

ARQUITETURA DE SEGURANÇA EM APLICAÇÕES BASEADAS EM WEB SERVICES

Reginaldo F. Silva¹ e José A. Cunha²

¹Gerência de Tecnologia da Informação – CEFET-RN.

naldo_ds@yahoo.com.br

²Professor da Gerência de Tecnologia da Informação – CEFET-RN.

jcunha@cefetrn.br

Recebido em outubro de 2005 e aceito em dezembro de 2005

RESUMO

Este artigo aborda conceitos e aspectos de segurança relativos aos Web Services, uma nova arquitetura de sistemas que permite que programas ofereçam suas funcionalidades a outros programas através da Internet. O fato de todas as suas operações estarem baseadas na Internet torna os Web Services muito sensíveis quanto à segurança: agentes não-autorizados podem interceptar e modificar dados em tráfego pela rede, interromper o funcionamento de serviços, utilizar recursos sem permissão. Por esse motivo, conceitos, mecanismos e padrões de implementação de segurança devem ser aplicados, para que a arquitetura de Web Services seja considerada confiável no que tange à segurança da informação.

Palavras-chave: segurança, web services, sistemas distribuídos.

ARQUITETURA DE SEGURANÇA EM APLICAÇÕES BASEADAS EM WEB SERVICES

INTRODUÇÃO

Informação vem se tornando cada dia mais importante e imprescindível no cotidiano da sociedade moderna em diversas áreas e de inúmeras maneiras. A quantidade de novas informações cresce constantemente, impulsionada pelos avanços tecnológicos que causam o barateamento e maior acessibilidade a novos meios de comunicação, fazendo com que pessoas e empresas possam interagir entre si mais rápida e facilmente do que antes.

Os sistemas de software também vêm se adequando a esse novo panorama de interatividade global, proporcionando soluções acessíveis de qualquer parte do mundo a qualquer instante, muitas vezes possuindo uma arquitetura distribuída, portátil, escalonável e ainda tendo capacidade de prover e requisitar serviços, ou seja, troca de informações entre aplicações.

Dentro desse novo cenário surgiram tecnologias de computação que se propõem ao suporte de todos esses requisitos. Web Services é uma dessas propostas. Mas esse panorama também traz algumas novas preocupações para os desenvolvedores de aplicações. A principal delas é relativa à segurança dos sistemas, pois toda a funcionalidade dos Web Services é feita on-line, e, portanto, sujeita a tentativas de ataques e outras práticas ilícitas do mundo virtual.

SEGURANÇA DA INFORMAÇÃO

A segurança da informação é um dos mais vastos, complexos e recorrentes temas dentro da tecnologia da informação por envolver diversos componentes e preocupações diferentes durante sua implantação e produção em um ambiente computacional (Moreira, 2001). Nessas fases é desejável a ocorrência da menor quantidade possível de vulnerabilidades e imprescindível o gerenciamento constante de como e por quem está sendo feito o acesso aos recursos disponibilizados aos usuários que estão cobertos pelo sistema de segurança (Nakamura, 2001).

Outro importante fator na aplicação da segurança da informação é o prévio conhecimento do que está sendo protegido e quanto custa protegê-lo. Se os gastos com segurança forem maiores do que uma possível perda dos recursos protegidos, talvez não seja uma boa idéia a adoção de medidas de segurança tão rígidas ou custosas (Wadlow, 2000).

Basicamente, a segurança da informação está fundamentada em sete conceitos que trabalham em conjunto para promover segurança a um sistema (O'Neill, 2003):

- **Confidencialidade:** garantia, através de criptografia, de que as informações armazenadas ou transmitidas não poderão ser vistas ou interpretadas por qualquer outra entidade diferente do usuário original e do destinatário.

- Autenticação: validação das credenciais de um cliente junto a uma autoridade certificadora.
- Autorização: verificação pelas autoridades certificadoras de que um cliente tem acesso ou não a um recurso que está tentando utilizar após já ter sido autenticado.
- Integridade: certificação de que as informações transmitidas não são alteradas acidental ou maliciosamente em nenhum ponto de seu trajeto entre um emissor e um receptor.
- Não-repúdio: garantia, através de criptografia, de que um emissor não pode negar que enviou uma mensagem que foi recebida por um receptor.
- Disponibilidade: garantia de que um serviço estará disponível aos usuários autorizados no momento em que necessitarem dele.

