) e certificados digitais (COSTA, 2005).

No nível de aplicação, a segurança é oferecida no próprio protocolo de troca de mensagens - SOAP (Simple Object Access Protocol), padrão de estrutura das mensagens XML usado na comunicação entre aplicações que fazem uso de serviços Web. Com este recurso é possível uma mensagem passar por vários serviços Web ao longo do seu processamento (ROSENBERG, 2004).

Cada nível de segurança possui suas características que contribuem para a segurança dos serviços Web. No entanto, nenhum nível de segurança é tão independente da integração de diferentes plataformas como a segurança no nível de aplicação. Uma das principais vantagens do uso de serviços Web nas aplicações web é garantir interoperabilidade entre plataformas distintas. O ideal nestes casos, para aplicações que usam serviços Web é implementar a especificação WS-Security que garante a segurança do serviço no nível de aplicação e interoperabilidade entre plataformas distintas.

A especificação WS-Security define um conjunto-padrão de extensões SOAP. Essas extensões podem ser usadas para implementar autenticação, integridade e confidencialidade no nível de mensagens SOAP (COSTA, 2005).

2.2 Visão Geral da Tecnologia Serviços Web

Os serviços Web fornecem um mecanismo de integração flexível e poderoso, que pode ser usado para expor funcionalidade e componentes existentes para outras empresas ou para novos aplicativos.

Podem ser vistos como a evolução mais recente e significativa do software. A programação procedural evoluiu para a programação orientada a objetos, melhorando a modelagem de elementos do sistema, o encapsulamento dos dados e a funcionalidade. O desenvolvimento com base em componentes fornece uma estrutura padronizada e rica em serviços, na qual a funcionalidade orientada a objetos pode ser implantada e transformada em aplicativos. Os serviços Web tiram proveito dos protocolos comuns da Web para tornar as instâncias de componentes facilmente acessíveis dentro e fora da empresa.

Um serviço Web é basicamente um componente aplicativo que pode ser acessado, usando-se protocolos da Web e mecanismos de codificação de dados, como HTTP e XML. Em alguns casos, isso será um componente de outro fornecedor contido de forma remota. A diferença entre um serviço Web e um componente tradicional reside não apenas nos protocolos usados para acessá-lo, mas também no fato de que o serviço pode trazer seus próprios dados e funcionalidades com ele.

Um exemplo disso seria um serviço de câmbio. Sob o modelo de componente, um componente de câmbio poderia trazer com ele um arquivo contendo um conjunto fixo de taxas de câmbio, que devesse ser atualizado regularmente. Entretanto, ficaria por conta da manutenção da aplicação garantir que essa informação fosse atualizada. Por sua vez, um serviço de câmbio assume a responsabilidade por essa atualização. O aplicativo apenas faz uso do serviço de câmbio e deixa os detalhes da obtenção dos dados exigidos e os serviços subsidiários para aqueles que implementam e hospedam o serviço (BOND, 2003).

No modelo serviços Web, identificam-se três importantes tipos de papéis com suas respectivas operações:

Tabela 2.1: Papéis no fluxo de mensagens dos serviços Web

Papel	Operação	Descrição
Provedor de Serviço Web	Publicar	Registra a descrição e localização do serviço em um site de Registro de Serviços que esteja disponível a todos ou aos interessados.
	Fornecer serviço	Disponibiliza uma funcionalidade que pode ser usada por diversas aplicações clientes do serviço Web.
Aplicativo Cliente do Serviço Web	Pesquisa	Recupera informação do serviço desejado do Registro do Serviço Web.
	Vincula	Faz a ligação da chamada remota e inicia a interação com o serviço.
Registro do Serviço Web	Armazena	Lugar em que os Provedores de Serviços Web publicam a descrição e localização de seus serviços.
	Distribui	Fornece informações do serviço em tempo de desenvolvimento (ligação estática) ou execução (ligação dinâmica).

Um modelo completo de interação entre um aplicativo baseado em serviço Web e o serviço Web em si é mostrado na figura abaixo:

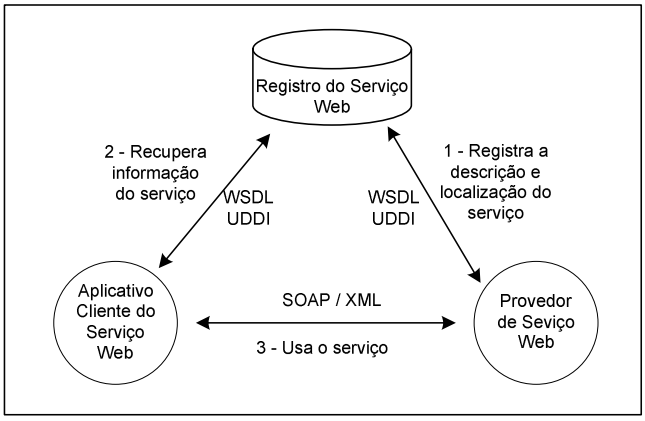


Figura 2.1: Arquitetura dos serviços Web

A interação inicia com o Provedor de Serviço Web registrando a descrição e localização do seu serviço. No passo 2 o Aplicativo Cliente do serviço Web recupera as informações do serviço desejado, necessárias para localizar e chamar o serviço remoto. Por último, no passo 3, o aplicativo cliente invoca o serviço Web no provedor de serviços toda vez que desejar usá-lo.

2.2.1 Tecnologias e Protocolos de serviços Web

A base dos serviços Web é formada por quatro tecnologias: eXtensible Markup Language (XML); Simple Object Acess Protocol (SOAP); Web Services Description

Language (WSDL); Universal Description, Discovery, and Integration (UDDI) (ROSENBERG, 2004).

O protocolo SOAP define o formato que as mensagens transportadas na rede devem ter para encaminhar requisições a serviços Web. Também define o formato das mensagens de resposta a requisições. Já o WSDL consiste em uma linguagem XML para descrição de interfaces de serviços, visando tornar essa descrição inteligível para programas que irão interagir com esses serviços. O UDDI, por sua vez, possibilita o armazenamento e recuperação de informações dos serviços Web.

A figura abaixo ilustra a pilha de protocolos utilizados na construção de um serviço Web (W3C, 2005).

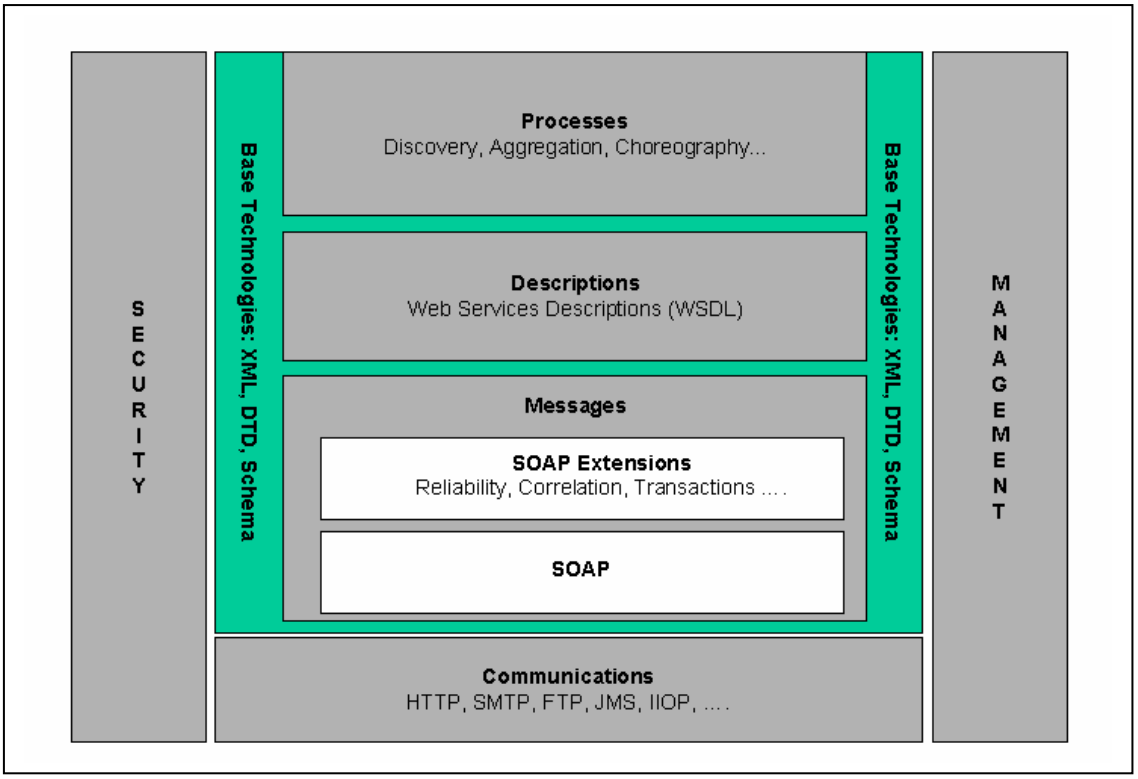


Figura 2.2: Pilha de protocolos serviços Web

A seguir será detalhado cada um destes protocolos XML, SOAP, WSDL e UDDI para dar um entendimento maior sobre o funcionamento dos serviços Web.

2.2.2 XML

XML é um formato de texto muito simples e flexível derivado do SGML (Standard Generalized Markup Language) (W3C, 2005). Originalmente projetado para encontrar chaves em documentos eletrônicos em larga escala, XML está também se tornando um importante padrão para a troca de uma vasta variedade de dados na web.

Com o advento do XML tornou-se mais fácil para sistemas de diferentes ambientes trocarem informações. A universalidade do XML tornou-se uma maneira muito atrativa para comunicação entre programas. Programadores podem usar diferentes sistemas operacionais, linguagens de programação etc. e ter seus softwares comunicando-se com outros de uma maneira ininterrupta.

Uma sintaxe para descrição de dados, XML é uma definição conduzida por uso de DTDs e schemas que permitem a manipulação de informações entre aplicações. Tags podem ser combinadas, interfaces podem ser definidas e processamentos podem ser padronizados. Serviços Web são componentes de programas reutilizáveis que utilizam XML como padrão de manipulação de informações entre aplicações, este padrão se enquadra em um *framework* extensível para facilitar a comunicação computador-a-computador.

XML faz um papel crucial integrando dados e coordenando a lógica de negócio. Tarefas específicas de negócio e serviços, inclusive lógica de *workflow*[2], lógica de negócio, lógica seqüencial de componente, lógica de transação e assim por diante, podem ser encapsulados em documentos XML e integrados a ambientes empresariais existentes.

SOAP é uma tecnologia que deriva do padrão de formatação de texto baseado em XML (XML-RPC) e aponta para um padrão emergente chamado ebXML (XML empresarial eletrônico). ebXML está em evolução, provendo definições inclusive de mensagens empresariais compartilhadas entre parceiros comerciais. SOAP é mais modesto em extensão e menos complexo em implementação (CLEMENTS, 2005).

2.2.3 SOAP

O protocolo SOAP foi criado para transportar mensagens XML de um computador para outro, via vários protocolos padrões de transporte. HTTP é o mais comum destes protocolos, obviamente porque predomina no uso da Web.

SOAP se define usando XML, que proporciona com simplicidade e coerência uma maneira de uma aplicação enviar mensagem XML para outra. SOAP é o que faz a integração entre aplicações ser possível, pois após a definição do conteúdo do XML, é o SOAP que transfere os dados de um lugar para outro pela rede. Permite enviar e receber documentos XML que suportam um protocolo comum de transferência de dados. Além disso, SOAP permite tratar mensagens XML retornadas de um serviço remoto e seu modelo possibilita de forma clara a separação entre os dados de processamento de infra-estrutura e processamento de mensagens de aplicação.

A figura abaixo apresenta a estrutura básica de uma mensagem SOAP de acordo com (ROSENBERG, 2004).

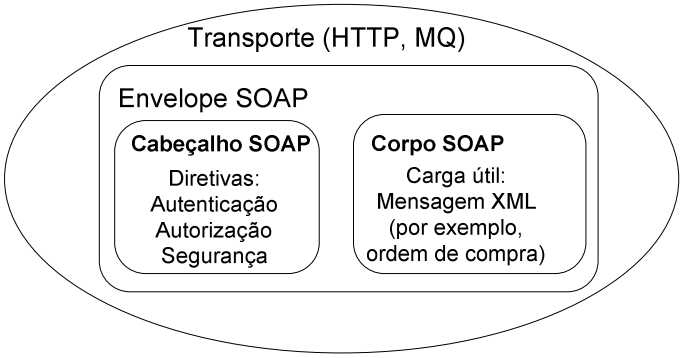


Figura 2.3: Estrutura básica da mensagem SOAP

[2] Processo pelo qual a informação flui por determinada organização, de maneira rápida e organizada, seguindo a seqüência pré-estabelecida de tramitação.

SOAP fornece um envelope (Envelope SOAP) para a mensagem XML, este envelope é um *container*[3] que protege os dados XML. O objetivo é criar um *container* uniforme para que mensagens SOAP possam ser transmitidas por qualquer protocolo de transporte (HTTP, MQ). O protocolo evita que aplicação conheça o protocolo de transporte e se mantém coerente com o envelope SOAP.

