UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TIAGO FARIAS MACARINI

ANÁLISE E IMPLEMENTAÇÃO DE MÉTODOS DE SEGURANÇA NO SISTEMA VISIONSCAN

CRICIÚMA, JUNHO DE 2009

TIAGO FARIAS MACARINI

ANÁLISE E IMPLEMENTAÇÃO DE MÉTODOS DE SEGURANÇA NO SISTEMA VISIONSCAN

Trabalho de Conclusão de Curso apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. MSc. Rogério Antônio Casagrande

CRICIÚMA, JUNHO DE 2009

Aos meus pais, Valmir e Niltéia, minha namorada Jaqueline e meus amigos pelo apoio concedido.

AGRADECIMENTOS

Agradeço a Deus pela sua presença constante em minha vida, pelo auxílio nas minhas escolhas e por me confortar nas horas difíceis.

Agradeço também:

A minha família, por todo amor, carinho, compreensão e pela sólida formação dada em minha juventude, o que proporcionou continuidade nos estudos até hoje.

A minha namorada Jaqueline por estar sempre transmitindo seu amor e compreensão.

Aos meus amigos Fernando, Tânia e Junior pelo apoio concedido durante a execução deste trabalho.

A todos os meus colegas de graduação, por compartilharem suas experiências de vidas e momentos de descontração e diversão durante todo o curso.

Agradeço profundamente o grande incentivador desta pesquisa que não mediu esforços para auxiliar na conclusão da mesma, meu professor e orientador Rogério, contribuindo também, além do conhecimento, com sua amizade.

Enfim, a todos que contribuíram para a execução desse trabalho, seja pela ajuda constante ou por uma simples palavra de amizade.

“O sucesso é uma consequência e não um objetivo”. (Gustave Flaubert).

RESUMO

Com o avanço dos recursos da Internet e também da necessitade de haver troca de informações entre sistemas computacionais diferentes, foi possível e necessário o desenvolvimento de *Web Services* para a realização deste tipo de tarefa. Assim como qualquer sistema computacional, sistemas baseados em *Web Services* possuem vulnerabilidades que devem ser tratadas para evitar possíveis ataques. A bomba XML, injeção Xpath ou anexos SOAP infectados por vírus são alguns exemplos de ataques XML que estes sistemas poderão sofrer. Pelo fato deste tipo de sistema estar disponível na Internet, as proteções a serem aplicadas nestes sistemas ou em sua infra-estrutura deverão ser indispensáveis. O VisionScan é um exemplo de sistema que é baseado em *Web Service* para a troca de informações entre o sistema cliente e servidor. Este sistema é uma aplicação de redes neurais artificiais voltada para a identificação de códigos de barras no padrão EAN 13 que utiliza serviços *web* para efetuar o processamento das informações. Verificou-se então neste trabalho as possíveis vulnerabilidades existentes em relação ataques XML e quais as formas de protegê-lo. Nos testes efetuados verificou-se que para deixar o sistema menos vulnerável contra estes ataques XML, é necessário que sejam implementados métodos para tratamento de informações e alterações na infra-estrutura do servidor.

Palavras-Chaves: Ataque XML, Segurança, *Web Service*, VisionScan

ABSTRACT

The development of the resources of the Internet and also the need of having an exchange of information between different computer systems, made it necessary and possible to the development of Web Services to implement this type of task. Like any computer system, Web Services-based systems have vulnerabilities that must be treated to prevent possible attacks. XML Bomb, XPath injection SOAP attachments or infected by viruses are some examples of XML attacks that these systems may suffer. Because this type of system is available on the Internet, the protections to be applied in these systems or their infrastructure will be essential. The VisionScan is an example of a system that is based on Web Service for the exchange of information between the client and the server system. This system is an application of artificial neural networks devoted to the identification of bar codes in the standard EAN 13, which uses web services to perform the processing of information. This report to checked these vulnerabilities in this system attacks on XML and what ways to protect it. In tests performed, it was found out that to make the system less vulnerable to these attacks XML, it must be implemented methods for processing information and changes in the infrastructure of the server.

Word-Keys: XML Attacks, Security, Web Service, VisionScan

LISTA DE ILUSTRAÇÃO

LISTA DE SIGLAS

CNPJ	Cadastro Nacional da Pessoa Jurídica
DTD	Document Type Definitions
ESB	Enterprise Service Bus
GPL	General Public License
HTTP	Hypertext Transfer Protocol
JME	Java Micro Edition
JAI	Java Advanced Imaging
OASIS	Organization for the Advancement of Structured Information Standards
SAML	Security Assertion Markup Language
SQL	Structured Query Language
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SSL	Secure Socket Layer
UDDI	Universal Description, Discovery and Integration
W3C	World Wide Web Consortium
WSDL	Web Services Description Language
XML	Extensible Markup Language
X-KISS	XML Key Information Service Specification
X-KRSS	XML Key Registration Service Specification

SUMÁRIO

1 INTRODUÇÃO

Em sistemas baseados em Web Services existem vários problemas de segurança, pelo fato de que as Infra-estruturas de Web Services não fornecem segurança implementada. Desta forma, padrões de segurança vão surgindo, sendo possível aplicar em um ambiente Web Service (FRAGA et al, 2006).

As tecnologias de segurança aplicadas em geral e com bastante intensidade na Internet, como é o caso do *Secure Socket Layer* (SSL), não conseguem fornecer uma segurança adequada a sistemas distribuídos (CUNHA; SILVA, 2005).

Um exemplo de um sistema que utiliza o Web Service seria o VisionScan. O sistema VisionScan é uma aplicação de redes neurais artificiais voltada para a identificação de códigos de barras no padrão EAN 13 que utiliza serviços *web* para efetuar o processamento das informações. Este sistema foi desenvolvido sob a linguagem Java e tem a finalidade de efetuar leitura de imagens em dispositivos móveis. Para a comunicação entre o dispositivo móvel e o servidor, foi implementado um *Web Service* para que a imagem captada pelo dispositivo móvel seja enviada ao servidor para o reconhecimento (CONTE et al, 2008).

Serão analisados e implementado métodos de segurança, sendo descritos durante o trabalho, para que este sistema se torne menos vulnerável.

Outra preocupação referente a segurança em Web Services, é anteceder a ação do *firewall*, pois existem ataques em que a ameaça se encontra no conteúdo do documento transportado na rede e não em sua origem. Desta forma o pacote que está transitando pelo ambiente não é bloqueado pelo *firewall* (FRAGA et al, 2006).

Em Web Services também existem os problemas de autenticação e autorização, onde serão descritos padrões existentes para que se possa validar estas informações de forma segura (CUNHA; SILVA, 2005).

1.1 OBJETIVO GERAL

Estudar e aplicar no sistema VisionScan métodos de segurança contra os ataques de Anexo SOAP, Bomba XML e Injeção XPath.

1.2 OBJETIVOS ESPECÍFICOS

a) Relatar conceitos sobre SOA e Web Service;
b) Enumerar alguns tipos de ataques de XML existentes em sistemas baseados em SOA;
c) Analisar os métodos de segurança para a proteção em ataques de XML no VisionScan;
d) Aplicar ao VisionScan os métodos de segurança para que seja minimizada a possibilidade de ataques de XML;
e) Testar as implementações de segurança que serão utilizados no VisionScan, para confirmar a proteção aplicada no sistema.

1.3 JUSTIFICATIVA

Nos últimos anos o uso da Internet se tornou mais comum nas empresas e também em residências. Com isso foram surgindo várias ferramentas e tecnologias que se beneficiaram e continuam se beneficiando da Internet. Dentre estas tecnologias, pode-se citar os sistemas baseados em serviços *Web* (FRAGA et al, 2006).

Como a Internet é uma rede que abrange o mundo inteiro, os problemas de segurança como invasões a servidores, computadores infectados com vírus e manipulação de informações que trafegam na rede, se tornam mais comuns de acontecerem, pois um computador conectado a Internet pode tornar-se um alvo destes ataques (POTTS; KOPACK, 2003).

Com a diversidade de sistemas e tecnologias existentes, também surgem diferentes tipos de ataques a estes sistemas. Como é o caso de sistemas que utilizam serviços *web*, que utilizando a Internet para a troca de informações, estão sujeitos a ataques e requisições maliciosas. Em *Web Services* existem os ataques de XML, como por exemplo a Bomba XML, as Injeções Xpath e os Anexos SOAP (JOSUTTIS, 2008).

O sistema VisionScan é um sistema que utiliza Web Services para a troca de informações entre o dispositivo móvel e o servidor. Neste sistema é necessário que exista implementado em seu código, métodos de segurança para que ataques XML sejam prevenidos, tanto no lado cliente quanto no servidor.

Com o desenvolvimento dos métodos necessários de segurança contra estes ataques, o sistema se torna mais confiável e também previne contra problemas de disponibilidade de serviços.

O conhecimento adquirido no estudo e implementação de métodos de segurança

em Web Services será muito útil, considerando que esta tecnologia está crescendo cada vez mais e a segurança é sempre muito importante, para que seja evitado que informações sejam visualizadas ou manipuladas.

2 ARQUITETURA ORIENTADA A SERVIÇO

Existem muitas definições sobre o que seria a Arquitetura Orientada a Serviço, do inglês *Service Oriented Architecture* (SOA). A SOA é um paradigma que conduz a uma arquitetura computacional concreta. Esta arquitetura faz com que diferentes aplicativos possam se comunicar entre si, independente da plataforma e da linguagem de programação em que estão sendo executadas ou construídas (JOSUTTIS, 2008).

Segundo Josuttis (2008) a SOA se baseia em três conceitos principais:

a) Serviço: Seria uma função ou componente do sistema responsável pela realização de alguma tarefa específica. Os serviços poderão estar ligados entre si, ou até mesmo reutilizadas em outros aplicativos de área distinta em que foi planejada, desde que sua funcionalidade não seja alterada;

b) *Enterprise Service Bus* (ESB): O ESB trata-se da infra-estrutura que possibilita a comunicação entre os aplicativos que utilizam a SOA, ou seja, fornecer a interoperabilidade. Este barramento virtual envolve a conectividade entre os sistemas, a transformação dos dados, o roteamento inteligente, a segurança, a confiabilidade, o gerenciamento de serviços, os monitoramentos e *loggins*. Desta forma pode-se considerar que seria a parte principal da arquitetura orientada a serviço;

c) Acoplamento Fraco: Seria o conceito de redução de dependências do sistema. Isto é bastante importante pelo fato de atualmente existirem variadas plataformas computacionais e também sistemas corporativos desenvolvidos em diferentes linguagens, sendo executadas em uma mesma empresa.

A interoperabilidade entre os aplicativos ocorre utilizando-se de troca de serviços,

onde um aplicativo envia um arquivo no formato *Extensible Markup Language* (XML) ao servidor de aplicativos, que é onde os serviços são executados, após o processamento da informação a resposta é enviada à máquina que fez a solicitação (máquina cliente). Basicamente seria desta forma o funcionamento da SOA (ABINADER; LINS, 2006).