WEB SERVICES

A arquitetura orientada a serviços dos Web Services permite construir aplicações modulares e independentes que são distribuídas facilmente em qualquer estrutura de redes TCP/IP, através da criação de servidores e clientes que independem da linguagem de programação e do sistema operacional em que são implementados. Os servidores descrevem seus próprios serviços através de uma linguagem pré-definida e, dessa forma, os clientes podem facilmente obter informações sobre estes servidores, tais como: estrutura, métodos e parâmetros exigidos, e, assim, usar seus serviços (Rosenberg, 2004).

A iniciativa para o desenvolvimento da lógica de Web Services é um grande esforço conjunto de gigantes da área de informática como *IBM, Microsoft, Oracle, Sun, Novell, HP*, entre outros, que tenta resolver alguns problemas que tecnologias de sistemas distribuídos como CORBA e COM+ possuem, como passagem por *firewalls*, complexidade dos protocolos e integração entre plataformas heterogêneas (Basiura, 2003).

Mas o grande poder existente na arquitetura dos Web Services é que, com sua implementação e infra-estrutura, é possível alavancar muitos dos recursos já existentes e funcionais na Internet, como protocolos de segurança e de transporte de pacotes, evitando assim a complexidade existente nos protocolos de outras tecnologias de sistemas distribuídos (Hartman, 2003).

XML

É o padrão utilizado para serialização e descrição dos dados enviados e recebidos pelos Web Services em um formato de texto plano estruturado, que permite aos documentos não ter a complexidade de um formato de codificação binária usado por CORBA e COM+, além de ser mais extensível e legível (Basiura, 2003).

SOAP

O protocolo SOAP foi criado como um meio de transmitir mensagens no formato XML sobre protocolos-padrão de transporte na Internet como HTTP e SMTP. A própria estrutura de uma mensagem SOAP é definida com o uso de XML (Rosenberg, 2004).

SOAP possui um envelope no qual a mensagem XML é inserida. Este envelope é composto de duas partes: um cabeçalho que fornece informações, algumas vezes opcionais, sobre a mensagem SOAP, como, por exemplo, credenciais de segurança, e um corpo que contém o *payload* da mensagem, que pode ser uma operação a ser executada pelo Web Service e seus parâmetros ou um conteúdo qualquer a ser transmitido. O protocolo SOAP foi projetado de uma maneira que o torna extensível a futuras necessidades, sendo que a maior área livre a extensões fica no cabeçalho do envelope (Rosenberg, 2004).

Por razões de segurança as mensagens SOAP devem ter seu conteúdo mantido secreto dos intermediários (se forem não-autorizados) até chegarem aos seus destinos, onde o destinatário de uma mensagem SOAP deve reconhecer quem está enviando os dados e o que o emissor está autorizado a acessar (Rosenberg, 2004).

WSDL

É o padrão usado para descrever em formato XML a estrutura lógica e a sintática de um Web Service. Nessa descrição está exposto aos consumidores o que podem consumir desse Web Service, como devem interagir com ele através de mensagens empacotadas no envelope SOAP, e onde procurar e encontrar esse Web Service (Rosenberg, 2004). A utilização de XML como padrão para troca de informações e referências para a localização de serviços torna a WSDL mais flexível e rica que a IDL usada em CORBA e COM+ (Scribner, 2001).

Por outro lado, WSDL pode ser o maior problema de segurança dentro de uma arquitetura de Web Services, justamente porque expõe ao mundo exterior a interface de um Web Service. Devem ser aplicados mecanismos de segurança para bloquear o acesso de agentes não-autorizados a WSDL, até mesmo para leitura (Rosenberg, 2004).

UDDI

É um repositório central de localização conhecida baseado em XML, onde os Web Services podem ser publicados, pesquisados em tempo de execução e consumidos por clientes interessados em seus serviços. UDDI é um componente opcional na implementação de uma arquitetura de Web Services, pois o consumidor pode já conhecer a localização do Web Service desejado e não necessitar do serviço de descoberta (Scribner, 2001).

Classe Proxy

A classe *proxy*, ou simplesmente *proxy*, permite estabelecer uma referência a um Web Service remoto e usar sua funcionalidade dentro de uma aplicação. A função principal de um *proxy* é abstrair os detalhes de interação com o Web Service, como invocação, codificação, empacotamento SOAP e transmissão, tornando mais natural a implementação da lógica Web Service nas aplicações. Também possuem a capacidade de interpretar WSDL e gerar automaticamente o código-fonte apropriado para chamar qualquer Web Service (Basiura, 2003).

