UNIVERSIDADE FEDERAL DO MARANHÃO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA DE ELETRICIDADE

WAGNER ELVIO DE LOIOLA COSTA

SISTEMA DE DETECÇÃO DE INTRUSÃO EM REDES BASEADO EM SOA (NIDS-SOA) PARA SUPORTAR A INTEROPERABILIDADE ENTRE IDS'S: APLICAÇÃO AO NIDIA

São Luís
2012

WAGNER ELVIO DE LOIOLA COSTA

SISTEMA DE DETECÇÃO DE INTRUSÃO EM REDES BASEADO EM SOA (NIDS-SOA) PARA SUPORTAR A INTEROPERABILIDADE ENTRE IDS'S: APLICAÇÃO AO NIDIA

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia de Eletricidade da UFMA, para a obtenção do título de mestre em Engenharia de Eletricidade, Área de Concentração Ciência da Computação.

Orientador: Prof. Ph.D Zair Abdelouahab
Co-Orientador: Prof. Dr. Denivaldo Lopes

Costa, Wagner Elvio de Loiola.

Sistema de detecção de intrusão em redes baseado em soa (NIDS-SOA) para suportar a interoperabilidade entre IDS'S:aplicação ao NIDIA/Wagner Elvio de Loiola Costa.-2012.
132f.
Impresso por computador (Fotocópia).
Orientador: Zair Abdelouahab
Co-Orientador: Denivaldo Lopes
Dissertação (Mestrado)-Universidade Federal do Maranhão, Programa de Pós-Graduação em Engenharia de Eletricidade, 2012.
1. Redes de Computadores - Barreira de segurança 2. IDS 3. SOA 4. Web - Composição de serviços I. Título
CDU 004.72.056.52

Wagner Elvio de Loiola Costa

SISTEMA DE DETECÇÃO DE INTRUSÃO EM REDES BASEADO EM SOA (NIDS-SOA) PARA SUPORTAR A INTEROPERABILIDADE ENTRE IDS'S: APLICAÇÃO AO NIDIA

Aprovada em 10 de Outubro de 2012.

Prof. Zair Abdelouahab, PhD
(Orientador)

Prof. Denivaldo Cicero Pavão Lopes, Dr.
(Co-Orientador)

Profa. Karla Donato Fook. Dra.
(Membro da Banca Examinadora)

Prof. Sofiane Labidi, Dr.
(Membro da Banca Examinadora)

Dedico este trabalho à minha mãe Rosa, a minha esposa Maria Virgínia, aos meus filhos José e Pedro e a minha Família.

Agradecimentos

À DEUS, pela minha vida e pela sua infinita presença em todas as minhas realizações.
À minha adorável e querida mãe Rosa de Loiola Costa e a minha família.
A minha esposa Maria Virgínia e aos meus filhos José e Pedro.
Ao meus orientadores, Prof. Ph.D. Zair Abdelouahab, pelo apoio, incentivo e confiança depositados em mim e ao Prof. Dr. Denivaldo Lopes, pela ajuda, apoio, colaboração e incentivo na realização deste trabalho.
A Profa. Dra. Maria da Guia pelo incentivo na finalização deste mestrado.
A Jéssica Bassani pelo auxílio na parte teórica desse trabalho.
A Wesley pelas dicas na elaboração deste trabalho digitado em LaTeX.
E a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.
A chefia do Departamento de Engenharia de Eletricidade por colaborar na realização deste Mestrado.
Ao Programa de Pós-Graduação em Engenharia de Eletricidade por esta oportunidade em realizar o Mestrado.

A vida é uma peça de teatro que não permite ensaios. Por isso, cante, chore, dance, ria e viva intensamente, antes que a cortina se feche e a peça termine sem aplausos.

Charles Chaplin

Resumo

Os sistemas antivírus e *firewall* são sistemas de proteção que visam impedir a execução de ações maléficas na rede, constituindo, portanto, uma barreira aos invasores (*e.g.vírus, worms e hackers*). Entretanto, não há como garantir uma proteção total da rede e dos computadores, podendo ocorrer invasões através da exploração de vulnerabilidades, já conhecidas, que permitem a execução de programas remotamente, a alteração de privilégios dentro do sistema e a divulgação de informações importantes. Neste caso, o Sistema de Detecção de Intrusão-IDS(*Intrusion Detection System*) permite a detecção de intrusões e a consequente notificação ao administrador da rede ou, em conjunto com o *firewall*, bloqueia a porta utilizada na invasão ou o endereço IP do atacante. Um fator importante para a detecção de intrusões é a qualidade da base de assinaturas. Entretanto os sistemas IDS são sistemas isolados e a interoperabilidade entre IDS de fornecedores diferentes é complexa e de difícil implementação. Os sistemas IDS existentes na literatura, incluindo o IDS-NIDIA (*Instrusion Detection System- Network Intrusion Detection System based on Intelligent Agents*), são sistemas isolados, não são facilmente reutilizados. Geralmente, estes se comunicam utilizando diferentes protocolos e são criados com paradigmas de programação diferentes. É proposta uma arquitetura baseado na filosofia SOA (*Service Oriented Architecture*) para suportar a interoperabilidade entre sistemas IDS. O sistema IDS-NIDIA será adaptado e estendido de acordo com esta filosofia SOA, contendo camadas de serviços web com o propósito de oferecer uma composição de serviço estática entre as camadas de aplicação e o reuso de informações com outros IDS's.

Palavras-chave: IDS, SOA(Arquitetura Orientada a Serviço), Composição de serviços *web*.

Abstract

The antivirus system and firewall are protection systems designed to prevent malicious work in the network, thus constituting a barrier to invaders (e.g. vírus, worms e hackers). However, there is no guarantee a full protection to network and computers, invasions can occur by exploiting vulnerabilities, known, and allow running programs remotely, changing privileges within the system and the dissemination of important information. In this case, Intrusion Detection System IDS (Intrusion Detection System) allows the detection of intrusions and subsequent notification to the network administrator or, in conjunction with the firewall blocks the port used in the invasion or the IP address of the attacker. An important factor for the intrusion detection is the quality of subscriber base. However IDS systems are isolated systems and the interoperability among different vendors' IDS is complex and difficult to implement. Existing IDS systems in the literature, including the IDS NIDIA (Instrusion-Detection System Network Intrusion Detection System based on Intelligent Agents) are isolated systems, are not easily reused. Generally, they communicate using different protocols and are designed with different programming paradigms. In this work it is proposed an architecture based on the philosophy SOA (Service Oriented Architecture) to support interoperability between IDS systems. The IDS-NIDIA will be adapted and extended according to the SOA philosophy, containing layers of web services in order to provide a static service composition between the layers of the application and reuse of information with other IDS's.

Keywords: IDS, SOA(Service Oriented Architecture), Web Services Composition.

Sumário

Lista de Abreviaturas **ix**

Lista de Figuras **x**

Lista de Tabelas **xii**

1 Introdução **1**
1.1 Definição do problema 2
1.2 Solução proposta: *Framework NIDS-SOA* 3
1.3 Metodologia utilizada 4
1.4 Objetivos 4
1.5 Contribuições 5
1.6 Organização do Trabalho 5

2 Fundamentos Teóricos **6**
2.1 Sistema de Detecção de Intrusão - Breve histórico 6
2.2 Descrição de um IDS 7
2.3 Metodologia de detecção de um IDS 8
2.3.1 Método de detecção baseado em Anomalia 9
2.3.2 Método de detecção baseado em Assinatura 9
2.4 Tipos de IDS 9
2.4.1 IDS na Rede - NIDS 10
2.4.2 IDS em *Host* 11
2.5 O *Framework* CIDF 12
2.6 O *Framework* IDS-NIDIA 13
2.6.1 Descrição do IDS-NIDIA 14
2.6.2 IDS-NIDIA-Trabalhos Relacionados 16
2.7 Serviços *web* 16
2.7.1 *WSDL* 17
2.7.2 *UDDI* 18
2.7.3 *SOAP* 19
2.8 XML - (*Extensible Markup Language*) 20

2.9 Arquitetura Orientada a Serviços (SOA) . . . 21
2.10 Composição de Serviços . . . 22
2.11 Síntese . . . 23

3 Arquitetura Orientada a Serviço 24
3.1 Introdução . . . 24
3.2 Definição de SOA . . . 26
3.3 Modelagem de processos em SOA . . . 27
3.4 Fundamentação teórica . . . 28
3.4.1 O que é Modelagem Orientada a Serviço? . . . 28
3.4.2 Visões das partes interessadas sobre a modelagem . . . 28
3.4.3 Disciplinas de Modelagem de Orientada a Serviço . . . 30
3.4.4 Estrutura de Modelagem Orientada a Serviço . . . 34
3.5 Gerenciamento de processos de negócios . . . 34
3.5.1 Definição de gerenciamento de processo de negócio . . . 35
3.6 A Linguagem WS-BPEL . . . 38
3.6.1 Atividades em um processo WS-BPEL . . . 38
3.6.2 Código de um processo BPEL . . . 40
3.7 Ciclo de vida do modelo orientado a serviço . . . 40
3.8 Notação dos diagramas de modelagem adotada . . . 43
3.8.1 Diagrama de transação circular . . . 44
3.9 Síntese . . . 45

4 *Framework:* NIDS-SOA 46
4.1 Introdução . . . 46
4.2 Infraestrutura de rede com o NIDS-SOA . . . 47
4.3 Objetivo do *Framework* NIDS-SOA . . . 47
4.4 Concepção do NIDS-SOA . . . 48
4.5 Projeto do *Framework* NIDS-SOA . . . 50
4.5.1 Adaptação do IDS-NIDIA ao *Framework* NIDS-SOA . . . 51
4.5.2 Serviços *web* e serviços em SOA . . . 52
4.5.3 Serviços do *Framework* . . . 53
4.5.4 Diagrama de Sequência do *Framework* NIDS-SOA . . . 54
4.5.5 Diagrama de Classe do *Framework* NIDS-SOA . . . 55
4.6 Modelagem do *Framework* NIDS-SOA . . . 57
4.6.1 Definição dos serviços candidatos . . . 57
4.6.2 Descrição da implementação - tecnologias utilizadas . . . 58
4.6.3 Descrição da composição de serviços no NIDS-SOA . . . 58
4.6.4 Atividades de transição circular . . . 59
4.6.5 Composição de serviços . . . 61

4.6.6 Considerações sobre o *Framework* 62
4.7 Síntese 62

5 Implementação e Resultados **63**
5.1 Introdução 63
5.2 Composição de serviço 65
5.3 *Software* utilizados 65
5.4 Teste com o serviço *WsAnalyser* 66
5.4.1 Projeto BPEL dessa composição 67
5.4.2 Código BPEL 67
5.5 Teste com o serviço *WsAnalyser* Consulta IP 68
5.5.1 Projeto BPEL dessa composição 68
5.5.2 Resultados obtidos - Teste 01 70
5.5.3 Resultados obtidos - Teste 02 71
5.6 Composição de Serviços:*WsAnalyser Consulta IP e WsReaction* 72
5.6.1 Projeto BPEL dessa composição 72
5.7 Considerações sobre os testes 73

6 Conclusão **74**
6.1 Contribuições do trabalho 74
6.2 Sugestões para Pesquisas Futuras 76

A Código exemplo de processo BPEL **77**
A.1 Partes do código do processo BPEL 77
A.2 Código do processo BPEL 80

B Código WSDL **81**
B.1 Código do WSAnalyser 81
B.2 Código do WsSniffer 86
B.3 Código do WsReaction 89
B.4 Código BPEL 92

C Código BPEL dos testes realizados **94**
C.1 Código da composição dos serviços WsSniffer e WsFormater 94
C.2 Código do teste WsAnalyser 97
C.3 Código do teste WsReaction 101
C.4 Código do teste WsAnalyser Consulta IP 104

D Código BPEL dos testes realizados **108**
D.1 Log de acesso a porta 22 - IP 101.78.202.18 108
D.2 Log de acesso a porta 22 - IP 222.221.2.210 110

Referências Bibliográficas **113**

Lista de Abreviaturas

CERT	Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CIDF	Common Intrusion Detection Framework
CISL	Common Intrusion Specification Language
CPU	Central Processing Unit
DMZ	DeMilitarized Zone
DNS	Domain Name System
IDS	Instrusion Detection System
IDS-NIDIA	Instrusion Detection System-Network Intrusion Detection System based on Intelligent Agents
NSM	Network Security Monitor
SPAN	Switched Port Analyzer
SOA	Service Oriented Architecture
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
HTTP	Hypertext Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
SMTP	Simple Mail Transfer Protocol
FTP	File Transfer Protocol
CORBA	Common Object Request Broker Architecture
REST	Representation State Transfer
RPC	Remote Procedure Call
SOAP	Simple Object Access Protocol
SMA	System Monitoring Agent
SGML	Standard Generalized
HTML	HTML Hypertext Markup Language
W3C	World Wide Web Consortium

Lista de Figuras

1.1 Estatísticas segundo o site CERT.br[1] . . . 2
1.2 IDS-NIDIA com e sem camadas de serviços web . . . 4

2.1 Trabalho em conjunto do Firewall com o IDS[2] . . . 8
2.2 NIDS - IDS baseado na análise de rede . . . 11
2.3 HIDS - IDS baseado na análise de host . . . 12
2.4 O Framework CIDF[3] . . . 13
2.5 Arquitetura do IDS NIDIA - Modelo de Camadas[4] . . . 14
2.6 Modelo do serviço *web* . . . 17
2.7 Representação de um documento WSDL[5] . . . 18
2.8 Interações com o UDDI [6] . . . 19
2.9 Estrutura de uma mensagem SOAP [6] . . . 20
2.10 Camada de protocolos dos serviços web [7] . . . 20
2.11 *Arquitetura Orientada a Serviços [7]* . . . 22
2.12 Orquestração e Coreografia[8] . . . 23

3.1 Modelo triangular SOA[9] . . . 25
3.2 Hierarquia de arquitetura SOA[10] . . . 26
3.3 Arquitetura Orientada a Serviços[7] . . . 27
3.4 Perspectiva da Modelagem Orientada a Serviço[11] . . . 29
3.5 Metamorfose Orientada a Serviço[11] . . . 30
3.6 Disciplinas Orientada a Serviço[11] . . . 31
3.7 Processo de modelagem conceitual orientada a Serviço[11] . . . 31
3.8 Processo de modelagem de descoberta orientada a serviço[11] . . . 32
3.9 Processo de modelagem de integração de negócios orientadas a serviço [11] . . . 32
3.10 Processo de modelagem de design orientado a Serviços [11] . . . 33
3.11 Processo de modelagem de arquitetura de serviços[11] . . . 33
3.12 Estrutura de modelagem do serviço orientado[11] . . . 34
3.13 Visualização de um processo de negócios[12] . . . 35
3.14 Cenário do fluxo de atividades de um processo de negócios[12] . . . 36
3.15 Arquitetura Orientada a Serviços[12] . . . 37
3.16 Design usando o BPEL Eclipse . . . 39

3.17 Ciclo de vida de desenvolvimento SOA[10] 41
3.18 Ciclo de vida de desenvolvimento-Thomas Erl[10] 42
3.19 Representação dos serviços atômico e composto[11] 43
3.20 Representação da Transição Circular[11] 44
3.21 Representação do serviço composto de formatação de pacotes[11] 44

4.1 Infraestrutura de rede com IDS-NIDIA 47
4.2 Infraestrutura de rede com o NIDS-SOA 48
4.3 Funcionalidades do NIDS-SOA 48
4.4 Serviços básicos do NIDS-SOA[11] 49
4.5 Servico Composto de Formatação de Pacotes 50
4.6 Arquitetura do IDS-NIDIA - Modelo de Camadas 50
4.7 Adaptação do IDS-NIDIA ao NIDS-SOA - Acrescido de uma camada web . 51
4.8 IDS-NIDIA adaptado para Framework NIDS-SOA 52
4.9 Diagrama de Sequência do NIDS-SOA 54
4.10 Diagrama de Classe do NIDS-SOA 55
4.11 Relação dos serviços candidatos 57
4.12 Serviços básicos do NIDS-SOA 59
4.13 Diagrama de transição circular - Serviço Composto de formatação de pacotes[11] 60
4.14 Diagrama de transição circular do NIDS-SOA[11] 60
4.15 Modelagem do Framework NIDS-SOA no Active BPEL 61

5.1 Código fonte do IDS-NIDIA 63
5.2 Interface do IDS-NIDIA no NetBeans 64
5.3 Serviços básicos do NIDS-SOA 65
5.4 Uso do BPEL Eclipse para composição dos serviços 67
5.5 Uso do BPEL Eclipse para composição dos serviços - Consulta IP 68
5.6 Uso do Eclipse BPEL - Consulta IP 70
5.7 Envelope da mensagem SOAP 70
5.8 Resposta da consulta no console do BPEL Eclipse 70
5.9 Uso do Eclipse BPEL - Consulta IP 71
5.10 Envelope da mensagem SOAP 71
5.11 Resposta da consulta no console do Eclipse BPEL 71
5.12 Uso do BPEL Eclipse para composição dos serviços - Consulta IP e Reaction 73

Lista de Tabelas

3.1 Atividades primitivas básicas usadas em uma composição de serviço[8] 38
3.2 Atividades estruturadas básicas usadas em uma composição de serviço[8] 39

5.1 Regras utilizadas na análise dos pacotes capturados 66

Capítulo 1

Introdução

Os benefícios oferecidos aos usuários pelos sistemas computacionais em uma rede de computadores são vastos, estes benefícios vão desde um simples envio de uma mensagem eletrônica, passando pela aquisição de produtos pela rede mundial de computadores com o uso do comércio eletrônico até o uso de operações bancárias em agências na Internet, os chamados *home-banking*. Entretanto, esses benefícios só ocorrem em um ambiente seguro, em uma rede de computadores segura.

Um ambiente computacional seguro ou uma rede de computadores segura pode ser definido como um ambiente ou sistema que possui os programas computacionais e o sistema operacional atualizados e adequadamente configurados bem como sistemas de proteção para este ambiente como sistema de antivírus, *firewall* e IDS.

Sistemas antivírus e *firewall* são sistemas de proteção que visam impedir a execução de ações maléficas na rede, constituindo, portanto, uma barreira aos invasores (e.g. vírus, *worms* e *hackers*). Um IDS é um sistema que visa detectar a intrusão uma vez que esta conseguiu passar pelas barreiras como antivírus e *firewall*. Um IDS é um sistema que fica atrás das barreiras de proteção e visa detectar o quanto antes possível uma invasão a fim de minimizar os prejuízos decorrentes da mesma. Partindo do pressuposto de que não há um ambiente completamente seguro ou uma rede de computadores completamente segura, minimizar o tempo de detecção de uma invasão e o tempo de tomada de contramedida é fundamental para os usuários, sejam estes pessoas jurídicas ou pessoas físicas. Vários IDS's estão disponíveis como produtos ou propostas acadêmicas contendo diferentes focos, diferentes mecanismos de detecção e tomada de contramedidas, diferentes paradigmas usados em sua concepção e implementação e diferentes protocolos de comunicação. Entretanto, os IDS's existentes são sistemas isolados e a reutilização de uma funcionalidade por um outro IDS é de difícil implementação. Uma arquitetura que permita a reutilização de funcionalidades entre sistemas de detecção de intrusão e que permita a interoperabilidade entre diferentes IDS's é um desafio que se transposto pode diminuir significativamente o tempo de detecção e a melhorar a tomada de contramedidas por uma rede de IDS.

A cada dia, novos registros de ataques a *sites* de organizações privadas e gover-

namentais são noticiados. Empresas de tecnologia, serviços bancários e financeiros, de cartão de crédito, meios de comunicação ou entretenimento e empresas de energia recebem a maior parte dos ataques diariamente. Estes ataques possuem o objetivo de obter dados sigilosos e posteriormente solicitar recursos financeiros através de chantagem virtual, uso de cartões de credito e débito, disseminação de vírus em uma rede corporativa, manifestar e ou chamar atenção sobre assuntos.

As estatísticas divulgadas pelo CERT-BR[1], apontam para um pouco mais de 201 mil registros de ataques no primeiro semestre do ano de 2012, um valor um pouco a mais da metade (50,39%) sobre o mesmo período registrado no primeiro semestre do ano de 2011. Apesar de apresentar um pequeno aumento no número de incidentes nesse semestre em relação ao ano passado, esses números só reforçam a necessidade de termos um ambiente computacional bastante seguro. A Figura 1.1 apresenta as estatísticas dos incidentes informados ao CERT.br desde o ano de 1999 até o mês de Junho de 2012.

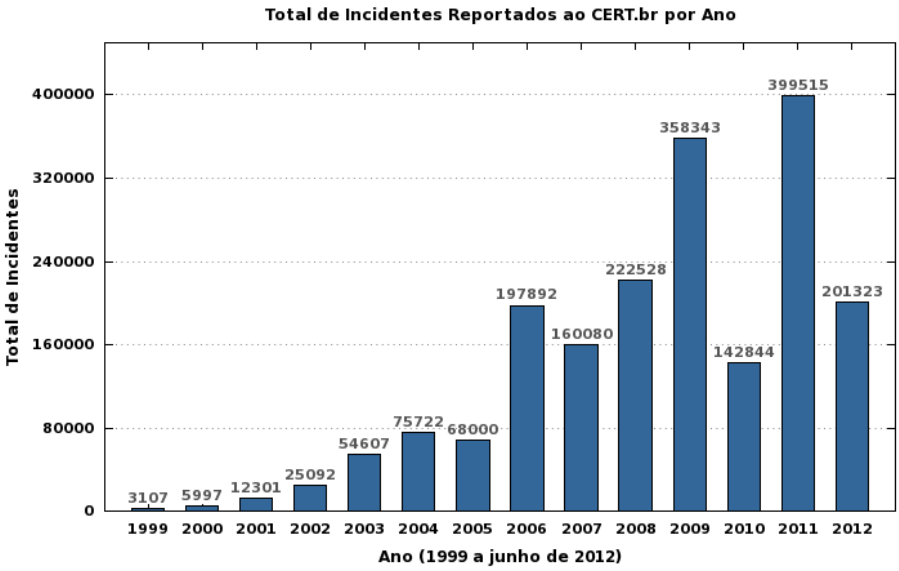


Figura 1.1: Estatísticas segundo o site CERT.br[1]

1.1 Definição do problema

Um ambiente computacional seguro para garantir os benefícios aos usuários exige itens com sistema de segurança com *firewall*, sistema antivírus, IDS e um equipe com profissionais qualificados para atuarem nessa área de segurança.

[1]O CERT-BR é um Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil é mantido pelo NIC.br, do Comitê Gestor da Internet no Brasil, e atende a qualquer rede brasileira conectada à Internet. O seu *site* é http://www.cert.br/.

O *firewall* protege o sistema atuando em portas legítimas de acesso aos programas em um sistema operacional mas não possui a capacidade de identificar a natureza do trafego que circula nessas portas legítimas, como tentativas de acesso, por exemplo. O IDS protege o sistema analisando o comportamento do trafego nas portas e tomando decisões para proteger o ambiente computacional. Entretanto essa análise faz-se com um grau de acerto levando em consideração a quantidade de assinatura em base de conhecimento do tráfego que é utilizado em uma rede de computadores. O IDS permite a detecção de intrusões e a consequente notificação ao administrador da rede ou, em conjunto com o *firewall*, bloqueia a porta utilizada na invasão ou o endereço IP do atacante.

Um fator importante para a detecção de uma intrusão é a qualidade da base de assinaturas. Os sistemas IDS são sistemas isolados e a interoperabilidade entre IDS de fornecedores diferentes é complexa e de difícil implementação. Os IDS existentes na literatura, incluindo o IDS-NIDIA (*Instrusion Detection System- Network Intrusion Detection System based on Intelligent Agents*), são sistemas isolados, não são facilmente reutilizados. Geralmente, estes se comunicam utilizando diferentes protocolos e são criados com paradigmas de programação diferentes. A solução proposta nesta dissertação de mestrado é uma arquitetura para suportar a interoperabilidade entre sistemas IDS seguindo uma filosofia SOA (*Service Oriented Architecture*). O IDS-NIDIA será adaptado e estendido de acordo com esta arquitetura para validá-la. Os serviços oferecidos pela arquitetura IDS-NIDIA serão transformados em serviços *web*, com o propósito de oferecer uma composição estática de IDS e reuso das informações de outro IDS.

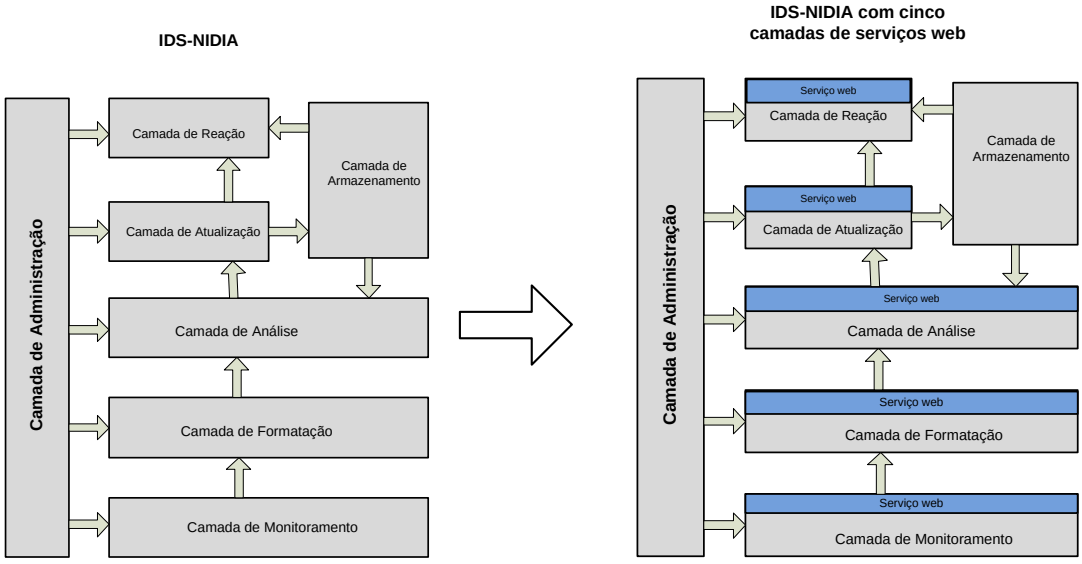
1.2 Solução proposta: *Framework NIDS-SOA*

A solução proposta é transformar as camadas, os agentes e/ou os agentes de um IDS(como camada de captura, análise, formatação, atualização e consulta em uma base de dados e reação) em serviços *web*.

A finalidade de transformar todas essas camadas e/ou agentes de um IDS em serviço *web*, permitirá que as funções de cada camada e/ou agente sejam transformadas posteriormente em serviços para que sejam utilizados em uma arquitetura orientada a serviço.

Esses serviços podem ser reutilizados com outros serviços de outros IDS's para criar uma ou mais composição de serviços de IDS. Um IDS, por exemplo, poderá utilizar o serviço básico de conhecimento de assinatura de invasões para compor um serviço básico de análise de um outro IDS, gerando assim, uma composição de serviços com dois ou mais IDS's distintos. Esses serviços básicos serão utilizados por outros IDS's que tenham a característica de realizar composição de serviços com outros IDS's com o obetivo de compartilhar informações através de uma arquitetura orientada a serviços.

A Figura 1.2 apresenta os diagramas do IDS-NIDIA sem as camadas de serviços *web* 1.2(a) e com os serviços *web* 1.2(b).



(a) IDS-NIDIA sem a camada de serviço web[4] (b) IDS-NIDIA com a camada de serviço web

Figura 1.2: IDS-NIDIA com e sem camadas de serviços web

1.3 Metodologia utilizada

A metodologia utilizada no desenvolvimento deste trabalho consiste:

- Inicialmente, a pesquisa bibliográfica sobre o sistema IDS, sistemas multiagentes, serviços *web*, arquitetura SOA e o sistema IDS-NIDIA;
- A Concepção de um *framework* de IDS baseado em SOA (NIDS-SOA). Assim as funcionalidades do IDS serão expostos e reutilizados como serviços;
- Adaptação e extensão do IDS-NIDIA para implementar o framework proposto, incluindo composição estática e reutilização;
- Testes da plataforma NIDS-SOA.