O envelope SOAP é composto por duas partes: o cabeçalho (Cabeçalho SOAP) e o corpo da mensagem (Corpo SOAP).

Cabeçalho SOAP (*header*) – contém informações sobre a mensagem SOAP. Estas informações são usadas para gerenciar ou prover segurança para o pacote.

Corpo SOAP (*body*) – contém a mensagem SOAP propriamente dita (Carga útil). Esta informação é enviada de uma aplicação para outra. Poderia ser um documento com uma ordem de compra ou um contrato, poderia ser uma descrição de uma classe com métodos remotos e seus parâmetros.

Um exemplo de mensagem SOAP é listado na figura abaixo (ROSENBERG, 2004):

```
<?xml version="1.O" ?>
<env:Envelope xmlns:env="http://www.w3.org/2001/12/soap-
              envelope">
   <env:Header>
     <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
       <n:priority>1</n:priority>
      <n:expires>2004-06-22T14:00:00-5:00</n:expires>
     </n:alertcontrol>
   </env:Header>
   <env:Body>
     <m:alert xmlns:m="http://example.org/alert">
       <m:msg>Pick up Bobby at school at 2PM</m:msg>
     </m:alert>
   </env:Body>
</env:Envelope>
```

Figura 2.4: Mensagem SOAP – envelope, cabeçalho e corpo

SOAP necessita ser seguro. A mensagem transportada deve ser de conhecimento somente dos seus receptores. O serviço remoto deve conhecer quem está requisitando seu serviço e se está autorizado. SOAP é um mecanismo de pacote para mensagens XML e documentos. Muitos pacotes necessitam descrever importantes informações sobre o que o pacote todo contém, por exemplo: o remetente, como o receptor vai validar o remetente, quais permissões o remetente possui e assim por diante. Isto basicamente representa a implementação de segurança na própria mensagem SOAP discutida mais adiante (ROSENBERG, 2004).

2.2.4 WSDL

WSDL é uma linguagem XML que define as operações que um serviço Web provê e a estrutura dessas operações em relação às mensagens SOAP. Isto é, define as estruturas de entrada e saída de um serviço Web, associação entre os parâmetros e tipos de dados.

WSDL contém informações de um serviço para que outros possam interagir com este serviço, como a localização do serviço, o que o serviço pode fazer e como poderá ser chamado.

---

[3] Área delimitada e protegida destinada para armazenar algo.

Um arquivo WSDL possui três seções: o quê, como e onde. Na primeira seção, um arquivo WSDL especifica as mensagens de entrada e saída, representando o que o serviço faz. A segunda seção define como as mensagens devem se empacotadas na mensagem SOAP e como devem ser transportadas. Além disso, define que informação deve estar no cabeçalho do SOAP. Na terceira e última seção descreve a implementação específica de um serviço Web e onde poderá ser encontrado (ROSENBERG, 2004).

2.2.5 UDDI

Na construção de serviços Web, esses serviços necessitam ser acessados em algum lugar na Web por uma aplicação cliente. Uma forma de acessar um serviço Web é fazer com que a aplicação cliente conheça a URI (Uniform Resource Identifier) do serviço, desta maneira caracteriza o modo estático de localizar e acessar um serviço. Entretanto, quando a aplicação cliente não detém a localização de um serviço Web, este pode ser descoberto antes de ser acessado, caracterizando o modo dinâmico de descobrir a localização de um serviço.

O UDDI provê um método padronizado para a publicação e descoberta de informações sobre serviços Web. É uma iniciativa de criar um *framework* padrão para descrição de serviços, descoberta de negócios e integração de serviços comerciais focando na pesquisa e na arquitetura orientada a serviços (SOA - Service Oriented Architecture) (CHAPPELL, 2002) .

2.3 Segurança em Serviços Web

Segurança em serviços Web é um dos assuntos mais importantes de serviços Web. A preocupação com segurança no uso serviços Web assemelha-se à segurança necessária para aplicações da Internet, aplicações baseadas em *middleware*[4] e em comunicação.

A segurança nesse tipo de aplicação é basicamente garantida pela autenticação do usuário, utilização de algoritmos de criptografia[5] e assinaturas digitais[6]. Na maior parte, os algoritmos de criptografia e assinaturas digitais são usados juntos, um em complemento de outro. Isto é, os algoritmos de criptografia procuram garantir a privacidade e as assinaturas digitais procuram garantir a integridade das mensagens (STALLINGS, 2000).

A figura abaixo ilustra um exemplo de um caixa de banco (cliente do serviço Web) que se conecta via Internet com o site central do banco (provedor do serviço Web) para executar suas transações bancárias sem segurança segundo (WHALI, 2005).

---

[4] Software de interface que permite interação de diferentes aplicações de software, geralmente sobre diferentes plataformas de hardware e infra-estrutura, para troca de dados.

[5] Embaralhar um conjunto de dados com código secreto, para que as informações deste conjunto não possam ser utilizadas ou lidas até serem decodificadas (decriptografar).

6 Conjunto de instruções matemáticas baseadas na criptografia, que permite conferir autenticidade, privacidade e inviolabilidade aos documentos digitais e transações comercias efetuadas pela Internet.

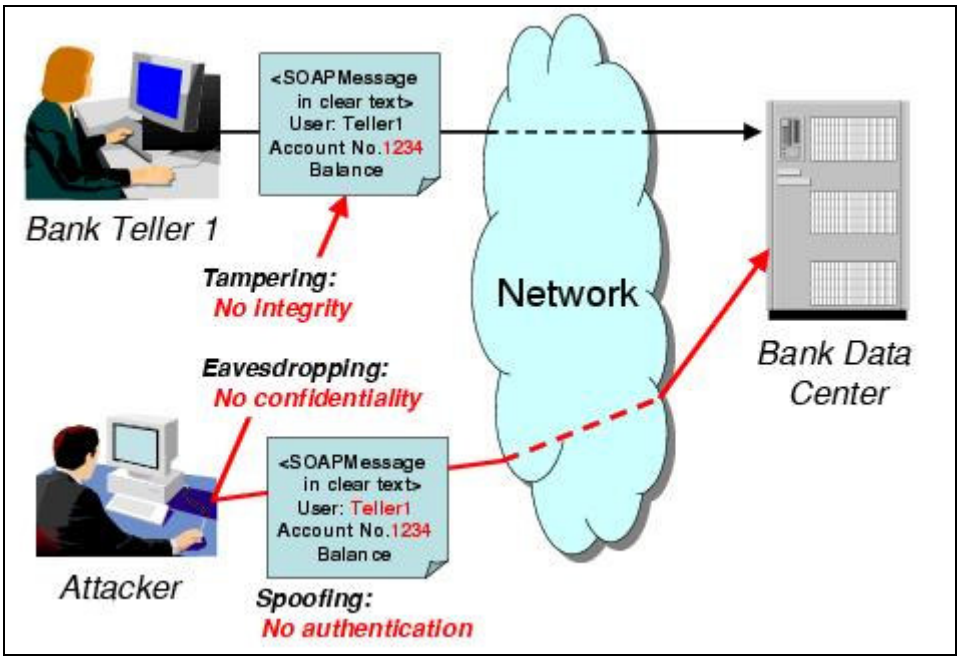


Figura 2.5: Exemplo de uso serviço Web

Os três principais fatores de risco neste exemplo são:

a) **Sem Autenticação (*no authentication- Spoofing*)** – um infrator (*Attacker*) poderia enviar uma mensagem SOAP modificada ao provedor de serviços (*Bank Data Center*), enquanto finge ser um caixa de banco (*Bank Teller*), acessa informações confidencias ou retira dinheiro da conta de outro cliente. Implementando a autorização ao serviço Web, poderia eliminar esta exposição de insegurança;

b) **Sem Integridade (*no integrity - Tampering*)** – uma mensagem é interceptada entre o cliente (*Bank Teller*) e o servidor (*Bank Data Center*) do serviço web. O infrator (*Attacker*) poderia modificar a mensagem, por exemplo, depositar o dinheiro em outra conta mudando o número da conta. Porque não há nenhuma restrição de integridade, servidor do serviço Web não confere se a mensagem é válida e aceitará a transação modificada. Implementando um mecanismo de integridade ao serviço Web, poderia eliminar esta insegurança;

c) **Sem confidencialidade (*no confidentiality - Eavesdropping*)** – um infrator pode interceptar a mensagem SOAP e ler toda a mensagem. Pois a mensagem não é criptografada, informações de confidenciais de clientes e do banco podem ir para pessoas erradas. Esta exposição existe porque as informações são enviadas em texto claro pela rede (WHALI, 2005).

Para prevenir as vulnerabilidades apresentadas, existem várias alternativas de solução para o problema da segurança em serviços Web.

2.3.1 Segurança no nível de rede

Uma das soluções para garantir segurança no nível de rede são as VPNs (Virtual Private Networks) que se baseiam, geralmente, em IPSec (Internet Protocol Security), um *framework* de padrões abertos desenvolvidos pela IETF (Internet Engineering Task Force), para garantir a privacidade e a autenticação dos dados, além da autenticação do usuário, em redes públicas (HENDRICKS, 2002).

As VPNs implementam a tecnologia de tunelamento que pode ser definido como processo de encapsular um protocolo dentro de outro. O uso do tunelamento incorpora um novo componente a esta técnica: antes de encapsular o pacote que será transportado, este é criptografado de forma a ficar ilegível caso seja interceptado durante o seu transporte. O pacote criptografado e encapsulado viaja através da Internet até alcançar seu destino onde é desencapsulado e decriptografado, retornando ao seu formato original (IEC- International Engineering Consortium, 2005).

As VPNs podem se constituir numa alternativa segura para transmissão de dados através de redes públicas ou privadas, uma vez que já oferecem recursos de autenticação e criptografia com níveis variados de segurança, possibilitando eliminar os links dedicados de longa distância, de alto custo, na conexão de WANs. Entretanto, em aplicações onde o tempo de transmissão é crítico, o uso de VPNs através de redes externas ainda deve ser analisado com muito cuidado, pois podem ocorrer problemas de desempenho e atrasos na transmissão sobre os quais a organização não tem nenhum tipo de gerência ou controle, comprometendo a qualidade desejada nos serviços corporativos (VIERA, 2005).

2.3.2 Segurança no nível de transporte

Segurança no nível de transporte significa proteger o protocolo de rede que o serviço Web utiliza para se comunicar. O Secure Sockets Layer (SSL) é um protocolo padrão para encriptar comunicações sobre TCP/IP. Neste modelo, um cliente abre um socket seguro para um serviço Web e então o utiliza para trocar mensagens SOAP via HTTPS.

Implementação SSL (Secure Socket Layer) é razoavelmente simples e a implementação de SSL garante segurança encriptando todo o tráfego de rede sobre o socket, mas não se deve esquecer da sobrecarga em relação ao desempenho do servidor Web.

Tal uso do SSL faz o servidor Web trabalhar muito para criptografar e decriptografar as informações que são enviadas através do HTTP. Se a aplicação on-line for usada por milhares de usuários, o SSL deve ser usado cautelosamente para que não ocorra um forte impacto sobre os usuários que teriam, assim, uma resposta lenta (HENDRICKS, 2002).

2.3.3 Segurança no nível de aplicação

No contexto de serviços Web, chamado de segurança no nível de mensagem XML, envolve a encriptação e decriptação de documentos XML. O World Wide Web Consortium (W3C), o qual mantém o padrão XML, tem criado grupos de trabalhos para definir padrões para segurança em XML, incluindo assinaturas digitais, encriptação e gerenciamento de chaves para XML (W3C, 2005).

As implementações de segurança no nível de mensagem podem ser usadas para garantir a confidencialidade e a integridade das mensagens, conforme elas passam por um número arbitrário de sistemas intermediários. As mensagens podem ser assinadas para fornecer integridade. Para a confidencialidade, pode-se escolher entre criptografar toda a mensagem ou parte de uma mensagem.

Há várias vantagens em proteger a mensagem em vez de usar o protocolo de transporte. Em primeiro lugar, esse método é mais flexível, pois partes da mensagem ao invés da mensagem inteira, podem ser assinadas ou criptografados. Isso significa que os intermediários conseguirão ver as partes da mensagem destinadas a eles. Um exemplo é

um serviço Web que roteia mensagens SOAP e é capaz de inspecionar partes não criptografadas de uma mensagem para determinar o local para onde a mensagem deverá ser enviada, enquanto outras partes da mensagem permanecem criptografadas. Depois, os intermediários poderão adicionar seus próprios cabeçalhos à mensagem e assiná-la para fins de registro de auditoria. Finalmente, a mensagem protegida poderá ser enviada através de vários protocolos diferentes, como HTML, SMTP, FTP e TCP, sem que seja preciso contar com o protocolo para garantir a segurança.

A figura abaixo ilustra a implementação de segurança no nível de mensagem, onde as mensagens XML podem ser transportadas sob qualquer protocolo, com informações de segurança: credenciais, assinaturas digitais e mensagens criptografadas.