A Figura 1 mostra a utilização de todos os elementos básicos apresentados acima durante o ciclo de vida de um Web Service: o desenvolvedor cria, testa e disponibiliza um Web Service (1), podendo opcionalmente publicá-lo em um catálogo UDDI (2), e então um consumidor de Web Services poderá procurar o serviço nesse catálogo (3). Após obter a localização do Web Service, o consumidor requisita sua descrição WSDL (4). De posse da

estrutura lógica e sintática do Web Service, o consumidor pode criar um *proxy* e um cliente para acessar a interface (5) e, finalmente, chamar algum método do Web Service (6). Todas as operações são feitas utilizando o formato XML (Basiura, 2003).

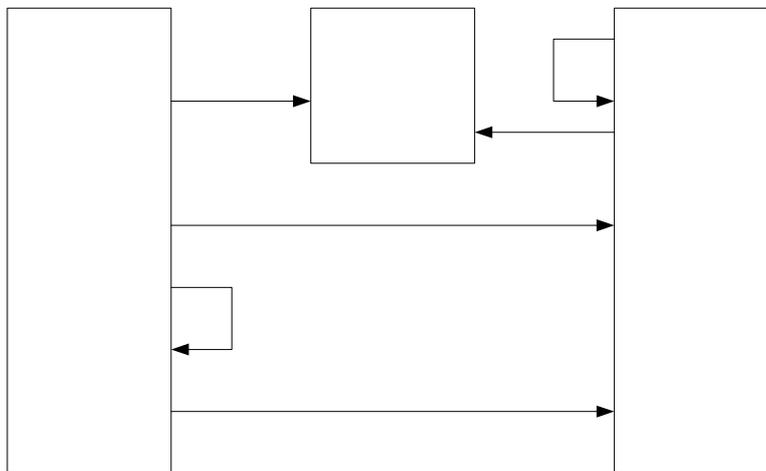


Figura 1. Ciclo de vida de um Web Service.

SEGURANÇA DE WEB SERVICES

3

Como já descrito anteriormente, os Web Services expõem suas funcionalidades através de uma arquitetura orientada a serviços que é bem mais aberta em virtude de sua característica distribuída e de sua natureza heterogênea quanto a plataformas de execução. Por prover essas facilidades de acesso e integração, a arquitetura de Web Services, é frágil em relação à segurança, que deve ser então considerada o foco principal durante o projeto de um sistema baseado em Web Services. O desafio é manter a eficiência dessas funcionalidades e ainda proporcionar um ambiente seguro (Hartman, 2003).

Consumidor
do Web
Service

Desafios da segurança de Web Services

5

1. *Segurança baseada no usuário final de um Web Service* – Web Services foram criados para serem consumidos por aplicações e não diretamente por usuários finais, isto é, humanos. Porém a permissão de acesso a um Web Service pode ser decidida a partir de informações a respeito de um usuário final. O desafio, então, é como embutir de forma segura as informações credenciais do usuário final dentro das mensagens SOAP que trafegam nas requisições a Web Services (O'Neill, 2003).
2. *Manutenção da segurança enquanto roteando entre múltiplos Web Services* – Dependendo da infra-estrutura do sistema distribuído, uma mensagem SOAP pode rotear entre diversos Web Services para atender a uma requisição de um cliente. Assim, por questões de confidencialidade, a informação contida na mensagem não deve ser acessível aos Web Services intermediários do roteiro, apenas ao Web Service que foi consumido pelo cliente e pelo Web Service que atendeu a requisição encadeada (O'Neill, 2003).

3. *Abstrair a segurança da rede subjacente* – A segurança aos Web Services não pode ser fixada nos mecanismos de segurança Web existentes atualmente. Deve ser flexível e dinâmica o bastante para se adaptar a qualquer novo protocolo de troca de mensagem que venha a surgir (O'Neill, 2003).