1.4 Objetivos

Temos por objetivo conceber e desenvolver um *framework* para suportar a interoperabilidade entre sistemas IDS seguindo uma filosofia SOA. Para validar este *framework*, o IDS-NIDIA será adaptado e estendido de acordo com esta arquitetura. Os serviços

oferecidos pela arquitetura IDS-NIDIA serão transformados em serviços *web*, com o propósito de oferecer uma composição estática de IDS e reuso das informações de outros IDS's.

1.5 Contribuições

As principais contribuições deste trabalho são as seguintes:

- Fazer com que IDS, através do uso de serviço *web* e composição de serviços, colaborem com outros IDS;
- Através do compartilhamento de informações de vários IDS's, que utilizem a composição de serviço, as informações sobre assinaturas e comportamento de tráfego ajudarão um IDS, que utilize a composição desse serviços, a melhorar o tempo de resposta a uma invasão em uma rede de computadores.

1.6 Organização do Trabalho

Esta dissertação está organizada em uma quantidade de seis capítulos e os mesmos estão descritos desta forma:

- No Capítulo 1, o estado atual dos sistema de segurança baseados em IDS e a sua importância serão apresentados. Uma proposta de um *framework* seguindo a filosofia SOA(*Service Oriented Architecture*) com a característica de poder compartilhar informações entre IDS's distintos é apresentado.
- No Capítulo 2, a teoria sobre o Sistema de Detecção de Invasão é apresentada, relatando um breve histórico, a importância do uso de um IDS em uma rede, os tipos de IDS existentes e uma breve descriçaõ sobre o IDS-NIDIA e os trabalhos já realizados sobre este IDS. Conceitos sobre serviços *web* e composição de serviço.
- No Capítulo 3, os conceitos sobre a modelagem para uma arquitetura orientada a serviço são apresentados.
- No Capítulo 4, o *framework* NIDS-SOA é apresentado. A metodologia utilizada para a criação desse *framework*, ferramenta computacional utilizada para o desenvolvimento do mesmo e a adequação do IDS-NIDIA a este *framework* são detalhados.
- No Capítulo 5, os resultados do *framework* NIDS-SOA são apresentados.
- No Capítulo 6, a conclusão deste trabalho é apresentada.

Capítulo 2

Fundamentos Teóricos

Neste capítulo, apresenta-se um breve histórico sobre o IDS, suas características, os tipos ou modelos de IDS existentes e a metodologia utilizada por esses IDS. O IDS-NIDIA, será também apresentado, que é um modelo de IDS baseado em agentes e as tecnologias utilizadas na elaboração do *framework* IDS-NIDS-SOA, que é o objetivo deste trabalho. Apresentaremos também conceitos sobre arquitetura orientada a serviço.

2.1 Sistema de Detecção de Intrusão - Breve histórico

Um IDS possui capacidade de detectar as atividades maliciosas em uma rede de computadores através da captura e análise dos dados que trafegam por essa rede bem como os dados contidos nos arquivos de *log* gerados pelos mais variados serviços em servidor dentro de um ambiente corporativo. Ele auxilia na proteção do ambiente corporativo e o seu ponto de instalação é de fundamental importância. sugerido inicialmente por James Anderson em 1980 em um artigo intitulado ***"Computer Security Threat Monitoring and Surveillance"***[13]. Este trabalho definia os conceitos de que uma auditoria em um sistema poderia fornecer informações importantes no uso inadequado do sistema por parte do usuário. Posteriormente, esses conceitos foram difundidos em 1987 por Dorothy Denning em um artigo intitulado ***"Um Modelo de Detecção de Intrusão"***[14]. Um ano depois Dorothy Denning desenvolveu um modelo para detecção de intrusão[15]. Tood Heberlein introduziu o conceito de detecção de intrusão de rede, chegando a desenvolver o Monitor da Segurança da Rede NSM (***Network Security Monitor***), como sendo o primeiro sistema de detecção de intrusão de rede. O uso do NSM em grandes redes fornecia uma grande quantidade de informações sobre o que era gerado pela rede. O conceito de detecção de intrusão de rede iniciado por Tood Heberlein atraiu um maior interesse das empresas de TI contribuindo para um aumento de investimentos nesse mercado[14].

O princípio de funcionamento de um IDS é capturar um tráfego de pacotes TCP/UDP que circula pela rede de computadores e logo em seguida, com base nas regras ou nas

descrições de anomalias que ele dispõe em sua base. Sendo assim, um IDS analisa o conteúdo do trafego capturado com o objetivo de encontrar alguma semelhança com um pacote que contenha características de vulnerabilidades ou de códigos maliciosos previamente conhecidos em sua base de dados que comumente chamamos de regras. Na eventualidade de ser encontrado algum pacote com características que se assemelhem com as características que se encontram em sua base de dados, o sistema de proteção contra invasão tomará ações para proteger o sistema, que vão desde um envio de um simples alerta para o *e-mail* do administrador do sistema como ações mais fortes como o bloqueio do remetente desses pacotes através do *firewall*.

2.2 Descrição de um IDS

Um IDS tem por finalidade analisar o comportamento em uma rede computadores. Ele trabalha com um sistema que alerta o administrator ou o próprio sistema quando da ocorrência de tentativas de intrusões na rede, com uso de conhecimentos listados em uma base de dados(como por exemplo assinatura de ataques, comportamento diferente do esperado)[2].

Um IDS junto com o *firewall* tem a capacidade de melhor proteger uma rede de computadores. O *firewall* limita-se a bloquear portas de serviços que não estejam em uso em uma rede, enquanto que um IDS realiza a analise do tráfego na rede em todas as portas de serviços disponíveis em um sistema, ele detecta os ataques realizados contras portas e serviços legítimos (como por exemplo, podemos citar as portas TCP: **25, 80, 110 e 443**[1] e as portas UDP: **53 e 123**[2]).

Os serviços listados para essas portas podem sofrer ataques que não serão barrados pelo *firewall*, haja vista que essas portas são legítimas. Entretanto, o tráfego de rede em todas essas portas pode ser capturado e analisado por um IDS e dependendo das regras que o IDS possuir, o mesmo poderá responder ao invasor com o bloqueio da porta para o IP do atacante.

Na Figura 2.1, mostra-se um cenário[2] com o comportamento do tráfego em uma rede corporativa, onde um *firewall* e um IDS trabalham em conjunto para melhor proteger a rede corporativa. Observa-se na Figura 2.1 que todo tráfego que vem da Internet passará pelo *firewall* que poderá bloquear a conexão, caso a porta(TCP e/ou UDP) não esteja liberada no *firewall* ou poderá permitir sua conexão, para portas (TCP e/ou UDP) liberadas no *firewall*.

A conexão liberada pelo *firewall* será enviada ao IDS que fará a análise desse tráfego

[1] A porta 25 oferece o serviço de SMTP, a porta 80 oferece serviço para o servidor *web*, a porta 110 oferece serviço para o servidor POP e a porta 443 fornece serviço para uma conexão segura para servidores *web*.

[2] As portas 53 e 123, fornecem os serviços de resolução de nomes e sincronização do horário nos relógios dos computadores, respectivamente

de rede e consequentemente permitirá a passagem do tráfego legítimo e aquele tráfego suspeito ou com anormalidades será analisado e logo em seguida será efetuada uma reação de proteção para a rede.

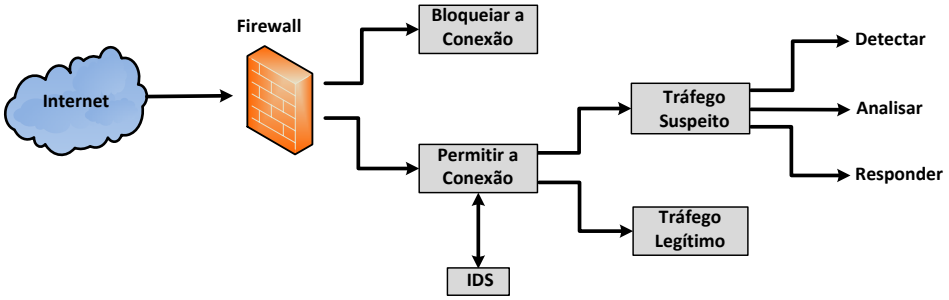


Figura 2.1: Trabalho em conjunto do Firewall com o IDS[2]

Um sistema IDS deve possuir as seguintes características [2]:

- **Captura de dados:** Necessita ter um sistema de captura de dados da rede ou em algum arquivo de *log* em um sistema;
- **Análise de dados:** Necessita ter um sistema que realizará a análise dos dados capturados com base em assinaturas de ataques já conhecidas e registradas em uma base de dados. Também realizará análise em atividades consideradas fora do padrão de normalidade de uso da rede;
- **Base de dados:** Necessita ter uma base de dados para registrar informações sobre tipos de assinatura de ataques já conhecidos e fornecer respostas a consulta realizadas pelo sistema de análise de dados;
- **Atualização na base de dados:** Fará a atualização em uma base de dados, das ocorrências capturadas e consideradas como atividades suspeitas;
- **Módulo de reação:** Esse módulo terá a capacidade de realizar ações de resposta a um ataque realizando ações como: alterações nas regras do *firewall*, emissão de alarme, geração dos eventos ocrridos em relatórios ou em um arquivo de registro desses eventos(Arquivo de *log*).

2.3 Metodologia de detecção de um IDS

Um IDS precisa ter métodos para analisar os dados capturados na rede e identificar atividades suspeitas e/ou maléficas na rede e/ou no sistema, para que uma ação contra essas atividades suspeitas e/ou maléficas possam ser tomadas o mais rápido possível. Essas análises baseiam-se em informações já conhecidas(assinaturas) e no comportamento anormal do tráfego analisado.

Os resultados da análise feita pelo IDS no tráfego de rede apresenta resultados e que podem ser os seguintes:

- **Comportamento Normal** - É o tráfego legítimo e que o IDS o analisa como sendo legítimo. Há também o tráfego suspeito e/ou maléfico e o que o IDS o analisa como suspeito e/ou maléfico;
- **Falso Negativo** - É o tráfego suspeito e que o IDS o analisa como sendo normal;
- **Falso Positivo** - É o tráfego legítimo que o IDS o analisa como sendo suspeito.

Os métodos utilizados por um IDS para analisar os dados são os de detecção baseado em anomalias e assinaturas.

2.3.1 Método de detecção baseado em Anomalia

Este método utiliza técnicas baseadas em padrões de uso normal e do registro das atividades dos usuários, *hosts* e conexões de rede [16]. Uma base de dados com essas informações é utilizada. Portanto, todo tráfego diferente desse padrão e já conhecido através dos registros guardados, será marcado como uma invasão. Com base nessas informações armazenadas em uma base de dados é que o sistema analisa o tráfego capturado e, em função do comportamento desse tráfego, o sistema gera um alerta, notificando ao administrador sobre essa anomalia [16]. A principal vantagem deste método de detecção é que ele é capaz de detectar qualquer tipo de novos ataques. Entretanto, a principal desvantagem desse método é a grande quantidade de informações que o mesmo guarda, podendo gerar alertas falsos positivos e a necessidade de atualizar o sistema com informações corretas [16].

2.3.2 Método de detecção baseado em Assinatura

Este método utiliza a pesquisa no tráfego de rede para encontrar byte ou seqüências de pacotes conhecidos como maliciosos que combinem com regras pré-estabelecidas [16]. Quando esse método encontra algum dado sobre o tráfego analisado que combinem com as regras pré-estabelecidas, um alerta para este dado é gerado. Como são muitas regras(assinaturas), uma base de dados é utilizada para esse propósito. Esse método possui a vantagem de gerar novas assinaturas baseadas no comportamento da rede em que se encontra o IDS [16].

2.4 Tipos de IDS

Os tipos de IDS existentes são os baseados em *host* denominado de **HIDS**(*Host-Based Intrusion Detection System*) e em rede denominado **NIDS** (*Network-Based Intru-*

sion Detection System).
Há o **IDS híbrido**, que reúne as melhores características dos IDS de rede e de *host*. O IPS(*Intrusion Prevention System*) foi criado devido à descoberta de pontos fracos nos IDS de rede e de *host*[2]. Na próximas seções, descreveremos as funcionalidades desses dois tipos de IDS.

2.4.1 IDS na Rede - NIDS

O IDS baseado em rede de computadores, o NDIS, consiste em instalar um IDS em um segmento de rede. Esse segmento de rede pode ser uma rede de um laboratório de pesquisa, uma rede de uma área administrativa ou em uma DMZ(*DeMilitarized Zone*), que é uma área onde se localizam todos os servidores onde os aplicativos da empresa são executados(como servidor *web*, servidor com banco de dados, servidor de arquivos, servidor de DNS, servidor de correio eletrônico e outros serviços em uma rede corporativa). Entretanto, pode-se instalar esse NIDS na entrada da rede em uma posição anterior ou posterior ao *firewall* da rede. Uma desvantagem nesse caso, reside na grande quantidade de dados a serem analisados, pois o NIDS estará escutando todo o tráfego da rede. O NDIS[3] pode facilmente ser instalado em um segmento de rede onde existe um *switch* com essa tecnologia de SPAN disponível, sem causar maiores impactos na performance da rede. Na Figura 2.2, mostra-se um NDIS instalado em um segmento de rede, localizado logo após o *firewall*, utilizando-se da tecnologia SPAN[17] de um *switch*. Nessa Figura observa-se que todo o tráfego de rede que sai do *firewall* e entra na **porta 1** do *switch* é copiado para a **porta 2**, onde encontra-se instalado o NDIS, que realizará a captura e análise dos pacotes da rede. Uma alternativa para esse caso apresentado na Figura 2.2, caso o *switch* não tivesse a tecnologia de espelhamento de tráfego (a *SPAN*), seria fazer com que todo o tráfego nesse segmento passasse pelo NDIS, o que poderia afetar o desempenho na rede, pois todo o tráfego de dados precisaria primeiro ser analisado pelo NDIS para depois ser liberado para a rede.

[3]Os novos *switches* possuem o recurso de SPAN (*Switched Port Analyzer*) ou *Port Mirroring*, que permitem fazer uma cópia do tráfego em uma outra porta.

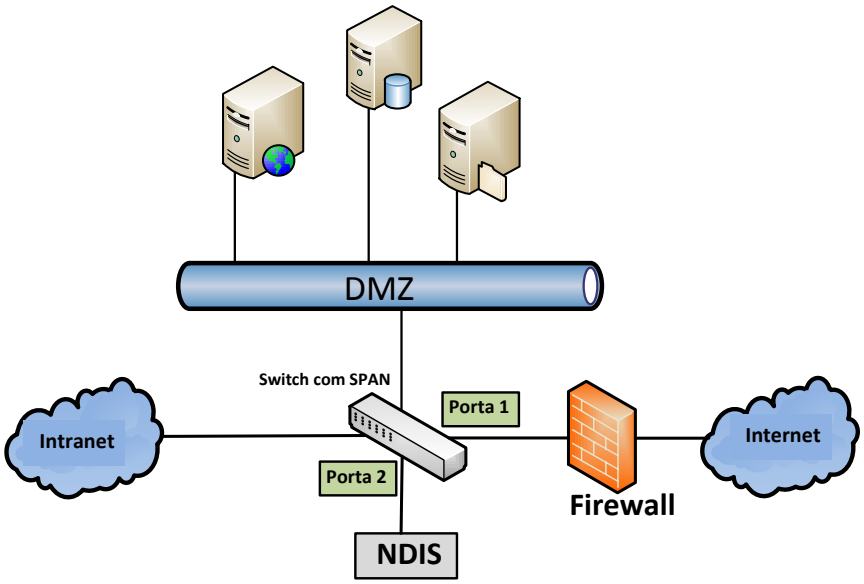


Figura 2.2: NIDS - IDS baseado na análise de rede

As atividades realizadas pelo NDIS são as seguintes:

- Monitorar o tráfego em segmento de rede ou em toda a rede;
- Capturar cada pacote de rede e fazer a análise dos cabeçalhos e do conteúdo do pacote;
- Registrar, baseado em regras de análises, a quantidade de vezes que um determinado endereço IP de origem e/ou destino acessou o sistema;
- Registrar, baseado em regras de análises, a quantidade de vezes que uma determinada porta (TCP e/ou UDP) foi acessada.

2.4.2 IDS em *Host*

O IDS baseado em análise de host, o HIDS, consiste na análise dos *logs* do *Hosts* monitorados. Esses *logs*, referem-se aos registros dos serviços ou *daemons* em um sistema operacional e no comportamento das atividades em um *host*.

A Figura 2.3 apresenta um HIDS.

As atividades realizadas pelo HIDS são as seguintes:

- Registro das atividades de autenticação de usuários no sistema;
- Atividades de acesso e erro em servidores *web*;
- atividades de acesso e erro em correio eletrônico e com os serviços associados ao controle de *spam*, filtro de spam e a sistema de antivirus;

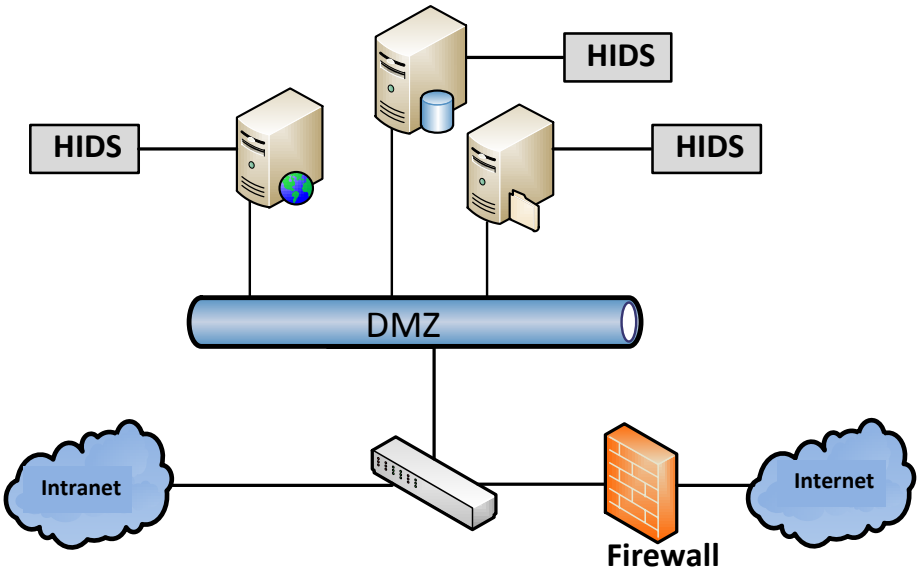


Figura 2.3: HIDS - IDS baseado na análise de host

- Atividades de acesso e erro em servidor de nomes (DNS);
- Atividades de acesso e integridade de arquivos nas pastas dos usuários;
- Atividades de varredura nas portas dos serviços (*daemons*);
- Análise de comportamento de processos no sistema e
- Análise do uso da CPU e da quantidade de memória por cada serviço e/ou usuário.

2.5 O *Framework* CIDF

O modelo do *Framework* CIDF (*Common Intrusion Detection Framework*) é um projeto da DARPA (*Defense Advanced Research Projects Agency*) [18]. Este modelo foi idealizado com a finalidade de desenvolver protocolos e interfaces de programação para que os projetos de pesquisas sobre IDS pudessem compartilhar informações e recursos [19]. O CIDF também tem como objetivo possibilitar a reutilização dos componentes de detecção de intrusão em outros sistemas [3].

Ele apresenta quatro componentes ou módulos, conforme apresentado na Figura 2.4, onde é mostrado as suas funcionalidades e o fluxo de informações entre os componentes.

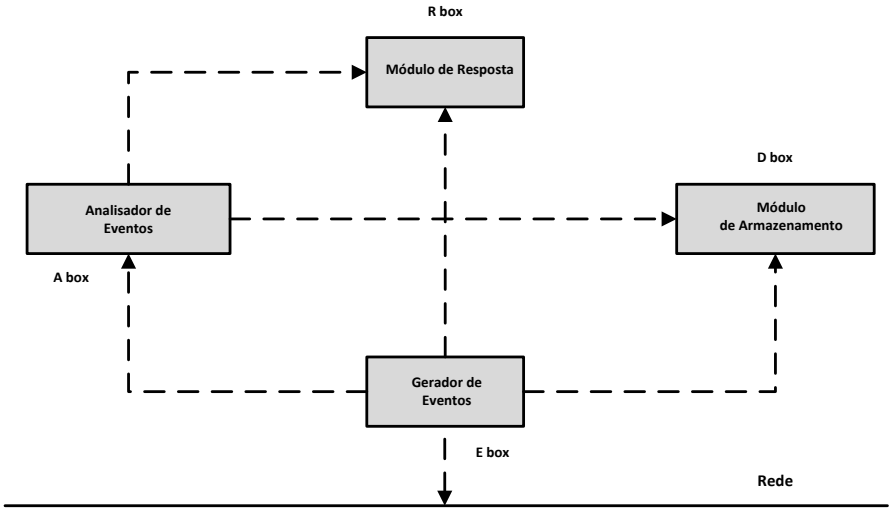


Figura 2.4: O Framework CIDF[3]

Os componentes do *Framework* CIDF são:

- **O gerador de eventos - E Boxes:** Este componente possui a finalidade de obter eventos do meio externo, através da captura de tráfego na rede de computadores ou em *logs* de uma máquina [6]. Esses dados capturados são repassados em seguida para o componente de análise do sistema;
- **O analisador de eventos - A boxes:** Este componente realiza a análise dos dados coletados pelo componente Gerador de Eventos (E-Box). Baseado em regras pré-estabelecidas (como por exemplo, assinatura de ataques) este componente classifica e identifica os registros coletados;
- **O módulo de armazenamento - D Boxes:** Este componente armazena os dados classificados pelo analisador de eventos (A-Box) e também armazena registro de ataques já conhecidos;
- **O módulo de respostas - R boxes:** Este componente realiza uma contramedida na ocorrência de uma invasão registrada pelo componente de análise.

2.6 O *Framework* IDS-NIDIA

O IDS-NIDIA é um sistema de detecção de invasão baseado em um conjunto de agentes. Este sistema foi baseado no modelo CIDF(*Common IntrusionDetection Framework*)

que desenvolve protocolos e interfaces de aplicativos como objetivo de que projetos de pesquisas sobre detecção de intrusão possam compartilhar informações[19], [20]. O IDS-NIDIA possui um conjunto de agentes que relatam ações de acordo com a sua funcionalidade. Há agentes sensores (rede e *hosts*) que capturam informações sobre a rede de computadores e sobre um determinado host; agentes que analisam informações coletadas pelos agentes sensores; agentes que tem o objetivo de armazenar informações já analisadas em uma base de dados para uso posterior; agentes para controlar as ações a serem executadas com base nas informações armazenadas na base de dados. As camadas do projeto do IDS-NIDIA[16], apresentado na Figura 2.5, são as seguintes:

- Camada de Monitoramento;
- Camada de Análise;
- Camada de Reação;
- Camada de Atualização;
- Camada de Armazenamento.

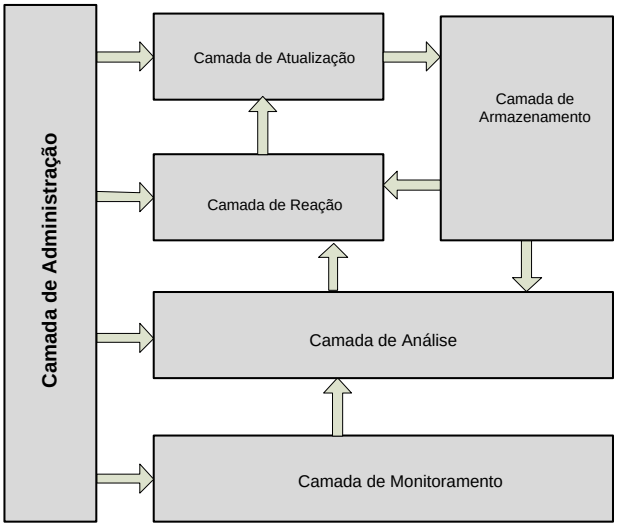


Figura 2.5: Arquitetura do IDS NIDIA - Modelo de Camadas[4]

2.6.1 Descrição do IDS-NIDIA

O IDS NIDIA possui camadas, cuja descrição é apresentada a seguir:

- **Camada de Monitoramento:** Esta camada(equivalente ao gerador de eventos do modelo CIDF) tem por objetivo capturar pacotes em uma rede de computadores ou em um arquivo de *log*. Nesta camada, encontram-se os agentes de monitoramento

do sistema(SMA) (*System Monitoring Agent*). Estes agentes SMA dividem-se em duas outras categorias: Os **agentes sensores de rede (NSA)** e os **agentes sensores de *host* (HSA)**[4], [21]. Os agentes sensores de rede capturam o tráfego de rede em um segmento de rede ou em um ponto estratégico dentro de uma rede corporativa enquanto os agentes sensores de *hosts* capturam dados em arquivo de *log* localizado em um servidor;

- **Camada de Análise:** Esta camada (equivalente ao analisador de eventos do modelo CIDF) tem por objetivo analisar os pacotes capturados pela camada de monitoramento(pacotes de rede ou de *logs* de *host*)[21]. Essa camada também realiza a formatação dos pacotes e os padroniza. Essa formatação permite que ataques realizados posteriormente possam ser identificados com maior precisão e segurança[4]. O **agente de avaliação de sistema** (SEA) possui a função de analisar e gerar grau de suspeita dos pacotes formatados. Neste modelo, são utilizadas as seguintes bases[4]:
  - Base de padrões de intrusão - IIDB (*Instrusion Information Data Base*);
  - Base de incidente de intrusão - DFDB (*Data Forense Data Base*);
  - Base de estratégias - STDB (*Strategy Data Base*);
  - Base de dados de respostas - RADB (*Response Action Data Base*).
- **Camada de Reação:** Esta camada(equivalente ao módulo de resposta do modelo CIDF) tem por objetivo realizar alguma ação (como bloqueio do invasor no *firewall* ou da porta acessada por este invasor)[4],[21]. O **agente controlador do sistema** (SCA), que se encontra nessa camada realiza as ações de contramedidas conforme os registros localizados nas bases de dados de estratégias (STDB) e de ações (RADB)[4];
- **Camada de Atualização:** Esta camada (equivalente ao módulo de armazenamento do modelo CIDF) tem por objetivo atualizar todas as bases de dados do sistema IDS-NIDIA (STBD, RADB, DFDB e IIDB). O **agente de atualização do sistema** (SUA) realiza o serviço de atualização de todas essas bases de dados além de manter a integridade dos dados armazenados em todas essas bases[20], [4];
- **Camada de Armazenamento:** Esta camada mantém os dados e informações das bases de dados utilizados pelo IDS-NIDIA[4];
- **Camada de Administração:** Esta camada é constituída pelo **agentes controladores principais** (MCA), cuja função é administrar todos os agentes do IDS-NIDIA [4] e realizar funções de como configurar políticas de segurança, inserindo regras na base de estratégias, administrar o sistema IDS-NIDIA e saber como o mesmo está se comportando [20].

2.6.2 IDS-NIDIA-Trabalhos Relacionados

O IDS-NIDIA é um sistema de detecção de invasão baseado em um conjunto de agentes com noção de sociedade de agentes inteligentes e redes neurais[19], [20]. Os últimos trabalhos relacionados a este sistema de IDS constou com as seguintes contribuições:

- Estudo de agentes inteligentes para detecção de intrusos em redes de computadores, baseado na noção de sociedade de agentes inteligentes e redes neurais[20];
- Um modelo de atualização automática do mecanismo de detecção de ataques de redes para um IDS[22];
- Um modelo com o uso de agentes para monitoração de ataque e respostas automatizadas[23];
- Um modelo para atualização automática de um IDS através de serviço *web*[3];
- Um modelo para gerenciar e integrar bases de dados do IDS-NDIA[24];
- Um modelo para o IDS-NIDIA ter uma comunicação segura e confiável adaptando especificações XML[25].