Para este trabalho pode-se destacar melhor os dois primeiros itens citados anteriormente.

2.1 SERVIÇO

A utilização de serviços em aplicativos cresceu muito com a utilização de Web Services nas empresas.

Sampaio (2006, p.15) define serviço da seguinte forma:

> Componente que atende a uma função de negócios específica para os clientes. O serviço recebe requisições e as respondem ocultando todo o detalhamento do seu processamento.

O serviço é responsável pelo processamento das informações recebidas do cliente através de arquivos XML. Cada serviço possui sua tarefa específica, que seria um passo natural da funcionalidade de negócio, onde deve corresponder a uma atividade de negócio do mundo real (JOSUTTIS, 2008).

Um serviço poderá executar a rotina completa de uma funcionalidade de negócio, como por exemplo, a reserva de uma vaga no hotel, a emissão de um bilhete de passagem aérea ou verificar o saldo da conta bancária em um determinado dia. Toda a rotina deverá ser executada independentemente do estado de outro serviço. O cliente, ao solicitar o serviço, fornece apenas os parâmetros necessários para a sua execução completa. As rotinas

intermediárias do processamento devem ser gerenciadas apenas pelo serviço (SAMPAIO, 2006).

2.2 *ENTERPRISE SERVICE BUS*

O ESB, ou o Barramento de Serviços Corporativos, seria a infra-estrutura responsável em fornecer a interoperabilidade entre os sistemas. Chappel (2004, p. 9) define o ESB como:

> A transformação de dados é inerentemente parte do barramento em uma distribuição de ESB. Os serviços de transformação que são especializados para as necessidades das aplicações individuais ligadas no barramento podem ser localizados em qualquer lugar e acessíveis de qualquer lugar do barramento. Pelo fato de a transformação de dados ser uma parte integrante de ESB, um ESB pode ser imaginado como sendo uma resolução de desencontros de impedância entre as aplicações.

Conforme descrito na citação acima, o ESB é responsável também pela transformação dos dados no barramento pelo fato de integrar sistemas de diferentes plataformas e linguagem de programação (JOSUTTIS, 2008).

Outra função do ESB é efetuar o roteamento de forma inteligente para a localização das aplicações, como por exemplo fazer as chamadas de serviço e também enviar a resposta do serviço para o cliente (JOSUTTIS, 2008).

Prover segurança no sistema distribuído baseado em SOA também é função do ESB. A autenticação, autorização, confidencialidade, integridade, disponibilidade, contabilização e auditoria são os pontos principais de segurança que deverão ser tratados em um sistema baseado em SOA. Como segurança é o foco deste trabalho, serão descritos posteriormente os tipos de ataques existentes em XML, como por exemplo as Injeções Xpath, que seria um tipo de injeção de código no XML, as Bombas XML, que fazem com que os

documentos XML sejam expandidos, prejudicando o sistema distribuído e também os Anexos SOAP, onde, assim como nos e-mails, o SOAP também poderá receber vírus, worms, etc. (JOSUTTIS, 2008).

Outras funções do ESB são o mapeamento de dados, a confiabilidade, o gerenciamento dos serviços, o monitoramentos e loggins, o monitoramento das atividades corporativas e o suporte para implementação de serviços (FRAGA et al, 2006).

2.3 SOA NA PRÁTICA

Basicamente o conceito de arquitetura orientada a serviço foi descrito nas seções anteriores. Com o surgimento da SOA, foram colocadas em práticas as definições sobre aplicativos baseados em serviço.

Vários autores, analistas e fabricantes recomendam apenas uma das práticas de realizar o SOA: com Web Service (JOSUTTIS, 2008).

3 WEB SERVICE

Como descrito anteriormente, Web Service seria a SOA na prática. Desta forma o Web Service realiza a comunicação e a integração entre aplicativos diferentes utilizando a Internet ou uma intranet. O *The World Wide Web Consortium* (W3C) define o Web Service como um sistema criado para apoiar software interoperável com interação máquina-máquina utilizando uma rede. Ele tem uma interface descrita em um formato processável por máquina (especificamente WSDL). Outros sistemas interagem com o serviço Web de uma forma prescrita pela sua descrição, usando mensagens SOAP, normalmente transmitida utilizando HTTP com uma serialização XML em conjunto com outras normas relacionadas com a web (W3C, 2008).

Existem cinco padrões de Web Services, sendo que dois são gerais, um é o XML, arquivo utilizado para registrar e transferir as informações entre os aplicativos. Outro padrão geral é o *Hypertext Transfer Protocol* (HTTP), que é um protocolo de baixo nível usado para a distribuição de informações e hipermídia utilizando a Internet. Estes dois padrões já eram comuns antes do surgimento do Web Services (POTTS; KOPACK, 2003).

Os três outros padrões existentes são:

a) *Web Service Definition Language* (WSDL): Seria uma linguagem no formato XML para descrever as interfaces de serviços, como as assinaturas, protocolos de ligação e localização. Isto faz com que os outros aplicativos possam efetuar a troca de informações entre si utilizando serviços;

b) *Simple Object Acces Control* (SOAP): Permite a comunicação e integração entre aplicativos em um ambiente Web Service, utilizando arquivos XML para a realização de troca de informações. Também existem parâmetros de

chamadas e dados de resultados como parte integrando do SOAP;

c) *Universal Description, Discovery and Integration* (UDDI): Esta especificação tem como objetivo definir um padrão para a descoberta de serviços. Utilizando a UDDI pode-se descobrir a localização do serviço requisitado para que este serviço faça parte do Web Service.

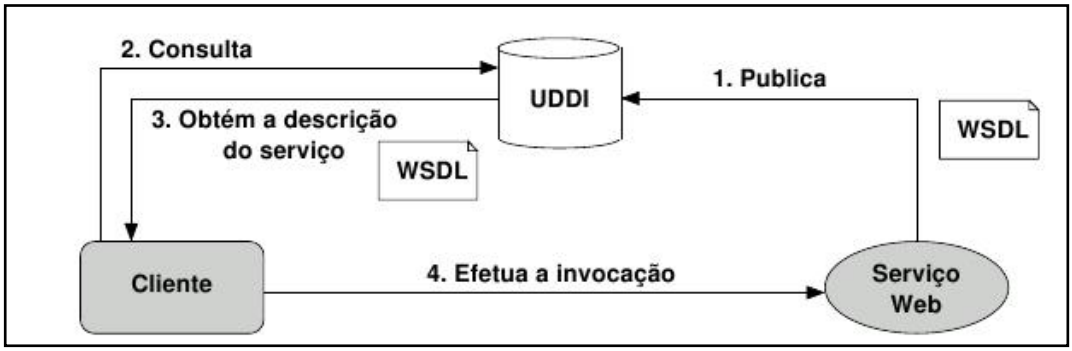

Figura 1: Funcionamento do Web Service com seus padrões
Fonte: FRAGA et al (2006)

Web Service foi a concretização dos benefícios de uma arquitetura que se baseia em serviços apara a comunicação entre aplicativos distintos, de diferentes plataformas e linguagens de programação. Isso beneficiou diretamente as empresas que possuem diferentes sistemas sendo executados e que necessitassem de integração (POTTS; KOPACK, 2003).

Como se pode notar nos padrões descritos acima, a transferência de informação é dada utilizando da linguagem XML, onde neste arquivo são contidos principalmente as informações dos protocolos existentes citados acima e parâmetros, para que o serviço seja executado. Na seção seguinte será descrito mais detalhadamente sobre a tecnologia XML.

Um exemplo de um sistema que utiliza o Web Service seria o VisionScan. O sistema VisionScan é uma aplicação que utiliza redes neurais artificiais para a identificação de códigos de barras no padrão EAN 13. Utiliza-se de serviços *web* para efetuar o processamento das informações. Este sistema foi desenvolvido com a linguagem Java e tem a finalidade de efetuar leitura de imagens em dispositivos móveis. Para a comunicação entre o dispositivo

móvel e o servidor, foi implementado e publicado um serviço em um *Web Service*, que recebe a imagem capturada pelo dispositivo móvel , reconhece a imagem e retorna as informações sobre o reconhecimento da imagem para o dispositivo móvel (CONTE et al, 2008).

Tratando-se então de um Web Service, a troca de informações entre o cliente e o servidor (sistemas consumidor e fornecedor respectivamente), é feita utilizando arquivos XML.

4 EXTENSIBLE MARKUP LANGUAGE (XML)

O XML é uma linguagem de marcação para organizar os dados dentro de um determinado documento, utilizando *tags* como um arquivo de *Hypertext Markup Language* (HTML), para classificar cada informação contida no arquivo (PITTS-MOULTIS, 2000).

A utilização de tags deixa o conteúdo do arquivo de maior entendimento. Não existe um padrão definido para as *tags.* Desta forma qualquer um poderá deixá-lo conforme a necessidade de uso (SAMPAIO, 2006).

Não é possível programar ou criar um site utilizando XML. Mas este tipo de documento pode ser utilizado para o transporte de informações, principalmente na utilização de Web Services ou pode também ser utilizado como banco de dados. Estes seriam apenas dois exemplos de utilização (SAMPAIO, 2006).

Para poder entender melhor sobre o que seriam as linguagens de marcação e as tags, que fazem parte do documento XML, será descrito abaixo os detalhes destes itens.

4.1 LINGUAGEM DE MARCAÇÃO

O termo marcação significa marcas ou anotações destinadas à identificação ou instrução de como proceder em um determinado evento ou de como o documento deverá ser interpretado (ALMEIDA, 2002).

Um código HTML é composto por marca ou *tags*. Estas tags são utilizados para determinar cada item do documento, no início e no fim, indicadas pelo sinal < (menor) no

início e > (maior) no fim, e o conteúdo desta tag geralmente são auto-explicativas (PITTS-MOULTIS, 2000).

Existem, para as linguagens de marcação, os sinais que indicam a tag de início e a tag de fim da instrução. A tag de início é formada apenas pelo conteúdo entre os sinais menor (<) e maior (>) e a tag de fim existe uma barra (/) antes do conteúdo. Por exemplo, para determinar que uma determinada informação seja a data de nascimento e que seu dia seria 16-04-1985. Então a tag que define a data de nascimento, juntamente com a sua instrução ficaria da seguinte forma: <DataNascimento>16-04-1985</DataNascimento>. Pode-se afirmar que este exemplo faz parte de um documento XML (PITTS-MOULTIS, 2000).

Ao contrário do HTML, o XML não possui marcações já definidas, como é o caso do HTML. A linguagem XML permite que o próprio criador do documento crie suas próprias marcações, geralmente com o nome sugestivo à função, como no exemplo citado acima, onde a data de nascimento do cliente é definida pela tag de marcação <DataNascimento> (ALMEIDA, 2002).