Como toda a comunicação com Web Services é feita através do protocolo SOAP em formato XML, suas mensagens são perfeitamente legíveis. Além disso SOAP não implementa segurança por questões de manutenção da simplicidade e portabilidade do protocolo. Então para que a comunicação seja segura implementações de segurança devem ser feitas ao nível de sistema (nas camadas de rede e de transporte) e ao nível de aplicação (através de processos e configurações personalizadas e pela aplicação de padrões de segurança em XML) (Scribner, 2001).

Segurança na camada de rede

A segurança nessa camada se reflete através dos mecanismos de comunicação e de segregação das redes interna e externa de uma organização (Ferreira, 2005).

1. *Firewall* – Conseguem bloquear ou liberar acessos partindo de determinados computadores ou aplicações aos Web Services, mas não possuem capacidade de análise da integridade do conteúdo das mensagens SOAP (Ferreira, 2005).
2. *VPN* – Permite proteger a confidencialidade e integridade dos dados trafegados e estabelecer perímetros que podem definir conjuntos de clientes confiáveis que têm acesso aos métodos dos Web Services. Por outro lado, uma VPN não controla o consumo inadvertido de Web Services por aplicações disparadas de computadores de dentro de seu domínio (Ferreira, 2005).

Segurança na camada de transporte

Dentro da camada de transporte a segurança de Web Services pode ser aplicada através da implementação de protocolos e mecanismos de segurança já existentes (Ferreira, 2005).

1. *Certificados digitais* – Validações digitais atribuídas por empresas certificadoras independentes e confiáveis a clientes *web*, *sites* ou usuários finais, através do uso de chaves de criptografia, para que possam ser autenticados em outras entidades da Internet (Basiura, 2003).
2. *PKI* – É uma implementação do conceito de criptografia assimétrica baseada em pares de chaves, uma pública, outra privada. A chave privada é armazenada no lado servidor e a chave pública está disponível no lado cliente. O servidor envia dados na forma codificada usando sua chave privada e o cliente recebe os dados e os decodifica usando sua chave pública. Se alguém entre o cliente e o servidor apanhar os dados, não conseguirá decifrá-los sem a chave pública da chave privada codificada. Da mesma maneira, o cliente envia os dados codificados usando a chave pública e o servidor a decodifica usando sua chave privada. Ninguém no meio do caminho pode decodificar os dados sem a chave privada (Basiura, 2003).
3. *SSL* – Permite a transmissão de dados com garantia da proteção de sua integridade e confidencialidade através de criptografia PKI e da autenticação de usuários e servidores por meio de certificados digitais. Essa segurança é obtida através da abertura de um canal

de comunicação seguro pelo cliente que o utiliza para trocar mensagens SOAP. Mas o grande problema existente em SSL é que não fornece uma segurança fim-a-fim. Em sistemas onde o servidor que armazena as informações sigilosas é diferente do servidor onde estão os Web Services, a vulnerabilidade é alta (Ferreira, 2005).

Segurança na camada de aplicação

Na camada de aplicação é possível levar a segurança ao nível da análise de integridade de cada mensagem SOAP enviada ou recebida por um Web Service, além do reconhecimento do usuário responsável por cada uma delas, através da autenticação de credenciais, certificados ou outras informações contidas nas mensagens (Ferreira, 2005).