2.7 Serviços *web*

Os serviços *web* surgiram como uma consequência natural do crescimento do uso da Internet [5]. Eles representam um grande avanço para facilitar a integração de sistemas distribuídos [26]. Eles apresentam vantagens para o desenvolvimento de aplicações, pois propiciam a agilidade requerida pelas empresas frente às rápidas mudanças no ambiente de negócios. A principal vantagem do uso dos serviços *web* está na sua facilidade de integrar diferentes aplicações e serviços, devido ao uso de protocolos e padrões amplamente difundidos na *web*, como por exemplo, SOAP, HTTP, XML, e outros[27].

Os serviços *web* são aplicações entre cliente e servidor e que se comunicam através da rede [28]. Como descrito pelo W3C[4](*World Wide Web Consortium*), os serviços *web* fornecem um padrão de interoperabilidade entre as aplicações que são executadas em diferentes plataformas e *frameworks* [28]. Os serviços *web* são caracterizados por sua grande interoperabilidade, extensibilidade graças ao uso do XML[28].

Estes são sistemas de *software* construídos para suportar interação entre máquinas em uma rede ou em sistemas distribuídos. Os serviços *web* são componentes ou uma aplicação que é acessível através de protocolos padrões da Internet. São aplicações autocontidas, que possuem interfaces baseadas em XML e que descrevem uma coleção

[4]A (W3C) é uma comunidade internacional, que trabalha para desenvolver padrões para a *web*. Essa comunidade é liderada por Tim Berners-Lee, um dos criadores da *web*.

de operações acessíveis através da rede, independente da tecnologia utilizada na implementação do serviço. Um serviço *web* então seria disponibilizado na Internet, possuindo suas definições de acesso descritas em WSDL (*Web Services Descrition Language*), seria registrado em uma UDDI (*Universal Description, Discovery and Integration*) e poderia ser acessado por qualquer cliente utilizando SOAP (*Simple Object Access Protoco*), e os dados seriam transmitidos e representados em XML. WSDL, UDDI e SOAP são baseados em XML. A Figura 2.6 apresenta a descrição de *web services* e a relação de interação ente os seus padrões [26].

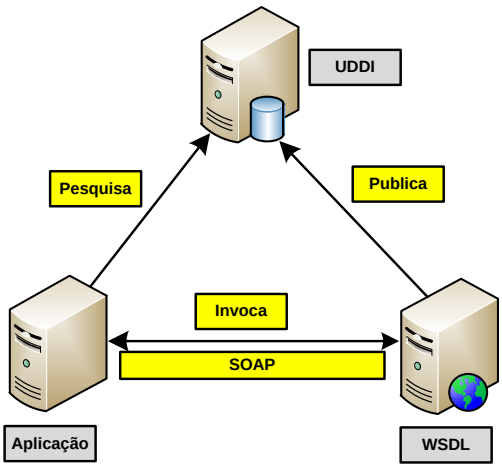


Figura 2.6: Modelo do serviço *web*

Nas próximas subseções, uma breve descrição será feita sobre esse conjunto de tecnologias utilizadas para melhor compreensão sobre a tecnologia de serviços *web*.

2.7.1 *WSDL*

O WSDL (*Web Services Description Language*) possui a função de informar ao consumidor o que o serviço realiza, de qual forma o consumidor pode acionar o serviço e como o consumidor pode diferenciar serviços similares oferecidos por diferentes fornecedores [5]. Ele apresenta as definições como: nome do serviço *web*, o nome dos métodos utilizados, os argumentos desses métodos e o protocolo de acesso. A principal função do WSDL é permitir que o serviço *web* possa ser utilizado automaticamente dispensando a intervenção do criador do serviço *web*.

A Figura 2.7 apresenta a representação conceitual de um documento WSDL, onde as setas representam as relações entre os elementos e as caixas de relacionamento do *container*

A descrição desse componentes são as seguintes:

- ***types*** - São os tipos de dados utilizados nas mensagens. Esses tipos podem ser inteiro, *String*, *boolean* e complexo;

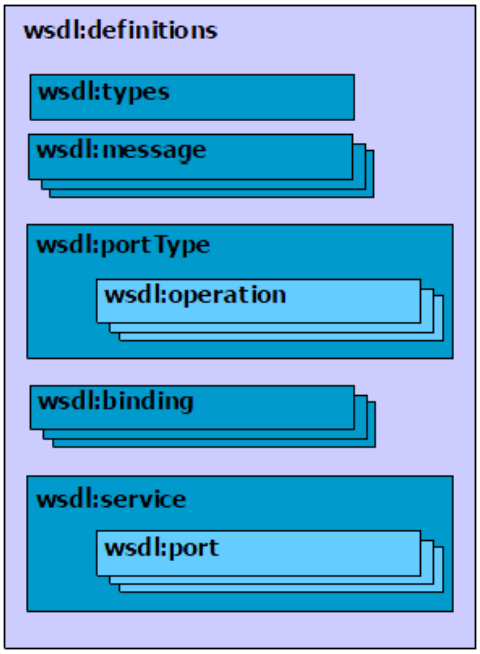


Figura 2.7: Representação de um documento WSDL[5]

- ***message*** - Representa as mensagens enviadas e recebidas, incluindo seus seus parâmetros de entrada e saída do serviço;
- ***operation*** - Representa a assinatura do método e o relacionamento entre os parâmetros de entrada e saída;
- ***porttypes*** - Representa o as funções suportadas, métodos disponibilizados pelo serviço;
- ***bindings*** - Representa a definição dos protocolos a serem utilizados para acessar o serviço na rede.

2.7.2 *UDDI*

É um protocolo que possui mecanismo para publicar, descobrir e integrar serviços. Esses mecanismos definem as características para uma descrição, classificação e a interface para acesso a essas informações[27]. Foi uma solução definida por um conjunto de empresas[5] para solução dos muitos problemas encontrados na execução de negócios utilizado na Internet em B2B(*Business to Business*) [5].

O UDDI (*Universal Description, Discovery, and Integration*) possui informações sobre os serviços *web* disponíveis, apresentando referencias para os mais variados segmentos

[5]IBM, Microsoft, Ariba e outras mais

A Figura 2.8 apresenta a estrutura de registro do UDDI com as ações que ocorrem nessa estrutura[6]:

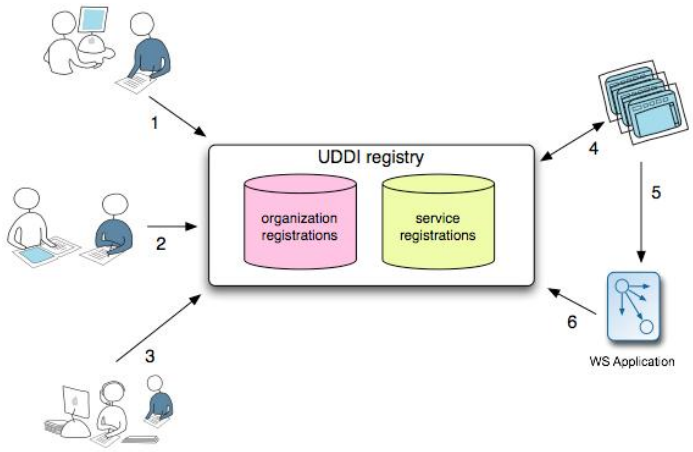


Figura 2.8: Interações com o UDDI [6]

- Arquitetos de negócios, usuários e desenvolvedores disponibilizam os serviços no UDDI (**Itens 1, 2 e 3**)
- Clientes e aplicações buscam no UDDI por serviços que atendam as suas necessidades (**Item 4**)
- Os serviços encontrados pelos clientes e aplicações (**item 4**) serão utilizados em suas aplicações bem como para gerar novos serviçso *web* (**item 5**)

2.7.3 *SOAP*

É um protocolo utilizado para troca de informações estruturadas, baseado na linguagem XML(*eXtensible Markup Language*), entre um serviço *web* e um cliente[5] e/ou para comunicação entre processos[8]. Suporta os protocolos HTTP e SMTP.

A Figura 2.9 apresenta a estrutura de uma mensagem SOAP.

A Figura 2.10 apresenta o modelo de camada dos serviços *web*, onde mostra-se as camadas de busca, descrição, comunicação em XML e transporte [7].

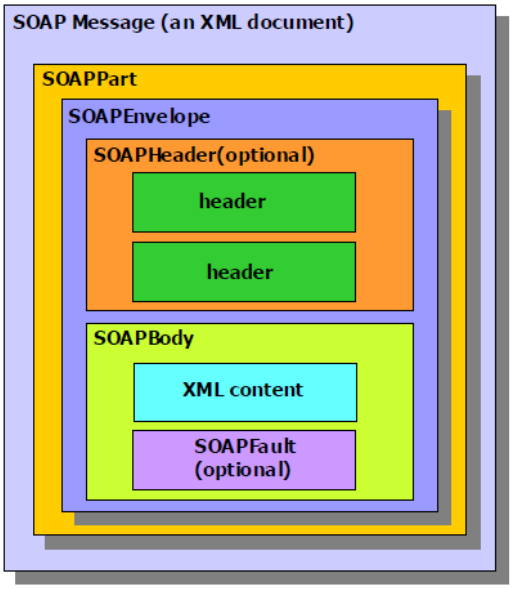


Figura 2.9: Estrutura de uma mensagem SOAP [6]

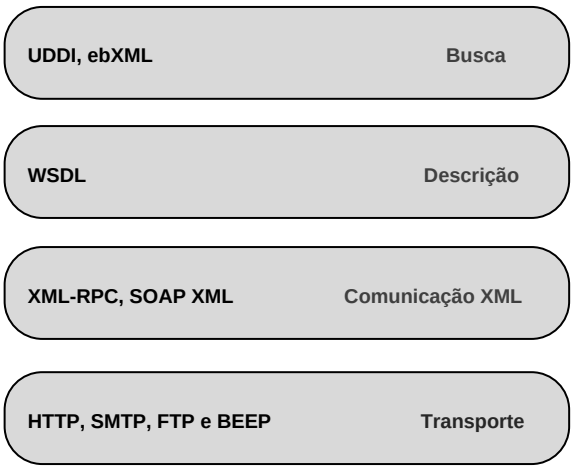


Figura 2.10: Camada de protocolos dos serviços web [7]

2.8 XML - (*Extensible Markup Language*)

A XML(*Extensible Markup Language*), é uma linguagem, recomendada pela W3C para gerar linguagens de marcação para as mais diversas finalidades[5]. Originou-se de duas fontes distintas: A SGML(*Standard Generalized Markup Language*) e a HTML(*Hypertext Markup Language*)[5]. A linguagem XML utiliza *tags*(marcadores) para identificar as informações em um documento com formato XML[5]. Há dois tipos de marcadores: os iniciais, onde são utilizados os sinais de maior e de menor(por exemplo: <**texto**>) e os marcadores finais que usam o sinal de maior e menor com a inclusão de uma barra(</**texto**>). Entretanto, essas *tags* estão relacionadas a descrição do textos que elas delimitam [3]. Utiliza-se um arquivo chamado esquema para descrição da estrutura de um documento XML [3]. O código abaixo mostra a listagem de um documento em formato XML, refe-

rente a informações de um pacote de rede com protocolo IPV4, onde mostram-se os elementos: porta de destino e de origem, IP de destino e de origem e uma informação sobre o tempo.

```
<pacote>
<dst__port>49195</dst__port>
<dst__ip>/192.168.177.103</dst__ip>
<src__port>80</src__port>
<src__ip>/129.128.5.191</src__ip>
<sec>1319653063</sec>
<usec>523886</usec>
</pacote>
```

2.9 Arquitetura Orientada a Serviços (SOA)

Uma Arquitetura Orientada a Serviço (SOA) pode utilizar serviços *web* para criação de novos serviços e reuso de serviços com uma maior eficiência. Os serviços *web* são aplicações auto-contidas que possuem interfaces baseadas em XML e que descrevem coleções de operações acessíveis através da rede, independente da tecnologia usada na implementação do serviço[8], comunica-se através de troca de mensagem com uso de XML [26].

SOA é baseada em uma metodologia que tem por objetivo integrar aplicações já existentes (sistemas legados) com novas aplicações. Essa integração pode ser feita com o uso de serviços *web*, DCOM[6](*Distributed Component Object Model*), SOAP (*Simple Object Access Protocol*), CORBA[7](*Common Object Request Broker Architecture*), REST (*Representation State Transfer*) e RPC (*Remote Procedure Call*)[8].

SOA tem três conceitos técnicos que são: serviços, interoperabilidade através de um barramento de serviços e um fraco acoplamento[8]. Um serviço é componente de *software* que representa um processo, uma tarefa de negócio ou uma atividade e que possui interface e que é invocada via mensagem. O barramento de serviço (*bus*) é uma infraestrutura que permite a interoperabilidade entre sistemas distribuídos, permitindo a comunicação entre diversos serviços ou processos de negócios de diferentes linguagens e tecnologia. O fraco acoplamento refere-se à redução da dependência de um serviço em relação a um outro serviço dentro de uma composição de serviço.

A Figura 2.11 apresenta um sistema orientado a serviço, onde tem-se um consumidor de serviço, um provedor de serviço e um repositório de serviço. A descrição desses componentes são as sseguintes:

[6]É a tecnologia da Microsoft utilizada para a computação distribuída[5]

[7]Este padrão constitui-se em uma tentativa, gerenciada pela OMG (*Object Management Group*), de desenvolver um padrão para aplicações distribuídas[5]

- **Provedor de Serviço:** É a entidade que cria, disponibiliza e descreve um serviço;
- **Registro de Serviço:** É a entidade que armazena e localiza a descrição dos serviços publicada pelo provedor de serviços;
- **Consumidor de Serviços:** É a entidade que solicita alguma função para realizar alguma tarefa ou solicita uma iteração com outro serviço.

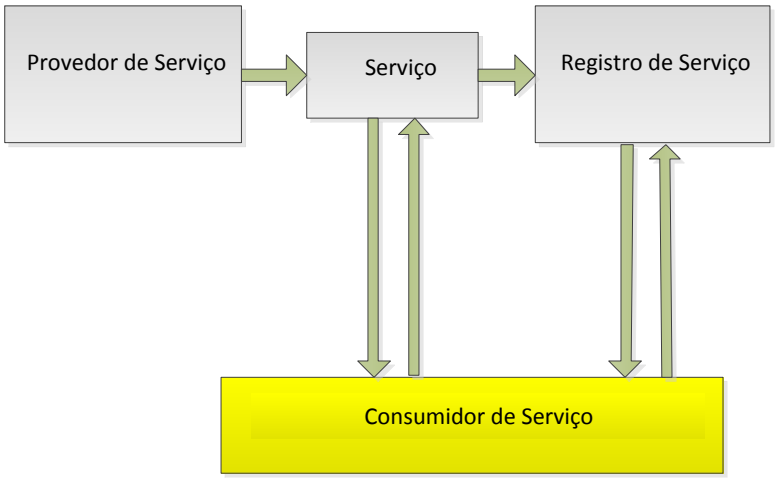


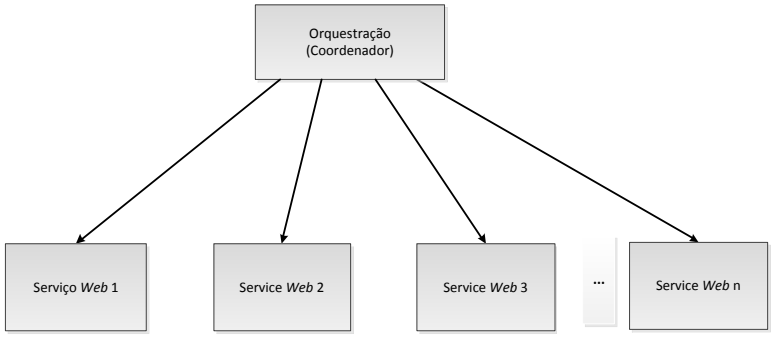
Figura 2.11: *Arquitetura Orientada a Serviços [7]*

2.10 Composição de Serviços

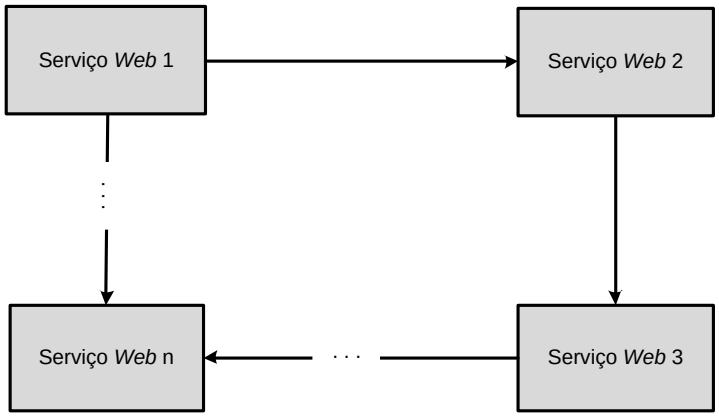
Com a evolução de SOA e o uso dos serviços *web*, surgiu nas empresas de TI, o papel do arquiteto de *software*, que é responsável por administrar os serviços *web* em uma empresa e alinhá-lo aos negócios de TI[8]. Para realizar essa tarefa foram criadas várias ferramentas para este propósito, todas baseadas em XML[8]. A ebXML(*eletronic bussiness XML*), o RosettaNet, a UBL (*Universal Bussiness Language*) e a BPEL(*Bussiness Process Execution Language*).

A linguagem BPEL é considerada como padrão para execução de processos de negócios com o uso de orquestração e coreografia de serviços *web*[8].

Na orquestração, há um serviço *web* central que controla e que gerencia as diferentes operações dos outros serviços *web* envolvidos nessa operação, enquanto que na coreografia, não há um serviço *web* que realiza esse serviço de coordenação, pois cada serviço *web* sabe sua funcionalidade e com quais serviços *web* deve interagir. Essas duas abordagens são apresentadas nas Figuras 2.12(a) e 2.12(b), respectivamente[8].



(a) Orquestração em composição de serviços[8]



(b) Coreografia em composição de serviços[8]

Figura 2.12: Orquestração e Coreografia[8]

2.11 Síntese

Neste capítulo, apresentamos a teoria sobre IDS, sua metodologia de detecção, os tipos de IDS e o *framework* CIDF e o IDS-NIDIA com os trabalhos já feitos sobre o IDS-NIDIA.

Apresentamos uma pequena introdução sobre os serviços *web*, arquitetura orientada a serviços e composição de serviços. No próximo capítulo, a teoria sobre a modelagem para arquitetura orientada a serviço será apresentado.

Capítulo 3

Arquitetura Orientada a Serviço

3.1 Introdução

A Arquitetura Orientada a Serviço(SOA) é baseada em uma metodologia que tem por objetivo integrar aplicações já existentes (sistemas legados) com novas aplicações, representando uma nova abordagem para utilização dos recursos disponíveis em TI com os negócios de uma empresa[9].

Nas organizações, novos desafios tem se apresentado decorridos de vários fatores como globalização, união de empresas decorrente da aquisição de empresas menores, concorrências de empresas nos ramos de negócios, busca por melhores índices de rendimento observando-se o custo/benefício de cada operação[29]. SOA apresenta a teoria de serviços que define conceitos, que estão transformando a maneira como as empresas desenvolvem e integram seus sistemas [9]. Ela está se consolidando como um diferencial estratégico para alinhamento do negócio com a TI [9], conseguindo desvincular o domínio de negócio de tecnologia e modelos específicos(Como linguagens de programação e/ou sistemas operacionais) e proporcionar a empresa vantagens de acompanhar as mudanças exigidas para seu campo de atuação(negócio) sem que sobrecarregue a utilização dos recursos de TI[9].

A origem do termo SOA, alguns dizem, que foi criado por Alexander Pasik[1], no ano de 1994[12]. Outros dizem que termo foi criado no ano de 2000, oriundo de estudos entre as companhias Microsoft, IBM e outras empresas[9].

O OASIS(*Organization for the Advancement of Structured Information Standards*) define SOA como um conjunto de operações que são acessíveis pela rede através de algum formato padronizado. Estes requisitos são ativados em qualquer lugar em um ambiente computacional dinâmico e/ou móveis onde provedores de serviços computacionais oferecem um conjunto de serviços. Estes requisitos podem estar sendo utilizados por mais de um domínios de aplicação[10]. Uma SOA facilita aos requisitantes identificarem, procurarem e utilizarem serviços apropriados em qualquer momento entre o conjunto de

[1] É um analista do Grupo Gartner (http://www.gartner.com/technology/home.jsp)

serviços oferecidos pelos provedores de serviços[10].

Em uma arquitetura SOA a integração de aplicações já existentes com as novas aplicações, pode ser feita com o uso de serviços *web*[30], DCOM (*Distributed Component Object Model*), SOAP (*Simple Object Access Protocol*), REST (*Representation State Transfer*), CORBA (*Common Object Request Broker Architecture*) e RPC (*Remote Procedure Call*)[8].

O modelo operacional triangular SOA[9], representa a iteração entre os três elementos na arquitetura SOA e que é apresentada na Figura 3.1. A descrição desses três elementos é a seguinte:

1. **Provedor de Serviço:** É responsável por fornecer o serviço e toda infraestrutura de acesso ao mesmo. Ele responde por requisições internas(em uma rede interna, uma Intranet) bem como para requisições externas(acesso à Internet)[9];

2. **Consumidor de Serviço:** Este consome o serviço disponível pelo **Provedor de Serviço** quando atende às suas necessidades. O consumidor de serviço pode ser uma aplicação(*software*) em busca de algum serviço complementar para concluir uma tarefa[9];

3. **Registro de Serviço:**É responsável por manter as informações sobre os serviços disponíveis em seus repositórios pelas organizações[9].

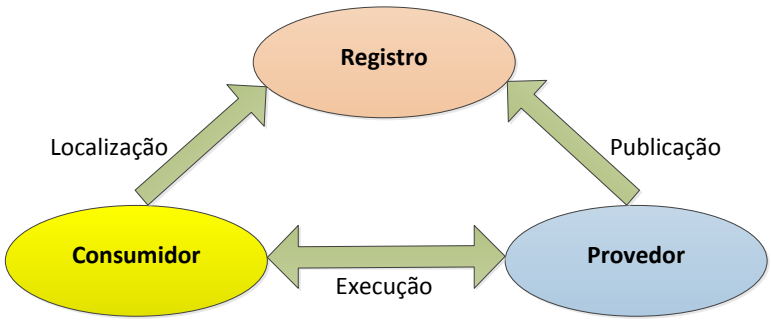


Figura 3.1: Modelo triangular SOA[9]

3.2 Definição de SOA

Há muitas definições para o termo SOA[12]. Alguns conceitos de SOA se confundem com os conceitos de serviços *web*. O autor Papazoglou define SOA, referindo-se ao termo de Computação Orientada a Serviço(SOC) e o define como um paradigma de computação e que usa serviços como os principais elementos para desenvolver soluções e aplicações em sistemas distribuídos[10].

Thomas Erl, um outro autor na área de computação orientada a serviço, dá a seguinte definição sobre SOA [31]:

> *"A SOA estabelece um modelo arquitetônico que visa aprimorar a eficiência, a agilidade e a produtvidade de uma empresa, posicionando os serviços como os principais meios para que a solução lógica seja representada no suporte à realização dos objetivos estratégicos associados à computação orientada a serviços."*

Na Figura 3.2[10], apresenta-se uma arquitetura orientada a serviço interpretada por diversos níveis de abstração ou domínios de negócios que reúnem processos com funções e tarefas em comum, um domínio é formado por processos de negócios que por sua vez são composições de diversos serviços de negócios que por sua vez são construídos sobre uma camada de serviços de infraestrutura e estes são implementados por componente de software[10]. O sistema de *back-end*, consiste de sistemas integrados de gestão empresarial como o ERP(*Enterprise Resource Planning*), CRM(*Customer Relationship Management*)e o SAP (*Statistical Analysis System*)

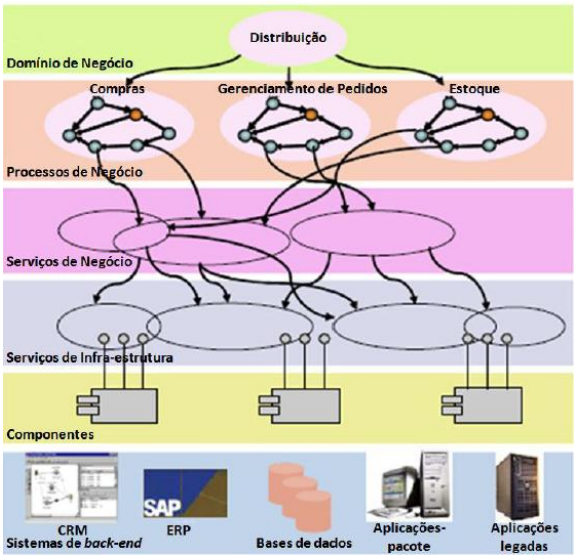


Figura 3.2: Hierarquia de arquitetura SOA[10]

3.3 Modelagem de processos em SOA

Nesse capítulo apresenta-se os procedimentos necessários para modelar processos de negócios ou processos de serviços para uma arquitetura orientada a serviço. Similar a UML (*Unified Modeling Language*), os procedimentos adotados, como análise do conjunto de requisitos de análise de processos orientados a serviços, análise e estudo de serviços legados e outros procedimentos que devem ser seguidos para a realização da análise de projetos de serviços ou modelagem orientada a serviço, haja vista, que há vários métodos de análise orientada a serviço mas entretanto não há ainda uma padronização para esse tipo de modelagem[10].

A Figura 3.3 ilustra um sistema orientado a serviço, onde tem-se um consumidor de serviço, um provedor de serviço e um repositório de serviço.

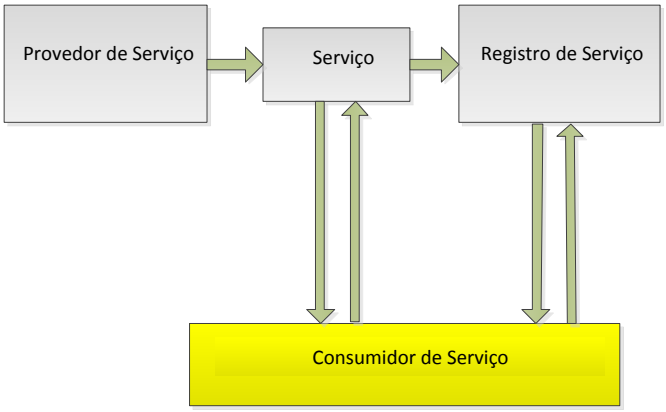


Figura 3.3: Arquitetura Orientada a Serviços[7]

A descrição desses componentes nessa Figura 3.3 são as seguintes:

- **Serviço:** É a entidade que realiza uma tarefa quando ela é solicitada por seu consumidor;
- **Provedor de Serviço:** É a entidade que cria, disponibiliza e descreve um serviço;
- **Registro de Serviço:** É a entidade que localiza a descrição dos serviços publicada pelo provedor de serviços e
- **Consumidor de Serviços:** É a entidade que solicita alguma função para realizar alguma tarefa ou solicita uma iteração com outro serviço.

3.4 Fundamentação teórica

A atividade de modelagem orientada a serviço faz-se necessário devido a dependências nos dias atuais do uso dos serviços dos sistema de informação ou da Tecnologia da Informação com a criação de novos serviços e uso de serviços já prontos ou sistemas legados. Diante deste novo cenário há uma necessidade de um planejamento, um estudo, uma modelagem dos serviços existentes com criação de novos[32].

Nesta seção será definido a modelagem orientada a serviço com as suas perspectivas e visões sobre a modelagem orientada a serviço e as disciplinas necessárias para realizar um boa modelagem orientada a serviço[11].