A W3C define como os principais objetivos do XML a ser diretamente utilizável na Internet, ser de fácil entendimento as pessoas, possibilitar um meio independente para a publicação eletrônica, permitir a definição de protocolos de transporte entre as empresas independentemente do *hardware* e do *software*. Facilitar o processamento de dados em *softwares* de baixo custo, facilitar a utilização de metadados que auxiliam na busca de informações e aproximar os produtores e consumidores de informação (servidor e cliente respectivamente) (BROWN; HAAS, 2008).

Os elementos são os blocos de construção do documento XML, como por exemplo as tags inicial e final, já descritos na seção anterior . Além destes dois tipos de elementos, existem outros, como as tags de elemento vazio e elementos filhos. Os elementos vazios são as tags que não possuem conteúdo algum. No documento XML, as tags são

representadas da seguinte forma: <CNPJ></CNPJ>, sendo que o nome CNPJ foi de escolha aleatória, não sendo obrigatório o uso deste nome para a representação de elementos vazio. Existe também a forma abreviada de representação de elementos vazios. Esta forma abreviada é composta pelo nome do elemento seguido da barra. Utilizando o mesmo elemento do exemplo anterior, no documento é indicado da seguinte forma: <CNPJ/> (ANDERSON, 2001).

Assim como no HTML, o XML poderá conter várias tags e elementos. Poderão existir os elementos principais e elementos filhos, que seriam dependentes do elemento principal. Pode-se imaginar então, uma árvore de elementos pai-filho, que é uma das propriedades importantes do XML (ANDERSON, 2001).

Na figura abaixo, será possível visualizar parte das tags de um XML, juntamente com seus elementos pai e filho:

```
<bacharelado>
        <descricao>Bacharéis em Ciência da Computação</descricao>
        <turma>
                <pessoa>
                        <nome>Pedro</nome>
                        <idade>22</idade>
                        <cidade>Criciúma</cidade>
                </pessoa>
                <pessoa>
                        <nome>Maria</nome>
                        <idade>23</idade>
                        <cidade>Araranguá</cidade>
                </pessoa>
                <pessoa>
                        <nome>João</nome>
                        <idade>25</idade>
                        <cidade>Sombrio</cidade>
                </pessoa>
        </turma>
</bacharelado>
```

Figura 2: Parte de um documento XML

Na figura acima, percebe-se as tags bacharelado, descrição, turma, pessoa, nome, idade e cidade. As tags nome, idade e cidade são filhos da tag pessoa, que é filho da tag turma,

que é filho da tag bacharelado. É visualizado então, neste exemplo, de forma bem definida, a hierarquia existente no documento XML. Esta figura é somente um exemplo. Pode-se considerar como somente uma parte de um documento XML (PITTS-MOULTIS, 2000).

4.2 ATRIBUTOS

Os elementos XML permitem que atributos sejam vinculados ao XML. Estes atributos servem para especificar uma propriedade do elemento. Os atributos são definidos como pares, de nome e valor, onde o valor deverá sempre estar entre aspas (ALMEIDA, 2002).

A Figura 3 mostra uma parte de um XML que utiliza atributos em alguns de seus elementos.

```
<produto>
        <nome língua="inglês">book</nome>
        <preco moeda="dólar">45,00</preco>
        <fornecedor formato="XLB56" língua="inglês">
                <rua>Penbridge Square</rua>
                <numero>30</numero>
                <cep>92310</cep>
                <pais>United Kingdom</pais>
        </fornecedor>
</produto>
```

Figura 3: Parte de um documento XML com seus atributos

Note que para as tags nome e fornecedor existe o atributo língua, que define o idioma do elemento. Os outros dois atributos são a moeda, que pertence a tag preco e o formato que pertence a tag fornecedor.

4.3 *DOCUMENT TYPE DEFINITIONS*

O Document Type Definitions (DTD) é bastante importante pois permite que o usuário que esteja criando o XML defina as suas próprias marcações, ou tags (ALMEIDA, 2002).

Para a construção de um DTD, é obrigatório obedecer uma série de regras de sintaxes básicas do XML, para que todo o vocabulário existente no XML seja reconhecido. Se algum vocabulário não estiver contido no DTD, não será reconhecido no processamento (ANDERSON, 2001).

A DTD poderá estar tanto internamente no próprio arquivo XML, como externamente, em um arquivo separado.

Conforme Pitts-Moultis (2000), a chamada do DTD externo, dentro de um arquivo XML é feito no cabeçalho, da seguinte forma:

<?xml version=”1.0”>

<DOCTYPE produto SYSTEM “myserver/decs/teste.dtd”>

A declaração DOCTYPE consiste no nome principal e no nome do elemento raiz do documento XML. Já a declaração SYSTEM consiste em indicar diretamente ao parser (processador XML) onde se encontra o arquivo DTD.

Basicamente um DTD para a parte do XML da figura 2 pode ser definida da seguinte forma:

<!ELEMENT produto(nome, preco, fornecedor)>

<!ELEMENT nome (#PCDATA)>

<!ELEMENT preco (#PCDATA)>

<!ELEMENT fornecedor (#PCDATA)>

A declaração ELEMENT indica o nome do elemento aceito pelo XML. Para o elemento produto, existe os elementos dependentes (filho), que seria o nome, preco e fornecedor, sendo que estes elementos poderão receber qualquer tipo de dados (indicados pelo #PCDATA).

Estes exemplos acima são bastante básicos, sendo apenas para demonstração. Mais informações poderão ser encontradas acessando o site http://www.w3schools.com/DTD/default.asp.

5 SEGURANÇA EM SISTEMAS BASEADOS EM WEB SERVICE

Assim como em qualquer software existente, segurança é sempre fundamental para que a integridade e a confidencialidade das informações sejam asseguradas, principalmente quando existe troca de informações que se utiliza da Internet ou mesmo da rede local.

Focando mais em sistemas que se baseiam em SOA, como é o caso do VisionScan, os problemas de segurança podem surgir pois muitas pessoas poderão ter acesso ao mesmo sistema, sendo que nem todas as pessoas tem a permissão de visualizar ou manipular os dados (JOSUTTIS, 2008).

Os sistemas que trocam informações entre si utilizando arquivos XML que percorrem a rede precisam se preocupar em autenticação, autorização, confidencialidade, integridade, disponibilidade, contabilização e auditoria. Alguns autores, como por exemplo Landwehr, indica que o não repudio é um dos pontos fundamentais de segurança em *Web Services* (FRAGA et al, 2006).

De acordo com Josuttis (2008) os itens abaixo descrevem o que cada categoria de segurança descrito acima significa:

a) **Autenticação:** Seria a verificação de uma identidade, sendo que essa identidade pode ser o usuário, um dispositivo físico. Seria descobrir quem está solicitando o serviço;

b) **Autorização:** Isso significa identificar se um determinado usuário, ou como citado acima, uma identidade, tem a permissão de executar ou visualizar o resultado de um determinado serviço;

c) **Confidencialidade:** Confidencialidade assegura que somente o usuário que chamou o serviço poderá visualizar os dados, enquanto os mesmo estão trafegando pela rede entre o fornecedor e consumidor;

d) **Integridade:** O ponto principal da integridade é garantir que os dados que estão trafegando pela rede não sejam alterados ou falsificados. Caso isso ocorra, as informações contidas no arquivo estarão incorretos;

e) **Disponibilidade:** Disponibilidade em um sistema que utiliza Web Service significa dizer que o serviço chamado esteja sempre disponível, pois existem ataques em servidores de serviços que fazem que os dados não sejam perdidos e nem corrompidos, tornando o sistema inoperante. Um ataque típico seria o *denial of service*, ou traduzindo para o português, negação de serviço;

f) **Contabilização:** O objetivo da contabilização seria guardar os rastros do consumo de serviço, ou seja, armazenar em arquivos, txt por exemplo, todo o histórico de cada serviço. Isso poderá ser útil para o gerenciamento, planejamento, financeiro, etc;

g) **Auditoria:** O conceito de auditoria é parecido com contabilização. Auditoria armazenaria as informações relevantes a segurança de cada serviço, como por exemplo o usuário requisitante, horário de execução e qualquer outro detalhe que interesse a segurança ou exigências legais;

h) **Não repúdio:** O não repúdio assegura que um determinado usuário não poderá negar a sua participação em um evento da execução do serviço ou transação.

Uma vulnerabilidade no sistema é causado por uma má implementação do sistema, ou uma falha na configuração ou uma falha na operação do sistema. Isso pode permitir que usuários não autorizados invadam, ou que usuários não autorizados manipulem o sistema, prejudicando o seu funcionamento (FRAGA et al, 2006).

Já uma ameaça consiste em uma possibilidade de ataque ao sistema. Isto compromete a confidencialidade, integridade, disponibilidade ou a autenticidade. O ataque seria a exploração de algumas vulnerabilidades existente no sistema, prejudicando o seu funcionamento e até mesmo roubando e modificando as informações importantes e confidenciais existentes (FRAGA et al, 2006).

Conforme FRAGA et al (2006) é importante destacar os tipos de ataques existentes em sistemas baseados em SOA:

a) **Interceptação:** Seria a visualização das informações por usuários não autorizados;

b) **Interrupção:** A interrupção é a não disponibilidade de um determinado serviço, ou que a informação fornecida pelo serviço não chegue ao seu destino;

c) **Modificação não-autorizada:** Algum usuário não autorizado faz alterações no arquivo que trafega pela rede em direção ao seu destino. Pode ser tanto no sentido consumidor-fornecedor ou fornecedor-consumidor;

d) **Personificação:** Uma entidade não autorizada transmite uma informação falsa entre os sistemas distribuídos se passando por uma entidade válida. Em muitos ataques o usuário do sistema acredita que esta se comunicando com um determinado usuário autorizado, ou servidor, mas na verdade está se comunicando com um criminoso.

Conforme descritos nas seções anteriores, os Web Services utilizam o protocolo SOAP para a transmissão das informações entre os aplicativos, sendo que este protocolo é baseado em XML. Então, para que um Web Service possua um grau confiável de segurança, é necessário que exista a criptografia nas transações efetuadas. Este tratamento poderá ser feito no nível de transporte das informações ou no próprio XML (SILVA, 2004).

A forma de segurança citada no parágrafo acima seria uma segurança em nível

lógico, onde o tratamento da segurança está sendo efetuada no sistema computacional em si. Outro nível de segurança que também é muito importante seria a segurança física, que destina-se em controlar o meio físico de onde se encontram os sistemas. As formas de controle poderá ser o controle de acesso as salas onde estão as máquinas que executam os sistemas e também medidas para evitar desastres como incêndio ou inundações, por exemplo (FRAGA et al, 2006).

Este trabalho abordará apenas a segurança em nível lógico, mais especificamente na camada de transporte. Serão descritos os tratamentos a serem feitos no XML gerado pelo VisionScan para que os servidores de serviços não recebam algum código malicioso, prejudicando toda a execução de qualquer serviço.