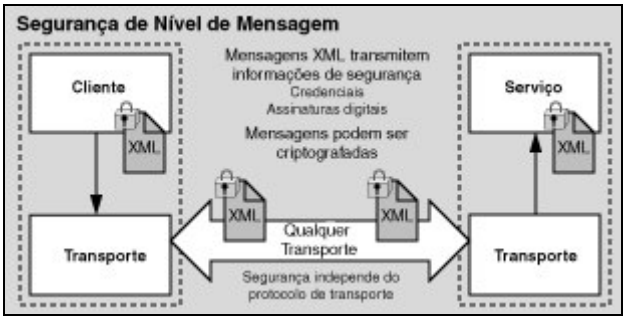


Figura 2.6: Segurança no nível de mensagem

Representa uma abordagem mais flexível e poderosa comparada aos níveis de rede e transporte, usada principalmente pela especificação WS-Security (MICROSOFT, 2005).

2.4 Especificação de Segurança WS-Security

Em abril de 2002, a Microsoft Corporation, a IBM Corporation e a VeriSign Inc. uniram-se e publicaram um conjunto de novas especificações de segurança denominadas WS-Security para serviços Web, com o objetivo de que as empresas pudessem criar e construir aplicações de serviços Web com ampla interoperabilidade.

Em março de 2004, uma nova especificação do padrão WS-Security foi publicada, como “Web Services Security: SOAP Message Security 1.0”, pelo grupo OASIS (Organization for the Advancement of Structured Information Standards) que descreve avanços na mensagem SOAP, ao especificar um perfil no uso da Assinatura XML (XML *Signature*) e da Criptografia XML (XML *Encryption*) para garantir integridade e confidencialidade para mensagens SOAP (OASIS, 2005).

WS-Security suporta, integra e unifica vários modelos, mecanismos e tecnologias de segurança em uso no mercado, permitindo que vários sistemas possam interoperar em plataformas e linguagens neutras.

As novas especificações de segurança definem um conjunto de padrões para extensões SOAP ou para cabeçalhos de mensagens, utilizados para oferecer maior integridade, confidencialidade e autenticação das mensagens com o sistema de transmissão de mensagens SOAP.

A autenticação está relacionada à identificação do chamador. O WS-Security usa *tokens*[7] de segurança para manter essas informações com um cabeçalho de segurança da mensagem SOAP. A integridade da mensagem é obtida com assinaturas digitais XML. Isso garante que partes da mensagem não tenham sido adulteradas após a assinatura do originador. A confidencialidade da mensagem é baseada na especificação de criptografia XML e garante que partes correspondentes da mensagem só possam ser compreendidas pelo(s) destinatário(s) desejado(s).

A segurança é oferecida no próprio protocolo de troca de mensagens – SOAP. Com este recurso é possível uma mensagem passar por vários serviços Web ao longo do seu processamento. Suponha que um cliente A faça uma chamada a um serviço Web B, que faz outra chamada a um serviço Web C. Com informações de segurança embutido no protocolo de comunicação é possível atender os requisitos de segurança para este exemplo e muitas aplicações exigem (ROSENBERG, 2004).

O exemplo abaixo mostra uma mensagem que usa segurança baseada em *tokens*, assinaturas digitais e criptografia.

```
(001) <?xml version="1.0" encoding="utf-8"?>
(002)  <S11:Envelope xmlns:S11="..." xmlns:wsse="..." xmlns:wsu="..."
                                    xmlns:xenc="..." xmlns:ds="...">
(003)   <S11:Header>
(004)    <wsse:Security>
(005)     <wsu:Timestamp wsu:Id="T0">
(006)      <wsu:Created>
(007)       2001-09-13T08:42:00Z</wsu:Created>
(008)     </wsu:Timestamp>
(009)
(010)     <wsse:BinarySecurityToken ValueType="...#X509v3"
                wsu:Id="X509Token" EncodingType="...#Base64Binary">
(011)             MIIEZzCCA9CgAwIBAgIQEmtJZc0rqrKh5i...
(012)     </wsse:BinarySecurityToken>
(013)     <xenc:EncryptedKey>
(014)      <xenc:EncryptionMethod Algorithm=
           "http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>
(015)      <ds:KeyInfo>
(016)       <wsse:KeyIdentifier EncodingType="...#Base64Binary"
                        ValueType="...#X509v3">MIGfMa0GCSq...
(017)       </wsse:KeyIdentifier>
(018)      </ds:KeyInfo>
(019)      <xenc:CipherData>
(020)       <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...
(021)       </xenc:CipherValue>
(022)      </xenc:CipherData>
(023)      <xenc:ReferenceList>
(024)       <xenc:DataReference URI="#enc1"/>
(025)      </xenc:ReferenceList>
(026)     </xenc:EncryptedKey>
(027)     <ds:Signature>
(028)      <ds:SignedInfo>
(029)       <ds:CanonicalizationMethod
            Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(030)       <ds:SignatureMethod
```

[7] É um segmento de texto ou símbolos que podem ser manipulados por um *parser* (interpretador de *tokens*).

```
             Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>
(031)        <ds:Reference URI="#T0">
(032)         <ds:Transforms>
(033)          <ds:Transform
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(034)         </ds:Transforms>
(035)         <ds:DigestMethod
               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(036)         <ds:DigestValue>LyLsF094hPi4wPU...
(037)         </ds:DigestValue>
(038)        </ds:Reference>
(039)        <ds:Reference URI="#body">
(040)         <ds:Transforms>
(041)          <ds:Transform
                Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#"/>
(042)         </ds:Transforms>
(043)         <ds:DigestMethod
               Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
(044)         <ds:DigestValue>LyLsF094hPi4wPU...
(045)         </ds:DigestValue>
(046)        </ds:Reference>
(047)       </ds:SignedInfo>
(048)       <ds:SignatureValue>
(049)        Hp1ZkmFZ/2kQLXDJbchm5gK...
(050)       </ds:SignatureValue>
(051)       <ds:KeyInfo>
(052)        <wsse:SecurityTokenReference>
(053)         <wsse:Reference URI="#X509Token"/>
(054)        </wsse:SecurityTokenReference>
(055)       </ds:KeyInfo>
(056)      </ds:Signature>
(057)     </wsse:Security>
(058)    </S11:Header>
(059)    <S11:Body wsu:Id="body">
(060)     <xenc:EncryptedData
           Type="http://www.w3.org/2001/04/xmlenc#Element"
           wsu:Id="enc1">
(061)      <xenc:EncryptionMethod
            Algorithm="http://www.w3.org/2001/04/xmlenc#tripledes-
            cbc"/>
(062)      <xenc:CipherData>
(063)       <xenc:CipherValue>d2FpbmdvbGRfE0lm4byV0...
(064)       </xenc:CipherValue>
(065)      </xenc:CipherData>
(066)     </xenc:EncryptedData>
(067)    </S11:Body>
(068)   </S11:Envelope>
```

Figura 2.7: Mensagem SOAP com WS-Security

Na seqüência estão descritas algumas seções deste exemplo:

As linhas (003)-(058) contêm o cabeçalho da mensagem SOAP.

As linhas (004)-(057) representam o bloco <*wsse:Security*> do cabeçalho. Este contém a segurança da informação para esta mensagem.

As linhas (005)-(008) especificam o momento da criação da mensagem.

As linhas (010)-(012) especificam o *token* de segurança associado à mensagem. Neste caso é especificado um certificado X.509 que está codificado como Base64.

As linhas (013)-(026) especificam a chave usada para criptografar o corpo da mensagem.

As linhas (027)-(056) especificam a assinatura digital. Neste exemplo, a assinatura é baseada num certificado X.509.

A linha (039) referencia o corpo da mensagem.

As linhas (048)-(050) indicam o atual valor da assinatura que foi especificada na linha (043).

As linhas (052)-(054) indicam a chave usada na assinatura.

O corpo da mensagem está representado pelas linhas (059)-(067).

As linhas (060)-(066) representam os metadados da criptografia e formas do corpo usando XML Encryption.

As linhas (063)-(064) contêm o corpo criptografado (IBM, 2005).

2.4.1 XML Digital Signature

O padrão XML Digital Signature especifica uma forma de criar assinaturas digitais para o uso em transações de XML. Este padrão define um esquema para pegar o resultado de uma operação sobre uma assinatura aplicada a dados no formato XML. Da mesma forma que as assinaturas digitais comuns, as assinaturas de documentos XML adicionam importantes características como a autenticação, a integridade dos dados e a sustentação do não-repúdio de dados previamente assinados.

Uma característica fundamental da assinatura de XML é a habilidade de assinar somente partes específicas da árvore de XML ao invés de assinar todo o documento original. Isto será relevante quando um único original de XML pode ter um longo histórico no qual os diferentes componentes são criados em momentos diferentes por pessoas diferentes, cada um assinando somente aqueles elementos relevantes para si. Esta flexibilidade também será crítica em situações nas quais é importante garantir a integridade de certas partes de um documento XML, enquanto deixa para outros a possibilidade de alterar outras partes do documento. Exemplo: um formulário XML assinado entregue a um usuário para ser preenchido. Se a assinatura for aplicada ao formulário XML completo, quaisquer alterações feitas pelo usuário sobre os valores padrão do formulário invalidarão a assinatura original do documento.

Uma assinatura XML pode assinar mais de um tipo de recurso. Por exemplo, uma única assinatura XML pode ser utilizada sobre dados de texto (um documento HTML), dados binários (uma imagem), dados no formato XML, assim como partes específicas de um documento XML.

A validação de assinaturas requer que o objeto assinado esteja disponível. A assinatura, por si só, indicará onde está o objeto original. Esta informação pode:

a) Ser passada através de uma URI com a assinatura XML;

b) Ficar dentro do mesmo recurso que a assinatura XML;

c) Estar embutida dentro da assinatura XML;

d) Ter sua assinatura XML embutida em si mesma (SOUZA, 2005).

A estrutura básica de uma assinatura XML segundo (Rosenberg, 2004) é listada na figura abaixo:

```
<Signature>
  <SignedInfo>
    (CanonicalizationMethod)
    (SignatureMethod)
    (<Reference(URI=)?>
      (Transforms)?
      (DigestMethod)
      (DigestValue)
    </Reference>)+
  </SignedInfo>
  (SignatureValue)
  (KeyInfo)?
  (Object)*
</Signature>
```

Figura 2.8: Sintaxe do elemento <*Signature*> assinatura XML

A informação assinada aparece dentro do elemento <*SignedInfo*>. O algoritmo usado no cálculo do elemento <*SignatureValue*> é mencionado dentro da seção assinada. O elemento <*SignatureMethod*> especifica o algoritmo usado para converter o *SignedInfo* canonizado[8] no *SignatureValue*. Esta é uma combinação de um algoritmo que depende da chave e de um algoritmo de resumo. O elemento <*KeyInfo*> indica a chave que é usada para validar a assinatura, possíveis formas de identificação dos certificados são: nomes de chaves, algoritmos de aceitação de chaves e informação.

Cada recurso deve assinar seu próprio elemento <*Reference*>, identificado pelo atributo URI. O elemento <*Transform*> especifica uma lista ordenada de processos que são aplicados ao conteúdo do recurso especificado antes de ser aplicada a função *hash*[9]. O (*DigestValue*) é o elemento que recebe o resultado da função *hash* aplicada ao recurso.

2.4.1.1 Geração e verificação de uma assinatura XML

Para gerar uma assinatura XML seguem-se os seguintes passos:

a) Determinar o que necessita ser assinado;

- poderão ser assinados documentos XML, elementos XML, imagem JPEG e documentos HTML que são especificados por elementos <*Reference*> da assinatura XML .

b) Cálculo do *hash*;

- em assinaturas de XML, cada recurso referenciado é especificado através de um elemento <*Reference*> e seu *hash* (calculado sobre o recurso identificado e não sobre o elemento <*Reference*> em si) é colocado num elemento <*DigestValue*>. O elemento <*DigestMethod*> identifica o algoritmo usado para calcular o *hash*.

---

[8] Algoritmo de verificação de inconsistência em conteúdos XML antes de extrair a representação em bits para posterior processamento de uma assinatura.

[9] Função matemática que gera um código representativo de uma mensagem.

c) Associação de cada elemento assinado com um elemento;

- todos os elementos <*Reference*> (e seus respectivos *hashes*) devem estar dentro de um elemento <*SignedInfo*>. O elemento <*CanonicalizationMethod*> especifica qual algoritmo foi usado para canonizar o elemento <*SignedInfo*>.

d) Assinatura de tudo;

- o elemento <*SignatureMethod*> identifica o algoritmo usado para produzir o valor da assinatura. Cálculo do *hash* do elemento <*SignedInfo*>, com o valor da assinatura colocado dentro de um elemento <*SignatureValue*>.

e) Inserir informação sobre a chave;

- a informação da chave se necessário pode ser incluída no <*KeyInfo*>.

f) Encapsular tudo num elemento <*Signature*>

- os elementos <*SignedInfo*>, <*SignatureValue*> e <*KeyInfo*> vão dentro do elemento <*Signature*>. O elemento <*Signature*> representa a assinatura XML.

Para verificar uma assinatura XML seguem-se os seguintes passos:

a) Verificar a assinatura do documento contida em <*SignedInfo*>;

- cálculo do *hash* do elemento <*SignedInfo*> (usando o algoritmo de *hash* especificado no elemento <*SignatureMethod*> e a chave pública de verificação) para verificar se o valor do elemento <*SignatureValue*> é o correto em relação ao *hash* do elemento <*SignedInfo*>.

b) Verificação da assinatura de cada elemento <*Reference*>.