3.4.1 O que é Modelagem Orientada a Serviço?

O autor Michael Bell[11], define atividade de modelagem orientada a serviço como:

> *"A atividade de modelagem está tipicamente encaixada na fase de planejamento de praticamente qualquer projeto ou iniciativa de desenvolvimento de software, ao qual uma organização é capaz de conduzir. O paradigma de modelagem abrange as disciplinas de análise, desenho e arquitetura que serão seguidas durante o projeto dado"*

O processo de modelagem orientada a serviço inicia-se com a construção de uma pequena cópia, que pode ser feita em uma folha de papel, como um rascunho, com um conjunto de pessoas participando da elaboração em sessões de estudos da análise, desenho e arquitetura. Podem ser utilizadas ferramentas de modelagens do *software* para ajudar na visualização de soluções. Em resumo, a modelagem orientada a serviço necessita de uma criação de um mundo virtual para que os componentes do *software* interajam e proporcionem soluções viáveis para os estudos realizados pela equipe de desenvolvimento[11].

3.4.2 Visões das partes interessadas sobre a modelagem

Segundo o autor Michael Bell[11], o processo de modelagem orientada a serviço é administrado pelo governo do SOA[2] e por partes, visões do centro de excelência SOA que produz boas práticas e orientações para um projeto de modelagem orientado a serviço.

[2]Governo do SOA entende-se como o termo utilizado como um conjunto de práticas, padrões e relacionamentos estruturados que os administradores usam na gerir modelos de serviços em SOA.

A Figura 3.4 apresenta a visão da modelagem de negócios e a visão da modelagem tecnológica. Nestas duas visões ou partes são apresentados conceitos e os profissionais que nela trabalham, destacando a necessidade das duas trabalharem em conjunto para executar a modelagem em serviço e alinhar a infraestrutura de TI as necessidades dos negócios de uma organização[11], [10]. Logo a seguir a descrição de cada visão é apresentada:

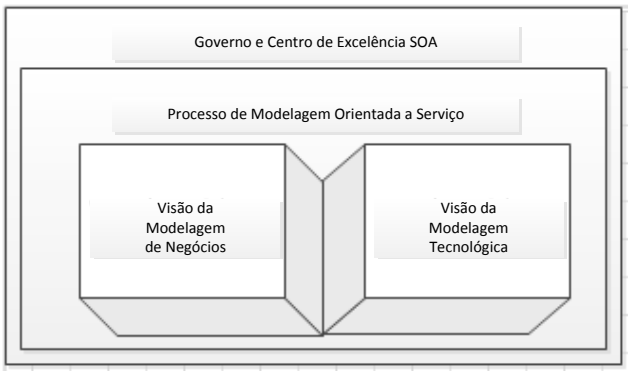


Figura 3.4: Perspectiva da Modelagem Orientada a Serviço[11]

- **A visão da modelagem interessada em negócios** - Esta visão representa o departamento de pessoal de uma empresa que não apenas compreende as implicações financeiras de um projeto ou uma iniciativa empresarial, mas tem controle sob os vários processos de negócios de uma organização. Esta visão discute sobre o domínio do problema, apresenta as exigências da organização e propõe as soluções para os negócios. Nessa perspectiva de visão observa-se que o profissional de negócio deveria ter participação e opinar e, lógico, estar familiarizado com uma linguagem de modelagem orientada a serviço e fornecer, na fase de análise de requisitos, informações importantes para a elaboração do projeto em estudo. Esta visão é representada por um conjunto de profissionais como: gerente de produtos, gerente de negócios, analistas financeiros, arquitetos de negócios, modeladores de negócios e os executivos de alto nível, como CIO(*Chief Information Officer*) e CEO(*Chief Executive Officer*)[11].

- **A visão da modelagem interessada em tecnologia** - Esta visão representa com os aspectos tecnológicos relacionados ao processo da modelagem orientada a serviço. Esta visão contribui para os aspectos de análise e desenho do serviço e também para as demais atividades de arquitetura da modelagem. Executa funções como estabelecimento de estratégias de integração dos recursos, análise de consumo, reutilização e prestação de serviços. Tem a finalidade de alinhar o a infraestrutura de TI com modelo de negócios da organização com estratégias de

negócios e exigências da empresa[10]. Esta visão é representada pela equipe de administração técnica de TI, arquitetos de aplicativos, analistas de sistemas, desenvolvedores, modeladores de serviços, modeladores de dados e arquitetos de banco de dados[11].

3.4.3 Disciplinas de Modelagem de Orientada a Serviço

As disciplinas de modelagem de orientada a serviço possibilitam oferecer as melhores práticas, padrões e políticas para facilitar as atividades de desenvolvimento do serviço-orientado durante o ciclo de vida. Essas disciplinas identificam processos centrais nos quais o grupo de funcionários dos negócios e da tecnologia da informação precisam estar envolvidos quando iniciarem a modelagem do serviço[11].

O processo de modelagem orientada a serviço faz com que o serviço desejado ou um conjunto de serviço passe da fase inicial de idealização até a fase de execução, com tudo pronto e funcionado como o planejado. Essas transformações são guiadas pela disciplina e métodos que devem ser seguidos para a conceituação e descoberta do serviço desejado, integração com novos serviços já existentes, projeto do serviço orientado já definido com análise e desenho e a definição da arquitetura conceitual e lógica.

A sequência de estados na qual a modelagem do serviço orientado passa está mostrado na Figura 3.5, onde observa-se que o estado de serviço conceitual passa para os próximos estados mas entretanto pode retornar para o estado o inicial ou o anterior com o propósito de melhorar agilidade nos negócios devido as necessidades de melhorar o produto desejado.

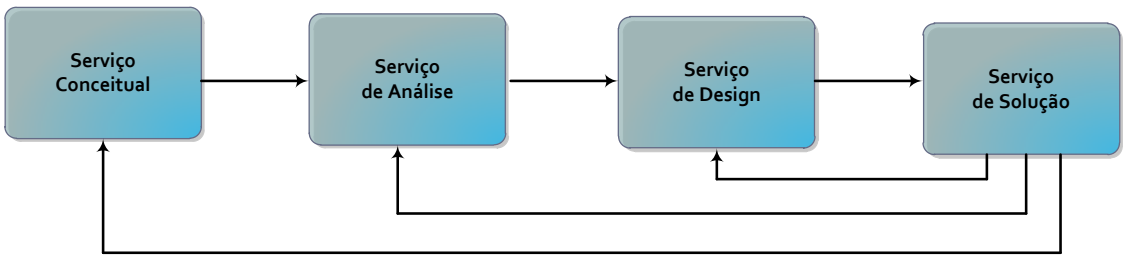


Figura 3.5: Metamorfose Orientada a Serviço[11]

Na Figura 3.6 apresenta as disciplinas necessárias para ter-se sucesso em uma modelagem orientada a serviço. A descrição dessas disciplinas são estas:

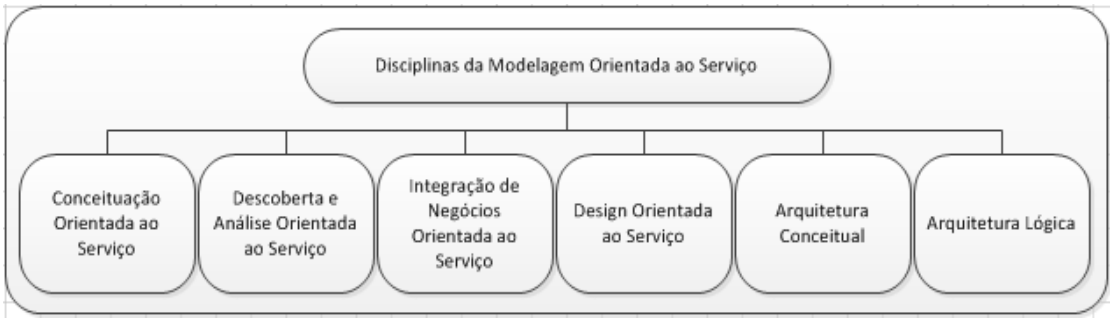


Figura 3.6: Disciplinas Orientada a Serviço[11]

- **Conceituação orientada a serviço** - Nesta etapa serão definidos as análises da atribuição e a identificação do serviço conceitual. A análise atribuição descreverá conscientemente as exigências, especificação e problemas do negócio(serviço) para que seja produzida uma documentação com os principais atributos que serão utilizados para identificar os principais serviços conceituais da modelagem orientada a serviço enquanto que a identificação de serviço conceitual servirá para descobrir a relacionamento dos conceitos e identificar os serviços conceituais[11]. A Figura 3.7 apresenta esses dois itens.

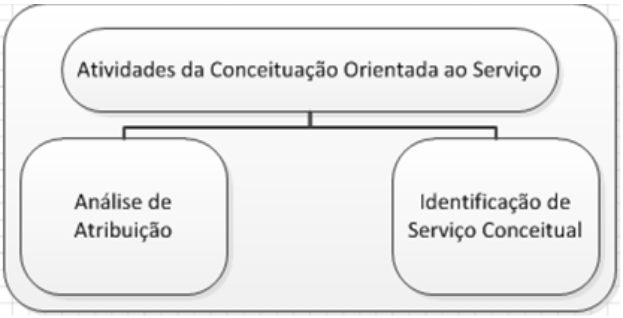


Figura 3.7: Processo de modelagem conceitual orientada a Serviço[11]

- **Descoberta e análise do serviço orientado** - Nesta etapa será feita a descoberta de serviços que irão contribuir para a solução dos negócios e tecnologias. Como pode ocorrer erros na etapa de conceituação de serviços e consequentemente não fornecendo uma solução completa para o projeto de descoberta do serviço desejado para o modelo planejado. Nesse caso será necessário a realização de três tarefas, conforme está mostrado na Figura 3.8. Essas tarefas são: **Tipificação e perfilação de serviços** que classifica o serviço desejado baseado na descrição que o mesmo desenvolve em uma organização, a **Análise dos serviços** que

determina onde o serviço é viável para facilitar a proposta da análise de serviço e a **Modelagem de análise de serviços** que conclui a descoberta e análise do serviço-orientado.

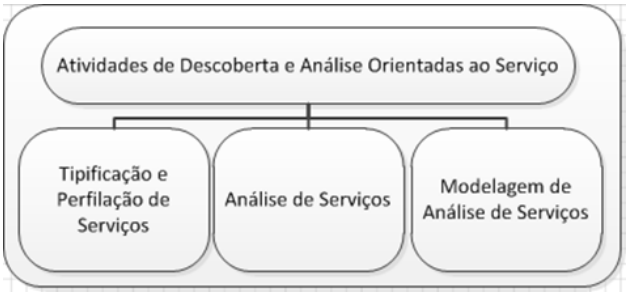


Figura 3.8: Processo de modelagem de descoberta orientada a serviço[11]

- **Integração de negócios orientada a serviço** - Nesta etapa será feita uma ação contínua para integrar as organizações dos negócios e a tecnologia[11] com o objetivo de alinhar os modelos de negócios da empresa à infraestrutura de TI[10]. Nesta etapa há três atividades principais: a integração contextual de negócios, a integração estrutural de negócios e a modelagem da integração de negócios, conforme está mostrado na Figura 3.9.

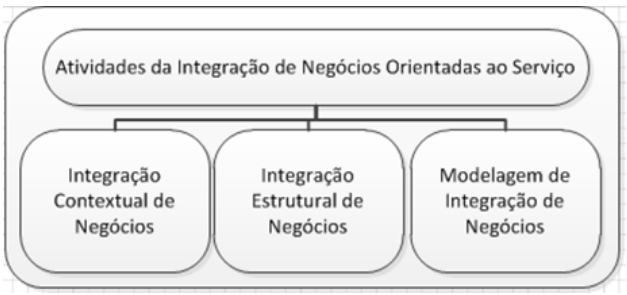


Figura 3.9: Processo de modelagem de integração de negócios orientadas a serviço [11]

- **Design orientado a serviço** - Nesta etapa será feito o desenho do serviço orientado com uma apresentação de projeto onde mostra-se os pontos de troca de informações entre os serviços já definidos com os seus consumidores correspondentes[11]. Nesta etapa, conforme o autor[11], é facilitado o estabelecimento do relacionamento lógico, da interface, interação, colaboração e aspectos relacionados ao consumidor e serviços definidos na etapa conceitual do serviço. Nesta etapa há três atividades principais, conforme está mostrado na Figura 3.10: a modelagem do relacionamento do serviço, a modelagem da estrutura lógica e a modelagem comportamental do serviço. A **Modelagem da relação dos serviços** ajuda a verificar a variedade de associações que o serviço mantém com seus serviços profissionais e consumidores. Trata de assuntos como de como podemos eficientemente coordenar a troca de mensagens e informações entre serviços e consumidores e como administrar a exposição pública dos serviços e garantir acesso seguro aos

consumidores assinantes. A **Modelagem da estrutura lógica** apresenta a solução que está sendo proposta a partir de uma perspectiva estrutural e a **Modelagem comportamental do serviço** apresenta a funcionalidade dos negócios e técnica que os serviços oferecem aos seus consumidores[11].

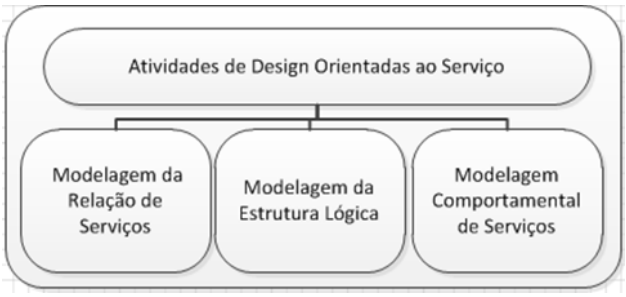


Figura 3.10: Processo de modelagem de design orientado a Serviços [11]

- **Arquitetura conceitual e lógica** - Nesta etapa, já com a definição os conceitos dos serviços, análise, integração de negócios e nas peças obtidas no *Design* orientado a serviço. São criados produtos com os serviços identificados com os consumidores já conhecidos. Nesta etapa há duas atividades principais: a modelagem da arquitetura conceitual e a modelagem da arquitetura lógica. A **Modelagem da arquitetura conceitual** ajuda a projetar um ambiente tecnológico que cumpre com as estratégias, melhores práticas e padrões para os serviços e a **Modelagem da arquitetura lógica** que trabalha com os aspecto lógico do ambiente de produção. A Figura 3.11 apresenta estas duas atvidades[11].

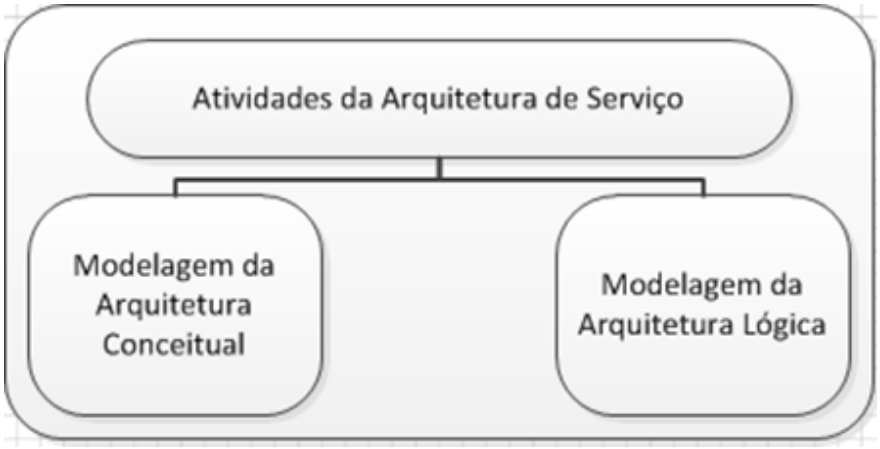


Figura 3.11: Processo de modelagem de arquitetura de serviços[11]

3.4.4 Estrutura de Modelagem Orientada a Serviço

A visão da estrutura de modelagem e serviço está mostrada na Figura 3.12 apresenta os vários componentes que ajudam na modelagem de serviço orientado a serviço[11].

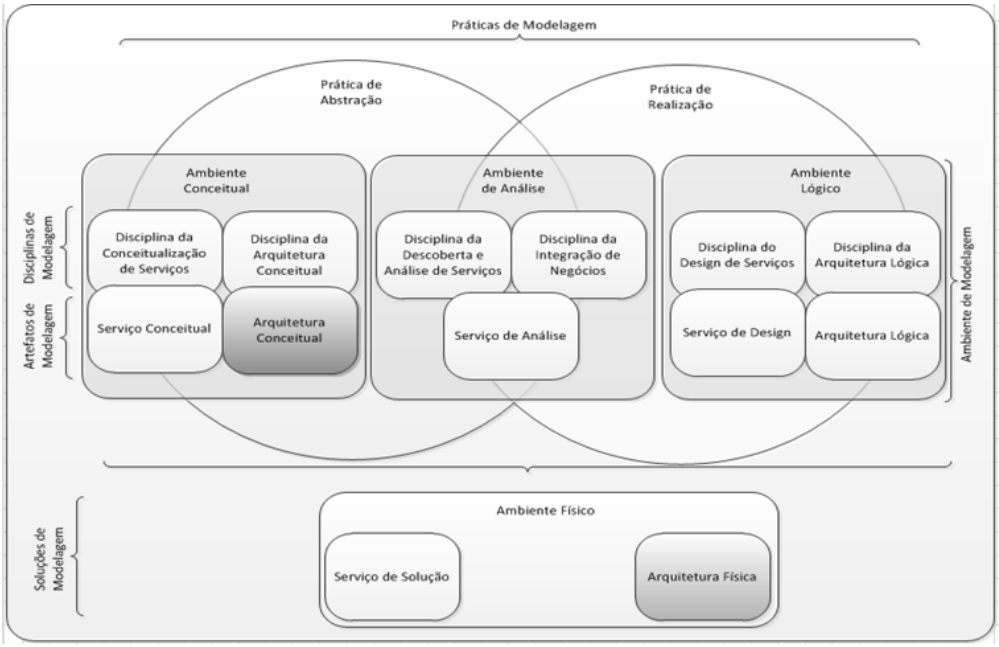


Figura 3.12: Estrutura de modelagem do serviço orientado[11]

3.5 Gerenciamento de processos de negócios

O gerenciamento de processos de negócio BPM (*Business Process Management*) é um conceito ou um termo para designar todas as atividades relacionadas ao gerenciamento dos processos de negócios em uma organização haja vista que os serviços utilizados em uma modelagem orientada a serviço vem de processos de negócios[12].

Estas atividades estão relacionadas à identificação, planejamento, monitoração, controle e melhoria dos processos de negócios que visam alcançar metas do melhor rendimentos em uma empresa atendendo o custo/benfício para cada processo[33] e seguindo as definições mencionadas nas seção 3.4 para realizar uma boa modelagem orientada a serviço. Em uma implementação SOA, os serviços fazem parte de um ou mais dos processos de negócios envolvidos [12]. A modelagem de processos é um conjunto de ações ou atividades que as corporações executam para que os seus processos de negócios tenham uma melhor visibilidade diante do problema em estudo. As ações tomadas por estas corporações diante desses processos de negócios fornecem uma diretriz, controle e pontos de tomada de decisão para o correto desenvolvimento do negócio em estudo[12].

Uma arquitetura de negócios é composta pela descrição de: funções de negócios em que esta descreve as funções da organização ou corporação com o meio externo e

dos processos de negócios que descreve o fluxo de trabalho que auxilia as ações das funções nas organizações.

3.5.1 Definição de gerenciamento de processo de negócio

O autor Nicolai M. JOSUTTIS[12], apresenta a seguinte definição para gerenciamento de processos de negócios:

> *"A modelagem de processos de negócios é um conjunto de práticas ou tarefas que as empresas podem executar para descrever visualmente todos os aspectos de um processo de negócio, incluindo o seu curso, controle e pontos de decisão, gatilhos e condições para execução das atividades, o contexto em que uma atividade executa e os recursos associados."*

Um serviço está diretamente associado a um ou mais processos de negócios. Esses processos por sua vez estão relacionados a modelagem de serviço orientado[11] e são utilizadas ferramentas para compor e/ou orquestrar vários processos de negócios ou serviços em uma modelagem orientada a serviço[12]. A seguir serão apresentados cenários de processos de negócios e a necessidade de ter-se uma ferramenta para gerir todo esse processo.

Um processo de negócios pode ser dividido em subprocesso de negócios e estes por sua vez em atividades ou serviços que foram conceituados ou descobertos no processo de modelagem orientada a serviço e projetado, conforme descrito na subseção 3.4.3, para participarem da modelagem adequada às necessidades da empresa, conforme está mostrado na Figura 3.13.

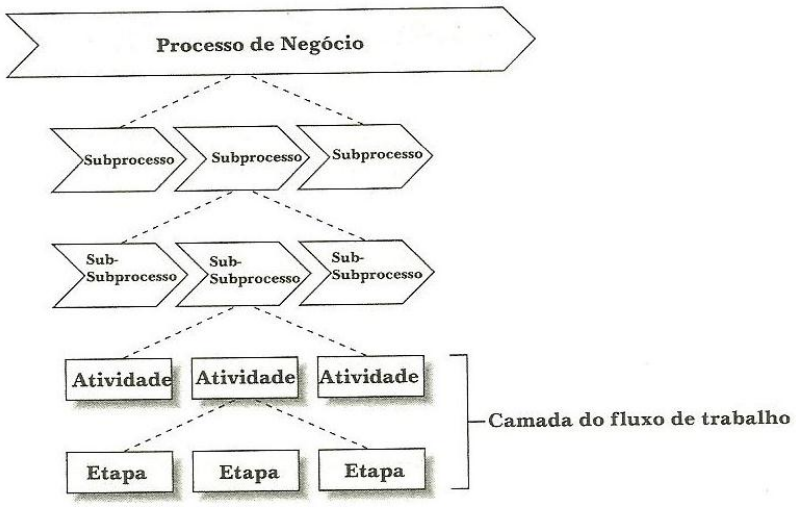


Figura 3.13: Visualização de um processo de negócios[12]

Na Figura 3.14 apresenta-se um cenário de atividades de um gerenciamento de processos de negócios com duas visões. Uma visão do lado do serviço e a outra do lado processo de negócio. Observa-se que alguns serviços da organização, (como ***CRM, Pedidos e logísticas e Faturamento***), tornaram-se após a conceituação e descobertas de serviços em serviços básicos de dados e serviços compostos e estes por sua vez tornaram-se subprocessos de negócios no processo **Pedido do Cliente** da modelagem orientado a serviço deste cenário.

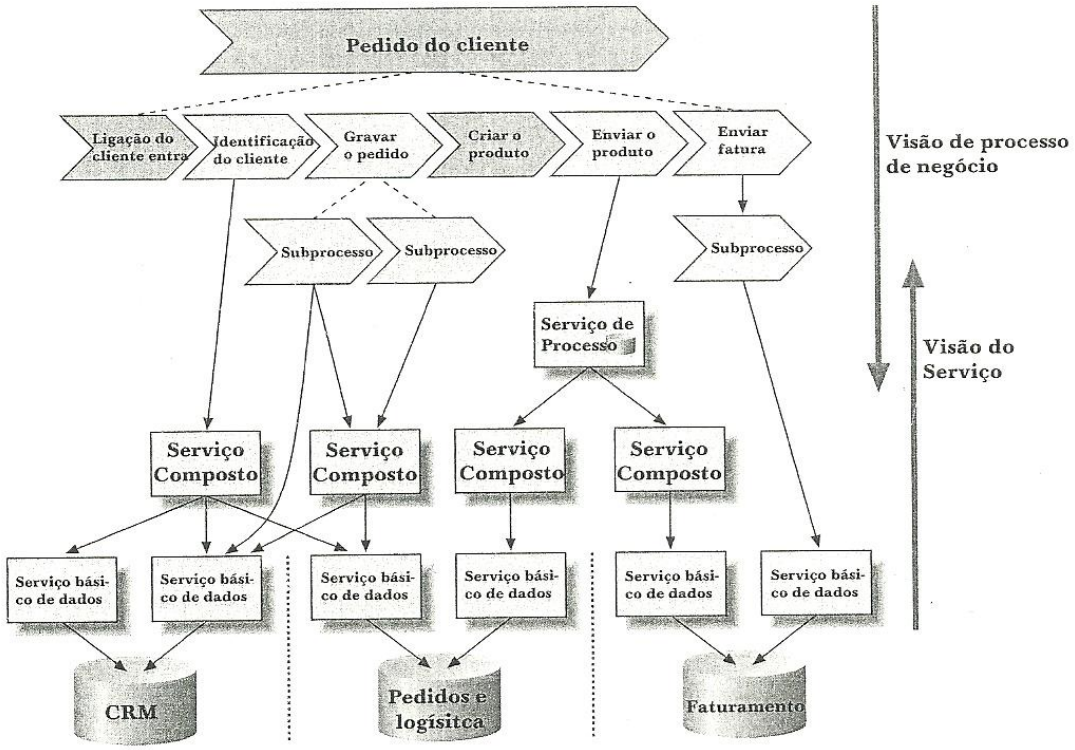


Figura 3.14: Cenário do fluxo de atividades de um processo de negócios[12]

Na Figura 3.15, apresenta-se um cenário idêntico ao cenário da Figura 3.14, onde mostra-se uma área onde atua a ferramenta de gerenciamento do processo de negócio. Há uma grande variedade de ferramentas para gerenciar um processo de negócios, ferramentas gratuitas e comerciais.

Há um *plugin* para a IDE *Eclipse* denominado *ActiveBPEL*[34] que junto com o *Apache* ODE(*Orchestration Director Engine*)[35] constituem uma aplicação que implementa e executa processos BPEL. Há também, o BonitaBPM[3] [36] e o *Oracle Business Process Management*[32], que é um conjunto completo de ferramentas para criar, executar e otimizar processos de negócios. Todas essas ferramentas utilizam a linguagem WS-BPEL (*Web Services Bussiness Process Execution Language*) para gerenciar os processos de negócio. Na seção 3.6 será descrita a linguagem WS-BPEL e as atividades de um processo BPEL.

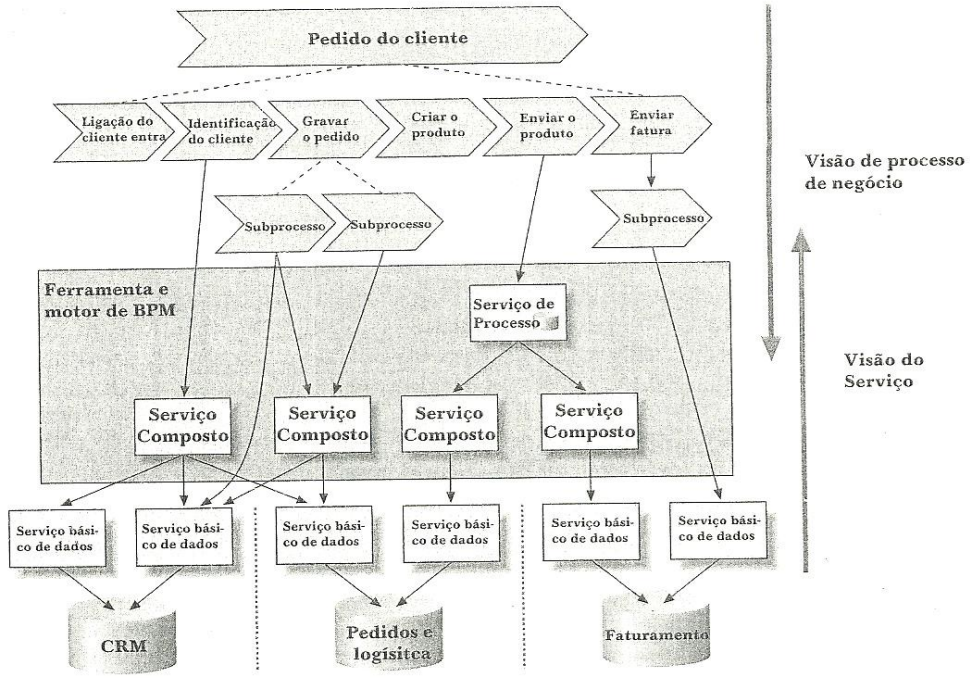


Figura 3.15: Arquitetura Orientada a Serviços[12]

[3] O BonitaSoft pertence ao consórcio OW2 (http://www.ow2.org/) e também possui apoio das empresas *Bull* (http://www.bull.com/) e a *France Telecom* (http://www.orange.com/)

3.6 A Linguagem WS-BPEL

A linguagem WS-BPEL(*Web Services Bussiness Process Execution Language*) é um padrão mantido pela OASIS(*Organization for the Advancement of Structured Information Standards*) para executar processos de negócios. Essa linguagem foi inicialmente projetada em agosto de 2002 pelas empresas IBM, Microsoft e BEA como uma combinação de Web Services FLow Language da IBM e XLANG (eXtensible LANGuage) que é uma notação para automação de processos de negócios usando serviços *web*[8].