Um sistema baseado em Web Service deverá ser fortificado ao ponto de que o custo do ataque seja bem superior ao lucro que o invasor terá atacando o sistema, independente de qual for o método utilizado para este ataque (POTTS, KOPACK, 2003).

Como a arquitetura de um Web Service é aberta em função de ser distribuída e heterogênea em relação a plataformas de execução, ela se torna frágil em relação a segurança. Isto é um ponto de maior importância quando se inicia um projeto de um sistema baseado em Web Service (CUNHA; SILVA, 2005).

Como descrito anteriormente, as informações que trafegam na rede oriundas de um sistema baseado em SOA, utilizam o protocolo SOAP. O maior desafio é fazer com que as informações confidenciais dos usuários sejam transmitidas de forma segura utilizando a rede em mensagens SOAP (O' NEILL, 2003).

Outro ponto que deverá ser bastante seguro em um ambiente de Web Service é garantir que somente o sistema cliente e o sistema servidor consigam visualizar e modificar as informações do documento SOAP. Isto porque durante a transmissão do conteúdo pela rede,

poderão existir diversos roteamentos e em algum roteamento o documento poderá ser visualizado ou modificado caso não exista algum tratamento (O' NEILL, 2003).

Os métodos de segurança a serem utilizados em *Web* Services deverão ser flexíveis e dinâmicos o bastante para poder ser adaptado a qualquer protocolo novo de troca de mensagens (O' NEILL, 2003).

6 ATAQUES DE XML EM SISTEMAS BASEADOS EM WEB SERVICES

As informações que transitam pela Internet não podem ficar de fácil entendimento para quem lê, ou seja, não poderá ficar com o texto claro. Isto já seria uma falha para que usuários não autorizados visualizem o conteúdo do documento (POTTS; KOPACK, 2003).

Conforme descrito na seção anterior, um dos métodos de ataque a servidores de serviço seria a falsificação da autenticação e também da autorização. Uma falha nestes itens pode permitir que invasores ataquem os servidores de serviços, visualizando e até mesmo alterando informações dos serviços. Este tipo de ataque interfere diretamente a confidencialidade e a integridade das informações (JOSUTTIS, 2008).

Uma requisição de serviço é solicitada utilizando um arquivo XML. Neste arquivo são contidas as informações de usuário, senha e quaisquer outros parâmetros necessários para o processamento desta requisição. Caso não exista tratamento nenhum no documento gerado pelo VisionScan, como criptografia por exemplo, algum interceptador poderá resgatar este documento quando o mesmo estiver trafegando na rede, e visualizar ou modificar as informações (JOSUTTIS, 2008).

Existe um problema bastante comum de acontecer em processos distribuídos que utilizam múltiplas camadas de abstração, que seriam quando um serviço depende de outro serviço para que se chegue ao resultado desejado. Isto poderá ser possível quando um cliente inicia um processo utilizando um portal de serviços, e este processo passa por diferentes outros processos, ou por diferentes *backends* e até mesmo chegar a algum ponto onde é necessário a intervenção de algum usuário. Por passar por diferentes camadas,poderão existir dois problemas: não está claro qual sistema autentica e autoriza o usuário a utilizar o serviço e

também garantir a confidencialidade nas diferentes conexões e nós durante a transmissão das informações (JOSUTTIS, 2008).

Para o primeiro problema fica a questão de qual sistema deverá autenticar e autorizar o usuário. Se é o *frontend* ou o *backend*. Isto porque um poderá permitir uma determinada funcionalidade ao usuário, onde outro não permite. Como o serviço poderá passar por vários *backends*, é necessário que todos os sistemas compartilhem estas identidades dos usuários. Já para o segundo problema que tem a ver com a integridade e confidencialidade, não basta que exista uma conexão física seguras entre o *frontend* e o *backend*. Isto é bastante importante, pois poderá trafegar na rede informações como número de documento, cartões de créditos ou dados confidenciais de empresas, mas o *Secure Socket Layer* (SSL) não é o suficiente. Métodos recomendados de segurança, assim como o SSL, serão descritos com mais detalhes no capítulo seguinte (JOSUTTIS, 2008).

Sistemas distribuídos possuem capacidades multiclientes e frequentemente as utilizam desta capacidade. Desta forma vários clientes acessam o mesmo *backend*, e um cliente não poderá acessar as informações de outro cliente que também esteja acessando o sistema. No caso de um determinado cliente de um banco acessar a sua conta bancária, provavelmente outro cliente deste mesmo banco também esteja acessando a sua conta bancária, ambos utilizando a Internet. Caso o primeiro cliente acesse a sua conta e faça uma requisição do seu saldo bancário, esta informação não poderá, em hipótese alguma, ser visualizada pelo segundo cliente (FRAGA et al, 2006).

Neste caso o serviço bancário poderia requerer que um banco apenas solicitasse os dados do cliente, como a agência e o número da conta, e esta regra ser aplicada a todo o tipo de serviço chamado. Mas neste caso não funcionaria com empresas de telefonia celular. Existem poucos fornecedores de rede celular, mas existem muitos provedores de serviços para

celular. Algumas das informações são relacionadas a esse provedores devem ser confidenciais e outras poderão ser acessíveis em todas as empresas da rede.

Atualmente os usuários têm a permissão de fazer a troca do provedor de serviço celular, continuando com o mesmo número do telefone. Poderá não ser apropriado negar o acesso as informações do cliente que está associado a um número de telefone celular para um determinado provedor de serviço, como por exemplo o provedor que o usuário possui anteriormente. O *backend* da rede não saberá antecipadamente a qual provedor de serviço um determinado número está relacionado. Desta forma então este *backend* terá que permitir o acesso para todas as empresas associadas a rede do cliente (JOSUTTIS, 2008).

Nas empresas onde estão sendo aplicadas sistemas distribuídos, a segurança poderá ser negligenciada ou ignorada por vários motivos, como por exemplo as pessoas envolvidas na implementação do projeto suporem que a própria infra-estrutura SOA já possua segurança suficiente, ou por que segurança requer muito esforço e planejamento para a sua implementação, ou pelo motivo de se saber que é impossível obter segurança absoluta com os métodos de segurança comuns para Internet como os *firewalls* e protocolos como *Secure Sockets Layer* (SSL) ou também por haverem conflitos entre a equipe de infra-estrutura e de negócios, decidindo quem deverá aplicar a segurança no sistema (FRAGA et al, 2006).

6.1 SEGURANÇA, DESEMPENHO E ESTADO

A aplicação de segurança em um sistema tem um custo. Fazer requisição de serviços adicionais para a tomada de decisões, encriptar ou decriptar informações, fazer validações de integridade e checar a disponibilidade de um serviço consome tempo de

processamento e recurso computacional. Mas esse seria o preço para manter o sistema seguro. O correto é sempre encontrar o equilíbrio entre o risco de falhas e o desempenho de processamento das informações nos sistemas distribuídos, como por exemplo os Web Services (LANDWEHR, 2001).

Como a aplicação de segurança afeta diretamente o desempenho, poderá afetar também os estados dos serviços.

Um serviço com estado seria um serviço que consegue manter o seu estado em suas múltiplas chamadas de serviço. O exemplo mais comum seria o de carrinho de compras de uma loja virtual. Cada item da loja que se deseja comprar, é adicionado no carrinho e cada item a mais que for comprando, o número de itens no carrinho também vai aumentando. Ao finalizar o pedido, todos os itens do carrinho ainda estarão lá. Nesta caso cada solicitação de compra que é adicionado no carrinho é requisitado um serviço para efetuar esta operação. A cada requisição o serviço foi mantendo o seu estado, possibilitando a adição de itens no carrinho (JOSUTTIS, 2008).

Para cada chamada de serviço que é requisitado, deverá ser verificado se o usuário que está fazendo a chamada de serviço possui a permissão de execução. Para que o serviço saiba se o usuário tem a permissão, a cada requisição deverá ser checada esta informação. Com isso o *backend* poderá resgatar as informações da identificação feita e utilizá-lo para carregar o perfil do usuário relacionado a estas informações (JOSUTTIS, 2008).

Caso os serviços sejam roteados para diferentes sistemas fornecedores, o perfil do usuário deverá ser carregado novamente em cada requisição ou pelo menos na primeira chamada para cada servidor em que os serviços serão executados (sistema fornecedor). Os perfis do usuário poderão ficar em cache do sistema fornecedor, assim cada vez que este mesmo usuário efetuar a requisição de algum serviço deste sistema, o seu perfil já estará disponível (LANDWEHR, 2001).

Neste caso só existe o problema em definir em quanto tempo os perfis dos usuários que estão em cache, continuarão sendo válidos. Para isso poderá ser utilizado um código de identificação para o *backend*, onde o mesmo poderá ser enviado posteriormente a todas as chamadas a serem feitas futuramente (LANDWEHR, 2001).

6.2 ATAQUES DE XML E WEB SERVICES

Em todo o sistema baseado em Web Service, como por exemplo o VisionScan, têm-se o risco de receber um ataque externo nos sistemas fornecedores, como por exemplo os ataques de negação de serviço. Muitas requisições de serviços, autorizadas ou não, podem prejudicar o fornecedor de serviço, e quando estes ataques são concretizados, é necessário reconhecer o ataque que está sendo efetuado e responder de acordo com as necessidades, para que o ataque não seja concluído (O'NEILL, 2003).

Mas não somente este tipo de ataque é possível de ser feito. Também é possível fazer o ataque com documentos XML e Web Services, conforme abaixo.

6.2.1 Bombas XML

Em algumas situações o interpretador de XML poderá expandir o documento recebido, com pouco conteúdo, em um documento enorme. Abaixo segue um exemplo:

<XML version="1.0"?>

```
<!DOCTYPE xmlbomb [
<!ELEMENT data (#PCDATA)>
<!ENTITY a “&b;&b;&b; ”>
<!ENTITY b “&c;&c;&c; ”>
<!ENTITY c “&d;&d;&d; ”>
<!ENTITY d “foo  ”>
]>
<data>&a;</data>
```

Neste caso a entidade que está sendo referenciada, que neste caso é a entidade d, expande o seu valor para as outras entidades. Desta forma então uma entidade é expandida três vezes para as outras entidades. Caso isto fosse visualizado em um navegador de Internet, ficaria da seguinte forma. (JOSUTTIS, 2008):

<data> foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo foo </data>

O exemplo citado acima é com pouco valores, mas isto poderá ser com mais entidades e mais valores em cada entidade. Caso exista dez expansões para dez entidades, isto o processador de XML expandirá em 10.000.000.000 (dez bilhões) de “*foo*”. Isto faz com que o fornecedor de serviço perca tempo e memória em seu processamento, sendo que em alguns casos poderá até parar a execução o serviço requisitado. (JOSUTTIS, 2008).

Este tipo de ataque de negação de serviço não pode ser detectado por um *firewall*, pois o gerador do ataque está no conteúdo do documento XML e não no número de mensagens enviadas.

Caso este tipo de ataque possa ocorrer um sistema distribuído, deverá de alguma maneira, restringir os processadores XML ou detectar estas situações antes que o documento seja processado.