- recálculo dos *hashes* dos elementos <*Reference*> contidos dentro do elemento <*SignedInfo*> e comparação destes com os valores dos *hashes* expressos em cada elemento <*DigestValue*> do seu correspondente elemento <*Reference*>. Se valores idênticos os dados não foram alterados, ou seja, estão de acordo com o que foram enviados (ROSENBERG, 2004).

2.4.2 XML Encryption

XML Encryption provê segurança no nível de privacidade por cifrar os dados evitando que terceiros vejam seu conteúdo. A segurança é realizada ponto a ponto para aplicações que requerem trocas seguras de dados estruturados. Criptografia baseada em XML é o caminho natural para segurança com troca de dados entre aplicações complexas.

Transport Layer Security (TSL) é de fato o padrão de comunicação segura na internet e é um protocolo de segurança ponto a ponto que segue o famoso Secure Socket Layer (SSL). XML Encryption não pretende substituir SSL/TSL, mas prover um mecanismo para requisitos de segurança que não é coberto pelo SSL como:

a) Criptografar parte dos dados;

b) Sessão segura entre mais de duas partes.

Com XML Encryption, cada parte pode manter estado de segurança ou não com qualquer das partes comunicantes. Ambos dados seguros ou não podem ser trocados no mesmo documento.

XML Encryption pode manusear dados no formato XML ou não (binário).

A tecnologia de criptografia poderá ser usada para criptografar um documento inteiro ou partes deste. Levando em consideração o consumo de tempo para o processo de criptografia e olhando a perspectiva de performance, é aconselhável que os dados não sejam cifrados a menos que comprometa a segurança (GALBRAITH, 2002).

Expressado numa forma curta, o elemento <*EncryptedData*> tem a seguinte estrutura (onde "?" significa zero ou uma ocorrência; "+" significa uma ou mais ocorrências; "*" significa zero ou muitas ocorrências; e um elemento *tag* vazio (<elemento/>) significa que o elemento será vazio):

```
<EncryptedData Id? Type? MimeType? Encoding?>
    <EncryptionMethod/>?
    <ds:KeyInfo>
      <EncryptedKey>?
      <AgreementMethod>?
      <ds:KeyName>?
      <ds:RetrievalMethod>?
      <ds:*>?
    </ds:KeyInfo>?
    <CipherData>
      <CipherValue>?
      <CipherReference URI?>?
    </CipherData>
    <EncryptionProperties>?
  </EncryptedData>
```

Figura 2.9: Sintaxe do elemento <*EncryptedData*> criptografia XML

Os elementos mais representativos de <*EncryptedData*> são: <*EncryptionMethod*> e <*CipherData*>. <*EncryptionMethod*> aponta para algoritmo utilizado na criptografia. <*CipherData*> contém os dados criptografados ou um ponteiro para as informações criptografadas (ROSEMBERG, 2004).

O processamento XML Encryption envolve dois processos: cifrar e decifrar mensagens. Os processos são descritos abaixo conforme (Rosenberg, 2004).

Passos para criptografar (cifrar):

a) Selecionar o algoritmo de criptografia (3DES, AES entre outros);

b) Escolha da chave criptográfica (opcional);

c) Serialização dos dados da mensagem;

d) Executa a criptografia dos dados;

e) Especificação do tipo de dado;

f) Processamento da correspondente estrutura <*EncryptedData*>.

Passos para decriptografar (decifrar):

a) Determinação do algoritmo, parâmetros e <*KeyInfo*>;

b) Localização da chave;

c) Decriptografa os dados;

d) Processamento dos elementos XML ou seus conteúdos;

e) Processamento dos elementos não XML (tipos não especificados).

Exemplo de um XML sem criptografia dos dados:

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <CreditCard Limit='5,000' Currency='USD'>
    <Number>4019 2445 0277 5567</Number>
    <Issuer>Example Bank</Issuer>
    <Expiration>04/02</Expiration>
  </CreditCard>
</PaymentInfo>
```

Figura 2.10: Mensagem XML sem criptografia

Exemplo da mensagem anterior usando criptografia:

```
<?xml version='1.0'?>
<PaymentInfo xmlns='http://example.org/paymentv2'>
  <Name>John Smith</Name>
  <EncryptedData
Type='http://www.w3.org/2001/04/xmlenc#Element'
   xmlns='http://www.w3.org/2001/04/xmlenc#'>
    <CipherData>
      <CipherValue>A23B45C56</CipherValue>
    </CipherData>
  </EncryptedData>
</PaymentInfo>
```

Figura 2.11: Mensagem XML criptografada

2.4.3 SAML

SAML (Security Assertion Markup Language) é um padrão de segurança para trocas de credenciais de autenticação e autorização baseadas em XML através de diferentes domínios por diferentes aplicações. SAML não impõe uma solução ou infra-estrutura centralizada, descentralizada ou federada, mas facilita a comunicação de autenticação, autorização e a informações de atributos. Não introduz qualquer nova forma ou método de autenticação nem uma alternativa para outros padrões de segurança como a WS-Security e não é limitada para aplicações legadas ou aplicações Web (GALBRAITH, 2002).

Este padrão provê meios para especificação de alegações de segurança, provas de propriedade e provas de possessão. Os *tokens* SAML são assertivas assinadas que fazem alegações baseadas em uma autoridade certificada. Estas assertivas permitem às organizações que troquem informações de segurança de forma neutra e padronizada.

Usado principalmente para inclusão, nas mensagens SOAP, de informações com o propósito de dar subsídios para que o receptor possa decidir sobre a validade das credenciais fornecidas através de algum outro mecanismo. Normalmente as informações SAML são usadas em conjunto com o WS-Security.

SAML suporta a tecnologia *Single Sign-On* (SSO) utilizada na autenticação de acessos a sites, a tecnologia SSO solicita as informações de senhas (*login*) somente uma vez e salva a autenticação para acessos de outros sites que utilizam a mesma autenticação.

SAML possuiu basicamente quatro componentes: assertivas, protocolos, *binding* e *profile*. Assertivas podem ser de 3 tipos: autenticação, atributos e autorização. Assertivas de autenticação – em que a identidade do usuário sempre é verificada. Assertivas de atributos – contêm informações específicas sobre o usuário tal como seus limites de crédito, seus níveis de acesso, seus créditos ou outras declarações de qualificação. Assertivas de autorização – identificam o que o usuário pode ou está autorizado a fazer.

Os protocolos definem como SAML troca mensagens de solicitação e resposta no processamento de assertivas.

Binding – define como mensagens SAML são transportadas em cima dos protocolos de mensagens e transporte padronizados.

Profiles - são regras para fixação, extração e integração. Um *profile* descreve como assertivas SAML são fixados ou combinados com outros objetos e são processados de uma origem Web para uma destinação Web (ROSENBERG, 2004).

3 CRIANDO SERVIÇOS WEB SEGUROS

A partir dos conceitos vistos na seção anterior, esta seção apresenta o que é necessário fazer para gerar serviços Web seguros. Inicialmente, a criação de serviços Web seguros exige uma análise cuidadosa das ameaças e ataques possíveis a partir do instante em que o serviço Web é publicado. Em seguida, para cada ameaça ou ataque conhecido, necessita de um estudo para aplicar mecanismos de segurança com o objetivo de eliminar ou reduzir possíveis pontos de insegurança passíveis de serem explorados pelos invasores.

3.1 Principais Ameaças e Medidas de Segurança

Em virtude da grande quantidade de ameaças e ataques possíveis a um serviço Web, estão destacados na figura abaixo os tipos de vulnerabilidades num contexto da arquitetura de um serviço Web. Dependendo do ponto em que o invasor poderá atacar há um tipo de vulnerabilidade, podendo ser atacado no consumidor do serviço Web, no próprio serviço Web ou mesmo em vários pontos do caminho entre o consumidor e serviço Web.

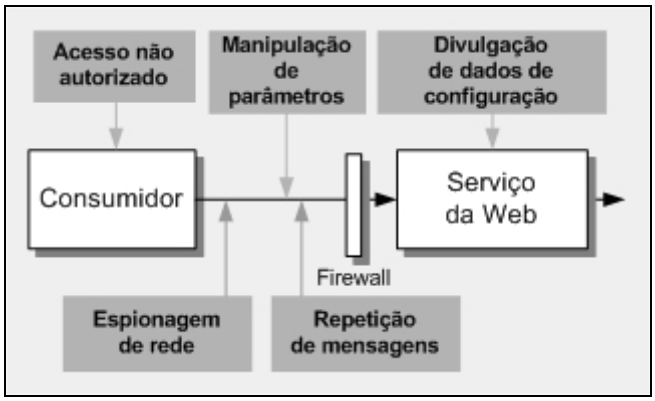


Figura 3.12: Principais ameaças e ataques aos serviços Web

Para cada tipo de ameaça ou ataque deverá haver uma contramedida para resolver a questão da insegurança que serão apresentados na seqüência (MICROSOFT, 2005).

3.1.1 Acesso não autorizado

Os serviços Web que possuem informações confidenciais e restritas devem autenticar e autorizar seus chamadores (usuários dos serviços). A autenticação e a autorização de baixa segurança podem ser exploradas de modo que seja obtido acesso não autorizado a operações e informações confidenciais.

3.1.1.1 Vulnerabilidades

As vulnerabilidades que podem causar o acesso não autorizado por meio de um serviço Web incluem:

a) Falta de autenticação;

b) Senhas de texto sem formatação passaram em cabeçalhos SOAP;

c) Autenticação básica usada por um canal de comunicação sem criptografia.

3.1.1.2 Contramedidas

Para evitar o acesso não autorizado, podem ser usadas as seguintes contramedidas:

a) Usar resenha de senhas em cabeçalhos SOAP para autenticação;

b) Usar permissões Kerberos10 em cabeçalhos SOAP para autenticação;

c) Usar certificados X.509 em cabeçalhos SOAP para autenticação;

d) Usar autorização baseada em função para restringir o acesso aos serviços Web. Isso pode ser feito com o uso da autorização de URL.

3.1.2 Manipulação de parâmetros

A manipulação de parâmetros refere-se à modificação não autorizada de dados enviados entre o consumidor de serviços Web e o serviço Web. Por exemplo, um invasor pode interceptar uma mensagem do serviço Web, talvez ao passar por um nó intermediário rumo ao seu destino. Em seguida, ele pode modificá-la antes de enviá-la ao ponto, de extremidade pretendido.

3.1.2.1 Vulnerabilidades

As vulnerabilidades que podem permitir a manipulação de parâmetros incluem:

a) Mensagens que não são assinadas digitalmente e, portanto, não oferecem proteção contra violação;

b) Mensagens que não são criptografadas e, portanto, não oferecem privacidade e proteção contra violação.

3.1.2.2 Contramedidas

Para evitar a manipulação de parâmetros, podem ser usadas as seguintes contramedidas:

a) Assinar a mensagem digitalmente. A assinatura digital é usada no destinatário para verificar se a mensagem não foi violada enquanto estava em trânsito;

b) Criptografar a carga da mensagem para oferecer privacidade e proteção contra violação.

---

[10] Método de autenticação de rede. Deixa com o usuário um “tiquete” criptografado que pode ser utilizado para requerer qualquer tipo de serviço, sem que a senha do usuário precise passar pela rede.

3.1.3 Espionagem na rede

Com a espionagem na rede, um invasor consegue exibir mensagens do serviço Web à medida que elas passam pela rede. Por exemplo, um invasor pode usar um software de monitoramento de rede para recuperar dados confidenciais contidos em uma mensagem SOAP. Isso pode incluir dados confidenciais no nível do aplicativo ou informações de credenciais.

3.1.3.1 Vulnerabilidades

As vulnerabilidades que podem ativar a espionagem na rede com êxito incluem:

a) Credenciais de texto sem formatação que passaram em cabeçalhos SOAP;

b) Falta de criptografia no nível da mensagem;

c) Falta de criptografia no nível do transporte.

3.1.3.2 Contramedidas

Para proteger mensagens SOAP confidenciais à medida que elas passam pela rede, podem ser usadas as seguintes contramedidas:

a) Usar criptografia no nível do transporte, como SSL ou IPSec. Isso se aplica somente se houver controle em ambos os pontos de extremidade;

b) Criptografar a carga da mensagem para oferecer privacidade. Essa abordagem funciona em situações nas quais a mensagem passa por rota de nós intermediários até o destino final.

3.1.4 Divulgação de dados de configuração

Há dois modos principais pelos quais um serviço Web pode divulgar dados de configuração. Primeiro, o serviço Web pode oferecer suporte à geração dinâmica do WSDL ou conter informações sobre o WSDL que se encontram disponíveis no servidor Web. Isso pode não ser o desejado, dependendo do caso.

Segundo, com exceções inadequadas, o tratamento do serviço Web pode divulgar detalhes confidenciais internos da implementação que sejam úteis para um invasor.

3.1.4.1 Vulnerabilidades

As vulnerabilidades que podem causar a divulgação de dados de configuração incluem:

a) Arquivos WSDL irrestritos, disponíveis para download no servidor Web;

b) Um Web Service irrestrito oferece suporte à geração dinâmica do WSDL e permite que consumidores não autorizados obtenham características do serviço Web;

c) Tratamento de exceções de baixa segurança.

3.1.4.2 Contramedidas

Para evitar a divulgação indesejada de dados de configuração, podem ser usadas as seguintes contramedidas:

a) Autorizar o acesso a arquivos WSDL através de permissões;

b) Remover arquivos WSDL do servidor Web;

c) Desativar os protocolos de documentação para impedir a geração dinâmica do WSDL;

d) Capturar exceções e descartar uma exceção do tipo “SoapException” ou “SoapHeaderException”, que retornam somente informações mínimas e inofensivas, de volta para o cliente.

3.1.5 Repetição de mensagem

As mensagens do serviço Web podem passar por vários servidores intermediários. Com um ataque de repetição de mensagem, um invasor captura e copia uma mensagem e a repete no serviço Web assumindo a identidade do cliente. A mensagem pode ou não ser modificada.

Os tipos mais comuns de ataques de repetição de mensagens incluem:

Ataque de repetição básico **-** o invasor captura e copia uma mensagem e, em seguida, repete a mesma mensagem e assume a identidade do cliente. Esse ataque de repetição não requer que o usuário mal-intencionado conheça o conteúdo da mensagem.

Ataque de interceptadores - o invasor captura a mensagem, altera algum conteúdo, por exemplo, um endereço para remessa e, em seguida, a repete no serviço Web.

3.1.5.1 Vulnerabilidades

As vulnerabilidades que podem ativar a repetição de mensagens incluem:

a) Mensagens não criptografadas;

b) Mensagens que não são assinadas digitalmente e, portanto, não impedem a violação;

c) Mensagens duplicadas não detectadas pela falta de identificação de mensagem exclusiva.

3.1.5.2 Contramedidas

Para solucionar a ameaça de repetição de mensagens, podem ser usadas as seguintes contramedidas:

a) Usar um canal de comunicação com criptografia, por exemplo, o SSL;

b) Criptografar a carga da mensagem para oferecer privacidade de mensagem e proteção contra violação. Embora não impeça os ataques de repetição básicos, isso impede os ataques de interceptadores, nos quais o conteúdo da mensagem é modificado antes de ser repetido;

c) Usar uma identificação de mensagem exclusiva ou um valor de uso único com cada solicitação para detectar duplicatas e assinar digitalmente a mensagem para oferecer proteção contra violação.

Um *valor de uso único* é um valor exclusivo usado criptograficamente para a solicitação. Quando o servidor responde ao cliente, ele envia uma identificação exclusiva e assina a mensagem, inclusive a identificação. Ao fazer outra solicitação, o cliente inclui a identificação com a mensagem. O servidor certifica-se de que a identificação enviada para o cliente na mensagem anterior seja incluída na nova

solicitação do cliente. Se ela for diferente, o servidor rejeitará a solicitação e irá supor que está sujeito a um ataque de repetição.

O invasor não pode falsificar a identidade da mensagem, pois ela está assinada. Observando que isso protege somente o servidor contra ataques de repetição iniciados pelo cliente usando a solicitação de mensagem e não oferece nenhuma proteção ao cliente contra respostas repetidas.

3.2 Análise das Ameaças Versus Mecanismos de Segurança

Na tabela abaixo estão relacionados as principais ameaças e ataques aos serviços Web, o tipo de segurança envolvida e os mecanismos de segurança que procuram combater essas ameaças e ataques sofridos pelos serviços Web.

Tabela 3.2: Ameaças e ataques versus mecanismos de segurança – Serviços Web

Ameaças e Ataques	**Tipo de Segurança**	**Mecanismos de Segurança**
Acesso não autorizado	Autenticação Não repúdio	Autenticação de usuários através de certificados digitais.
Manipulação de parâmetros	Privacidade Integridade	Utilização de criptografia e assinaturas digitais nas mensagens.
Espionagem na rede	Privacidade	Utilização da criptografia nas camadas de transporte e aplicação.
Divulgação dos dados de configuração	Autorização	Autorizar acessos controlados aos arquivos WSDL (configuração) e tomar cuidados com o tratamento de exceções do serviço Web.
Repetição de mensagem	Privacidade Integridade	Utilização de criptografia e assinaturas digitais nas mensagens, além de utilizar uma identificação única de mensagens para evitar duplicação.

4 ESTUDO DE CASO

Com objetivo de demonstrar uma utilização prática da especificação WS-Security e como parte do trabalho desta monografia, implementou-se um protótipo de aplicação e um serviço Web, onde o protótipo de aplicação é o cliente do serviço Web implementado. O fluxo de execução inicia no momento em que o cliente invoca o serviço Web enviando ao mesmo uma mensagem com os parâmetros da quantidade de itens e o preço individual de um determinado produto, o serviço Web, por sua vez, se encarrega de retornar o preço total, multiplicando a quantidade de itens por seu preço unitário. Em toda a implementação realizada neste trabalho são utilizados os principais conceitos relacionados a segurança de serviços Web nas mensagens SOAP apresentados neste trabalho.

4.1 Apresentação

Para implementar o protótipo, utilizou-se a IDE (Integrated Development Enviroment) WebLogic Workshop, uma ferramenta completa com ambiente de desenvolvimento em java, servidor Web compatível com as especificações J2EE e WS-Security, viabilizando a implementação desejada.

Esta ferramenta está disponível[11] com licença *free*[12] para desenvolvimento e criação de protótipos de software com fins não comerciais. Pertence a empresa BEA Systems, empresa de software focada em infra-estrutura para aplicações, fornecedora de soluções abertas e baseadas em padrões de mercado, o que permite que o software do usuário dessas soluções rode na maioria das plataformas de hardware e sistemas operacionais hoje existentes.

Além disso, a ferramenta integra um *framework* de segurança que suporta três categorias de segurança: HTTP Transport Security, segurança na mensagem e segurança baseada em papéis. A figura a seguir ilustra o funcionamento do *framework* de segurança do servidor WebLogic:

---

[11] Disponível em: http://commerce.bea.com/index.jsp - versão 8.1.

[12] Sem restrições de uso.

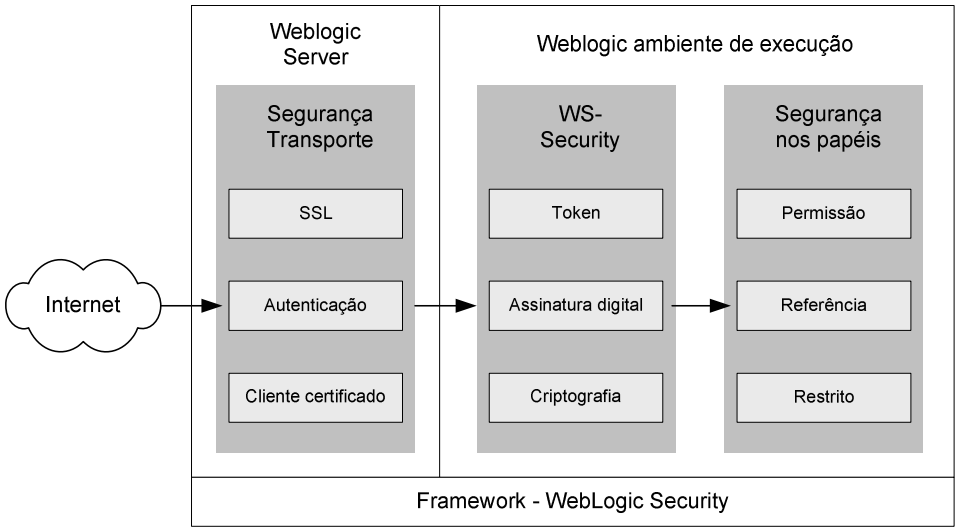


Figura 4.13: Camadas de segurança no ambiente WebLogic

Apesar das três camadas de segurança oferecidas pela ferramenta, as mesmas podem ser implementadas de forma independente uma da outra (ROSENBERG, 2004).

Para focar a segurança na mensagem, o estudo de caso deste trabalho implementa a camada do WS-Security, possibilitando usar três artefatos de segurança: *tokens*, assinaturas digitais e criptografia de documentos XML .

4.2 Metodologia