Em 2003 foi lançada a versão 1.1, junto com as empresas SAP e SIEBEL. No ano de 2003, o OASIS, recebeu uma primeira versão para ser padronizada. A versão 2.0 foi lançada no ano de 2007.

É uma linguagem padrão baseada em XML para a execução de processos de negócios que podem ser feitos com orquestração ou coreografia de serviços. Um processo WS-BPEL pode ser síncrono ou assíncrono. Um processo WS-BPEL síncrono bloqueia o cliente até que o processo finalize o seu processamento. Já em um processo assíncrono o cliente não é bloqueado.

3.6.1 Atividades em um processo WS-BPEL

Há um conjunto de sintaxe que descreve as atividades(instruções) que são realizadas para manipular os elementos envolvidos em um processo WS-BPEL. São instruções como uma invocação (<**invoke**>) de serviço *web*, manipulação variáveis (<**assign**>), recepção de dados de um serviço *web* em modo assíncrono(<**receive**>), envio de resposta em modo assíncrono(<**reply**>) e encerramento do processo BPEL(<**exit**>). Uma lista com as atividades primitivas encontram-se na Tabela 3.1 e as atividades estruturadas encontram-se na Tabela 3.2.

Primitiva	**Significado**
`<invoke>`	Invoca um serviço *web*
`<receive>`	Aguarda a resposta de um cliente *web*
`<reply>`	Gera resposta síncrona *web*
`<assign>`	Manipula dados *web*
`<throw>`	Indica faltas ou exceções *web*
`<wait>`	Espera por certo tempo *web*
`<terminate>`	Finaliza um processo *web*
`<flow>`	Usada para definir um conjunto de atividades que podem ser invocadas em paralelo
`<pick>`	Aguarda por um evento

Tabela 3.1: Atividades primitivas básicas usadas em uma composição de serviço[8]

Estruturada	Significado
<sequence>	Seqüência de atividades a serem invocadas *web*
<flow>	Define um grupo de atividades que serão invocadas em paralelo
<if>	Define uma condição
<while>	Define uma repetição com uma condição no início
<repeatUntil>	Define uma repetição com uma condição no final
<forEach>	Define uma repetição programada
<switch>	Permite aplicar alguma condição na atividade
<pick>	Permite selecionar um de vários caminho para a atividade

Tabela 3.2: Atividades estruturadas básicas usadas em uma composição de serviço[8]

Na Figura 3.16 tem-se o projeto de um processo de negócio com o uso do BPEL *Designer* da IDE Eclipse. Nesse processo observa-se os seguintes serviços *web*: *WsSniffer, WsFormater, WsAnalyser e WsSender*. Nesse processo de negócio os serviços *web* utilizados precisam de uma seqüência para fornecer o resultado desejado.

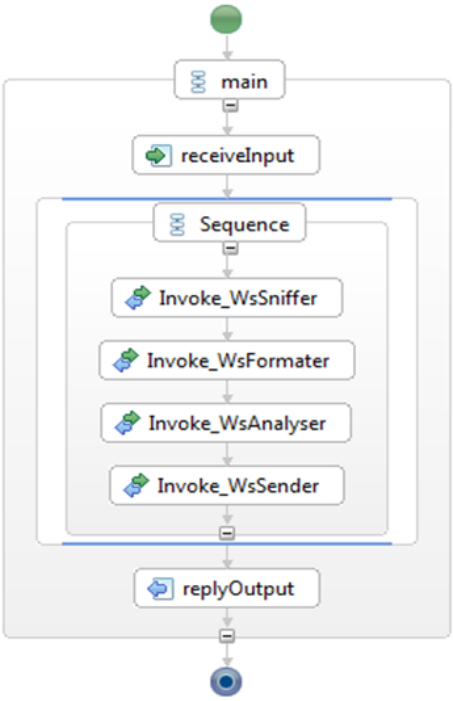


Figura 3.16: Design usando o BPEL Eclipse

O exemplo de projeto do processo BPEL na Figura 3.16, mostra o início do processo BPEL com a entrada de uma mensagem de um fonte externa que é recebida em (***receiveInput***) e logo em seguida uma atividade estruturada (***sequence***) define a ordem de execução dos serviços *web*: ***WsSniffer, WsFormater, WsAnalyse e WsSender***. Após a execução desse serviços *web*, será executada a atividade básica (***replyOutput***) que enviará uma resposta sobre esse processo para uma fonte externa e logo em seguida encerrará o processo BPEL.

3.6.2 Código de um processo BPEL

Um arquivo de um processo BPEL está mostrado abaixo, onde pode se observar o uso de algumas primitivas das atividades de um processo BPEL. Tem-se também o ***partnerLink*** e ***port type***. Um ***partnerLink*** define um canal de comunicação entre parceiros de negócios através do mapeamento entre um processo BPEL e o ***port type*** do serviço definido na linguagem de descrição do serviço *web*.

No apêndice A.1 está listado partes do código BPEL, com o nome do processo, a definição do ***partnerlink***, do ***port type*** e a declaração dos serviços *web* utilizados e que são listados nesse apêndice.

No apêndice A.2 está listado o código completo do processo BPEL mostrado na Figura 3.16.

3.7 Ciclo de vida do modelo orientado a serviço

Na modelagem de um desenvolvimento SOA faz-se necessário planejar e implementar os passos e as etapas que os serviços deverão passar para que haja êxito na execução dos serviços em conjunto. Esse conjunto de serviço terá uma definição e uma sintonia de como deve funcionar e quais serviços precisam ser escolhidos para participarem do conjunto completo da modelagem. Esse conjunto de ações chama-se de ciclo de vida e este serve para guiar e determinar passos para construção, implantação e manutenção de *software*. Assim como em qualquer processo de desenvolvimento de software, o paradigma da arquitetura SOA possui ciclo de vida [11],[31]. O desenvolvimento de uma arquitetura SOA pode ser descrita por meio do ciclo de vida de desenvolvimento de serviço[31].

Os autores Papazohlou e Van Den Heuvel[10] descrevem que uma arquitetura orientada a serviço consiste de sistemas em múltiplas camadas e com várias tecnologias envolvidas, o que exige uma grande quantidade de requisitos de negócios o que leva a um processo de desenvolvimento baseado em serviços para realizar a especificação, construção e refinamento dos processos de negócios. Para esses autores, o método de desenvolvimento orientado a serviço deve atentar-se aos processos de negócios. O baixo acoplamento e alta coesão entre serviços são os dois itens relacionados por esses autores, para que os processos de negócios utilizados com base nos serviços conceituados e ou definidos na fase inicial de modelagem ou os serviços já existentes sejam úteis e confiáveis[10].

O ciclo de vida apresentado por Papazohlou e Van Den Heuvel[10] apresenta um método de análise que faz parte de um processo cíclico e que é executado em iterações.

Na Figura 3.17 é apresentado o ciclo de vida de um projeto de solução orientada a serviço [10].

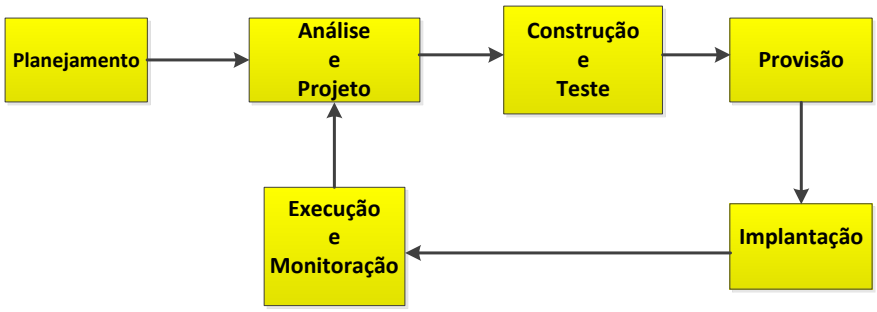


Figura 3.17: Ciclo de vida de desenvolvimento SOA[10]

Abaixo uma breve descrição de cada etapa:

1. **Planejamento** - Esta etapa, consiste do planejamento do(s) serviço(s) desejado(s) com o levantamento da viabilidade do mesmo[10]. Mais atividades são desenvolvidas nessa etapa: Análise do serviço desejado, definição de requisitos, análise do ambiente tecnológico, análise de custo quanto a viabilidade financeira do serviço desejado e definição de tempo para a execução desse serviço[10];

2. **Análise** - Nesta etapa, são definidos os requisitos do(s) serviço(s) desejado(s). Observando-se os serviços já existentes dentro da organização e quais deverão ser criados[10];

3. **Projeto** - Nesta etapa, os processos e serviços conceituais obtidos na fase de **Análise** serão transformados em especificações de interfaces que serão produzidas independentes da plataforma;

4. **Construção** - Nesta etapa, serão desenvolvidos os serviços definidos na fase de **Projeto** bem como suas interfaces;

5. **Teste** - Nesta etapa, realiza-se a validação dos requisitos, averiguando-se se os artefatos estão conforme exigido nas fases de **Análise** e **Projeto**;

6. **Provisão** - Nesta etapa, o(s) serviço(s) que geram receitas devem ser medidas e realizadas sua devida cobrança pelo seu uso;

7. **Implantação** - Nesta etapa, os serviços planejados serão publicados e estarão pronto para serem executados;

8. **Execução** - Nesta etapa, o serviço planejado e desejado na fase inicial estará em execução;

9. **Monitoração** - Nesta etapa, serão medidos os níveis de serviço dos processos com o objetivo de garantir uma boa qualidade em sua execução com um retorno(*feedback*) do desempenho dos processos de serviço com o objetivo de corrigir os erros e melhorar a performance dos níveis de serviço.

Na Figura 3.18 é apresentado o ciclo de vida de um projeto de solução orientada a serviço segundo o autor **Thomas Erl**[10],[31].

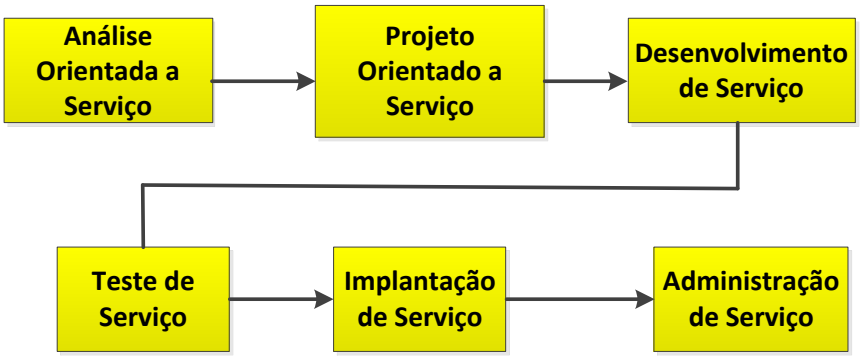


Figura 3.18: Ciclo de vida de desenvolvimento-Thomas Erl[10]

Abaixo uma breve descrição de cada etapa:

1. **Analise Orientada a serviço** - Nesta etapa, será determinado o escopo da Arquitetura SOA. As camada de serviço(orquestração, negócio e aplicação) são mapeadas e serviços preliminares são modelados na forma de serviços candidatos[10];

2. **Projeto orientado serviço** - Nesta etapa, será feito um planejamento de como esses serviços serão construídos. Essa etapa é baseada em padrões e diretamente influenciada pelos princípios de orientação a serviços. Podendo ser utilizadas quatro diferentes tipos de estratégias para projetar um serviço:(Serviço baseado em Entidades, Serviços de Aplicação, Serviços baseados em Tarefas e Serviços de Processos);

3. **Desenvolvimento de serviço** - Nessa etapa, serão considerados os aspectos relacionados ao desenvolvimento e implementação dos serviços desejados, como a linguagem de programação a ser utilizada e outros ambientes de desenvolvimento computacional;

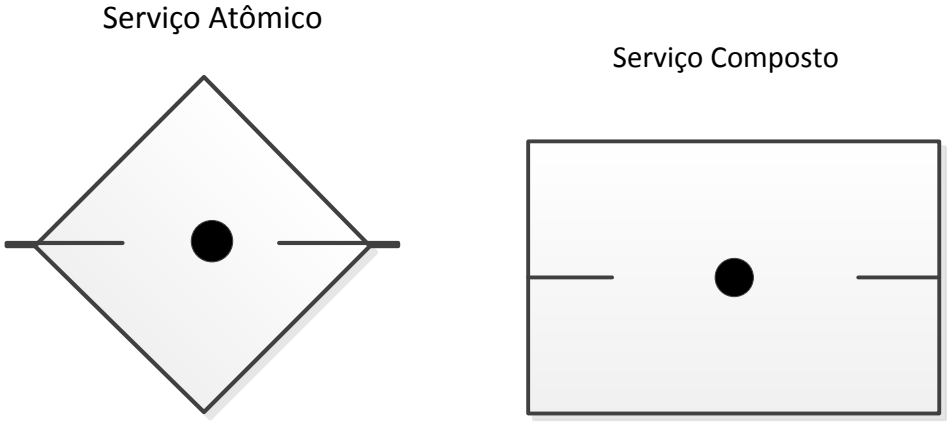
4. **Teste de serviços** - Nessa etapa, os serviços serão testados para obedecer um série de requisitos(Aderência aos padrões, comunicação com diversos consumidores, aspectos de qualidade de serviço-QoS e tratamento de exceções e compensação), haja vista que muitos serviços serão reusados para participar da composição de serviço;

5. **Implantação de serviços** - Nessa etapa os componentes distribuídos que implementam os serviços serão instalados e configurados, junto com as interfaces de serviços e produtos de *middleware* associados em ambientes de produção. Define-se a alocação física de cada componente, avalia-se a capacidade de infraestrutura de atender aos requisitos de desempenho esperados. Define-se também as configurações de segurança e verifica-se as integrações com sistemas e aplicações;

6. **Administração de serviços** - Nessa etapa, será acompanhada a execução dos serviços, com monitoração de seus serviços, seu uso e desempenho com controle da versão dos artefatos relacionados e fornecendo manutenção para que todos os serviços funcionem conforme o planejado.

3.8 Notação dos diagramas de modelagem adotada

A notação da modelagem para orientação a serviço, seguiu a notação apresentada por [11]. A Figura 3.19 apresenta a simbologia de um serviço atômico 3.19(a) e de um serviço composto 3.19(b). Há mais representações de notações utilizadas na linguagem de modelagem em SOA apresentada pelo autor Michael Bell[11], entretanto apenas as notações de serviço atômico e serviço composto serão utilizadas na modelagem SOA nesse trabalho.



(a) Serviço atômico (b) Serviço composto

Figura 3.19: Representação dos serviços atômico e composto[11]

3.8.1 Diagrama de transação circular

As atividades sequenciais exigidas em um processo de negócio faz com que seja necessário o uso de diagrama de transação circular[11]. Esse diagrama mostra a sequência das atividades dos serviços utilizados em uma cadeia de serviços. Um conjunto de serviços atômicos que executam um conjunto de atividades simples e/ou sequenciais podem ser reunidos e serem representados por um serviço composto. Na Figura 3.20, tem-se a representação do diagrama de transação circular, onde observa-se os serviços atômicos(serviços simples) de captura de pacotes na rede mais o de formatação de pacotes, o que resulta em um serviço composto de formatação de pacotes.

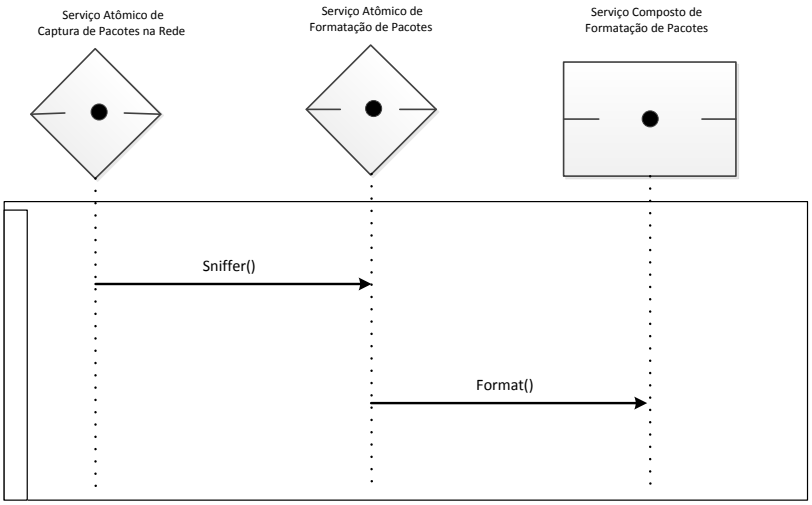


Figura 3.20: Representação da Transição Circular[11]

Em etapas posteriores de um processo de modelagem, a representação do conjunto de serviço atômicos simples como mostrado na Figura 3.20 poderá ser substituída apenas por uma representação do serviço composto desses conjuntos de serviços atômicos em estudo. O serviço composto de formatação de pacotes ou apenas serviço de formatação de pacotes será representado pela Figura 3.21.

Figura 3.21: Representação do serviço composto de formatação de pacotes[11]

3.9 Síntese

Neste capítulo, foram apresentados a teoria sobre Arquitetura Orientada a Serviço - SOA, sua definição, os aspectos que envolvem uma modelagem orientada a serviço, a linguagem WS-BPEL e as ferramentas utilizadas para realizar uma composição de serviços dentro de um processo de negócio.

No próximo capítulo será apresentado o *Framework NIDS-SOA* com os seus conceitos, sua concepção oriunda do sistema IDS-NIDIA.

Capítulo 4

Framework: NIDS-SOA

4.1 Introdução

Para se ter um ambiente computacional seguro em uma empresa de médio e/ou grande porte, implica na utilização de recursos que garantam essa proteção. Esses recursos constam de profissionais qualificados e equipamentos de segurança aliados a adoção de uma política de segurança[1] para a infraestrutura da rede corporativa, com atualização dos sistemas operacionais utilizados na infraestrutura dos servidores (*Windows Server*, *Linux*, BSD(*Berkeley Software Distribution*)) e dos programas utilizados onde são executados os aplicativos de uma rede corporativa(Linguagem PHP, máquina virtual *Java*, Banco de dados relacionais, servidor *web* (HTTP), servidor de nomes BIND(*Berkeley Internet Name Domain*) e outros programas), faz-se também necessário o uso de recursos de proteção como *firewall*, IDS e/ou um *Appliance*[2] de segurança e um *software* que contenha proteção contra vírus, *spyware* e *phishing*.

Todos esses procedimentos de segurança estariam dentro da governança de TI (Tecnologia da Informação)[3].

A Figura 4.1 apresenta um cenário com a infraestrutura de rede de duas **empresas: A e B** constituídas de uma infraestrutura de rede corporativa(Rede *Intranet*, DMZ, *Firewall* e IDS) e um usuário doméstico utilizando um *firewall* em seu computador pessoal. As duas redes da duas empresas e o usuário doméstico se conectam a grande de rede de computadores, a Internet, para buscar dados com o uso de alguma ferramenta de busca, fazer compras na Internet, acessar os serviços de um *Home Banking*, rede sociais, sites de notícias, enviar *e-mails*, acessar dados da empresa em outro local e outras

[1] O uso das normas NBR ISO/IEC 27001, 27002 e 27005 já possibilitam às organizações um guia para uma implementação de um Sistema de Gestão da Segurança da Informação-SGSI, através do estabelecimento de uma política de segurança, controles e gerenciamento de riscos.

[2] *Appliance* é um equipamento onde já se encontram instalados e pré-configurados sistemas de proteção como *firewall*, IDS e sistemas com antivírus.

[3] Governança de TI é um conjunto de práticas, padrões e relacionamentos estruturados que os administradores de redes, gestores e técnicos de uma organização assumem para gerir a infraestrutura da organização com o objetivo de melhorar o custo/benefício na implantação de cada aplicação ou serviço na infraestrutura de rede e conseqüentemente unir os negócios da empresa à TI.

atividades na grande rede.

As redes das empresas **A** e **B** possuem sistema de proteção com *firewall* e IDS-NIDIA. O *firewall*, localizados nas empresas **A** e **B** e no usuário doméstico, protegem suas respectivas redes contra ataques às portas já conhecidas, eles possuem uma proteção limitada, enquanto que o IDS trabalha com base no comportamento do tráfego, baseando-se assinaturas já conhecidas, conforme descrito na seção 2.3. Entretanto estes IDS-NIDIA, localizados na infraestruturas das empresas A e B não compartilham informações entre si[6].

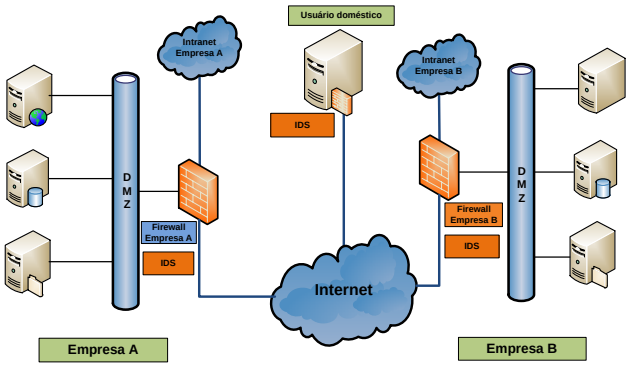


Figura 4.1: Infraestrutura de rede com IDS-NIDIA

4.2 Infraestrutura de rede com o NIDS-SOA

A Figura 4.2 apresenta o mesmo cenário da Figura 4.1 com a substituição do IDS-NIDIA pelo NIDS-SOA que é o *framework* proposto neste trabalho. A vantagem da utilização do NIDS-SOA sobre o IDS-NIDIA é que o primeiro poderá compartilhar informações entre os NIDS-SOA instalados nas empresas A e B, proporcionado dessa forma uma maior proteção contra ataques da rede graças ao compartilhamento de funcionalidades e informações. O compartilhamento dessas informações entre esses NIDS-SOA se dará com o uso da arquitetura orientada a serviço. Nas próximas seções, esse *framework* NIDS-SOA será explicado.

4.3 Objetivo do *Framework* NIDS-SOA

O objetivo de conceber o *framework* NIDS-SOA foi de aumentar as funcionalidades dos IDS existentes com auxílio da arquitetura SOA. Os sistemas IDS's são sistemas isolados e a interoperabilidade entre IDS's de fornecedores diferentes é complexa e de difícil implementação. Os IDS's existentes na literatura, incluindo o IDS-NIDIA são sistemas isolados e não são facilmente reutilizados[20]. Geralmente, estes se comunicam utilizando diferentes protocolos e são criados com paradigmas de programação diferentes.

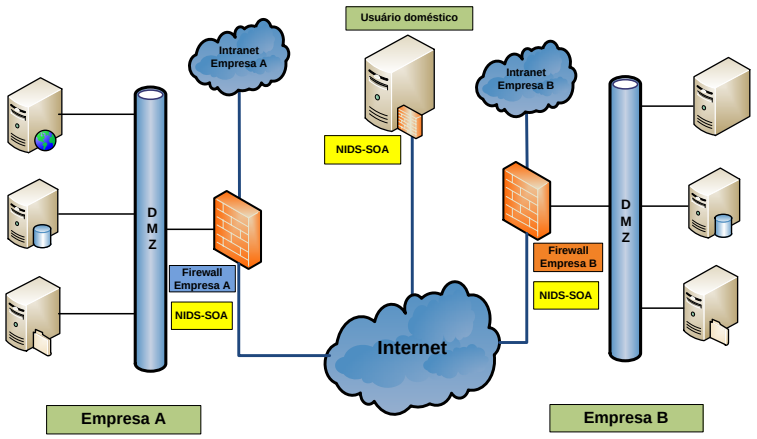


Figura 4.2: Infraestrutura de rede com o NIDS-SOA

A solução proposta é uma arquitetura para suportar a interoperabilidade entre sistemas IDS seguindo uma filosofia SOA. Para validar esta arquitetura, o IDS-NIDIA será adaptado e estendido de acordo com esta arquitetura. Os serviços oferecidos pela arquitetura IDS-NIDIA serão transformados em serviços *web*, com o propósito de oferecer uma composição estática de IDS e reuso das informações de outros IDS's.

4.4 Concepção do NIDS-SOA

Esta seção apresenta o *framework* NIDS-SOA(*Network Intrusion Detection System based on SOA*) idealizado para fornecer as funcionalidades de uso e reuso das informações de um IDS. A Figura 4.3 apresenta a arquitetura do *framework* NIDS-SOA, que tem a finalidade de detectar o comportamento de ataques em uma rede de computadores e realizar um medida para sanar este ataque.

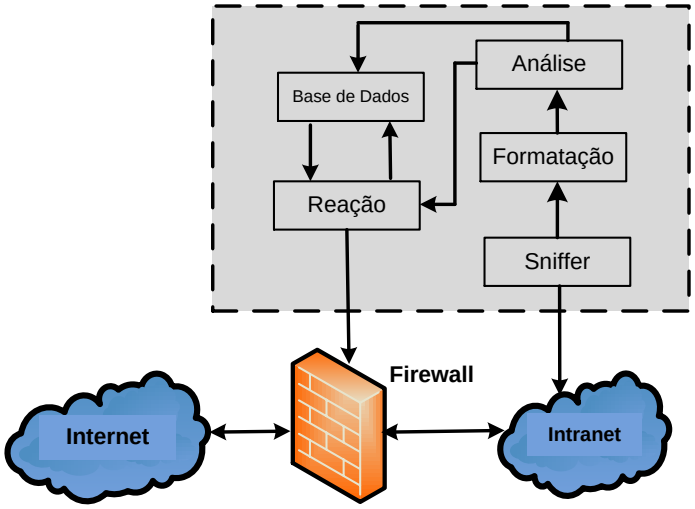


Figura 4.3: Funcionalidades do NIDS-SOA

Essa arquitetura possui os seguintes componentes: ***sniffer*** que realiza a captura de pacotes na rede e/ou em um arquivo de *log* dos servidores. Esses pacotes serão passados para o próximo componente que é o **formatador de pacotes** que formata os pacotes de rede em uma estrutura de dados XML, contendo, por exemplo, algumas informações importantes como IP de origem, IP de destino, portas TCP e UDP e o tempo desse acesso. Os pacotes já formatados são analisados, pelo componente **análise**, com base em algumas regras já definidas (em uma base de dados) e caso encontre alguma regra, uma ação (como o bloqueio do IP ou da porta acessada) é realizada pelo componente de **reação** com o objetivo de negar o acesso desse IP ao sistema. O componente ***Update-DB*** é responsável pela inserção na base de dados sobre esse IP e/ou porta que o mesmo acessou.

A Figura 4.4 apresenta uma arquitetura do *framework* NIDS-SOA com os seus serviços atômicos. Os componentes apresentados na Figura 4.3 e que correspondem às camadas de serviços (captura, formatação, análise, atualização e reação) são agora serviços (*Sniffer* de Rede ou *Host*, *Analyser, Formater, Sender* e *Reaction*).

A representação de serviços mostrados na Figura 4.4 é adotada pelo autor Michael Bell[11], conforme descrito na seção 3.8.