6.2.2 Injeções XPath

Como o próprio nome já sugere, injeções Xpath é uma espécie de injeção de código. Da mesma forma que acontece em injeções SQL, uma injeção Xpath engana a implementação feita para a leitura de entrada do cliente no sistema (digitação de um valor em um campo qualquer do sistema), fazendo com que o significado original do valor digitado seja alterado (KLEIM, 2004).

Para melhor compreensão, será demonstrado primeiramente como este tipo de ataque funciona no SQL.

Originalmente, o seguinte SQL é efetuado quando um usuário digita o seu código de usuário:

SELECT cidade, cep, rua FROM empregados WHERE id = 42

Na consulta acima suponhamos que o código do usuário seja 42.

Então desta forma, simplesmente está sendo requisitado a cidade, o CEP e a rua de onde o usuário mora. Esta consulta será feita na tabela empregados, sendo que o código do usuário é 42.

Mas ao invés de ser digitado somente o número 42 no campo Código de Usuário, seja digitado o seguinte:

42 UNION SELECT usuario, senha, 'a' FROM usuario

Caso não exista tratamento algum no campo de digitação de código, a consulta a ser efetuada no banco de dados será a seguinte:

SELECT cidade, cep, rua FROM empregados WHERE id = 42 UNION SELECT usuario, senha, 'a' FROM usuario

Verificando na consulta feita acima, serão retornados todos os usuários e senhas existente na tabela de usuários. Isto é muito perigoso para o sistema.

Este mesmo procedimento também é possível quando se utiliza o XML. Com o XPath, é possível navegar e buscar valores em arquivos XML. Um determinado campo do sistema poderá lhe encaminhar em um determinado local do documento XML, mas alterando o valor original da consulta, poderá também ser levado para um outro local do arquivo, podendo, desta forma, obter ou manipular outros dados do arquivo (KLEIM, 2004).

O seguinte código irá buscar as informações de usuário e senha em um documento XML:

```
string(//senha/usuario[email/text()=' tfmacarini@gmail.com' and senha/text()='123456'])
```

Então a consulta feita acima irá buscar o seguinte trecho do arquivo XML:

```
<senha>
    <usuario>
        <email>tfmacarini@gmail.com</email>
        <senha>123456</senha>
    </usuario>
</senha>
```

Mas agora ao invés de digitar o nome de usuário como tfmacarini@gmail.com, for digitado como *' or 1=1 or ''='*, a consulta XPath ficará da seguinte forma:

```
string(//senha/usuario[email/text()='' or 1=1 or ''='' and senha/text()='sfsdcsc'])
```

Como a segunda expressão (1=1), que sempre é combinada com o *or*, desta forma sempre será verdadeiro, o resultado da consulta mostrará o usuário e senha do primeiro usuário do arquivo (KLEIM, 2008).

6.2.3 Anexos SOAP

Assim como os *emails*, o protocolo SOAP pode receber anexos de qualquer tipo. Desta forma poderá ser uma porta de entrada para arquivos maliciosos, como por exemplo vírus e *worms* (JOSUTTIS, 2008).

7 INFRA-ESTRUTURAS DE SEGURANÇA PARA SERVIÇOS WEB

Infra-estruturas já utilizadas e prontas para Internet, Web Services ou qualquer outro *middleware* não asseguram que o sistema estará bastante seguro. Os protocolos desenvolvidos para fornecer interoperabilidade não fornecem segurança. Foram desenvolvidos especificamente para trabalharem com sistemas distribuídos. Os padrões XML, SOAP, WSDL e UDDI foram feitos para transitarem de forma eficiente entre os *firewalls*, por isso não existe nenhum aspecto de segurança implementado (FRAGA et al, 2006).

Mesmo que uma infra-estrutura já viesse com segurança embutida, seria necessário um estudo para verificar se os métodos de segurança existente na infra-estrutura seria o necessário para obter a máxima segurança no sistema implantado e garantir a autorização, autenticação, confidencialidade, integridade, disponibilidade, contabilização e auditoria (CUNHA; SILVA, 2005).

O foco neste trabalho é obter o máximo de segurança possível (não absoluta) nos ataques mais conhecidos para evitar ataques de documentos XML, como por exemplo os ataques citados na seção anterior. Mas para isso é bastante importante que seja descrito alguns conceitos de segurança em outras camadas do ESB.

A seguir será descrito a segurança na camada de transporte, e logo após a segurança na camada de mensagem, que seria o tratamento dos documentos para reduzir problemas de segurança.

7.1 SEGURANÇA NA CAMADA DE TRANSPORTE

Na camada de transporte de mensagens, abaixo da infra-estrutura de SOA, geralmente é utilizado a criptografia de nome *Security Socket Layer* (SSL), ou traduzindo, Camada de Socket Seguro, bastante comum para transações via Internet (SILVA, 2004).

Este tipo de segurança serve somente quando a comunicação entre os aplicativos é feita de forma direta, entre o consumidor e o fornecedor, sem passar por nenhum roteamento. Mas caso a comunicação passe por vários outros pontos de roteamento, este tipo de criptografia não é segura, porque a mesma é desfeita quando passa por um roteador. Desta forma o arquivo XML que está transitando pela rede poderá ser visualizado em algum ponto entre o consumidor e o fornecedor (SILVA, 2004).

7.2 SEGURANÇA NA CAMADA DE MENSAGENS

Na camada de mensagem, a segurança é aplicada no conteúdo do documento a ser transmitido, neste caso sendo o XML. É utilizado o protocolo da infra-estrutura do sistema de forma que nenhum outro usuário não autorizado seja capaz de visualizar ou modificar o conteúdo do documento XML sem que seja percebido (SILVA, 2004).

Neste método de segurança, deverão ser definidas algumas restrições no conteúdo da mensagem para que todos os pontos finais da rede que envolvem o sistema distribuído sejam capazes de se integrar uns com os outros. Caso o sistema origem encripte ou certifique uma mensagem, ou alguns campos da mensagem, ao recebê-la, o sistema destino deverá

poder decriptar a mensagem ou validar o certificado. Um exemplo seria utilizar o padrão WS-Security, mas que não será o foco deste trabalho (SILVA, 2004).

Comparando a segurança em nível de transporte e de mensagens, nota-se que a segurança em nível de mensagens torna bastante seguro o Web Service pelo fato de existir segurança do início ao fim da transmissão do documento, ao contrário da segurança em nível de transporte, que é de ponto a ponto da rede. A única desvantagem que a camada de mensagem tem em relação a de camada de transporte é que a esta camada possui um desempenho pior, em relação a camada de transporte (CUNHA; SILVA, 2005).

7.3 SEGURANÇA COM XML

Esta seção é a parte principal do trabalho, onde serão discutidas as formas existentes de segurança para serem aplicadas em sistemas que utilizam Web Service. Conforme descrito anteriormente, alterando o XML, ou a entrada de valores para a consulta de um documento XML, pode-se visualizar e também alterar conteúdos sigilosos, podendo, desta forma, prejudicar um ou vários usuários do sistema (FRAGA et al, 2006).

Existem padrões específicos para a segurança em XML, como por exemplo a Criptografia XML a Assinatura XML e a Linguagem de Marcação para Atribuição de Segurança (SAML). A vantagem de lidar diretamente com estes padrões específicos para XML é que eles lêem e escrevem arquivos XML, então o resultado de uma encriptação ou assinatura poderá ser processado usando processadores comuns de XML. Na Figura 4 visualiza-se a segurança nas diferentes camadas do sistema distribuído.

Figura 4: Segurança em diferentes camadas
Fonte: MELLO (2004)

Abaixo serão descritos de forma detalhada cada padrão de segurança existente para arquivos XML.

7.3.1 Linguagem de Marcação para Atribuição de Segurança (SAML)

Este padrão geral de segurança SAML para XML é mantida pela *Organization for the Advancement of Structured Information* Standards (OASIS) e se trata de uma linguagem que é baseada no XML para o gerenciamento e a troca de informações entre diversos sistemas (JOSUTTIS, 2008).

Qualquer Web Service seguro necessita reconhecer o usuário que deseja trocar informações entre os sistemas, ou também se um determinado usuário tem permissão de utilizar o Web Service ou algum serviço pertencente a ele, que no caso seria a autenticação e a autorização (FRAGA et al, 2006).

Existe o problema de que algum usuário necessite acessar mais de um Web Service para concluir a sua operação. No caso poderia existir um Web Service para mostrar os itens de uma loja na tela, outro para armazenar os itens no carrinho e outro para concluir a compra. Neste caso, o usuário poderá ter um nome de usuário para um Web Service, já para

outro Web Service seria outro nome de usuário, e para o terceiro Web Service poderia ser outro usuário diferente. Neste caso seria bastante complicado o acesso em cada Web Service, pois os usuários, e também as senhas podem ser diferentes um do outro (STARLIN; NOVO, 2000).

O padrão SAML surgiu para resolver este tipo de problema. O objetivo deste padrão é fornecer um protocolo para que os usuários possam realizar confirmações e os Web Services possam autenticar (STARLIN; NOVO, 2000).

O SAML insere o XML de confirmação no corpo da mensagem SOAP, geralmente no cabeçalho SOAP. As atribuições inseridas servem como identificação e permissão para que os Web Services possam reconhecer o usuário que está utilizando o Web Service (STARLIN; NOVO, 2000).

7.3.2 XML-Signature

O que preocupa qualquer usuário de um sistema distribuído é saber se realmente quem mandou o documento XML é realmente quem o arquivo indica. Se o arquivo chega ao seu destino indicando que quem enviou o arquivo foi o usuário 'macarini', o Web Service precisa confirmar que foi este mesmo usuário que solicitou a resposta (JOSUTTIS, 2008).

Uma assinatura digital seria um conjunto de tags que tem o objetivo de identificar o usuário que criou a mensagem, evitar a modificação não autorizada do documento ou evitar que uma entidade não autorizada transmita informações pelo Web Service. Caso o sistema fornecedor ou o consumidor recebam um documento XML com informações alteradas, o

sistema deverá reconhecer esta alteração e reagir de acordo com a situação, neste caso negando o arquivo (STARLIN; NOVO, 2000).

Como a criptografia é a base do XML-Signature, um usuário necessita de um par de chaves, sendo uma pública e outra privada, que serão usadas para criptografar ou decriptografar o documento enviado ou recebido.

Este padrão também é mantido pela OASIS. O conjunto das tags XML do XML-Signature, têm o objetivo de conter informações suficientes para que um usuário seja verificado, utilizando-se de um par de chaves pública e privada para que possa transmitir os documentos XML pelo sistema distribuído (JOSUTTIS, 2008).

Este padrão permite que somente parte do documento seja assinado digitalmente. Isso é bastante importante porque existe a possibilidade de que uma mensagem SOAP possa adquirir mais informações ao passarem por diversas camadas de serviços. Caso o documento fosse criptografado inteiramente, com a adição de novas informações no documento, o tornaria inválido, no sentido criptográfico. Então como o usuário inicial assinou apenas parte do documento, uma nova adição de informações manteria a integridade do documento (STARLIN; NOVO, 2000).