A ferramenta IDE WebLogic utilizada neste estudo de caso foi instalada em uma máquina com sistema operacional Windows XP, no modo padrão de instalação da ferramenta em questão. Todos os arquivos e configurações gerados para este estudo de caso foram através dessa ferramenta.

A implementação teve como primeiro passo, a criação de um projeto na IDE Weblogic. Em seguida foi codificado um serviço Web no arquivo *Servico.jws*.

Na seqüência gerou-se automaticamente o arquivo *ServicoControl.jcx*, utilizado pela classe cliente para acessar o serviço Web. O cliente por sua vez, foi codificado no arquivo *Cliente.jws*.

Para configurar o servidor Web foram acrescentados alguns parâmetros de autenticação e segurança para o protocolo HTTP nos arquivos *web.xml* e *weblogic.xml*.

Para finalizar a construção de um serviço Web seguro no padrão WS-Security, ainda restavam a criação dos arquivos que policiam as mensagens SOAP de entrada e saída do serviço Web. Para isso, foi necessário criar um arquivo de segurança, chamado *ServicoSeguro.wsse*, com a finalidade de proteção ao serviço Web, este referenciado no código de *Servico.jws*. Para a proteção do lado do cliente foi criado o arquivo *ServicoControlSeguro.wsse*, referenciado no código do arquivo *ServicoControl.*

Com a conclusão da codificação do serviço Web e do cliente, foi necessário cadastrar um usuário no servidor Web. Foram usados como nome “UsuarioValido” e como senha “0123456789”. Os mesmos dados de nome e senha do usuário também foram incluídos nos arquivos com extensão *wsse,* que determinam a segurança do serviço.

Assim, com toda a codificação concluída e usuário cadastrado no servidor, foi executado o código *Cliente.jws*, gerando uma tela que solicita o nome do usuário e senha. Em seguida, submetendo para autenticação do usuário e execução do serviço web, a aplicação gerou uma nova tela com duas mensagens SOAP, uma de envio pelo cliente e outra de retorno do serviço web. As duas telas geradas podem ser vistas no anexo B.

A comprovação de que a transação executou com segurança no padrão WS-Security, foi possível pela utilização de um aplicativo de monitoração da rede (exemplo: tcpmon) na porta 7001 ou 7002, portas em que o servidor WebLogic atende as requisições de aplicações clientes, conforme configuração padrão dos servidores WebLogic. No anexo C, estão listados alguns dados que trafegaram entre o serviço Web e a aplicação cliente, ambos criados neste estudo de caso.

4.3 Implementação

Este estudo de caso[13] procura enfatizar o uso do padrão de segurança WS-Security nos serviços Web. A funcionalidade simplificada implementada no serviço Web deste estudo de caso, poderá ser modificada para um serviço que forneça uma solução com necessidades mais complexas, aproveitando-se a mesma infra-estrutura de implementação usada aqui.

O serviço Web implementado, no arquivo *Servico.jws*, recebe um determinado produto com a quantidade de itens e o preço unitário do item. Retorna ao cliente do serviço, o resultado da multiplicação da quantidade de itens pelo preço unitário.

O cliente do serviço Web, por sua vez implementado no arquivo *Cliente.jws*, envia determinado produto com descrição, quantidade de itens e preço unitário ao serviço Web. Após a invocação do serviço, o cliente recebe o preço total do produto informado.

Além disso, o cliente necessita de um arquivo gerado a partir do serviço Web, que viabilize a utilização do serviço pelo cliente. Este arquivo, chamado *ServicoControl.jcx*, foi gerado de forma automática a partir do arquivo *Servico.jws*. A principal função desse arquivo é fornecer a interface e a descrição WSDL do serviço Web ao cliente.

A comunicação entre o serviço Web e seu cliente utiliza como padrão de segurança o WS-Security. A implementação deste padrão, foi realizada da seguinte forma: na necessidade de criar um arquivo que policia a comunicação tanto do lado do cliente como do lado do serviço, criou-se no cliente o arquivo *ServicoControlSeguro.wsse* e no servidor o arquivo *ServicoSeguro.wsse*. Ambos definem a segurança desejada da mensagem SOAP do serviço Web em estudo.

Os elementos de saída <*wsSecurityOut*> do arquivo de segurança *ServicoControlSeguro.wsse* correspondem aos elementos de entrada <*wsSecurityIn*> do arquivo de segurança *ServicoSeguro.wsse*. Igualmente os elementos de saída <*wsSecurityOut*> deste arquivo correspondem aos elementos de entrada <*wsSecurityIn*> do arquivo *ServicoControlSeguro.wsse*.

Os arquivos *web.xml* e *weblogic.xml* são utilizados pelo servidor para determinar a autenticação do usuário com o servidor Web. Em seguida, a chave e senha do usuário

---

[13] Todos os arquivos gerados para este estudo de caso estão listados no anexo A.

são repassadas para o serviço Web executar seus procedimentos de segurança, conforme configuração do arquivo *ServicoSeguro.wsse.*

Para ilustrar o funcionamento, a figura a seguir mostra a arquitetura da solução:

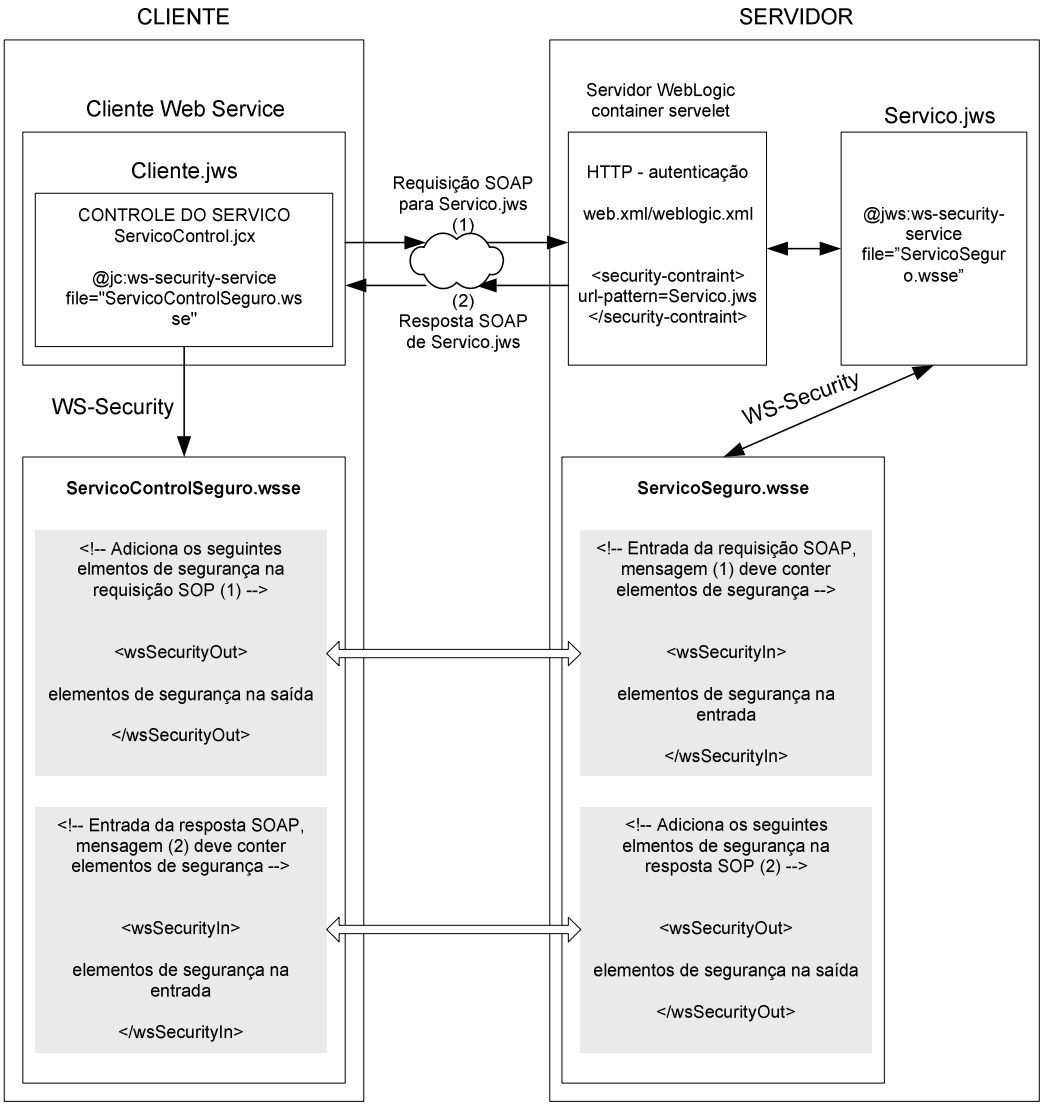


Figura 4.14: Arquitetura de segurança usada neste estudo de caso.

4.4 Avaliação do Estudo de Caso

O SSL permite a autenticação do cliente por meio de uma senha ou certificado digital. No entanto, o SSL não permite o uso de assinaturas digitais para autenticar as mensagens.

O SSL somente protege os dados enquanto estão em trânsito, não fornecendo segurança quando as informações são armazenadas ou repassadas através de outro computador. Esta arquitetura não funciona bem no cenário típico de serviços Web, no qual:

a) Os dados são gerados pelo computador A;

b) A os envia para o computador B;

c) B envia parte dos dados para o computador C;

Isso gera vários problemas:

a) Como C pode ter certeza de que os dados vieram de A?

b) Como A pode evitar que B veja os dados enviados a C?

Esses requisitos de segurança não são atendidos pelo SSL, nem a aplicação de segurança no nível de transporte pode ser empregada para atendê-los. Para fornecer tais funções, é necessário aplicar segurança na camada de mensagem.

A arquitetura da Internet é organizada em camadas. As três camadas de protocolo relevantes para as técnicas de segurança da informação descritas neste trabalho são: camada de rede, camada de transporte e com maior detalhamento a camada de aplicação.

Há vantagens e desvantagens na aplicação de segurança em cada camada. Em geral, a aplicação de segurança nas camadas superiores de protocolo permite que mais funções de segurança sejam executadas. A aplicação de segurança nas camadas inferiores de protocolo tem menor impacto sobre as aplicações. As comunicações podem ser roteadas através de uma Rede Privada Virtual IPSEC, sem modificar uma única linha de código em qualquer das aplicações. A segurança pode ser obtida dessa forma, porém, ela será limitada a decidir quem pode conectar-se à rede e quem terá o acesso negado. Estas vantagens e desvantagens são apresentadas na figura a seguir.

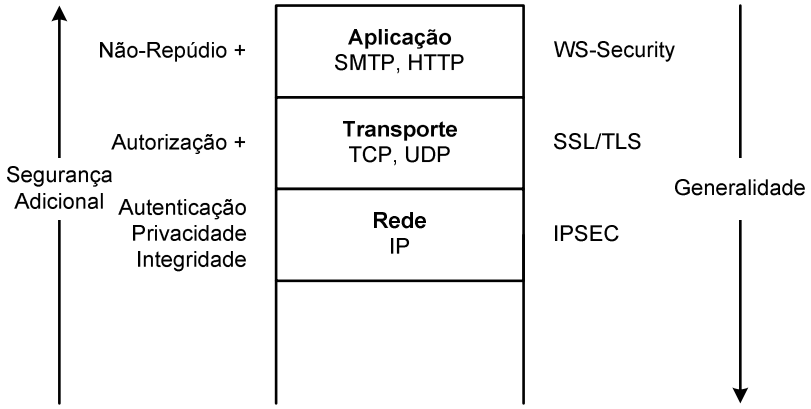


Figura 4.15: Vantagens e desvantagens da segurança aplicada em diferentes camadas

O desafio é implementar a maior parte da segurança necessária nas camadas superiores de protocolo, com o mínimo de impacto sobre as aplicações. A tabela a seguir apresenta uma avaliação demonstrativa das principais características dos padrões de segurança vistos neste trabalho e as respectivas ameaças combatidas por cada padrão, conforme avaliações do autor desta monografia.

Tabela 4.3: Avaliação demonstrativa - padrões de segurança para serviços Web

Padrões de Segurança	**Objetivo**	**Segurança**	**Nível**	**Características**	**Ameaças Combatidas**
IPSec – Rede Virtual (VPN)	Garantir a privacidade e autenticação através de uma VPN.	Autenticação Privacidade Integridade	Rede	Tempo de transmissão considerável.	Qualquer ameaça vinda de fora da rede virtual.
SSL/TLS	Transporte de mensagens com criptografia SSL	Autenticação Privacidade Integridade Autorização	Transporte	Baixo desempenho; Segurança fim-a-fim serviços Web não disponível.	Todas, porém ponto-a-ponto.
XML Digital Signature	Protocolo para assinaturas digitais	Autenticação Integridade Não repúdio	Aplicação	Assinatura digital de documentos todo / parte.	Acesso não autorizado.
XML Encryption	Protocolo para criptografia	Privacidade Integridade	Aplicação	Criptografia de documentos todo / parte.	Manipulação de parâmetros Espionagem na rede Repetição de mensagem.
SAML	Possibilitar trocas de credenciais de autenticação e autorização	Autenticação Autorização	Aplicação	Uso do SSO, onde diversas aplicações compartilham o mesmo login	Acesso não autorizado Divulgação dos dados de configuração.
WS-Security	Segurança nas mensagens SOAP	Autenticação Integridade Privacidade Não repúdio	Aplicação	Padronização da OASIS	Acesso não autorizado Manipulação de parâmetros Espionagem na rede Repetição de mensagem.

5 CONCLUSÃO

A importância crescente dos serviços Web como forma de integração entre organizações, aplicações e processos de negócio torna os aspectos relacionados à segurança destes serviços foco de atenção de seus usuários. Com o padrão SOAP e o uso do HTTP como protocolo de transporte, houve a quebra de alguns paradigmas de segurança importantes em ambientes corporativos, com a possibilidade de tratar a segurança na própria mensagem, permitindo o compartilhamento de informações de segurança entre intermediários, a análise do conteúdo das mensagens trocadas e a identificação do usuário responsável por cada uma delas.

A utilização de determinadas estratégias de segurança, depende do nível de segurança desejado ou característica da aplicação. No entanto, com a evolução das aplicações distribuídas, com o uso da Internet e padrões abertos, determinadas estratégias, principalmente no nível de rede e transporte não são mais suficientes, necessitando das estratégias de segurança no nível de aplicação. Os principais padrões de segurança neste nível fazem parte da especificação WS-Security, que aos poucos as ferramentas de desenvolvimento de serviços Web estão incorporando nos seus ambientes.

No estudo de caso foi usada a ferramenta WebLogic Workshop 8.1, que implementa a especificação J2EE e os principais padrões de segurança abordados neste trabalho. O protótipo criado possibilitou demonstrar a aplicação prática destes padrões e fazer uma avaliação, através da execução e observação das mensagens trocadas entre o serviço Web e a aplicação cliente, ambos implementados no estudo de caso deste trabalho.

A tecnologia de serviços Web, por sua infra-estrutura pública, sujeita a ataques, questões de segurança na Internet são importantes. Contudo, já existem padrões e produtos promissores para a segurança de serviços Web como visto neste trabalho.

Para trabalhos futuros, o que não foi abordado neste trabalho, é importante destacar a necessidade de estudar os ataques através de falhas em aplicações ao invés de falhas em infra-estrutura, exigindo que a segurança seja encarada como um processo de mitigar riscos, não apenas com o uso de ferramentas ou abordagens técnicas, mas também, na observação do comportamento das aplicações em situações de exceção.

Outra sugestão para trabalhos futuros é comparar o desempenho das ferramentas de desenvolvimento que implementam *frameworks* de segurança WS-Security e que permitem desenvolver serviços Web seguros. Além disso, é importante realizar estudos comparativos que apresentem o comportamento das tecnologias J2EE e .NET no uso dos padrões de segurança para os serviços Web, uma vez que a principal característica desses serviços é promover a interoperabilidade entre aplicações, independente das plataformas de hardware e software adotados.

REFERÊNCIAS

BOND, Martin et al. **Aprenda J2EE em 21 dias**. São Paulo: Pearson Education do Brasil, 2003.

CHAPPELL, David; JEWEL, Tyler. **Java Web Services**. [S.I.]: O'Reilly, 2002.

CLEMENTS, Tom. **Overview of Soap**. SUN. Disponível em: <http://developer.java.sun.com/developer/technicalArticles/xml/webservices/>. Acesso em: set. 2005.

COSTA, L.A.G. Segurança e Web Services em Java. **Revista MundoJava**, Rio de Janeiro, ano 2, n.9, p.50-54, 2005.

GALBRAITH, Ben et al. **Professional Web Services Security**. Birmingham: Wrox Press, 2002.

HENDRICKS, Mack et al. **Profissional Java Web Services**. Rio de Janeiro: Alta Books, 2002.

IBM. **Web Services Security (WS-Security)**. Disponível em: <http://www-128.ibm.com/developerworks/webservices/library/ws-secure/index.html>. Acesso em: ago. 2005.

IEC- International Engineering Consortium. **Tutorial Virtual Private Networks**. Disponível em: <http://www.iec.org/online/tutorials/vpn/index.html>. Acesso em: set. 2005.

MAHMOUND, Qusay H. **Securing Web Services and the Java WSDP 1.5 XWS-Security Framework**. SUN. Março 2005. Disponível em: <http://java.sun.com/developer/technicalArticles /WebServices/security/>. Acesso em: ago. 2005.

MICROSOFT. **Módulo 10 – Segurança de Web Services**. Centro de Orientações de Segurança. Disponível em: <http://www.microsoft.com/brasil/security/guidance/topics/devsec/secmod10.mspx>. Acesso em: set. 2005.

OASIS. **Web Services Security: Soap Message Security 1.0**. Disponível em: <http://www.oasis-open.org/committees/download.php/5531/oasis-200401-wss-soap-message-security-1.0.pdf>. Acesso em: jun. 2005.

PAMPLONA, Vitor Fernando. **Web Services. Construindo, disponibilizando e acessando Web Services via J2SE.** Disponível em: <http://www.javafree.org/content/view.jf?idContent=4>. Acesso em: set. 2005.

ROSENBERG, Jothy.; REMY, David. **Securing Web Services with WS-Security.** USA. Sams Publishing, 2004.

SOUZA, Gustavo de. **Assinatura Digital de Documentos XML.** Disponível em:<http://www.inf.ufsc.br/%7Edesouza/seguranca.htm>. Acesso em: set. 2005.

STALLINGS, Willian. **Network Security Essentials: applications and standards**. Upper Saddle River, NJ: Prentice-Hall, 2000.

VIEIRA, Pedro da Fonseca. **Virtual Private Networks**. Disponível em: <http://www.gta.ufrj.br/grad/00_1/pedro/vpns.htm> . Acesso em: set. 2005.

W3C. **Web Services Architecture.** W3C Working Group Note 11 February 2004. Disponível em: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/> Acesso em: jul. 2005.

_____. **Extensible Markup Language (XML) 1.0.** W3C Recommendation 04 February 2004. Disponível em: <http://www.w3.org/TR/2004/REC-xml-20040204/> Acesso em: out. 2005.

WHALI, U. et al. **WebSphere Version 6 Web Services Handbook Development and Deployment.** IBM RedBooks. Disponível em: <http://www.redbooks.ibm.com/redbooks/SG246461/wwhelp/wwhimpl/java/html/wwhelp.htm>. Acesso em: set. 2005.

ANEXO A ARQUIVOS FONTES

Arquivo *Cliente.jws* - este arquivo faz parte da aplicação cliente que invoca o serviço Web do estudo de caso, o arquivo nada mais é do que uma classe java que contém basicamente uma referência à interface pública do serviço Web (`servicoControl`) e o método *submitPO*. Com referência ao serviço Web, a classe *Cliente* cria um objeto *poBean*, que faz parte do serviço Web, e povoa os atributos do objeto *poBean* e em seguida invoca o serviço Web. Toda essa implementação descrita é realizada dentro do método *submitPO* da classe *Cliente*.