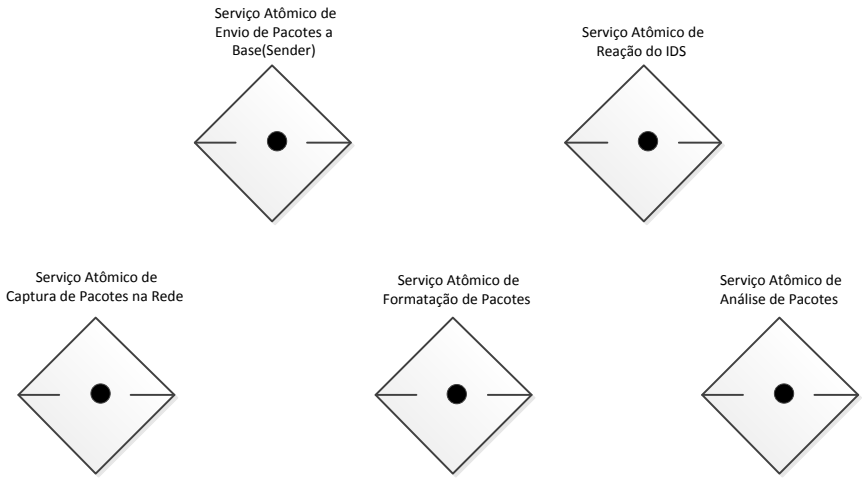


Figura 4.4: Serviços básicos do NIDS-SOA[11]

Cada serviço apresentado nesse *framework* é considerado um serviço básico. Cada um desses serviços pode se unir com outros serviços básicos ou com serviços de outros IDS's, proporcionando assim serviços composto, conforme é mostrado na Figura 4.5, onde tem-se a composição dos serviços atômicos de captura de pacotes da rede (*Sniffer*) com o serviço atômico de formatação desses pacotes, proporcionado o serviço composto de formatação de pacotes.

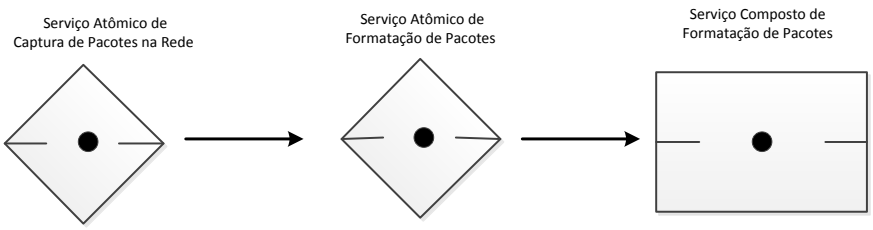


Figura 4.5: Servico Composto de Formatação de Pacotes

4.5 Projeto do *Framework* NIDS-SOA

O *Framework* NIDS-SOA foi projetado baseando-se no IDS-NIDIA[20] e que foi descrito na seção 2.6. A arquitetura do IDS-NIDIA que foi projetada em modelos de camadas, como está mostrado na Figura 4.6, possui as camadas de monitoramento, análise, reação, atualização, armazenamento e administração. Todas essas camadas, exceto a de admintração, possuem um conjunto de agentes que controlam e executam todas as atividades necessárias para o funcionamento do IDS-NIDIA.

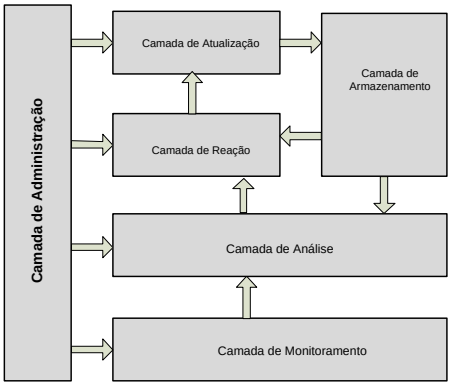


Figura 4.6: Arquitetura do IDS-NIDIA - Modelo de Camadas

4.5.1 Adaptação do IDS-NIDIA ao *Framework* NIDS-SOA

Para a criação do *Framework* NIDS-SOA, o IDS-NIDIA foi acrescentado de uma camada a mais, a camada de formatação. Além disto, em cada uma dessas camadas foi adicionado uma camada de serviço *web* para que o IDS-NIDIA se adptasse às características do *Framework* NIDS-SOA com disponibilidade de fornecer serviço e consequentemente está apto a ser utilizado em uma arquitetura SOA.

A adaptação do IDS-NIDIA ao *framework* NIDS-SOA, com a camada de formatação adicionada e com uma camada de serviço *web* em cada uma das camadas de captura, análise, formatação, reação e envio de dados para uma base dados está mostrado na Figura 4.7.

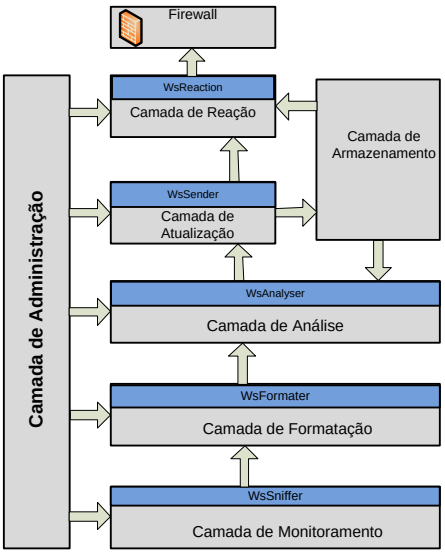


Figura 4.7: Adaptação do IDS-NIDIA ao NIDS-SOA - Acrescido de uma camada web

O *Framework* NIDS-SOA utiliza cinco serviços *web* para realizar as atividades que uma arquitetura orientada a serviço necessite. Cada componente ou módulo da Figura 4.3 terá um serviço *web* associado a sua função. Portanto, o NIDS-SOA mostrado na Figura 4.7 será adaptado e estendido por cinco serviços *web*, cada serviço *web* terá um WSDL atuando dessa forma como contrato de serviço para as outras camadas do *framework*. Nesse diagrama, há uma camada de administração que será realizada pelo administrador do sistema quando este achar necessário.

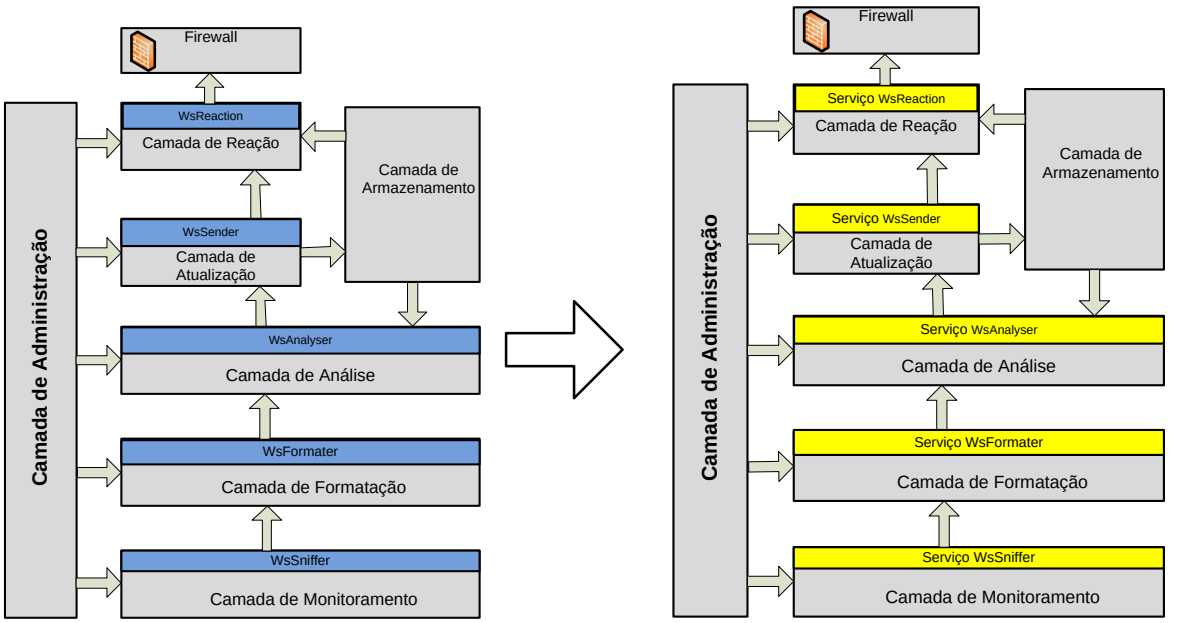
Esses serviços *web* associados a sua linguagem de descrição de serviço em cada camada, conforme está apresentado na Figura 4.7, serão chamados de serviços para atender às características de arquitetura orientada a serviço do *framework* NIDS-SOA. Na próxima seção será descrita a função de cada serviço para este *framework*.

4.5.2 Serviços *web* e serviços em SOA

A camada de serviço *web* acrescentada em cada camada do modelo IDS-NIDIA tem por objetivo fazer com que as funcionalidades de cada camada funcionem no *Framework* NIDS-SOA, observando-se os princípios de uma arquitetura orientada a serviço. Cada camada com o serviço *web* proporcionará a integração das funcionalidades entre os IDS's.

Como visto na seção 2.9 outras tecnologias poderiam ser utilizada para realizar essa integração e proporcionar as funcionalidades exigentes em uma arquitetura orientada a serviço.

A Figura 4.8 apresenta um comparativo do modelo de camadas do IDS-NIDIA, apresentado os serviços *web* 4.8(a) em cada uma das camadas e o mesmo modelo de camadas do IDS-NIDIA adaptado para o *Framework* NIDS-SOA apresentando os serviços que serão utilizado na arquitetura orientada a serviço do *Framework* NIDS-SOA, conforme descrito na Figura 4.8(b).



(a) IDS-NIDIA com a camada de serviço web[4]

(b) IDS-NIDIA com o serviço para SOA-NIDS-SOA

Figura 4.8: IDS-NIDIA adaptado para Framework NIDS-SOA

4.5.3 Serviços do *Framework*

Os serviços já prontos para serem utilizados no *Framework* NIDS-SOA, conforme mostrados na Figura 4.8(b), são os seguintes:

- **WsSniffer** - Esse serviço fica escutando e capturando todos os pacotes na rede e/ou *host*. Para iniciar a captura dos pacotes, deve-se instanciar o serviço, passando os parâmetros *"Número do dispositivo"* (inteiro) e *"opções"* (Lista de inteiros) e utilizar o seu método *start*. Esse método, *start*, o parâmetro *"Numero do dispositivo"* é passado para classe do **JpCap**, que é responsável por receber e manipular os pacotes;

- **WsFormatter** - Esse serviço formata os pacotes em arquivos XML, com o objetivo de facilitar a atualização da base de dados bem como uma comparação com outros pacotes em um serviço de análise. Ele recebe como parâmetro o pacote que se deseja formatar no formato XML, com os atributos "Porta de destino", "IP de destino", "Porta de origem", "IP de origem", "Segundos"e "Milisegundos". Esses dois últimos são passados para representar a hora exata que o pacote foi capturado;

- **WsAnalyser** - Esse serviço analisa cada pacote. Ele recebe o pacote já formatado em XML e o analisa de acordo com as regras de perigo pré-estipuladas. O pacote passa por essas regras, e se for reprovado, ele é passado para o **WsSender**, senão é ignorado e a captura de pacotes continua;

- **WsSender** - Esse serviço atualiza as bases de informações. O serviço **WsSender** recebe o pacote reprovado na análise do serviço **WsAnalyzer** e o salva no repositório de pacotes (Base de Dados), para ser utilizado futuramente em comparações com possíveis pacotes perigosos.

- **WsReaction** - Esse serviço tem a função de tomar medidas com inserção de regras no *firewall* da rede corporativa. Essas medidas são tomadas em função das análises feitas pelo serviço **WsAnalyser** e/ou com base nas assinaturas disponíveis nas bases de dados, inserir uma proteção no *firewall*(bloqueado a porta de acesso ou barrando o IP de origem do invasor).

4.5.4 Diagrama de Sequência do *Framework* NIDS-SOA

Na Figura 4.9, tem-se o diagrama de sequência do *framework* NIDS-SOA. A sua descrição é a seguinte:

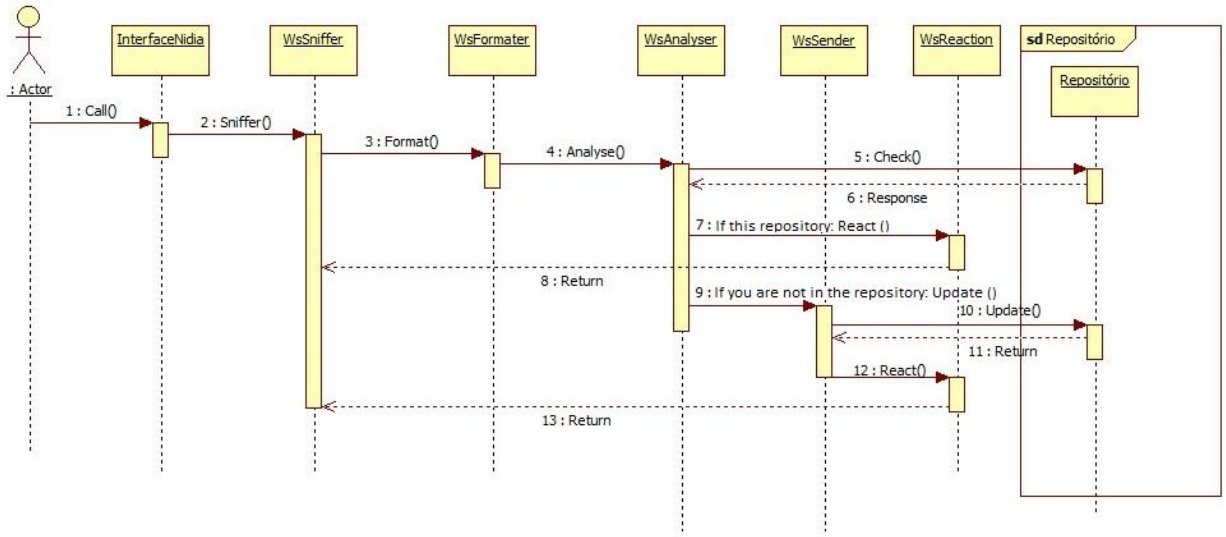


Figura 4.9: Diagrama de Sequência do NIDS-SOA

1. A interface de rede (***InterfaceNidia***) fornecerá os pacotes de rede a serem capturados. Foi utilizado a ***jpcap*** que é uma API (*Application Programming Interface*) para a captura de pacotes na redes escrita em Java. O serviço *web* (***WsSniffer*** (**2:Sniffer**)) captura os pacotes na rede através da biblioteca ***jpcap***;

2. Os pacotes capturados são formatados pelo serviço *web* (***WsFormater***(**3:Format()**)) em formato de arquivo XML para facilitar a atualização e comparação posterior por outro serviço *web*. Nesse arquivo XML, tem-se informação sobre o IP de origem e de destino, porta acessada, tempo no qual foi capturado o pacote e a quantidade de vezes que esse IP tentou acessar essa porta;

3. O serviço *web* (***WsAnalyser***(**4:Analyser()**)) realiza uma consulta (**5:Check()**) na base de dados (**Repositório**) para verificar se há informações com as informações dos pacotes já formatados pelo serviço *web* ***WsFormater***;

4. Caso a resposta da consulta (**6:Response**) seja positiva, (**7:If this Repository: React()**), uma mensagem será enviada ao serviço *web* (***WsReaction***) e este por sua vez passará o controle (**8:Return**)para o serviço *web* (***WsSniffer***) que continuará a capturar os pacotes da rede, iniciando assim todo o processo. Entretanto, caso a resposta da consulta (**6:Response**) seja negativa, (**9:If you are not in the**

Repository: Update()), uma ação (**10:Update()**) para atualizar a base de dados com essa informação será feita pelo serviço *web* (***WsSender***). Após a base de dados ser atualizada (**11:Return**), o serviço ***web*** (***WsSender***) enviará uma ação (**12:React()**) para o serviço *web* (***WsReaction***) tome ações necessárias para impedir o acesso desse IP ao sistema com bloqueio do IP e/ou da porta no *firewall*;

5. Após as ações do serviço *web* (***WsReaction***) o controle (**13:Return**)será repassado ao serviço *web* (***WsSniffer***) que continuará a capturar os pacotes da rede, iniciando assim todo o processo.

4.5.5 Diagrama de Classe do *Framework* NIDS-SOA

A Figura 4.10 apresenta o diagrama de classe do NIDS-SOA.

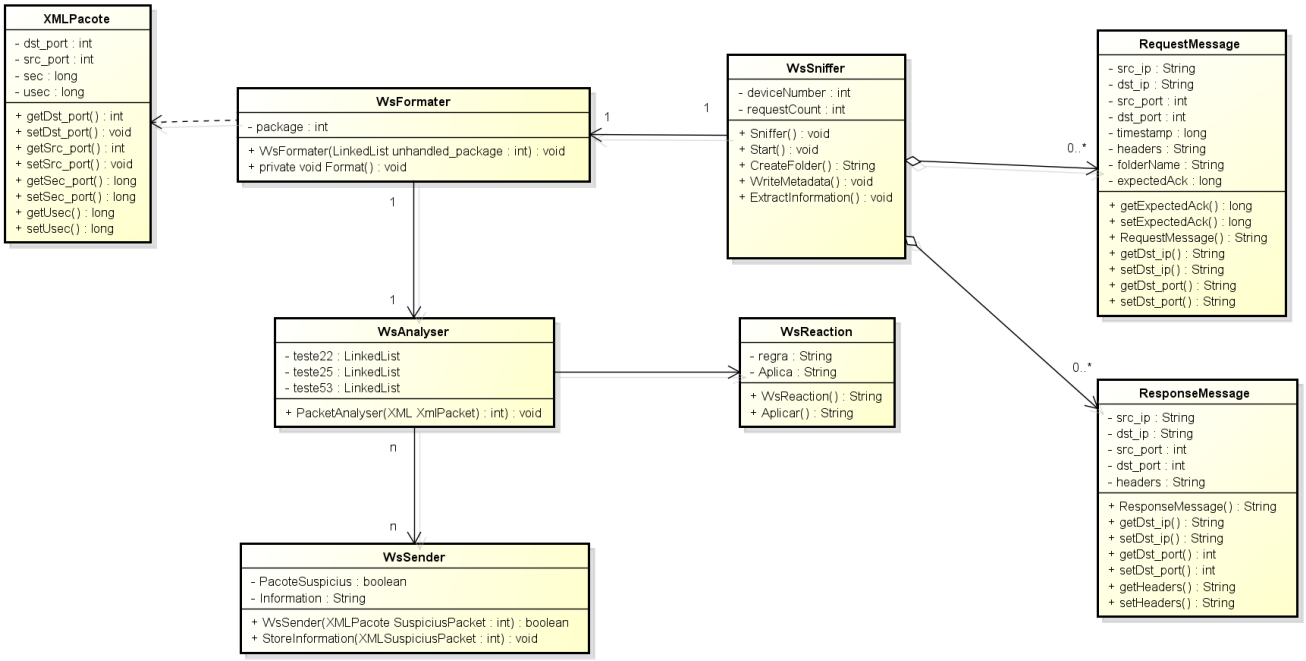


Figura 4.10: Diagrama de Classe do NIDS-SOA

Esse diagrama possui as seguintes classes:

- **WsSniffer** - Esta classe captura os pacotes na rede, com auxílio da API ***jpcap***. Ela recebe o número da posição do dispositivo de roteamento em public Sniffer(int device) e inicia o capturador de pacotes. A API public WsFormater(LinkedList unhandledpackage) que abre o dispositivo de roteamento é utilizada, para permitir que os pacotes sejam analisados. Logo depois, a própria classe ***Sniffer*** é reinstanciada, para garantir que ela rode em *loop* e capture todos os pacotes da rede. O método ***public void start()*** recebe os pacotes e extrai suas informações chamando o método ***extractInformation(Packet packet)***;

- **WsFormater** - Esta classe é chamada através da classe ***WsSniffer***. A ***WsFormater*** recebe uma lista com as informações dos pacotes capturados na rede e instancia a classe com o pacote não manuseado e chama a função ***Format()***. O método ***public WsFormater(LinkedList unhandledpackage)*** recebe a lista ligada e pega os seus elementos, transformado-os em um pacote XML e logo em seguida instancia a classe ***WsAnalyser***;

- **WsAnalyser** - Esta classe recebe o pacote já formatado em XML e chama a classe ***PacketAnalyser***. O método ***public WsAnalyser(XML XmlPacket)*** analisa as informações do pacote recebido testando as regras de segurança do sistemas e instancia a classe ***WsSender*** caso o pacote analisado com base nas regras do sistema, seja considerado nocivo, caso o pacote analisado seja seguro, o processamento será repassado para o próximo pacote formatado;

- **WsSender** - Esta classe será instaciada chamando a função ***StoreInformation*** quando o pacote analisado pela classe ***WsAnalyse*** for classsificado como nocivo. Este pacote será armazenado na base de dados. Caso esse pacote considerado como nocivo já se encontre na base de dados, será retornado um ***true*** pelo método ***private boolean StoreInformation(XML SuspiciusPacket)***;

- **XMLPacote** - É uma classe auxiliar que é útil na formatação de pacotes em arquivos XML. Através do uso da biblioteca ***XStream***, que é composta por classes que auxiliam a transformação de qualquer classe que segue um certo padrão em um formato XML. Esta classe ***XMLPacote*** transforma pacote capturados em pacotes no formato XML. A classe deve ter atributos globais, métodos de modificações desses atributos, e métodos que retornam esses atributos globais. A XMLPacote segue esse padrão, onde os métodos que modificam o valor dos atributos globais são preceditos por "***Set***"e os métodos que retornam são precedidos por "***Get***". Cada pacote rastreado pela classe ***WsSniffer*** é formatado na classe ***WsFormater***, que instancia a classe ***XMLPacote*** passando os dados do pacote para o objeto. Isso tudo é feito para facilitar a manipulação, consulta e modificação dos atributos de cada pacote;

- **RequestMessage e ResponseMessage** - Estas duas classes são bem similares, já que uma complementa a outra. Essas são as classes auxiliares da classe ***WsSniffer***. A classe ***RequestMessage*** e a ***ResponseMessage*** utilizam um formato semelhante da classe XMLPacote, com os atributos globais e os métodos de acesso. A RequestMessage é utilizada na comunicação da classe ***WsSniffer*** com o plugin ***Jpcap***, que é usado na captura de pacotes da rede. Especificamente, a classe ***RequestMessage*** é chamada somente quando é feito uma requisição *HTTP GET*, quando um *site* é acessado, por exemplo. Ela se comunica diretamente com a

classe ***ResponseMessage***, onde essa é usada pra retornar o pacote solicitado pela classe ***RequestMessage***.

4.6 Modelagem do *Framework* NIDS-SOA

A modelagem orientada a serviço do *Framework* NIDS-SOA se dará com o uso da definição do os conceitos apresentados na seção 3.4, das visões das partes interessadas, conforme visto na seção 3.4.2 bem como as disciplinas de modelagem orientada a serviço visto em 3.4.3 e seguindo os conceitos apresentados de ciclo de vida do modelo orientado a serviço e que foi descrito na seção 3.7. Será também apresentado um diagrama de transação circular[11] para apresentar as atividades sequencias dos serviços neste *framework*.

4.6.1 Definição dos serviços candidatos

Diferentemente dos estudos de casos sobre modelagem de composição de serviços, onde os estudos preliminares exigem uma definição da relação dos serviços candidatos que poderão participar da composição de serviço, neste *Framework* NIDS-SOA, todos os serviços participarão da composição de serviço.

A definição dos serviços candidatos no *Framework* NIDS-SOA são todos os serviços envolvidos e que executam tarefas necessárias ao funcionamento do IDS-NIDIA, conforme mostrado na Figura 4.11.

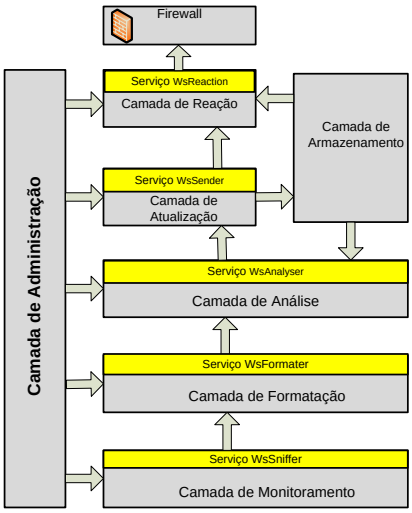


Figura 4.11: Relação dos serviços candidatos

Esses serviços já foram definidos em 4.5.3 e são estes:

- Serviço ***WsSniffer***;
- Serviço ***WsFormater***;
- Serviço ***WsAnalyser***;
- Serviço ***WsSender***;
- Serviço ***WsReaction***.

4.6.2 Descrição da implementação - tecnologias utilizadas

Seguindo os procedimento mencionados em 3.7 que descrevem o ciclo de vida em uma modelagem orientada a serviço, nesta seção descreveremos as características da fase de desenvolvimento de serviços, relatando as tecnologia de implementação e as linguagens de programação utilizadas nesse *Framework*.

Foram utilizados as seguintes ferramentas:

1. Foi escolhido o serviço para realizar a composição de serviço neste *Framework*;
2. Utilizou-se o *plugin Axis2 Support* na IDE do NetBeans 6.9 para gerar o arquivo de linguagem de descrição do serviço *web*, o **WSDL** a partir da classe do programa fonte;
3. Utilizou-se *Active BPEL* na IDE do Eclipse para gerenciar os processos de negócios.

4.6.3 Descrição da composição de serviços no NIDS-SOA

A composição de serviço que será feita no *framework* NIDS-SOA, utilizará os serviços definidos em 4.6.1. Este serviços, possuem a descrição idêntica a descrita na seção 4.5.4.

O serviço ***WsSniffer*** realizará a captura dos pacotes na rede, o serviço ***WsFormater*** formatará esses pacotes capturados, o serviço ***WsAnalyser*** fará a análise desses pacotes e caso encontre algum pacote suspeito (no caso um invasor) ele utilizará o serviço ***WsSender*** para atualizar uma base de dados com informações sobre esse invasor e também utilizará o serviço ***WsReaction*** para inserir informações no *firewall* da rede bloqueando seu acesso e/ou a porta na qual ele teve acesso.

Estes serviços, em conjunto ou em composição de todos ou de ao menos dois serviços farão parte de um processo de negócio WS-BPEL, conforme já descrito em 3.6.

A representação do serviços básicos do NIDS-SOA para realizar a modelagem da composição de serviços está mostrado na Figura 4.12.

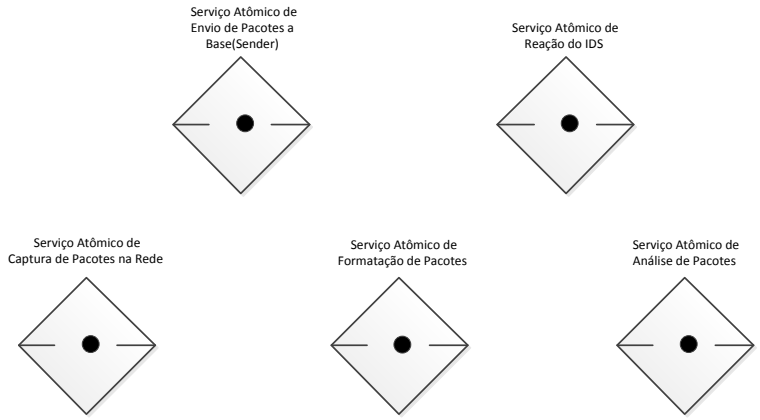


Figura 4.12: Serviços básicos do NIDS-SOA

4.6.4 Atividades de transição circular

As atividades de um conjunto de alguns serviços neste *Framework* são sequencias, como por exemplo o serviço ***WsSniffer*** que realiza o serviço de captura de pacotes na rede, logo em seguida esse pacote é formatado pelo serviço ***WsFormater***. Logo após ser formatado esse pacote é analisado pelo serviço ***WsAnalyser*** que consulta uma base de dados para verificar se o pacote analisado consta nesta base e caso se encontre uma ação de reação será enviada ao serviço web ***WsReaction***, caso não conste na base de dados e esse pacote seja considerado como um pacote oriundo de um IP conhecido como invasor, este pacote será inserido em uma base de dados. Essa inserção na base de dados será feito pelo serviço ***WsSender*** e todo processo do *Framework* NIDS-SOA é reiniciado. Como observa-se há uma sequencia de execução desses serviços. O diagrama de transição circular, conforme descrito em 3.8.1, será utilizado nesse *Framework* para melhor entender o funcionamento e a ação de cada serviço.