1. *Nativa em sistemas operacionais* – Através da política de segurança nativa de um sistema operacional pode-se autorizar os privilégios de uma conta ou grupo de usuário em relação a um Web Service ou a seu diretório (Basiura, 2003).
2. *IP e DNS* – Restrições feitas sobre IP e DNS permitem ceder ou negar acesso requisitado ao Web Service a partir de um endereço IP específico e de nomes DNS simples ou múltiplos (Basiura, 2003).
3. *HTTP*
 - a. *Basic Authentication* – O cliente de um Web Service deve fornecer ao servidor web as credenciais de um usuário autorizado. Os pontos positivos desse tipo de autenticação são a facilidade de implementação, a compatibilidade com a maioria dos servidores web e navegadores, a rapidez da autenticação e o suporte por todos os *proxies* e *firewalls*. Os aspectos negativos são a forma não segura de envio das credenciais, a necessidade de reenvio dessas credenciais a cada solicitação feita e a impossibilidade do uso em conjunto com uma autorizadora certificadora personalizada (Basiura, 2003).
 - b. *Basic Authentication com SSL* – O problema do envio de nome e senha de usuário em formato não seguro existente em *Basic Authentication* pode ser corrigido através do uso de um canal seguro SSL (Basiura, 2003).
 - c. *Digest Authentication* – Tem as mesmas características da *Basic Authentication*, mas transmite as credenciais de uma maneira diferente e mais segura, através de um processo chamado *hashing* de via única. Nesse processo são acrescentadas outras informações a senha com a finalidade de impedir a captura e posterior reutilização da senha *hash* por um agente não autorizado. As vantagens são as mesmas da *Basic Authentication*, além do nome do usuário e da senha nunca serem enviados como texto limpo. Porém só é suportada pelo *Windows 2000* com *Active Directory*, *IIS 5*, *Internet Explorer 4*, *Netscape 4* (e suas versões posteriores) e também não pode ser usada com uma autorizadora certificadora personalizada (Basiura, 2003).
 - d. *Integrated Windows Authentication* – Similar a *Digest Authentication*, utiliza um *hashing* mais sofisticado, através do suporte ao protocolo de autenticação *Kerberos V5* (se o *Active Directory* estiver instalado e o navegador for compatível) ou do protocolo *challenge/response*. Os pontos positivos são os mesmos da *Basic* e *Digest Authentication*, com a vantagem da senha nunca ser enviada na mesma mensagem. Porém só é suportada pelo navegador *Internet Explorer*, não funciona com servidores *proxy*, necessita da abertura de portas TCP adicionais no *firewall* e

também não trabalha em conjunto com uma autorizadora certificadora personalizada (Basiura, 2003).

- e. *Client Certification Authentication* – Os certificados de cliente são emitidos por empresas certificadoras autorizadas com a finalidade de validar um cliente junto a um servidor que tenha suporte a certificados. A grande vantagem do certificado de cliente é que existe a garantia de que o certificado é válido, pois as certificadoras autorizadas verificam a identidade do usuário antes de emití-lo (Basiura, 2003).

4. ASP.NET

- a. *Forms Authentication* – Atualmente é a forma de autenticação mais utilizada por permitir gerenciamento mais facilitado de um grande número de usuários e ser totalmente personalizável. Mas suas desvantagens são a necessidade de mais memória e poder de processamento para tratamento das autoridades personalizadas, e a obrigatoriedade de o usuário ter que aceitar *cookies* para poder se autenticar (Basiura, 2003).
- b. *Passport Authentication* – É uma proposta da *Microsoft* para um sistema centralizado de login. É semelhante a *Forms Authentication*, mas com a diferença de que as credenciais ficam armazenadas em servidores próprios da *Microsoft* (Hartman, 2003). O ponto forte é a possibilidade de se manter apenas uma credencial de usuário para autenticação em diversos *sites* que suportem *Passport*. A desvantagem é que muitos usuários não confiam no serviço, por se tratar de um sistema proprietário centralizado que pode falhar e deixar todos sem condições de autenticação (O'Neill, 2003).
- c. *Controle de acesso* – O ASP.NET pode decidir se um usuário autenticado pode ou não acessar determinado Web Service ou outro recurso através da configuração do XML *web.config* (Basiura, 2003).

- 5. *XML* – O protocolo SOAP não implementa nenhum mecanismo segurança, apenas alavanca mecanismos já existentes na infra-estrutura de rede que desempenhem essa função. SSL é o principal desses mecanismos, mas o fato de não prover segurança fim-a-fim, criptografia seletiva, autenticação, integridade de dados e não-rejeição, o torna não-aplicável a todas as situações em que Web Services precisem ser seguros. Por essas razões novas especificações de segurança têm sido propostas com o intuito de promover segurança específica a Web Services. Mas falta ainda um acordo sobre qual desses mecanismos devem ser adotados como padrão (Ferreira, 2005).

- a. *XML Signature* – É um padrão aberto que permite especificar regras de processamento para criação e representação de assinaturas digitais em porções selecionadas de um documento XML, através de *tags* XML específicas e de criptografia assimétrica, com o intuito de promover integridade e não-repúdio a uma mensagem SOAP (Potts, 2003).
- b. *XML Encryption* – Especifica um processo para criptografia de dados e sua representação em formato XML. Os dados podem ser arbitrários (incluindo um documento XML completo), elementos XML, ou conteúdos de elementos XML. Um documento XML que utiliza a XML Encryption pode ser visto por qualquer utilizador, mas apenas o proprietário da chave de decodificação conseguirá compreender o conteúdo do que foi criptografado (Potts, 2003).
- c. *SAML* – É um padrão emergente para a troca de informação sobre autenticação e autorização, solucionando um importante problema das aplicações, que é a

possibilidade de consumidores transportarem seus direitos entre diferentes Web Services. Isto é importante para aplicações que integrem um número de Web Services para formar uma aplicação unificada (Ferreira, 2005).