```
package pacoteSeguro;

import pacoteSeguro.ServicoControl.Bean;

//classe cliente do servico web
public class Cliente implements com.bea.jws.WebService
{
    //referencia para o servico web
    /**
     * @common:control
     */
    private pacoteSeguro.ServicoControl servicoControl;

    static final long serialVersionUID = 1L;

    //método para submeter nome e senha do usuário ao servidor
    /**
     * @common:operation
     */
    public double submitPO(String nomeUsuario, String senha)
    {
        //objeto que será enviado ao servidor
        Bean poBean = new Bean();
        poBean.itemNome = "Produto de teste";
        poBean.itemQuantidade = 2;
        poBean.itemPreco = 100.00;

        //código para especificar o nome e senha do usuario
        servicoControl.setUsername(nomeUsuario);
        servicoControl.setPassword(senha);

        return servicoControl.submitPO(poBean);
    }
}
```

Arquivo *Servico.jws* – este arquivo representa a classe java *Servico* que é serviço Web propriamente dito. A classe *Servico* possui uma classe interna chamada *Bean*, classe esta vista pela classe *Cliente*. Além disso, a classe *Servico* implementa o método

submitPO publicado como interface do serviço Web, através deste método o serviço Web executa sua funcionalidade e retorna o resultado para quem o invocou.

```
package pacoteSeguro;

//aponta para o arquivo de seguranca - padrao WS-Security
/**
 * @jws:ws-security-service file="ServicoSeguro.wsse"
 */
```

```
//classe que implementa o servico web
public class Servico implements com.bea.jws.WebService
{
    static final long serialVersionUID = 1L;

    //classe dados recebidos do cliente
    public static class Bean
    {
        //construtor que recebe os dados do cliente
        public Bean(String itemNome, int itemQuantidade, double itemPreco)
        {
            this.itemNome = itemNome;
            this.itemQuantidade = itemQuantidade;
            this.itemPreco = itemPreco;
        }

        public Bean(){};

        //atributos de dados do servico web
        public String itemNome;
        public int itemQuantidade;
        public double itemPreco;
    }

    /**
     * metodo que vai ser exportado para ser chamado pelo cliente
     * @common:operation
     */
    public double submitPO(Bean poBean)
    {
        //codigo que retorna o valor total
        return poBean.itemQuantidade * poBean.itemPreco;
    }
}
```

Arquivo *ServicoControl.jcx* – este arquivo é gerado após a construção da classe *Servico,* que representa o serviço Web. Este arquivo contém a interface do serviço Web vista pela classe *Cliente*, a URL do serviço Web e toda a descrição na linguagem WSDL do serviço Web.

```
package pacoteSeguro;

//3 linhas abaixo: localizacao do servico web, arquivo WSDL que descreve o
servico e
// o arquivo de seguranca padrao WS-Security
/**
 * @jc:location http-
url="https://localhost:7002/SeguroAppWeb/pacoteSeguro/Servico.jws" jms-
url="Servico.jws"
 * @jc:wsdl file="#ServicoWsdl"
 * @jc:ws-security-service file="ServicoControlSeguro.wsse"
 */
```

```
//classe que exporta a interface do servico web.
public interface ServicoControl extends com.bea.control.ControlExtension,
com.bea.control.ServiceControl
{
    public static class Bean
        implements java.io.Serializable
    {
        public java.lang.String itemNome;
        public int itemQuantidade;
        public double itemPreco;
    }
    /**
     * metodo que vai ser exportado para ser chamado pelo cliente
     * @jc:protocol form-post="false" form-get="false"
     */
    //metodo chamado pelo cliente e invocando o servico web.
    public double submitPO (Bean poBean);

    static final long serialVersionUID = 1L;
}

//definicao em WSDL do servico WEB.
/** @common:define name="ServicoWsdl" value::
    <?xml version="1.0" encoding="utf-8"?>
    <definitions xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:conv="http://www.openuri.org/2002/04/soap/conversation/"
xmlns:cw="http://www.openuri.org/2002/04/wsdl/conversation/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:jms="http://www.openuri.org/2002/04/wsdl/jms/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:s="http://www.w3.org/2001/XMLSchema" xmlns:s0="http://www.openuri.org/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
targetNamespace="http://www.openuri.org/">
      <types>
        <s:schema elementFormDefault="qualified"
targetNamespace="http://www.openuri.org/"
xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:ope="http://www.openuri.org/">
          <s:element name="submitPO">
            <s:complexType>
              <s:sequence>
                <s:element name="poBean" type="ope:Bean" minOccurs="0"/>
              </s:sequence>
            </s:complexType>
          </s:element>
          <s:element name="submitPOResponse">
            <s:complexType>
              <s:sequence>
                <s:element name="submitPOResult" type="s:double"/>
              </s:sequence>
            </s:complexType>
          </s:element>
          <s:complexType name="Bean">
            <s:sequence>
              <s:element name="itemNome" type="s:string" minOccurs="0"/>
              <s:element name="itemQuantidade" type="s:int"/>
              <s:element name="itemPreco" type="s:double"/>
            </s:sequence>
          </s:complexType>
        </s:schema>

      </types>
      <message name="submitPOSoapIn">
        <part name="parameters" element="s0:submitPO"/>
      </message>
      <message name="submitPOSoapOut">
```

```
        <part name="parameters" element="s0:submitPOResponse"/>
      </message>
      <portType name="ServicoSoap">
        <operation name="submitPO">
          <documentation>metodo que vai ser exportado para ser chamado pelo
cliente</documentation>
          <input message="s0:submitPOSoapIn"/>
          <output message="s0:submitPOSoapOut"/>
        </operation>
      </portType>
      <binding name="ServicoSoap" type="s0:ServicoSoap">
        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
style="document"/>
        <operation name="submitPO">
          <soap:operation soapAction="http://www.openuri.org/submitPO"
style="document"/>
          <input>
            <soap:body use="literal"/>
          </input>
          <output>
            <soap:body use="literal"/>
          </output>
        </operation>
      </binding>
      <service name="Servico">
        <port name="ServicoSoap" binding="s0:ServicoSoap">
          <soap:address
location="http://localhost:7001/pacoteSeguro/Servico.jws"/>
        </port>
      </service>
    </definitions>
 * ::
 */
```

Arquivo *ServicoSeguro*.wsse- este arquivo e gerado a partir da classe *Cliente* e algumas tags XML no padrão WS-Security são incluídas para definir o tipo de segurança que será adotado na comunicação entre a aplicação cliente e o serviço Web.

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
    xmlns="http://www.bea.com/2003/03/wsse/config"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <!-- Especifica os elementos de segurança na mensagem SOAP de entrada no
    serviço web-->
    <wsSecurityIn>
        <!-- Mensagem SOAP deve acompanhar um username e senha válidos-->
        <token tokenType="username"/>

        <!-- Encripta a mensagem SOAP com a chave pública de Servico.jws.
        O alias e password são usados para acessar a chave privada de
Servico.jws
        em Keystore, são elementos de decryptionKey-->
        <encryptionRequired>
            <decryptionKey>
                <alias>mycompany</alias>
                <password>password</password>
            </decryptionKey>
        </encryptionRequired>

        <!-- Na mensagem SOAP da origem, a chave privada deve ser assinada
digitalmente.
        A chave pblica da origem é usada pelo destinatário para validar a
assinatura-->
```

```
        <signatureRequired>true</signatureRequired>
    </wsSecurityIn>

    <!-- Especifica a segurança da mensagem SOAP de saída do serviço web. -->
    <wsSecurityOut>

        <!-- Nome e senha do usuário deve acompanhar a mensagem SOAP-->
        <userNameToken>
            <userName>UsuarioValido</userName>
            <password type="TEXT">0123456789</password>
        </userNameToken>

        <!-- Mensagem SOAP criptografada com a chave pública do receptor.
Somente
        o receptor com a chave privada poderá decriptograr a mensagem.
        Com isso, garante a confidencialidade da mensagem SOAP -->
        <encryption>
            <encryptionKey>
                <alias>client1</alias>
            </encryptionKey>
        </encryption>

        <!-- Mensagem SOAP assinada digitalmente pelo enviador com a chave
privada.
        Somente com a chave pública do enviador, validará a assinatura.
        Com isso, garante a autenticidade da origem da mensagem.-->
        <signatureKey>
            <alias>mycompany</alias>
            <password>password</password>
        </signatureKey>
    </wsSecurityOut>

    <!-- Localização do arquivo de chaves e certificados digitais usados para
    criptograr, decriptograr, assinar e validar mensagens-->
    <keyStore>
        <keyStoreLocation>samples_mycompany.jks</keyStoreLocation>
        <keyStorePassword>password</keyStorePassword>
    </keyStore>

</wsSecurityPolicy>
```

Arquivo *ServicoControlSeguro*.wsse – este arquivo é gerado a partir do arquivo de interface do serviço Web, são adicionados algumas tags do padrão WS-Security para definir o tipo de segurança adotado pelo serviço Web.

```
<?xml version="1.0" ?>
<wsSecurityPolicy xsi:schemaLocation="WSSecurity-policy.xsd"
    xmlns="http://www.bea.com/2003/03/wsse/config"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

       <!-- Especifica a segurança da mensagem SOAP de saída do cliente. -->
       <wsSecurityOut>

       <!-- Nome e senha do usuário deve acompanhar a mensagem SOAP-->
        <userNameToken>
            <userName>UsuarioValido</userName>
            <password type="TEXT">0123456789</password>
</userNameToken>

        <!-- Mensagem SOAP criptografada com a chave pública do receptor.
Somente
        o receptor com a chave privada poderá decriptograr a mensagem.
        Com isso, garante a confidencialidade da mensagem SOAP -->
        <encryption>
```

```
            <encryptionKey>
                <alias>mycompany</alias>
            </encryptionKey>
        </encryption>

        <!-- Mensagem SOAP assinada digitalmente pelo enviador com a chave
privada.
        Somente com a chave pública do enviador, validará a assinatura.
        Com isso, garante a autenticidade da origem da mensagem.-->
        <signatureKey>
            <alias>client1</alias>
            <password>password</password>
        </signatureKey>
</wsSecurityOut>

     <!-- Especifica os elementos de segurança na mensagem SOAP que retorna do
     serviço web-->
     <wsSecurityIn>

            <!-- Mensagem SOAP deve acompanhar um username e senha válidos-->
            <token tokenType="username"/>

            <!-- Encripta a mensagem SOAP com a chave pública de Servico.jws.
           O alias e password são usados para acessar a chave privada de
Servico.jws
           em Keystore, são elementos de decryptionKey-->
            <encryptionRequired>
                <decryptionKey>
                    <alias>client1</alias>
                    <password>password</password>
                </decryptionKey>
            </encryptionRequired>

            <!-- Na mensagem SOAP da origem, a chave privada deve ser assinada
digitalmente.
        A chave pblica da origem é usada pelo destinatário para validar a
assinatura-->
            <signatureRequired>true</signatureRequired>
    </wsSecurityIn>

    <!-- Localização do arquivo de chaves e certificados digitais usados para
    criptograr, decriptograr, assinar e validar mensagens-->
    <keyStore>
            <keyStoreLocation>samples_client.jks</keyStoreLocation>
            <keyStorePassword>password</keyStorePassword>
    </keyStore>

</wsSecurityPolicy>
```