Na Figura 4.13, apresenta o diagrama de transição circular do serviço composto de formatação de pacotes.

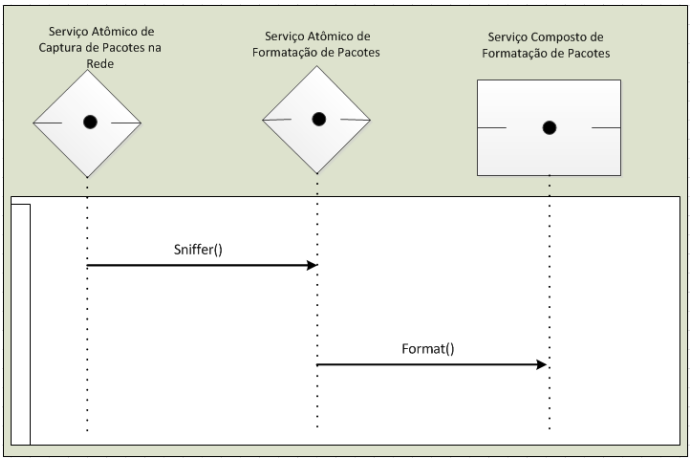


Figura 4.13: Diagrama de transição circular - Serviço Composto de formatação de pacotes[11]

Na Figura 4.14 apresenta o diagrama de transição circular do *Framework* NIDS-SOA, onde apresentam-se todos os serviços do *Framework* NIDS-SOA.

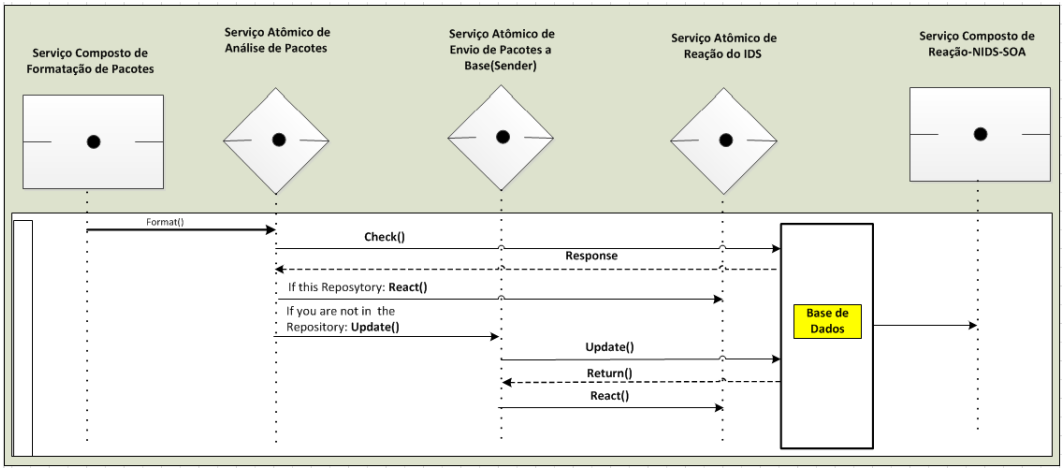


Figura 4.14: Diagrama de transição circular do NIDS-SOA[11]

4.6.5 Composição de serviços

A composição de serviços neste *Framework* finaliza a modelagem deste *Framework*. Os serviços que atuam nesse *Framework* possuem os seus respectivos arquivos de linguagem de descrição de serviço, o chamado ***WSDL***. Esses arquivos ***WSDL*** foram gerados a partir do código fonte do IDS-NIDIA e os arquivos ***WSDL*** encontram-se no apêndice B. Há uma variedade de ferramentas para realizar a composição de serviço e consequentemente gerenciar o processo de negócios, conforme descrito em 3.5.1. O *Active BPEL da IDE do Eclipse* foi utilizado para modelar e gerenciar todos esses serviços. A Figura 4.15 apresenta o projeto do processo de negócio do *Framework* NIDS-SOA com o uso do *Active BPEL da IDE do Eclipse*.

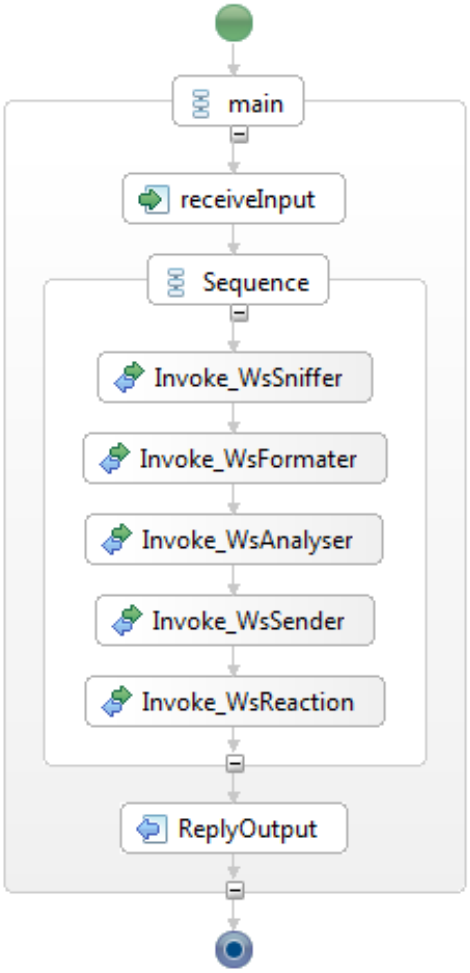


Figura 4.15: Modelagem do Framework NIDS-SOA no Active BPEL

4.6.6 Considerações sobre o *Framework*

O *Framework* NIDS-SOA possui algumas considerações:

1. Alguns ou todos os serviços apresentados em 4.6.1 podem participar durante a execução deste *Framework*. É desejável que o serviço *WsSniffer* participe quando o *Framework* estiver sendo executado pelo fato deste executar a captura dos pacotes de rede;

2. Alguns serviços deste *Framework* poderão compartilhar informações com outros IDS proporcionando deste modo as características de uma arquitetura orientada a serviço;

3. O *Framework* poderá também receber informações de outros funcionalidades de outros IDSt's desde que estas informações sejam fornecidas através de um serviço;

4. O serviço *Analyser* poderá ter mais itens para analisar(por exemplo, porta de acesso), pois nos teste realizados com este *Framework* foram considerados apenas os itens como porta de origem e destino, número IP de origem e de destino e tempo de ocorrência deste pacote na rede.

4.7 Síntese

Neste capítulo, o *Framework* NDIS-SOA foi apresentado com a sua descrição, sua concepção, como ele foi idealizado utilizando o IDS-NIDIA, as camadas utilizadas, os serviços que foram incluídos no *Framework*, a modelagem orientada a serviço, os diagramas utilizados e o diagrama de transação bem como o processo de modelagem WS-BPEL desse *Framework*.

No próximo capítulo, os testes realizados neste *Framework NIDS-SOA* serão apresentados.

Capítulo 5

Implementação e Resultados

5.1 Introdução

O *Framework* NIDS-SOA, inicialmente, foi planejado com base nas funcionalidades que IDS-NIDIA já possuia. Foi escolhida a linguagem de programação Java, devido a facilidade que a mesma proporciona em adicionar uma camada de serviço *web* no código fonte feito nesta linguagem de programação. A Figura 5.1 mostra todos os pacotes e classes desse código fonte do IDS-NIDIA.

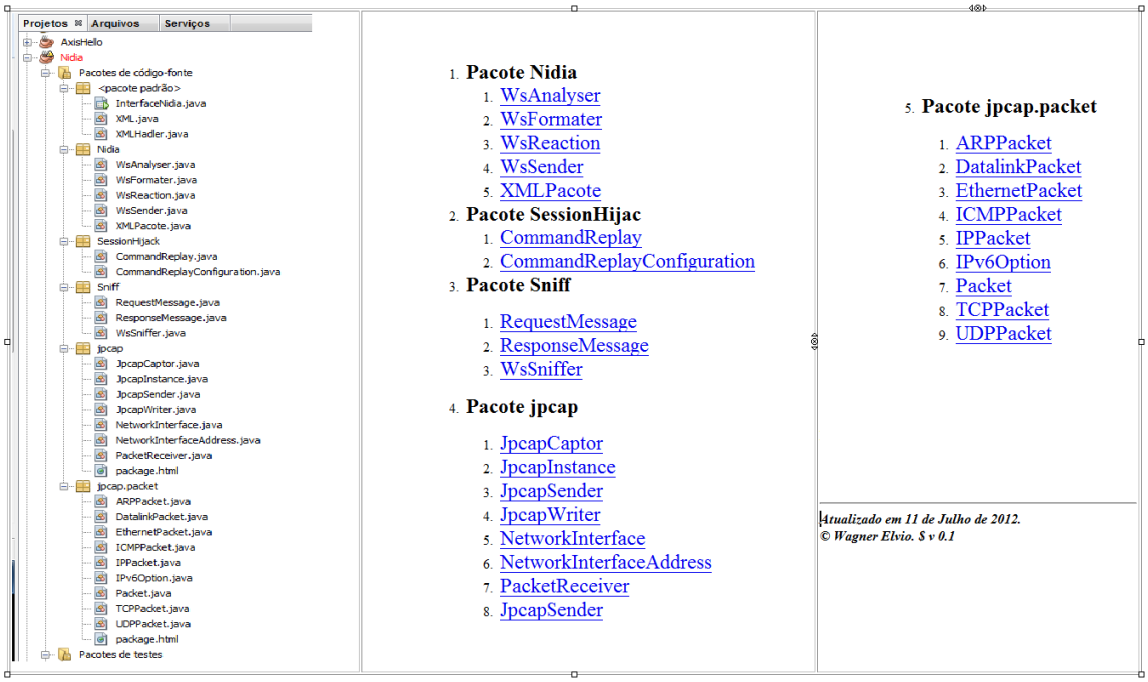


Figura 5.1: Código fonte do IDS-NIDIA

Foi utilizado a *IDE do NetBeans 6.9* para o desenvolvimento desse código fonte, o Apache Axis2 e o *plug-in* de suporte a Axis2[1].

[1] Esses dois *software* permitem transformar um código fonte escrito na linguagem Java em um arquivo *WSDL* que é a linguagem de descrição de serviço.

A Figura 5.2 apresenta a interface do IDS-NIDIA na *IDE do NetBeans*. Observa-se que tem-se os seguintes serviços disponíveis:

1. *Sniffer*;
2. *Formater*;
3. *Analiser*;
4. *Sender* e
5. *Reaction*

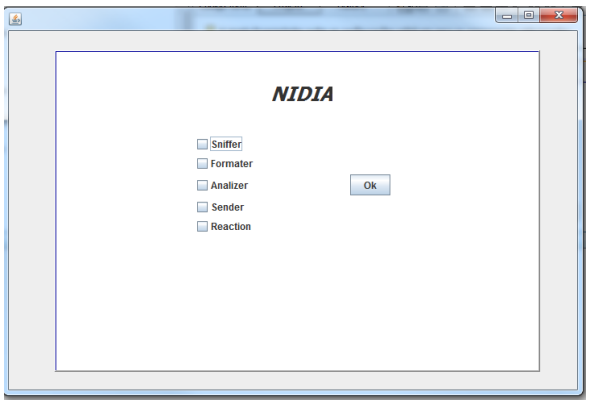


Figura 5.2: Interface do IDS-NIDIA no NetBeans

Ao ser selecionado os itens *Sniffer* e *Formater*, na tela em que está mostrado na Figura 5.2, o programa gera os pacotes capturados na rede de computadores local e os apresentam em um formato XML conforme mostrado logo abaixo.

```
<pacote>
  <dst__port>80</dst__port>
  <dst__ip>/200.17.202.197</dst__ip>
  <src__port>38845</src__port>
  <src__ip>/200.137.130.41</src__ip>
  <sec>1331303390</sec>
  <usec>187195</usec>
</pacote>
```

```
<pacote>
  <dst__port>80</dst__port>
  <dst__ip>/200.17.202.197</dst__ip>
  <src__port>38848</src__port>
  <src__ip>/200.137.130.41</src__ip>
  <sec>1331303390</sec>
  <usec>690533</usec>
</pacote>
```

A funcionalidade *Analiser* apresenta algumas regras[2] nesse código e caso essa funcionalidade fosse habilitada ela iria "ficar"analisando os pacotes já coletados com as regras inseridas no código e se quando houvesse uma ocorrência do uso da regra o programa apresentava uma tela alertando que um número IP coletado no pacote estava em uma ação maléfica ou suspeita na rede local.

5.2 Composição de serviço

O uso das ferramentas computacionais como Apache *Axis2* e o *plug-in* de suporte a *Axis2* possibilitou a geração dos arquivos *WSDL*(Linguagem de descrição de serviços) do código fonte do IDS-NIDIA. Alguns desses códigos gerados encontram-se no apêndice B.

Os testes realizados com o *framework* NIDS-SOA foram utilizados os serviços básicos deste *framework* e que estão descritos no item 4.6.1.

Os serviços básicos deste *Framework* NIDS-SOA, conforme definidos na subseção 4.6.3, estão apresentados na Figura 5.3.

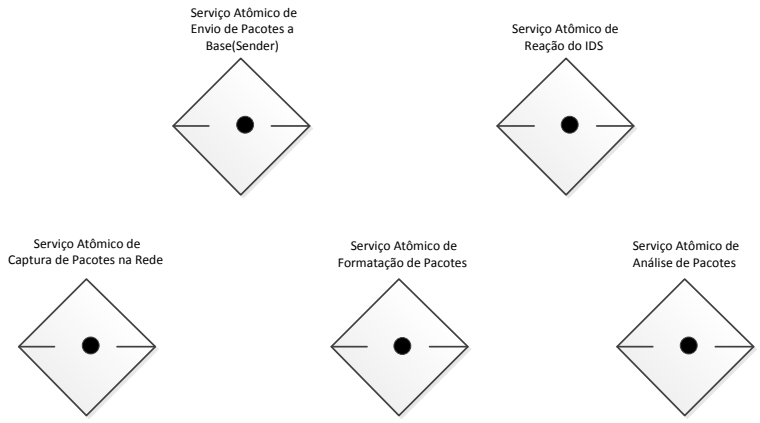


Figura 5.3: Serviços básicos do NIDS-SOA

5.3 *Software* utilizados

Os *software* utilizados para executar e validar os testes são todos *open source*.
Logo abaixo a relação dos mesmos:

1. IDE Eclipse versão Juno;
2. Apache tomcat versão 7.0;

[2]Essas regras serão relatadas na seção 5.4.

3. Apache ODE 1.3.5 (*Orchestration Director Engine*);

4. Eclipse SOA-BPEL 2.0.

5.4 Teste com o serviço *WsAnalyser*

Os testes realizados com o serviço **WsAnalyser**, constitui na análise dos pacotes capturados pelo serviço **WsSniffer** e formatados pelo serviço **WsFormater**. Os pacotes formatados em XML, conforme pode ser visto na seção 5.1, apresenta algumas informações como por exemplo: O número IP e porta do destinatário e da origem e informações sobre o tempo de atuação desse número IP. Essas informações sobre IP e porta de destino são utilizadas para criar os parâmetros de análise e que são utilizados pelo serviço *web WsAnalyser*. Neste *Framework* utilizou-se os seguintes parâmetros conforme descrito na Tabela 5.1.

Porta	Protocolo	Regras de Análise
22	TCP	Acesso do mesmo IP (*atacante*) em menos de 20 s
53	UDP	Acesso do mesmo IP (*atacante*) em menos de 20 s
80	TCP	Acesso do mesmo IP (*atacante*) em menos de 50 s

Tabela 5.1: Regras utilizadas na análise dos pacotes capturados

Os valores das regras de análise contidos na Tabela 5.1 foram estimados das observações feitas em *logs* de um servidor de produção. Nesse arquivo de *log*, localizado no apêndice D.1 observa-se que o IP do atacante ***101.78.202.18*** apresenta 30 registros em menos de 43 segundos. Um valor considerado bem elevado para uma acesso de um serviço nesse servidor.

Nesse outro arquivo de *log*, localizado no apêndice D.2, observa-se que o IP do atacante ***222.221.2.210*** apresenta 30 registros em menos de 60 segundos. Um valor também bastante elevado para uma acesso de um serviço nesse servidor.

Para efeitos de testes considerou-se apenas os valores que estão mostrados na Tabela 5.1 e de acordo com os valores que são realmente executados em uma ambiente de produção pode-se facilmente ajustar esse valores.

A depender das análises baseadas nas regras contidas na Tabela 5.1, o serviço *WsAnalyser* utilizará o serviço *WsReaction*, para inserir uma regra no *firewall* com o objetivo de bloquear o número IP do atacante e utilizará também o serviço *WsSender* para atualizar a base de dados com informações sobre esse atacante, conforme já descrito na sub seção 4.6.3.

5.4.1 Projeto BPEL dessa composição

A Figura 5.4 apresenta o projeto BPEL dessa composição.
Nesse projeto tem-se os seguintes atividades pertencentes no *Eclipse BPEL*:

1. Esse projeto BPEL possui o serviço *web WsAnalyser*
2. Possui a instrução ***AssigInputWsAnalyser*** que manipula os dados de entrada para o serviço *web WsAnalyser*.
3. Possui a instrução ***InvokeWsAnalyser*** que invoca o serviço *web WsAnalyser* de forma síncrona.
4. Possui a instrução ***AssignOutWsAnalyser*** que coleta os dados realizados pelo serviço *web WsAnalyser*.

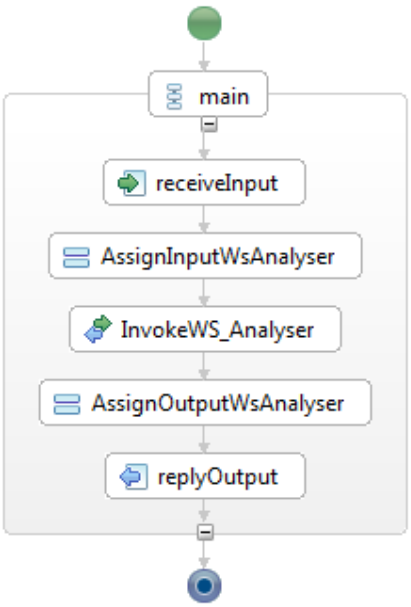


Figura 5.4: Uso do BPEL Eclipse para composição dos serviços

5.4.2 Código BPEL

No apêndice C.2 está listado o código completo do processo BPEL mostrado na Figura 5.4.

Na próxima seção será realizado o teste com o serviço ***WsAnalyser Consulta IP***.

5.5 Teste com o serviço *WsAnalyser* Consulta IP

Como descrito na seção 4.5.4, o serviço ***WsAnalyser*** realiza um consulta na base de dados para verificar se o número IP apresentado pelo serviço ***WsFormater*** já está na base de dados. Esta consulta é realizada pelo serviço ***WsAnalyser ConsultaIP***. Esse serviço realizará uma consulta em uma lista de IP's que constam em uma base de dados[3] do sistema. O resultado desta consulta retornará apenas uma resposta sobre a existência desse IP nessa base de informações.

Caso a resposta seja verdadeira ou seja o IP atacante já reside na base de informações do sistema nenhuma ação será tomada pois os números IP's desta lista já possuem regras de bloqueio no *Firewall* E no caso da resposta ser negativa ou seja o IP atacante não consta na base de informações do sistema, o serviço ***WsReaction*** irá inserir uma regra no *Firewall* para bloquear esse número IP.

Nas próximas subseções serão apresentados o projeto BPEL e os resultados obtidos com esse serviço.

5.5.1 Projeto BPEL dessa composição

A Figura 5.5 apresenta o projeto BPEL dessa composição.

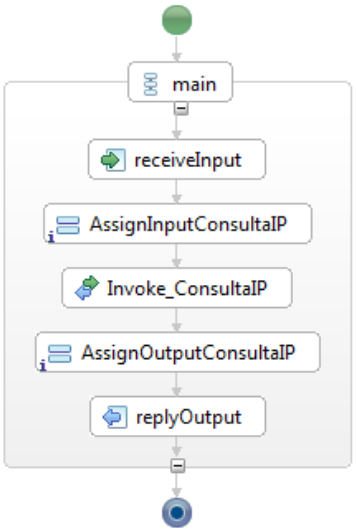


Figura 5.5: Uso do BPEL Eclipse para composição dos serviços - Consulta IP

Nesse projeto tem-se as seguintes atividades pertencentes no*Eclipse BPEL*:

1. Esse projeto BPEL possui o serviço *ConsultaIP* que realiza um consulta sobre um número IP em base de dados;

[3]Para simplicação com a modelagem SOA utilizou-se uma lista de IP's ao invés de uma base de dados SQL.

2. Possui a instrução ***AssigInputConsultaIP*** que manipula os dados de entrada para o serviço *web ConsultaIP*;

3. Possui a instrução ***InvokeConsultaIP*** que invoca o serviço *web ConsultaIP* de forma síncrona para que uma ação de escrita no *firewall* seja realizada.

4. Possui a instrução ***AssignOutConsultaIP*** que coleta os dados realizado do serviço *web ConsultaIP*.

No apêndice C.4 está listado o código completo do processo BPEL mostrado na Figura 5.5.

5.5.2 Resultados obtidos - Teste 01

Os procedimentos efetuados neste primeiro teste constou na consulta de um número IP já existente na base de dados e a sua execução teve os seguintes passos:

1. Busca-se o número IP desejado, digitando-se o valor deste número IP (***192.168.0.1***) no campo *Input* conforme mostrado na Figura 5.6 e executa-se o programa;
2. O envelope da mensagem **SOAP** é mostrado na Figura 5.7, onde observa-se o valor do número IP digitado na Figura 5.6;
3. A resposta aparece no console do *Eclipse BPEL*, (resposta ***true*** para esta consulta) conforme mostrado na Figura 5.8.

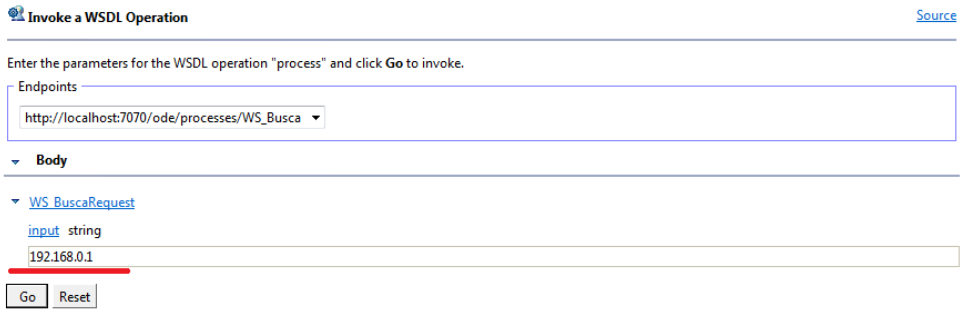


Figura 5.6: Uso do Eclipse BPEL - Consulta IP

Figura 5.7: Envelope da mensagem SOAP

Figura 5.8: Resposta da consulta no console do BPEL Eclipse

5.5.3 Resultados obtidos - Teste 02

Os procedimentos efetuados neste segundo teste constou na consulta de um número IP que não consta na base de dados e a sua execução teve os seguintes procedimentos:

1. Busca-se o número IP desejado, digitando-se o valor deste número IP (***192.168.2.200***) no campo *Input* conforme mostrado na Figura 5.9 e executa-se o programa;

2. O envelope da mensagem **SOAP** é mostrado na Figura 5.10, onde observa-se o valor do IP digitado na Figura 5.9;

3. A resposta aparece no console do *Eclipse BPEL*,(resposta ***false*** para esta consulta) conforme mostrado na Figura 5.11

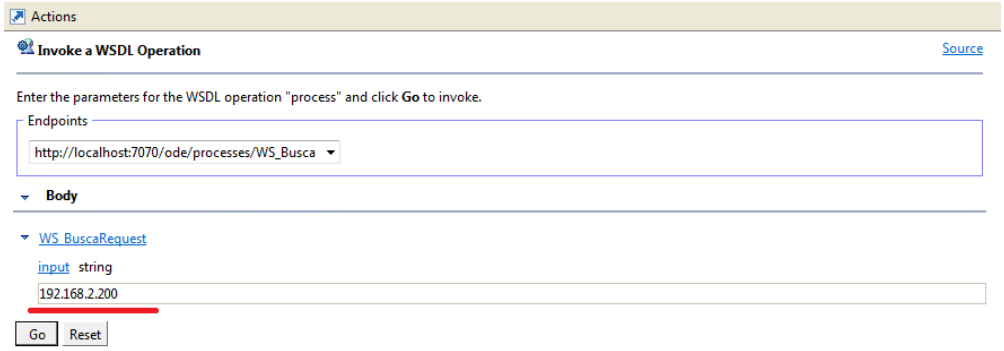


Figura 5.9: Uso do Eclipse BPEL - Consulta IP

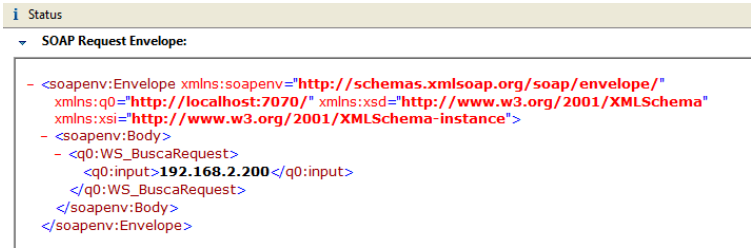


Figura 5.10: Envelope da mensagem SOAP

Figura 5.11: Resposta da consulta no console do Eclipse BPEL

5.6 Composição de Serviços: *WsAnalyser Consulta IP e WsReaction*

A composição dos dois serviços ***WsAnalyser Consulta IP*** e ***WsReaction***, apresentará uma ação de resposta no *Framework* NIDS-SOA para um invasor.

Como já descrito na seção 5.5, o serviço ***WsAnalyser Consulta IP*** realiza uma consulta em base de dados sobre um determinado número IP que foi obtido após o mesmo se enquadrar em algumas das regras de análise apresentadas na Tabela 5.1.

O serviço ***WsReaction*** só atuará se a consulta feita pelo serviço ***WsAnalyser Consulta IP*** encontrar uma resposta negativa, indicando desta forma que o número do IP invasor que se enquadrou em uma das regras apresentadas na Tabela 5.1 não se encontra na base de informações do sistema e conseqüentemente precisa ser tomada uma ação para bloqueá-lo no sistema com a seguinte regra:

```
1 iptables -A OUTPUT -p tcp -d 192.168.1.2 --dport 22 -j DROP
```

5.6.1 Projeto BPEL dessa composição

A Figura 5.12 apresenta o projeto BPEL dessa composição.

Nesse projeto tem-se os seguintes atividades pertencentes no*Eclipse BPEL*:

1. Esse projeto BPEL possui o serviço *ConsultaIP* que realiza um consulta sobre um número IP em base de dados;
2. Possui a instrução ***AssigInputConsultaIP*** que manipula os dados de entrada para o serviço *ConsultaIP*;
3. Possui a instrução ***InvokeConsultaIP*** que invoca o serviço *ConsultaIP* de forma síncrona para que uma ação de escrita no *firewall* seja realizada.
4. Possui a instrução ***AssignOutConsultaIP*** que coleta os dados realizado do serviço *web ConsultaIP*.
5. Possui a instrução ***AssigInputReaction*** que manipula os dados de entrada para o serviço *WsReaction*.
6. Possui a instrução ***InvokeReaction*** que invoca o serviço *WsReaction* de forma síncrona para que uma ação de escrita no *firewall* seja realizada.
7. Possui a instrução ***AssignOutWsReaction*** que coleta os dados realizado do serviço *WsReaction*.