Na Figura 5 é demonstrado um arquivo XML criptografado utilizando da XML-Signature.

```
[s01] <Signature Id="MyFirstSignature"
 xmlns="http://www.w3.org/2000/09/xmldsig#">
[s02]  <SignedInfo>
[s03]  <CanonicalizationMethod
 Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
[s04]  <SignatureMethod
 Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
[s05]  <Reference
 URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
[s06]   <Transforms>
[s07]    <Transform
Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-
20010315"/>
[s08]   </Transforms>
[s09]   <DigestMethod
Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
[s10]
<DigestValue>j61wx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
[s11]  </Reference>
[s12]  </SignedInfo>
[s13]  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
[s14]  <KeyInfo>
[s15a]   <KeyValue>
[s15b]    <DSAKeyValue>
[s15c]     <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
[s15d]    </DSAKeyValue>
[s15e]   </KeyValue>
[s16]  </KeyInfo>
[s17] </Signature>
```

Figura 5: Arquivo XML criptografado
Fonte: Silva (2004)

Verificando na Figura 5 o elemento SendInfo representa as informações da assinatura, como por exemplo o método de assinatura utilizado e o valor da assinatura. O elemento KeyInfo representa a chave a ser utilizada para ter a assinatura (SILVA, 2004).

Quando o usuário final recebe o documento assinado, é aplicado a chave pública. Caso o algoritmo indicar que a chave combina com a chave incluída, o usuário final poderá ter certeza que o documento recebido não foi modificado durante a transição, caso contrário, o documento é falso ou foi modificado (SILVA, 2004).

7.3.3 XML Key Management Specification

Utilizar as chaves públicas e privadas podem apresentar o seguinte problema: Primeiramente, como montar um XML necessário para checar o conteúdo das tags KeyInfo das assinaturas digitais? O segundo problema, seria como conseguir se comunicar com as autoridades de certificação de forma automática? (POTTS; KOPACK, 2003).

O XML Key Management Specification é dividido em duas partes: O XML Key Information Service Specification (X-KISS) e o XML Key Registration Service Specification (X-KRSS). O X-KISS consiste em um protocolo que permite que uma aplicação controle a validação da chave que foi transmitida para um outro Web Service. Já o X-KRSS é um protocolo que autoriza os usuários e os Web Services se comunicarem com uma entidade de certificação, para relacionar a uma chave pública nomes ou atributos (POTTS; KOPACK, 2003).

Utilizando estes dois protocolos, a comunicação utilizando chaves se torna mais eficiente e transparente, resolvendo o problema citado acima.

7.3.4 XML Encryption

Neste padrão de segurança com XML, o documento XML recebe algum tipo de codificação antes de ser enviado. O sistema que receber este arquivo enviado basta reconhecer a técnica de como reverter a codificação para poder visualizar o documento (POTTS; KOPACK, 2003).

A criptografia que o arquivo receber poderá ser tanto simétrica como assimétrica. A criptografia simétrica utiliza a mesma chave para encriptar e decriptar o documento. Já o assimétrico utiliza uma chave para encriptar e outra chave diferente, mas matematicamente relacionada para decriptar o documento. A criptografia assimétrica é mais segura, porém a criptografia simétrica é mais rápida (POTTS; KOPACK, 2003).

O XML Encryption é um padrão proposto e avaliado pelo W3C. Este padrão é bastante simples. As informações do documento são criptografadas e colocadas entre as tags *EncrypteData.* Somente parte do documento é criptografado (POTTS; KOPACK, 2003).

Estes padrões citados acima são os utilizados para que exista a segurança eficiente em documentos XML, evitando que possíveis ataques possam ocorrer na camada de XML, considerando que está sendo aplicado em um Web Service, tanto no lado do consumidor quanto na do fornecedor de serviços.

8 SEGURANÇA NO SISTEMA VISIONSCAN APLICADO EM DOCUMENTOS XML

Conforme descrito anteriormente, o sistema VisionScan faz o reconhecimento de imagem de código de barras. Utilizando um celular para registrar a imagem, a informação é enviada a um servidor para que o reconhecimento seja feito.

O sistema VisionScan divide-se na parte cliente, que seria o dispositivo móvel, e servidor, onde o reconhecimento da imagem é feito e o resultado enviado ao cliente.

Toda a troca de informações entre cliente e servidor é feito por meio de arquivos XML.

8.1 INTERFACE CLIENTE

Desenvolvida sobre a plataforma *Java 2 Micro Edition* (J2ME), sua tarefa principal é enviar a imagem de código de barras ao servidor e também receber o resultado do processamento da imagem. O padrão de codificação de código de barras utilizado pelo VisonScan é o EAN13, não merecendo mais detalhes neste trabalho (CONTE, 2006).

Como a interface cliente é um dispositivo móvel, como um celular por exemplo, uma outra tarefa que este dispositivo pode exercer é a captura da imagem, visto que hoje em dia é comum os celulares terem câmeras fotográficas embutidas. Para efetuar o teste do software, foi utilizado um emulador de celular, onde o sistema na parte cliente é executado. Na figura 6 é demonstrado o emulador com uma imagem de código de barras capturado.

Figura 6. Captura da imagem no celular

Com a imagem capturada, é feito o envio desta informação para o servidor para o reconhecimento da imagem. Após o reconhecimento da imagem, a parte cliente (celular) recebe o resultado, conforme mostrado na figura 7.

Figura 7. Resposta do servidor

8.2 INTERFACE SERVIDOR

A parte servidora do sistema VisionScan basicamente recebe a imagem enviada pela aplicação cliente, efetua o processamento da imagem e envia o resultado deste processamento ao cliente (CONTE, 2006).

Foi utilizada para a implementação desta interface a tecnologia J2EE. Esta plataforma possibilita a criação de componentes multicamadas distribuídos, a transferência de

dados utilizando XML, um modelo de segurança unificado e um flexível controle transacional (SUN MICROSYSTEMS, 2009).

A biblioteca Java utilizada para a manipulação de imagens no servidor é a *Java Advanced Imaging* (JAI), que permite o processamento da imagem com uma boa performance, onde o mesmo possui algumas funcionalidades específicas para este fim (CONTE, 2006)..

8.3 POSSÍVEIS ATAQUES NO XML DO SISTEMA VISIONSCAN

Alguns do ataques em arquivos XML existentes em *Web Services* são o Anexo SOAP, Injeção *Xpath* e Bomba XML.

O sistema VisionScan também poderá sofrer estes ataques, e para que o sistema fique menos vulnerável é necessário que sejam implementados métodos de segurança.

8.3.1 Anexo SOAP

Conforme já descrito no item 6.2.3, em um *Web Service* também poderá existir envio de anexo, assim como em um *email*.

No sistema VisionScan, existe o envio da imagem do celular para o servidor. A imagem é enviada ao servidor como um conjunto de bytes.

É possível que vírus, *worms*, ou qualquer outro tipo de código malicioso esteja embutido na imagem. Quando a imagem é enviada ao servidor, este código malicioso poderá ser ativado, danificando, de alguma forma, o servidor. Segue abaixo alguns vírus de imagem (SYMANTEC, 2009):

a) **JPEG *of Death*** – O computador é infectado com cavalo de tróia, podendo ser controlado remotamente;

b) **Perrum** - Modifica o registro do sistema operacional Windows e também aumenta em 11 Kb o tamanho dos arquivos de extensão jpg existentes no diretório corrente do arquivo originalmente contaminado;

c) **Bloodhound.Exploit.13** - Também faz com que seja possível um *hacker* assumir o controle da máquina. Os programas infectados por este cavalo de tróia não são possíveis de serem executados.

Estes são alguns vírus existentes que utilizam da imagem para invadir e ser executado no computador.

Para que os computadores estejam protegidos contra este tipo de ataque, é de extrema importância que tenha instalado na máquina um bom programa de anti-vírus, e que este esteja sempre atualizado. Assim como o anti-vírus, o sistema operacional utilizado no servidor também deverá estar sempre atualizado, evitando que existam possíveis falhas no sistema, diminuindo o risco de ataque ao sistema.

O anti-vírus poderá estar rodando na própria máquina servidora ou em alguma máquina dedicada a tarefa de evitar vírus no ambiente computacional do servidor (*appliance*).

Para o lado servidor do sistema VisionScan não é diferente. Para evitar ataque via Anexo SOAP, deve-se seguir as mesmas recomendações descritas anteriormente.

Quando o conjunto de bytes enviado pelo cliente chega ao servidor, o anti-vírus faz uma varredura no código recebido, detectando possível ameaça ao servidor e impedindo o ataque.

8.3.2 Injeção Xpath

Assim como já descrito no item 6.2.2, a injeção Xpath seria uma técnica utilizada para explorar as vulnerabilidades existentes em consultas feitas em bases de dados XML.

Pode-se fazer um teste utilizando a base de dados fictício criado, de nome senhas.xml, anexo neste trabalho. Deve-se fazer o download da ferramenta de consulta Xpath de nome Xpather, que seria um *Add-on* para o navegador Firefox. O caminho para *download* é <https://addons.mozilla.org/pt-BR/firefox/addon/1192>.

Após esta ferramenta ser instalada, abre-se a base de dados de nome senhas.xml com o Firefox. As instruções de busca são inseridas no campo Xpath. Já os resultados da busca são demonstrados no campo *Matching Nodes*.

Caso o desejo seja de verificar se na base de dados está cadastrado o usuário alex@bol.com.br com a senha yuiopx, o comando a ser executado é o seguinte: //senha/usuario[email/text()='alex@bol.com.br' and senha/text()='yuiopx'].

Caso seja demonstrada alguma informação no campo *Matching Nodes*, é porque a sentença é verdadeira, caso contrário é falsa.

Um usuário mal intencionado poderá se aproveitar da vulnerabilidade existente na consulta Xpath para buscar informações confidenciais de outros usuários. Nesta mesma base

de dados, poderemos explorar esta vulnerabilidade executando o seguinte comando Xpath: //senha/usuario[email/text()=" or "=" or 1=1 and senha/text()="].

Com o comando citado acima, é mostrado no campo *Matching Nodes*, todos os usuários e senhas existentes na base de dados. Caso for solicitado que o resultado seja do tipo String, será demonstrado o primeiro usuário e senha existente na base de dados. Para definir que o resultado será do tipo String, basta digitar a consulta Xpath dentro da função string(), como por exemplo string(//senha/usuario[email/text()=" or "=" or 1=1 and senha/text()="]).

Pode ser visto o ataque na prática, utilizando-se dos dois aplicativos disponíveis também neste trabalho. Estes aplicativos criados em java, possuem o nome de Insegura e ConsultaSegura, ambos com a extensão de arquivo .class.