- d. *XKMS* – Fornece um protocolo, baseado em SOAP, para gerenciamento de chaves públicas, incluindo funções para obtenção de informações sobre chaves, registro, verificação e revogação de permissões. O XKMS tem a função principal de fornecer suporte ao gerenciamento de chaves para tecnologias como XML Signature e XML Encryption, mas também pode ser utilizado para suportar outras tecnologias de chaves públicas (Potts, 2003).
- e. *WS-Security* – WS-Security propõe um conjunto de extensões no cabeçalho do envelope SOAP para inserir dados relativos a segurança através de especificações como XML Signature e XML Encryption, bem como os padrões que surgirem futuramente. O objetivo do WS-Security é fornecer a estrutura de uma solução de segurança de Web Services completa, que possa ser instanciada às necessidades do implementador (O'Neill, 2003). Mas o maior valor que a especificação WS-Security traz é a organização do cabeçalho SOAP, imprescindível para o estabelecimento de um padrão que transforme a segurança de Web Services de uma série de soluções personalizadas em um método mais unificado de transferir informações com segurança por meio da Internet (Potts, 2003).

CONCLUSÃO

Em mecanismos de segurança nativos de sistemas operacionais, como permissões de usuários e grupos e controle de acesso por IP e DNS, a solução de segurança fica totalmente dependente do sistema operacional e somente a autorização de acesso para usuários pode ser implementada.

Na segurança aplicada ao protocolo HTTP e sobre o ASP.NET existem problemas de incompatibilidade com alguns servidores *web*, navegadores e sistemas operacionais, além de só permitirem a implementação de autenticação e autorização.

Padrões de segurança baseados na camada de transporte sobre protocolos já estabelecidos, como SSL e PKI, fornecem confiabilidade na transmissão das mensagens, através de segurança básica ponto-a-ponto entre um cliente e um servidor, mas não garantem a integridade de uma comunicação fim-a-fim, nem provêm recursos para autenticação e autorização.

Nos padrões baseados em XML não existe o problema de dependência de plataforma, servidores *web* ou navegadores, pois XML é um padrão aberto e totalmente compatível com os Web Services. Além disso, os padrões de segurança baseados em XML oferecem segurança fim-a-fim, fornecendo mecanismos para confidencialidade e integridade do conteúdo das mensagens e capacidade de autenticação e autorização de usuários, o que possibilita uma solução completa para comunicação segura entre consumidor, intermediários e provedor.

REFERÊNCIAS

- Basiura, Russ et al. Professional ASP.NET Web Services. 1. ed. São Paulo: Pearson Education, 2003. 701p.
- Ferreira, Lucas de Carvalho. Relatório Técnico Segurança de Web Services. 1. ed. Rio de Janeiro: Cipher Segurança da Informação, 2005. 12p.
- Hartman, Bret et al. Mastering Web Services Security. 1. ed. Indianapolis: Wiley, 2003. 436p.
- Moreira, Nilton Stringasci. Segurança Mínima: uma visão corporativa da segurança de informações. 1. ed. Rio de Janeiro: Axcel, 2001. 211p.
- Nakamura, Rodolfo Reijiro. E-commerce na Internet: fácil de entender. 1. ed. São Paulo: Érica, 2001. 240p.
- O'Neill, Mark et al. Web Services Security. 1. ed. Berkeley: McGraw-Hill, 2003. 312p.
- Potts, Stephen et al. Aprenda em 24 horas Web Services. 1. ed. Rio de Janeiro: Campus, 2003. 367p.
- Rosenberg, Jothy; REMY, David L. Securing Web Services with WS-Security. 1. ed. Indianapolis: Sams, 2004. 408p.
- Scribner, Kenn; STIVER, Mark C. Applied SOAP: implementing .NET XML Web Services. 1. ed. Indianapolis: Sams, 2001. 432p.
- Wadlow, Thomas A. Segurança de Redes: projeto e gerenciamento de redes seguras. 1. ed. Rio de Janeiro: Campus, 2000. 269p.