Arquivo *web.xml* – arquivo de configuração do servidor Web do ambiente WebLogic. Neste arquivo são configuradas as propriedades de segurança usados na autenticação do HTTP e HTTPS.

```
...
<!-- caracteristicas de seguranca usados na autenticação HTTP e HTTPS -->
    <security-constraint>
        <display-name>Security Constraints</display-name>
        <web-resource-collection>
            <web-resource-name>Secure Resources </web-resource-name>
            <url-pattern>/pacoteSeguro/Servico.jws </url-pattern>
            <http-method>GET</http-method>
            <http-method>POST</http-method>
        </web-resource-collection>
        <auth-constraint>
```

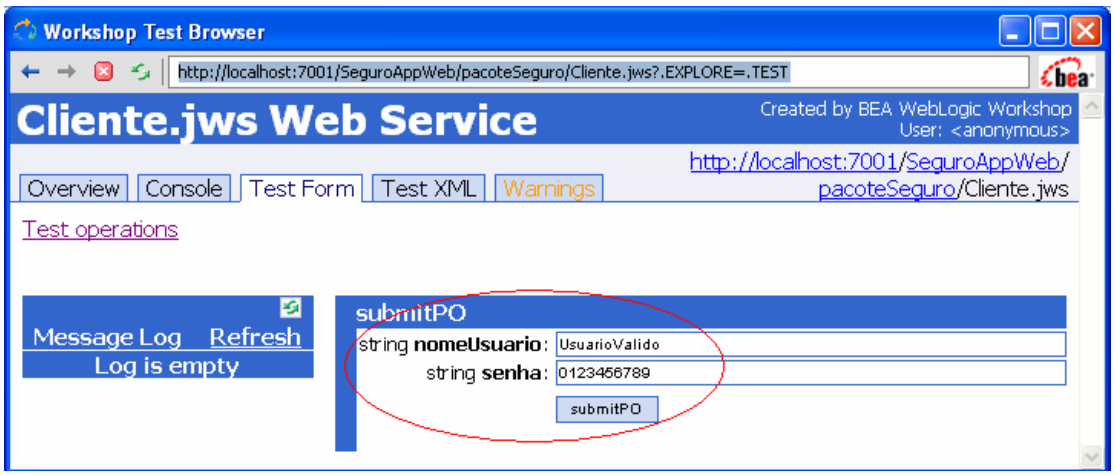
```
            <role-name>GoodRole</role-name>
        </auth-constraint>
        <user-data-constraint>
            <transport-guarantee>CONFIDENTIAL</transport-guarantee>
        </user-data-constraint>
    </security-constraint>

    <security-role>
        <description>Role description</description>
        <role-name>GoodRole</role-name>
    </security-role>
   <!-- fim seguranca ao nivel de transporte de HTTP E HTTPS -->
...
```

Arquivo *weblogic.xml* – também faz parte da configuração do ambiente de execução do servidor WebLogic. Basicamente define a segurança na camada de transporte através de papéis do usuário, isto é, controla os acessos de determinado usuário.

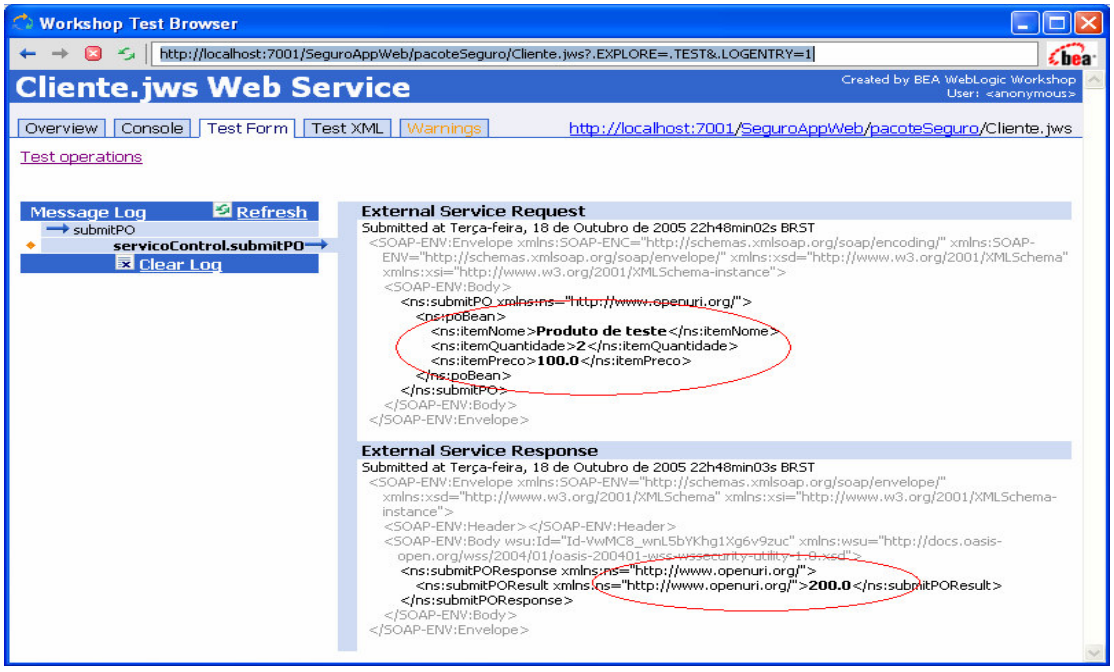
```
...
<!--seguranca camada de transporte -->
    <security-role-assignment>
        <role-name>GoodRole</role-name>
        <principal-name>UsuarioValido</principal-name>
    </security-role-assignment>
...
```

ANEXO B TELAS

Formulário de autenticação (tela de *debug* da ferramenta WebLogic) – tela que solicita o nome e senha do usuário e submete ao servidor.



Resultado da execução – após submeter o nome e senha do usuário ao servidor Web, visto na figura acima, o servidor Web retorna no padrão SOAP o resultado da execução do serviço Web implementado no estudo de caso.



ANEXO C DADOS NO PADRÃO WS-SECURITY

A listagem abaixo é uma mensagem SOAP usando o padrão WS-Security, capturada em tempo de execução da aplicação cliente e o serviço Web. Especificamente é uma mensagem de requisição do serviço Web.


```
<SOAP-ENV:Envelope  xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"                    xmlns:SOAP-
ENV="http://schemas.xmlsoap.org/soap/envelope/">

<SOAP-ENV:Header>

<wsse:Security       xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-secext-1.0.xsd" SOAP-ENV:mustUnderstand="1">

<xenc:EncryptedKey xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">

<xenc:EncryptionMethod Algorithm="http://www.w3.org/2001/04/xmlenc#rsa-1_5"/>

<dsig:KeyInfo xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">

<dsig:KeyName>CN=MyCompany,  OU=Development,  O=MyDevTeam,  L=Sealand,  ST=WA,
C=US</dsig:KeyName>

</dsig:KeyInfo>

<xenc:CipherData>

<xenc:CipherValue>NPstojvmM+RE9OtVaVaxMUn3HAR5Qm9Wz6NI7q+XPNfZUc/55/bQM58coFNa
d6PtF9voIS06dAwqvPOg9hwSyT9SfKtMgA4/WcFbo73SVz9ITaPCAQ9dn8USkz7s0oza3NyIlC6VdI
mBkGu87dbmLA2AKvfob7s2lnDTxSrc1eg=</xenc:CipherValue>

</xenc:CipherData>

<xenc:ReferenceList>

<xenc:DataReference URI="#Id-6F7LceaMAxS3wyRRENVS3dit"/>

</xenc:ReferenceList></xenc:EncryptedKey>

<wsse:BinarySecurityToken                       xmlns:wsu="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd"
ValueType="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-x509-token-
profile-1.0#X509v3"                          EncodingType="http://docs.oasis-
open.org/wss/2004/01/oasis-200401-wss-soap-message-security-1.0#Base64Binary"
wsu:Id="Id-
KODO_eyXF4HcN77_hWnLtnXC">MIICRTCCAa6gAwIBAAIEPrBeaTANBgkqhkiG9w0BAQUFADBnMQsw
CQYDVQQGEwJVUzELMAkGA1UECBMCTlkxFjAUBgNVBAcTDU5ldyBZb3JrIENpdHkxDDAKBgNVBAoTA2
9yZzETMBEGA1UECxMKY2xpZW50MU9yZzEQMA4GA1UEAxMHY2xpZW50MTAeFw0wMzA0MzAyMzM4MTda
Fw0wNDA0MjkyMzM4MTdaMGcxCzAJBgNVBAYTAlVTMQswCQYDVQQIEwJOWTEWMBQGA1UEBxMNTmV3IF
lvcmsgQ2l0eTEMMAoGA1UEChMDb3JnMRMwEQYDVQQLEwpjbGllbnQxT3JnMRAwDgYDVQQDEwdjbGll
bnQxMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQDF/Q/4VGVOb0fdrXELYh1JzKC76eICnJLrCC
h6nBfpKjZUBBiDlLhphB52arGonEUIBHHO9n68N1hoN/uz5j6H5/KmLRdcA1huAIlcNoWmxC61XjCx
EDT+agvrg2D6suyzElusWCrvpIEsWEtCcCD0x/MOVcQLK3q9oMg4ihj4ewIDAQABMA0GCSqGSIb3DQ
EBBQUAA4GBAKgcU99Prrz37UgiTp5NTX4oLDPM+HBmETQB9EnQPDPZ829tsHsPymM42Pe2Qk4TNM/+
ZIdbrFRSft64WWHYjr8K8uBR9F7/a1WyJmiNPE3wkiZlM140HjV8l0fAfwR2d+cdB0RvJpwLx/onTx
FcnMlCzJfUUp5mFHzebkw19/WD</wsse:BinarySecurityToken>

<dsig:Signature xmlns:dsig="http://www.w3.org/2000/09/xmldsig#">
```

```
<dsig:SignedInfo>

<dsig:CanonicalizationMethod      Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/>

<dsig:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"/>

<dsig:Reference URI="#Id-FRvBBqzkzp27T8MlOBAwunHg">

<dsig:Transforms>

<dsig:Transform                    Algorithm="http://www.w3.org/2001/10/xml-exc-
c14n#"/></dsig:Transforms>

<dsig:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>

<dsig:DigestValue>kTKQ8N+fLAMnqQFNzDp60alJjLQ=</dsig:DigestValue>

</dsig:Reference>

</dsig:SignedInfo>

<dsig:SignatureValue>E/W8kk2f4s/lBqZYxejRxN9xeiKwE1bE7oqFCJZ9miBUa06mva59h3spg
LLMrBQm+FccT/pj7Bf6ZoIfP8j4CCv2zNcyBm8IHaWg7yP+diVZtUczrK9mcuf6RQsmxtwo1S3HgYC
OEATQ3/5AnsJvVp+GWnmjufOkGYf7Ee1xN/s=</dsig:SignatureValue>

<dsig:KeyInfo>

<wsse:SecurityTokenReference>

<wsse:Reference URI="#Id-KODO_eyXF4HcN77_hWnLtnXC"/>

</wsse:SecurityTokenReference>

</dsig:KeyInfo>

</dsig:Signature>

wsse:UsernameToken     xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-
200401-wss-wssecurity-utility-1.0.xsd" wsu:Id="Id-VHr3d0FmPfnDjtLYTDaICdkZ">

<wsse:Username>UsuarioValido</wsse:Username>

<wsse:Password   Type="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
username-token-profile-1.0#PasswordText">0123456789</wsse:Password>

</wsse:UsernameToken>

</wsse:Security>

</SOAP-ENV:Header>

<SOAP-ENV:Body  xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-
wss-wssecurity-utility-1.0.xsd" wsu:Id="Id-FRvBBqzkzp27T8MlOBAwunHg">

<xenc:EncryptedData    xmlns:xenc="http://www.w3.org/2001/04/xmlenc#"    Id="Id-
6F7LceaMAxS3wyRRENVS3dit" Type="http://www.w3.org/2001/04/xmlenc#Element">

<xenc:EncryptionMethod      Algorithm="http://www.w3.org/2001/04/xmlenc#aes128-
cbc"/>

<xenc:CipherData>

<xenc:CipherValue>6V1NCg8jq7U6tq5JPLYvbfjXi7f3sk64RzT6GCkxNeAcM5uUNMvym0ZMKVey
lu1MWLbFA5MN50NkwnOa11olysZPHHnNuei6h6foQ9c24BQVLV8mJWjc0HiMrR80hJ6TCVtMEE54Np
1QmPOF/ydhPA7QTweiFUvbImp1Hwj0OhYwU4jn7BZTywp+yUAVJ8ODhpeaDPeMXrLI/wTFgnBbfmVq
lLvLTngft/IHbKP69r8IFrEUr+ZZSdl0BWj8Hr8x09jiFAgkxw73ZbYay57wMFeM94Xj8FTXYwQSgA
KSMNiWAWxNKmGBfA38InrybSbgtWWEX4EKFHMs890lmudoNwW1JdekDNLuFmE3AVOfWHqXurplKlCa
xfynzhL2WixTk7/bAJfa6ayA+7OuRTXg6V+tAthaDI2AfRxYEmM3lCn46xeDZAV+W3N3RF8TDtvhId
SSU/UXoamQYpJMxh6xGA==</xenc:CipherValue>

</xenc:CipherData>

</xenc:EncryptedData>

</SOAP-ENV:Body>

</SOAP-ENV:Envelope>
```