Como o próprio nome já sugere, o aplicativo de nome Insegura, possui uma forma de consulta ao banco de dados (senhas.xml) sem tratamento algum, sendo bastante vulnerável ao ataque de injeção Xpath.

Um teste semelhante ao descrito anteriormente poderá ser feito tanto no aplicativo Insegura como no ConsultaSegura.

Abrindo o aplicativo Insegura, é solicitado um nome de usuário e posteriormente a senha. Caso o usuário seja encontrado e a senha for verdadeira, é mostrada uma janela com o usuário e senha do usuário, caso contrário o aplicativo é fechado.

Como neste aplicativo não existe tratamento algum de segurança, quando é solicitado o nome de usuário, ao invés de ser digitado o nome do usuário, digita-se a expressão ‘ or 1=1 or ‘’=’ (inclui-se as aspas simples) e na senha pode ser digitada qualquer sequência de caracteres. O resultado desta consulta será o primeiro usuário e senha existente na base de dados em XML. O mesmo resultado poderá ser obtido digitando esta expressão no campo de senha e digitando qualquer conjunto de caracteres no campo destinado ao nome do usuário.

Para que o ataque de injeção Xpath possa ser evitado, deve-se fazer um tratamento de caracteres nos campos destinados a digitação de usuário e senha.

No caso desta base de dados fictícia, o nome do usuário será sempre um endereço de *email*, e a senha só poderá ter letras e números. Então no campo usuário do aplicativo de consulta a base de dados, obrigatoriamente deverá existir um *email* válido. Já no campo destinado a digitação de senha, somente serão permitidas letras e números. Com estas regras já definidas, poderá ser implementado métodos de proteção contra injeção Xpath. Para fazer estas validações de usuário e senha, foram utilizadas expressões regulares.

Expressões regulares são uma composição de símbolos e caracteres que juntamente formam uma expressão. Esta expressão, após interpretada, indica se o conjunto de caracteres a ser analisado é válido ou não (JARGAS, 2001).

A expressão regular utilizada para a validação de email é ^[-_a-z0-9]+(\\.[-_a-z0-9]+)*\\@([-a-z0-9]+\\.)*([a-z]{2,4})$ e a expressão regular para a verificação do campo de digitação de senha é [a-zA-Z0-9]+.

Na validação que verifica o campo de usuário, o valor digitado no campo deverá iniciar com hífen, *underline*, letra minúscula, letra maiúscula ou números. Não poderá ser usado o sinal de ponto no início, mas poderá ter o ponto do segundo caractere em diante, antes do arroba. Obrigatoriamente deverá existir um arroba após os caracteres citados anteriormente, não podendo ser o primeiro caractere digitado. Após o arroba poderá ser digitado mais uma sequência de caracteres, podendo ser hífen, letra minúscula e número. Após estas sequências deverá existir um caractere de ponto e posteriormente deverá existir de 2 a 4 caracteres digitados, devendo ser letra minúscula.

Já na validação feita para o campo de senha, basicamente a expressão regular indica que apenas poderá ser utilizada letras minúsculas, letras maiúsculas ou números.

Analisando o sistema VisionScan, verifica-se que existe uma base de dados para armazenamento das informações para relacionar o valor do código de barras com o produto correspondente. Mas o formato de arquivo desta base de dados é texto, não sendo possível efetuar consultas Xpath nesta base.

Em projetos futuros poderá ser proposta uma base de dados XML para o armazenamento das informações. Desta forma poderá ser criado um método para tratamento de informações recebidas da parte cliente do sistema, como por exemplo, a verificação de usuário e senha.

A classe java desenvolvida que possui o método de tratamento das informações para evitar o ataque de Injeção XPath possui o nome de Tratamento e faz parte do pacote br.com.tfmacarini.visionscan.sec. Poderá ser utilizado logo após o recebimento das informações da parte cliente do sistema VisionScan. Já o método que faz a checagem das informações recebidas possui o nome de verifica.

A implementação deste método foi feita na parte servidora do sistema, pois é o local onde deverá existir uma base de dados XML com os usuários e suas respectivas senhas. Caso esta base de dados seja feita na parte cliente do sistema, as informações contidas nesta base poderão ser visualizadas por qualquer pessoa, tornando o sistema vulnerável a nível físico, não existindo controle de acesso sobre o arquivo.

Este método recebe, dois parâmetros: usuario, do tipo String e outro parâmetro de nome senha, também do tipo String. Para fazer a validação de usuário e senha foram utilizadas expressões regulares, igualmente escritas conforme sistema ConsultaSegura, citada nos parágrafos anteriores. O resultado deste método será um valor booleano, sendo verdadeiro quando a expressão é válida ou falsa quando uma das expressões está incorreta.

Futuramente, quando este método ser utilizado no sistema VisionScan para validação de informações digitadas pelo usuário, as checagens de valores recebidos da parte

cliente do sistema poderão ser adaptadas pelo tipo de informação alterando as expressões regulares existentes.

8.3.3 Bomba XML

Este tipo de ataque é feito utilizando DTDs de arquivos XML. Assim como descrito no item 6.2.1, ao ser processado, o arquivo XML infectado faz com que o servidor sofra um ataque de negação de serviço, onde há um consumo de todos os recursos da máquina que está processando as informações, no caso do sistema VisionScan, a parte servidora do sistema.

Mais uma vez analisando o sistema VisionScan, foi detectado que não existe DTD incorporado nos arquivos XML para fazer o reconhecimento das tags. Ao invés de DTD, é utilizado o XML *Schema*.

O XML *Schema* permite que sejam realizadas regras de validação para as *tags* XML. São fornecidos meios para a definição da estrutura, conteúdo e semântica nos documentos XML. Se tornou uma recomendação da W3C em 2001 (W3C).

Estas regras deverão estar contidas em um arquivo de extensão .xsd. Neste arquivo poderão estar declarados os elementos do XML, utilizando a *tag* element, também os atributos, utilizando a *tag* attribute e também poderá ser especificado os tipos de dados de cada elemento ou atributo, utilizando a *tag* type.

No XML *Schema*, a entidade é tratada como um tipo de um atributo, onde na definição da regra do atributo é recebido o tipo como entidade, conforme abaixo:

<xsd: attribute name=”location” type=”xsd:ENTITIES”/>

Nas pesquisas efetuadas, não houveram citações sobre o ataque de Bomba XML e também não foi possível simular o ataque utilizando XML *Schema*. Desta forma, o sistema VisionScan, até o atual momento, estaria livre deste tipo de ataque.

Fazendo uma observação sobre a existência do DTD no sistema VisionScan, existe apenas um arquivo XML que utiliza DTD, que é o web.xml, encontrado no arquivo visionscan.war, na pasta WEB-INF. Este DTD é padrão nos arquivos war, que seria o *Servlet 2.3 Web Application*, DTD este desenvolvido pela Sun Microsystems.

Embora o termo *Web Service* seja novo, existem *Firewalls* XML que fazem o tratamento dos arquivos XML antes de chegar ao servidor. Caso o documento XML possua alguma ameaça, o *firewall* XML faz o tratamento no arquivo XML recebido, evitando que o ataque ocorra.

Uma das empresas que desenvolve este tipo de solução é Vordel. Esta empresa possui um *firewall* XML para ser utilizado em servidores. A Figura 8 mostra o funcionamento deste *firewall*.

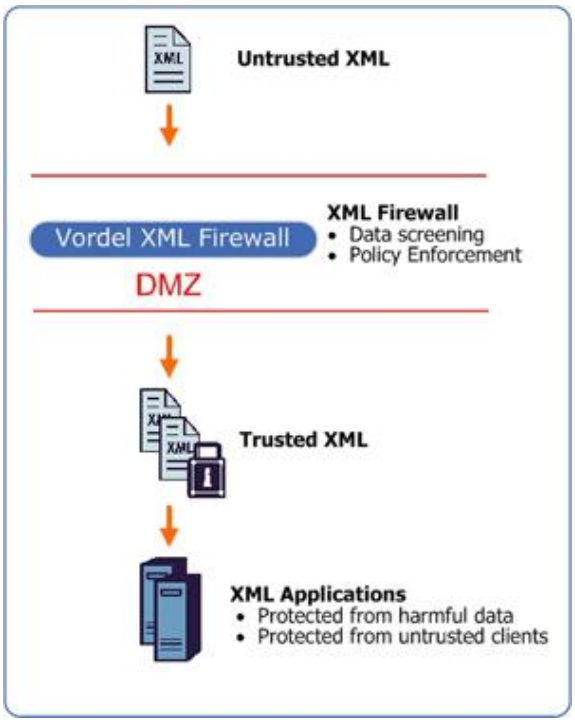

Figura 8: Firewall XML
Fonte: Vordel (2009)

Outras empresas que possuem este tipo de *software* são, por exmplo, a Xtradyne, Layer7Tech, F5 e ModSecurity. Todas estas empresas possuem o *firewall* XML proprietários, exceto a ModSecurity que possui a Licensa Pública Geral (GPL).

Vistas as informações acima sobre a Bomba XML, seria importante então que no servidor do sistema VisionScan exista instalado um *firewall* XML para uma maior proteção do sistema, evitando possíveis ataques.

8.4 METODOLOGIA

O desenvolvimento e análise dos métodos de segurança a serem implantados no sistema VisionScan fundamentou-se metodologicamente pelas seguintes etapas: levantamento

bibliográfico, análise do sistema VisionScan e sua implementação, estudo sobre formas de segurança para sistemas baseados em *Web Service*, aplicação dos métodos de segurança no sistema VisionScan ou sua infra-estrutura contra ataques XML e a realização de testes.

8.5 IMPLEMENTAÇÃO E TESTES DOS MÉTODOS DE SEGURANÇA

Para proteger o sistema contra o ataque de injeção Xpath é necessário que as informações recebidas da parte cliente sejam tratadas, e para isso foi desenvolvido o método verifica, contido na classe Tratamento do pacote br.com.tfmacarini.visionscan.sec.

O desenvolvimento desta classe foi feita utilizando a tecnologia Java e do ambiente de desenvolvimento Eclipse 3.4.2, disponível para *download* em http://www.eclipse.org/downloads.

Procurou-se também desenvolver e analisar o sistema VisionScan com os sistemas mais atuais. Foi atualizado o servidor de aplicação JBoss para a versão 4.0.3, originalmente desenvolvido na versão 4.0.2, e a ferramenta de desenvolvimento J2ME foi atualizada para a versão 2.5.2_01, originalmente desenvolvida na versão 2.2, ambos disponíveis em http://sourceforge.net/project/showfiles.php?group_id=22866&package_id=16942&release_id=365509 e também http://java.sun.com/products/sjwtoolkit/download.html respectivamente.

Desenvolvida a classe para tratamento das informações recebidas do lado cliente do sistema, conforme item 8.3.2 deste trabalho, os testes foram realizados utilizando-se do sistema ConsultaSegura, criado especificamente para teste, visto que no sistema VisionScan este tipo de teste não seria possível pois a base de dados existente não é do tipo XML. Desta

forma, é viável então que seja implementado no sistema um método para tratamento das informações, caso futuramente seja disponibilizado uma base de dados XML.

Sendo o método de segurança implementado no VisionScan semelhante ao método do sistema Insegura, os testes foram feitos utilizando a base de dados de nome senhas.xml, que possuem informações de usuários e senhas, sendo estes fictícios.

Para ataque de Anexo SOAP com vírus, o mesmo pode ser combatido com um anti-vírus atualizado. Para a realização dos teste, seria necessário que as imagens dos códigos de barras estivessem infectados com algum vírus e este teste não foi possível ser feito por completo, por não poder ter sido encontrado uma imagem de código de barras infectado por vírus. O anti-vírus instalado no computador durante os estudo foi o Kaspersky Internet Security 7.0.1.325, na versão de avaliação.

O teste também não pode ser feito por completo para a bomba XML por não existir DTD na composição dos arquivos de execução no servidor. Foi possível simular o ataque executando o arquivo Bomba.xml, existente na pasta BombaXML. Este arquivo foi criado conforme instruções descritas no item 8.3.3. Ao ser executado utilizando o navegador Internet Explorer, o sistema utiliza toda a capacidade de processamento e também consome todo o recurso de memória, prejudicando o funcionamento do computador. Como o sistema VisionScan utiliza XML Schema para validação de arquivos XML, este tipo de ataque não ocorre.

O sistema operacional utilizado nos testes foi o Windows XP Professional com *Service Pack* 3 em um computador com processador AMD Turion 64 Mobile 2.2Ghz e 960 MB de memória RAM.

8.6 RESULTADOS OBTIDOS

ATAQUE	POSSÍVEL	SOLUÇÃO
Anexo SOAP	Sim	Instalação de anti-vírus no servidor e manter o sistema operacional sempre atualizado.
Bomba XML	Não	Se desejado, poderá ser instalado um *Firewall XML* no servidor.
Injeção XPath	Não	Mesmo não sendo possível o atauqe atualmente no VisionScan, foi desenvolvido um método de proteção contra este tipo de ataque.

Os resultados foram obtidos por meio da análise feita no sistema VisionScan e também pelos testes realizados no método de segurança implementado.

Foi possível identificar, durante a pesquisa realizada, que existem vários tipos de ataques possíveis em sistemas baseados em *Web Service*, não sendo diferente no sistema VisionScan.

No sistema VisionScan verificou-se que existia vulnerabilidade para o ataque em Anexo SOAP. Recomendou-se então a instalação de anti-vírus e atualização do sistema operacional. Para o ataque de injeção Xpath, foi concluído que na atual estrutura do sistema este ataque não poderá ocorrer, mas como prevenção foi desenvolvido um método para tratamento de dados, que poderá ser utilizado quando for criado uma base de dados XML no sistema. O ataque com bomba XML também não poderá ocorrer no sistema VisionScan devido a atual estrutura do sistema não possuir validações XML com DTD.

O estudo sobre estes ataques se tornou complexo pelo fato de ser um assunto

novo, e consequentemente existirem poucos títulos (livros, artigos, etc) que detalhem sobre isto.

9 CONCLUSÃO

A implementação de métodos de segurança em sistemas baseados em *Web Services* é fundamental para que o sistema mantenha sua integridade e também proteger o usuário contra roubo de informações confidenciais.

A sua aplicação se dá por meio de implementações de novas funcionalidades no sistema e também pela utilização de ferramentas na infra-estrutura do servidor.

O estudo se concentrou em vulnerabilidades existentes em processamentos de arquivos XML, como ataques de bomba XML, injeções Xpath e também anexos SOAP infectados.

Verificou-se nos testes que no sistema VisionScan alguns destes ataques não são possíveis de ocorrer, mas também foi descrito as formas de proteção para que o sistema se torne mais seguro, dependendo do tipo de ataque.

Os testes realizados para a prevenção do ataque de injeção Xpath foram satisfatórios, onde a solução a ser tomada contra este ataque é simples.

Na realização desta pesquisa concluiu-se que são inúmeros os tipos de ataques existentes, e que o sistema VisionScan poderá possuir mais métodos de proteção ao sistema contra estes ataques, assim como a estrutura de armazenamento de dados, onde atualmente é feito utilizando arquivos texto.

Como sugestão, poderá ser implementado no sistema VisionScan a prevenção do sistema contra outros ataques em *Web Services* como por exemplo o *Cross-Site Scripting*, URL Previsível e a Indexação de diretório. Também poderá ser criada uma base de dados XML para que as informações existentes no sistema sejam melhor armazenadas, existindo a facilidade nas consultas das informações nela contida com o XPath.

REFERÊNCIAS

ABINADER, Jorge Abílio; LINS, Rafael Dueire. **Web services em Java**. Rio de Janeiro: Sociedade Brasileira de Telecomunicações, 2006. 288 p.

ALMEIDA, Maurício Barcellos. **Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares.** Disponível em: <http://www.scielo.br/pdf/ci/v31n2/12903.pdf>. Acesso em: 25 set. 2008.

ANDERSON, Richard. **Professional XML**. Rio de Janeiro: Ciência Moderna, 2001. 1266 p.

BROWN, Allen; HAAS, Hugo. **Web Services Glossary.** Disponível em: <http://www.w3.org/TR/ws-gloss/>. Acesso em: 25 set. 2008.

CONTE, Márcio José. **Reconhecimento de Código de Barras em Imagens Digitais por meio do Modelo Perceptron Multicamadas de Redes Neurais Artificiais**. 2006. 72 p. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade do Extremo Sul Catarinense, Criciúma.

CONTE, Márcio José et al. **Reconhecimento de Padrões de Código de Barras a partir da Utilização de Redes Neurais**. In: VII SIMPÓSIO DE INFORMÁTICA DA REGIÃO CENTRO/RS (SIRC 2008), 2008, Santa Maria. Anais do VII Simpósio de Informática da Região Centro/RS (SIRC 2008)

CHAPPEL, David A.. **Enterprise Service Bus.** Sebastopol: O'reilly Media, 2004. 239 p.

CUNHA, José A.; SILVA, Reginaldo F.. **Arquitetura de Segurança em Aplicações Baseadas em Web Services.** Disponível em: <http://www2.ifrn.edu.br/ojs/index.php/HOLOS/article/view/77/82>. Acesso em: 25 out. 2008.

FRAGA, Joni da Silva ; MELLO, Emerson Ribeiro de ; WANGHAM, Michelle S. ; CAMARGO, Edson . **Segurança em Serviços Web**. In: Lau C. Lung. (Org.). Minicursos do SBSEG 2006. : , 2006, v. , p. 1-48.

JARGAS, Aurélio Marinho. **Expressões Regulares:** Guia de Consulta Rápida. São Paulo: Novatec, 2001.

JOSUTTIS, Nicolai M. **SOA na Prática**: A arte da Modelagem de Sistemas Distribuídos. Rio de Janeiro: Alta Books, 2008. 259 p.

KLEIN, Amit. **Blind XPath Injection.** Disponível em: <http://www.packetstormsecurity.org/papers/bypass/Blind_XPath_Injection_20040518.pdf>. Acesso em: 02 ago. 2008.

LANDWEHR, Carl E.. Computer security. **International Journal Of Information Security,** Mclean, p. 3-13. 27 jul. 2001.

O'NEILL, Mark et al. **Web Services Security**. 1.ed. Berkeley: McGraw-Hill, 2003. 312p.

PITTS-MOULTIS, Natanya. XML black book. São Paulo: Makron Books, 2000. 627 p.

POTTS, Stephen; KOPACK, Mike. **Aprenda Web services em 24 horas**. Rio de Janeiro: Campus, 2003. 367 p.

SAMPAIO, C. **SOA e Web Services em Java**. Rio de Janeiro: Editora Brasport, 2006. 168 p.

SILVA, Jandson Almeida da. **Segurança em Web Services.** Aracaju: 2004.

STARLIN, Gorki; NOVO, Rafael. **Segurança contra hacker**. Rio de Janeiro: Book Express, 2000. 342 p.

SYMANTEC. **Threat Explorer.** Disponível em: <http://www.symantec.com/norton/security_response/threatexplorer/azlisting.jsp>. Acesso em: 20 abr. 2009.

W3C. **Extensible Markup Language (XML):** activity statement. Disponível em: <http://www.w3.org/XML/Activity>. Acesso em: 25 set. 2008.

W3C. **Extensible Markup Language (XML):** W3C recommendation 6. 2. ed. versão 1.0. Oct. 2000. Disponível em: <http://www.w3.org/TR/REC-xml/>. Acesso em: 24 set. 2008.

W3C. **XML in 10 points.** Disponível em: <http://www.w3.org/XML/1999/XML-in-10-points-translations.htm8>. Acesso em: 20 ago. 2008.

VORDEL. **Vordel XML Firewall:** Threat protection for XML Applications. Disponível em: <http://www.vordel.com/products/vx_firewall/>. Acesso em: 10 maio 2009.

BIBLIOGRAFIA COMPLEMENTAR

BEECH, David et al. **XML Schema Part 1**: Structures Second Edition. Disponível em: <http://www.w3.org/TR/xmlschema-1/>. Acesso em: 25 abr. 2009.

BATES, Bert; SIERRA, Kathy. **Use a Cabeça! Java.** 2. ed. Rio de Janeiro: Alta Books, 2007. 470 p.

DEITEL, H.M.; DEITEL, P. J. **Java**: como programar. 6.ed. São Paulo: Pearson Education do Brasil, 2005. 1110 p.

MCGRATH, Sean. **XML**: aplicações práticas : como desenvolver aplicações de comércio eletrônico. Rio de Janeiro: Ed. Campus, 1999. 368 p.

MSDN (Org.). **WebServices:** Perguntas e Respostas. Disponível em: <http://msdn.microsoft.com/pt-br/library/cc564894.aspx>. Acesso em: 02 abr. 2009.

SUN MICROSYSTEMS. **J2EE DTDs.** Disponível em: <http://java.sun.com/dtd/>. Acesso em: 28 abr. 2009.

SUN MICROSYSTEMS. **The Java Web Services Tutorial:** For Java Web Services Developer's Pack, v1.6. Disponível em: <http://java.sun.com/webservices/docs/1.6/tutorial/doc/>. Acesso em: 16 abr. 2009.

W3SCHOOLS. **XML Schema Tutorial.** Disponível em: <http://www.w3schools.com/Schema/default.asp>. Acesso em: 25 abr. 2009.

TIAGO FARIAS MACARINI

Análise e Implementação de Métodos de Segurança no Sistema VisionScan

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Prof. MSc. Rogério Antônio Casagrande
Coordenador Adjunto do Curso de Ciência da Computação

Banca Examinadora:

Prof. MSc. Rogério Antônio Casagrande
Orientador

Prof. Fabrício Giordani (UNESC)

Esp. Fabrizio Colombo Machado (Depto de TI - UNESC)