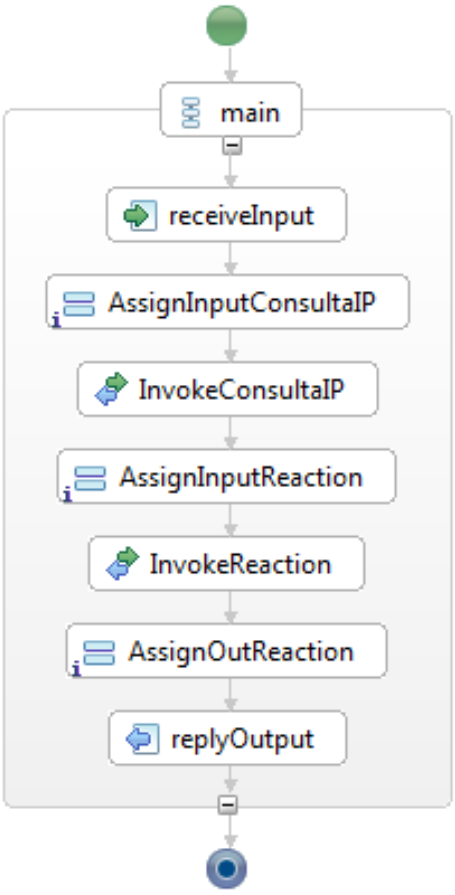


Figura 5.12: Uso do BPEL Eclipse para composição dos serviços - Consulta IP e Reaction

5.7 Considerações sobre os testes

A utilização do serviço *WsFormater* requer que o serviço *WsSniffer* forneça os dados para que o serviço *WsFormater* proceda com o tratamento e formatação dos dados em XML.

O serviço *WsAnalyser* necessita apenas dos pacotes já formatado em XML para serem analisados. Este serviço poder funcionar em modo isolado atuando na análise dos pacotes que são entregues pelo *WsFormater*. Da mesma forma, o serviço *WsReaction* necessita apenas ser executado quando IP e/ou uma porta de acesso do sistema operacional for considerada como sendo ameaçada por um atacante(*hacker*).

Uma outra observação é que o serviço *WsAnalyser* poderia trabalhar em conjunto com o serviço *WsSniffer* não se fazendo necessário a utilização do serviço *WsFormater* mas em compensação teríamos uma desvantagem em não termos as informações analisada pelo serviço *WsAnalyser* no formato XML.

Capítulo 6

Conclusão

Neste capítulo, a conclusão deste trabalho é apresentado, contendo os objetivos alcançados, as contribuições para o modelo do IDS-NIDIA já em estudo por outros alunos, alguns pontos a serem considerados e sugestões para trabalhos futuros na área de segurança em IDS-NIDIA com uso da arquitetura orientada a serviço.

O uso do *Framework* NIDS-SOA apresenta grande vantagem pelo fato de poder unir informações de sistemas de detecção de intrusão, que a priori, trabalham de forma isolada. A união dessas informações ajuda na defesa de sistemas computacionais contra ações maléficas. Os resultados alcançados são satisfatórios, pois as funcionalidades do IDS-NIDIA podem ser reutilizados por outros IDS's através da arquitetura orientada a serviço.

6.1 Contribuições do trabalho

É indiscutível a importância do uso de ferramentas computacionais no auxílio e às vezes até na proteção de dados e/ou informações em um rede corporativa ou *sites* de empresas que utilizam a Internet para transferir dados e informações sobre as atividades em que atuam.

O Comitê Gestor da Internet no Brasil - CGI.br[1], a CIS (*Center for Internet Security*) e outros órgãos governamentais e particulares que não têm medido esforços para garantir uma maior segurança dos dados que trafegam pela Internet, apresentam informações e estatísticas de ataques e as ameaças mais ocorridas[2] na área de segurança de redes em ambientes corporativos.

Com as atividades cotidianas cada vez mais dependentes de uma infra estrutura de

[1] Este comitê é composto por membros com representantes do Governo Federal, do setor privado e da comunidade científica e tecnológica.

[2] O SANS(*SysAdmin, Audit, Network, Security*) criado em 1989 é uma organização que trabalha com segurança em redes. Com mais de 165 mil profissionais que compartilham informações sobre questões de segurança em redes e constantemente revisam sua lista TOP-20, com as vinte ameaças mais ocorridas em redes de computadores[37].

rede segura e confiável, o uso de um sistema de proteção em uma rede de computadores com toda a segurança disponível (Com uso de *firewall*, IDS e sistemas antivirus e contra pragas da Internet) faz-se necessário. Um sistema de proteção de uma rede de computadores com um *firewall* limita-se apenas a bloquear tráfegos maliciosos em portas conhecidas, fazendo necessário a utilização de sistema que detecte o comportamento anormal do tráfego de uma rede de computadores como a utilização de um IDS.

Entretanto, como visto na seção 2.3, os IDS's podem gerar falsos positivos e comprometer desta forma a proteção segura das informações em uma rede corporativa. Uma solução para esse problema é termos uma base de assinatura[3] considerável que sirva de consulta com o objetivo de diminuir essa quantidade de falsos positivos.

Uma outra deficiência ainda encontrada nesse sistemas IDS's é que os mesmos não compartilham informações com outros sistemas de IDS's, ou seja um sistema de IDS com uma base de assinatura considerável não pode compartilhar suas informações com um outro IDS o que representa um desvantagem na proteção de uma rede corporativa.

Com todas essas desvantagens sobre o não compartilhamento de informações entre os sistemas IDS's faz-se necessário compartilhar essas informações entre esses sistemas de IDS para garantir uma maior eficiência na proteção das redes corporativas.

Uma maneira de realizar esse compartilhamento seria tornar a funcionalidade de consulta a uma base de dados de assinatura de um sistema de IDS como um serviço para ser utilizado em uma arquitetura SOA.

A arquitetura orientada a serviço (SOA) fornece essas características fazendo com que alguma ou todas as suas funcionalidades de um sistema de IDS compartilhem e/ou troquem informações com outros sistemas de IDS distintos baseado na arquitetura SOA, conforme descrito na seção 3.2.

Portanto, o presente trabalho apresenta essa contribuição para o compartilhamento de informações em sistemas IDS's distintos através do uso da arquitetura orientada a serviço (SOA). Também apresenta as funcionalidades (*Sniffer, Analyser, Formatter, Update e Sender*) do IDS-NIDIA como serviço para ser utilizado em arquitetura orientada a serviço.

Uma outra contribuição foi a modelagem do IDS-NIDIA em uma arquitetura orientada a serviço (SOA), proporcionando que as funcionalidades do IDS-NIDIA sejam agora serviços prontos para serem utilizados em uma arquitetura orientada a serviço, conforme descrito na sub seção 4.5.3.

[3]Uma assinatura é um conhecimento, uma informação sobre ataques conhecidos. Essas informações são armazenadas em base de dados[2].

6.2 Sugestões para Pesquisas Futuras

Como sugestão para trabalhos futuros pode-se citar:

1. Utilizar o NIDS-SOA para extender outros IDS de forma a criar uma rede cooperativa entre diferentes IDS, graças a interoperabilidade e suporte a composição fornecidos pela abordagem SOA e
2. Implementar o *Framework* NIDS-SOA por completo com o uso de uma ferramenta computacional que tenha suporte nativo a todas as funcionalidades deste *Framework* incluindo acesso a uma base de dados.

Apêndice A

Código exemplo de processo BPEL

A.1 Partes do código do processo BPEL

Código de um processo BPEL

```
<!-- NidiaUm BPEL Process [Generated by the Eclipse BPEL Designer] -->
<!-- Date: Tue May 01 21:25:49 BRT 2012 -->
<bpel:process name="NidiaUm"
          targetNamespace="http://localhost:8080/bpel/sample"
          suppressJoinFailure="yes"
          xmlns:tns="http://localhost:8080/bpel/sample"
          xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
              executable"
          >

    <!-- Import the client WSDL -->
   <bpel:import location="NidiaUmArtifacts.wsdl" namespace="http://
       localhost:8080/bpel/sample"
            importType="http://schemas.xmlsoap.org/wsdl/" />

    <!-- ================================================================
         -->
    <!-- PARTNERLINKS
                                                                    -->
    <!-- List of services participating in this BPEL process
                       -->
    <!-- ================================================================
         -->
    <bpel:partnerLinks>
        <!-- The 'client' role represents the requester of this service.
            -->
        <bpel:partnerLink name="client"
                     partnerLinkType="tns:NidiaUm"
                     myRole="NidiaUmProvider"
                     />
```

```
24      </bpel:partnerLinks>
25
26      <!-- ================================================================
            -->
27      <!-- VARIABLES
                                                                        -->
28      <!-- List of messages and XML documents used within this BPEL process
            -->
29      <!-- ================================================================
            -->
30      <bpel:variables>
31          <!-- Reference to the message passed as input during initiation --
               >
32          <bpel:variable name="input"
33                    messageType="tns:NidiaUmRequestMessage"/>
34
35          <!--
36            Reference to the message that will be returned to the requester
37            -->
38          <bpel:variable name="output"
39                    messageType="tns:NidiaUmResponseMessage"/>
40      </bpel:variables>
41
42      <!-- ================================================================
            -->
43      <!-- ORCHESTRATION LOGIC
                                                            -->
44      <!-- Set of activities coordinating the flow of messages across the
             -->
45      <!-- services integrated within this business process
                          -->
46      <!-- ================================================================
            -->
47      <bpel:sequence name="main">
48
49          <!-- Receive input from requester.
50               Note: This maps to operation defined in NidiaUm.wsdl
51              -->
52          <bpel:receive name="receiveInput" partnerLink="client"
53                   portType="tns:NidiaUm"
54                   operation="process" variable="input"
55                   createInstance="yes"/>
56
57          <bpel:flow name="Flow">
58              <bpel:sequence name="Sequence">
59                  <bpel:invoke name="Invoke_WsSniffer"></bpel:invoke>
60                  <bpel:invoke name="Invoke_WsFormater"></bpel:invoke>
61                  <bpel:invoke name="Invoke_WsAnalyser"></bpel:invoke>
```

```
62                  <bpel:invoke name="Invoke_WsSender"></bpel:invoke>
63              </bpel:sequence>                        </bpel:flow>
64
65          <!-- Generate reply to synchronous request -->
66          <bpel:reply name="replyOutput"
67                  partnerLink="client"
68                  portType="tns:NidiaUm"
69                  operation="process"
70                  variable="output"
71                  />
72      </bpel:sequence>
73  </bpel:process>
```

Listing A.1: Código BPEL de exemplo da Figura 3.16.

A.2 Código do processo BPEL

Código de um processo BPEL

```
<bpel:process name="Nome_do_Processo_BPEL"
    <!-- Import the client WSDL -->
  <bpel:import location="NidiaUmArtifacts.wsdl" namespace="http://
      localhost:8080/bpel/sample"
          importType="http://schemas.xmlsoap.org/wsdl/" />

    <bpel:partnerLinks> ...   </bpel:partnerLinks>
    <!-- Definicao de variaveis          -->
    <bpel:variables>  ...   </bpel:variables>
  <!-- ORCHESTRATION LOGIC              -->
    <bpel:sequence name="main">
        <bpel:receive name="receiveInput" partnerLink="client"
                portType="tns:NidiaUm"
                operation="process" variable="input"
                createInstance="yes"/>

        <bpel:flow name="Flow">
            <bpel:sequence name="Sequence">
                <bpel:invoke name="Invoke_WsSniffer"></bpel:invoke>
                <bpel:invoke name="Invoke_WsFormater"></bpel:invoke>
                <bpel:invoke name="Invoke_WsAnalyser"></bpel:invoke>
                <bpel:invoke name="Invoke_WsSender"></bpel:invoke>
            </bpel:sequence>                    </bpel:flow>

        <!-- Generate reply to synchronous request -->
        <bpel:reply name="replyOutput"
               partnerLink="client"
               portType="tns:NidiaUm"
               operation="process"
               variable="output"
               />
    </bpel:sequence>
</bpel:process>
```

Listing A.2: Partes do código BPEL de exemplo da Figura 3.16.

Apêndice B

Código WSDL

B.1 Código do WSAnalyser

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:axis2="http://Nidia/" xmlns:ns1="http://org.apache.axis2/xsd"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:http="
    http://schemas.xmlsoap.org/wsdl/http/" xmlns:ns0="http://Nidia/xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:mime="http://schemas.
    xmlsoap.org/wsdl/mime/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/
    soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    targetNamespace="http://Nidia/">
3     <wsdl:types>
4         <xs:schema xmlns:ns="http://Nidia/xsd" attributeFormDefault="
              qualified" elementFormDefault="qualified" targetNamespace="
              http://Nidia/xsd">
5             <xs:element name="isBlackListedPackage">
6                 <xs:complexType>
7                     <xs:sequence>
8                         <xs:element minOccurs="0" name="XmlPacket"
                              nillable="true" type="ns0:XMLPacote"/>
9                     </xs:sequence>
10                </xs:complexType>
11            </xs:element>
12            <xs:complexType name="XMLPacote">
13                <xs:sequence>
14                    <xs:element minOccurs="0" name="dst_ip" nillable="true
                          " type="xs:string"/>
15                    <xs:element minOccurs="0" name="dst_port" type="xs:int
                          "/>
16                    <xs:element minOccurs="0" name="sec" type="xs:long"/>
17                    <xs:element minOccurs="0" name="sec_port" type="
                          xs:long"/>
```

```
18                  <xs:element minOccurs="0" name="src_ip" nillable="true
                        " type="xs:string"/>
19                  <xs:element minOccurs="0" name="src_port" type="xs:int
                        "/>
20                  <xs:element minOccurs="0" name="usec" type="xs:long"/>
21              </xs:sequence>
22          </xs:complexType>
23          <xs:element name="isBlackListedPackageResponse">
24              <xs:complexType>
25                  <xs:sequence>
26                      <xs:element minOccurs="0" name="return" type="
                            xs:boolean"/>
27                  </xs:sequence>
28              </xs:complexType>
29          </xs:element>
30          <xs:element name="isBlackListedSnortPackage">
31              <xs:complexType>
32                  <xs:sequence>
33                      <xs:element minOccurs="0" name="XmlPacket"
                            nillable="true" type="ns0:XMLPacote"/>
34                  </xs:sequence>
35              </xs:complexType>
36          </xs:element>
37          <xs:element name="isBlackListedSnortPackageResponse">
38              <xs:complexType>
39                  <xs:sequence>
40                      <xs:element minOccurs="0" name="return" type="
                            xs:boolean"/>
41                  </xs:sequence>
42              </xs:complexType>
43          </xs:element>
44      </xs:schema>
45  </wsdl:types>
46  <wsdl:message name="isBlackListedSnortPackageRequest">
47      <wsdl:part name="parameters" element="
            ns0:isBlackListedSnortPackage"/>
48  </wsdl:message>
49  <wsdl:message name="isBlackListedSnortPackageResponse">
50      <wsdl:part name="parameters" element="
            ns0:isBlackListedSnortPackageResponse"/>
51  </wsdl:message>
52  <wsdl:message name="isBlackListedPackageRequest">
53      <wsdl:part name="parameters" element="ns0:isBlackListedPackage"/>
54  </wsdl:message>
55  <wsdl:message name="isBlackListedPackageResponse">
56      <wsdl:part name="parameters" element="
            ns0:isBlackListedPackageResponse"/>
57  </wsdl:message>
```

```
58      <wsdl:portType name="WsAnalyserPortType">
59          <wsdl:operation name="isBlackListedSnortPackage">
60              <wsdl:input message="axis2:isBlackListedSnortPackageRequest"
                    wsaw:Action="urn:isBlackListedSnortPackage"/>
61              <wsdl:output message="axis2:isBlackListedSnortPackageResponse"
                     wsaw:Action="urn:isBlackListedSnortPackageResponse"/>
62          </wsdl:operation>
63          <wsdl:operation name="isBlackListedPackage">
64              <wsdl:input message="axis2:isBlackListedPackageRequest"
                    wsaw:Action="urn:isBlackListedPackage"/>
65              <wsdl:output message="axis2:isBlackListedPackageResponse"
                    wsaw:Action="urn:isBlackListedPackageResponse"/>
66          </wsdl:operation>
67      </wsdl:portType>
68      <wsdl:binding name="WsAnalyserSOAP11Binding" type="
            axis2:WsAnalyserPortType">
69          <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document"/>
70          <wsdl:operation name="isBlackListedSnortPackage">
71              <soap:operation soapAction="urn:isBlackListedSnortPackage"
                    style="document"/>
72              <wsdl:input>
73                  <soap:body use="literal"/>
74              </wsdl:input>
75              <wsdl:output>
76                  <soap:body use="literal"/>
77              </wsdl:output>
78          </wsdl:operation>
79          <wsdl:operation name="isBlackListedPackage">
80              <soap:operation soapAction="urn:isBlackListedPackage" style="
                    document"/>
81              <wsdl:input>
82                  <soap:body use="literal"/>
83              </wsdl:input>
84              <wsdl:output>
85                  <soap:body use="literal"/>
86              </wsdl:output>
87          </wsdl:operation>
88      </wsdl:binding>
89      <wsdl:binding name="WsAnalyserSOAP12Binding" type="
            axis2:WsAnalyserPortType">
90          <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document"/>
91          <wsdl:operation name="isBlackListedSnortPackage">
92              <soap12:operation soapAction="urn:isBlackListedSnortPackage"
                    style="document"/>
93              <wsdl:input>
94                  <soap12:body use="literal"/>
```

```
95              </wsdl:input>
96              <wsdl:output>
97                  <soap12:body use="literal"/>
98              </wsdl:output>
99          </wsdl:operation>
100         <wsdl:operation name="isBlackListedPackage">
101             <soap12:operation soapAction="urn:isBlackListedPackage" style=
                    "document"/>
102             <wsdl:input>
103                 <soap12:body use="literal"/>
104             </wsdl:input>
105             <wsdl:output>
106                 <soap12:body use="literal"/>
107             </wsdl:output>
108         </wsdl:operation>
109     </wsdl:binding>
110     <wsdl:binding name="WsAnalyserHttpBinding" type="
            axis2:WsAnalyserPortType">
111         <http:binding verb="POST"/>
112         <wsdl:operation name="isBlackListedSnortPackage">
113             <http:operation location="WsAnalyser/isBlackListedSnortPackage
                    "/>
114             <wsdl:input>
115                 <mime:content type="text/xml" part="
                        isBlackListedSnortPackage"/>
116             </wsdl:input>
117             <wsdl:output>
118                 <mime:content type="text/xml" part="
                        isBlackListedSnortPackage"/>
119             </wsdl:output>
120         </wsdl:operation>
121         <wsdl:operation name="isBlackListedPackage">
122             <http:operation location="WsAnalyser/isBlackListedPackage"/>
123             <wsdl:input>
124                 <mime:content type="text/xml" part="isBlackListedPackage"/
                        >
125             </wsdl:input>
126             <wsdl:output>
127                 <mime:content type="text/xml" part="isBlackListedPackage"/
                        >
128             </wsdl:output>
129         </wsdl:operation>
130     </wsdl:binding>
131     <wsdl:service name="WsAnalyser">
132         <wsdl:port name="WsAnalyserSOAP11port_http" binding="
                axis2:WsAnalyserSOAP11Binding">
133             <soap:address location="http://localhost:8080/axis2/services/
                    WsAnalyser"/>
```

```
134         </wsdl:port>
135         <wsdl:port name="WsAnalyserSOAP12port_http" binding="
                axis2:WsAnalyserSOAP12Binding">
136             <soap12:address location="http://localhost:8080/axis2/services
                    /WsAnalyser"/>
137         </wsdl:port>
138         <wsdl:port name="WsAnalyserHttpport" binding="
                axis2:WsAnalyserHttpBinding">
139             <http:address location="http://localhost:8080/axis2/services/
                    WsAnalyser"/>
140         </wsdl:port>
141     </wsdl:service>
142 </wsdl:definitions>
```

Listing B.1: Código do WsAnalyser

B.2 Código do WsSniffer

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:axis2="http://Sniff/" xmlns:ns1="http://packet.jpcap/xsd"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:http="
    http://schemas.xmlsoap.org/wsdl/http/" xmlns:ns0="http://Sniff/xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:mime="http://schemas.
    xmlsoap.org/wsdl/mime/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/
    soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    targetNamespace="http://Sniff/">
3     <wsdl:types>
4         <xs:schema xmlns:ns="http://Sniff/xsd" attributeFormDefault="
              qualified" elementFormDefault="qualified" targetNamespace="
              http://Sniff/xsd">
5             <xs:element name="receivePacket">
6                 <xs:complexType>
7                     <xs:sequence>
8                         <xs:element minOccurs="0" name="packet" nillable="
                              true" type="ns1:Packet"/>
9                     </xs:sequence>
10                </xs:complexType>
11            </xs:element>
12        </xs:schema>
13        <xs:schema xmlns:ax21="http://packet.jpcap/xsd"
              attributeFormDefault="qualified" elementFormDefault="qualified"
               targetNamespace="http://packet.jpcap/xsd">
14            <xs:complexType name="Packet">
15                <xs:sequence>
16                    <xs:element minOccurs="0" name="datalinkPacket"
                          nillable="true" type="ax21:DatalinkPacket"/>
17                    <xs:element minOccurs="0" name="packetData" nillable="
                          true" type="xs:base64Binary"/>
18                    <xs:element minOccurs="0" name="packetHeader" nillable
                          ="true" type="xs:base64Binary"/>
19                    <xs:element minOccurs="0" name="data" nillable="true"
                          type="xs:base64Binary"/>
20                    <xs:element minOccurs="0" name="header" nillable="true
                          " type="xs:base64Binary"/>
21                    <xs:element minOccurs="0" name="caplen" type="xs:int"/
                          >
22                    <xs:element minOccurs="0" name="len" type="xs:int"/>
23                    <xs:element minOccurs="0" name="datalink" nillable="
                          true" type="ax21:DatalinkPacket"/>
24                    <xs:element minOccurs="0" name="sec" type="xs:long"/>
25                    <xs:element minOccurs="0" name="usec" type="xs:long"/>
26                </xs:sequence>
```

```
27              </xs:complexType>
28              <xs:complexType name="DatalinkPacket">
29                  <xs:sequence/>
30              </xs:complexType>
31          </xs:schema>
32      </wsdl:types>
33      <wsdl:message name="startRequest"/>
34      <wsdl:message name="receivePacketRequest">
35          <wsdl:part name="parameters" element="ns0:receivePacket"/>
36      </wsdl:message>
37      <wsdl:portType name="WsSnifferPortType">
38          <wsdl:operation name="start">
39              <wsdl:input message="axis2:startRequest" wsaw:Action="
                    urn:start"/>
40          </wsdl:operation>
41          <wsdl:operation name="receivePacket">
42              <wsdl:input message="axis2:receivePacketRequest" wsaw:Action="
                    urn:receivePacket"/>
43          </wsdl:operation>
44      </wsdl:portType>
45      <wsdl:binding name="WsSnifferSOAP11Binding" type="
            axis2:WsSnifferPortType">
46          <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document"/>
47          <wsdl:operation name="start">
48              <soap:operation soapAction="urn:start" style="document"/>
49              <wsdl:input>
50                  <soap:body use="literal"/>
51              </wsdl:input>
52          </wsdl:operation>
53          <wsdl:operation name="receivePacket">
54              <soap:operation soapAction="urn:receivePacket" style="document
                    "/>
55              <wsdl:input>
56                  <soap:body use="literal"/>
57              </wsdl:input>
58          </wsdl:operation>
59      </wsdl:binding>
60      <wsdl:binding name="WsSnifferSOAP12Binding" type="
            axis2:WsSnifferPortType">
61          <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
                style="document"/>
62          <wsdl:operation name="start">
63              <soap12:operation soapAction="urn:start" style="document"/>
64              <wsdl:input>
65                  <soap12:body use="literal"/>
66              </wsdl:input>
67          </wsdl:operation>
```

```
68          <wsdl:operation name="receivePacket">
69              <soap12:operation soapAction="urn:receivePacket" style="
                    document"/>
70              <wsdl:input>
71                  <soap12:body use="literal"/>
72              </wsdl:input>
73          </wsdl:operation>
74      </wsdl:binding>
75      <wsdl:binding name="WsSnifferHttpBinding" type="
            axis2:WsSnifferPortType">
76          <http:binding verb="POST"/>
77          <wsdl:operation name="start">
78              <http:operation location="WsSniffer/start"/>
79              <wsdl:input>
80                  <mime:content type="text/xml" part="start"/>
81              </wsdl:input>
82          </wsdl:operation>
83          <wsdl:operation name="receivePacket">
84              <http:operation location="WsSniffer/receivePacket"/>
85              <wsdl:input>
86                  <mime:content type="text/xml" part="receivePacket"/>
87              </wsdl:input>
88          </wsdl:operation>
89      </wsdl:binding>
90      <wsdl:service name="WsSniffer">
91          <wsdl:port name="WsSnifferSOAP11port_http" binding="
                axis2:WsSnifferSOAP11Binding">
92              <soap:address location="http://localhost:8080/axis2/services/
                    WsSniffer"/>
93          </wsdl:port>
94          <wsdl:port name="WsSnifferSOAP12port_http" binding="
                axis2:WsSnifferSOAP12Binding">
95              <soap12:address location="http://localhost:8080/axis2/services
                    /WsSniffer"/>
96          </wsdl:port>
97          <wsdl:port name="WsSnifferHttpport" binding="
                axis2:WsSnifferHttpBinding">
98              <http:address location="http://localhost:8080/axis2/services/
                    WsSniffer"/>
99          </wsdl:port>
100     </wsdl:service>
101 </wsdl:definitions>
```

Listing B.2: Código do WsSniffer

B.3 Código do WsReaction

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <wsdl:definitions xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:axis2="http://Nidia/" xmlns:ns1="http://org.apache.axis2/xsd"
    xmlns:wsaw="http://www.w3.org/2006/05/addressing/wsdl" xmlns:http="
    http://schemas.xmlsoap.org/wsdl/http/" xmlns:ns0="http://Nidia/xsd"
    xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:mime="http://schemas.
    xmlsoap.org/wsdl/mime/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/
    soap/" xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/"
    targetNamespace="http://Nidia/">
3     <wsdl:types>
4         <xs:schema xmlns:ns="http://Nidia/xsd" attributeFormDefault="
              qualified" elementFormDefault="qualified" targetNamespace="
              http://Nidia/xsd">
5             <xs:element name="aplicar">
6                 <xs:complexType>
7                     <xs:sequence>
8                         <xs:element minOccurs="0" name="regra" nillable="
                              true" type="xs:string"/>
9                     </xs:sequence>
10                </xs:complexType>
11            </xs:element>
12            <xs:element name="aplicarResponse">
13                <xs:complexType>
14                    <xs:sequence>
15                        <xs:element minOccurs="0" name="return" type="
                              xs:boolean"/>
16                    </xs:sequence>
17                </xs:complexType>
18            </xs:element>
19        </xs:schema>
20    </wsdl:types>
21    <wsdl:message name="aplicarRequest">
22        <wsdl:part name="parameters" element="ns0:aplicar"/>
23    </wsdl:message>
24    <wsdl:message name="aplicarResponse">
25        <wsdl:part name="parameters" element="ns0:aplicarResponse"/>
26    </wsdl:message>
27    <wsdl:portType name="WsReactionPortType">
28        <wsdl:operation name="aplicar">
29            <wsdl:input message="axis2:aplicarRequest" wsaw:Action="
                  urn:aplicar"/>
30            <wsdl:output message="axis2:aplicarResponse" wsaw:Action="
                  urn:aplicarResponse"/>
31        </wsdl:operation>
32    </wsdl:portType>
```

```
33    <wsdl:binding name="WsReactionSOAP11Binding" type="
          axis2:WsReactionPortType">
34        <soap:binding transport="http://schemas.xmlsoap.org/soap/http"
              style="document"/>
35        <wsdl:operation name="aplicar">
36            <soap:operation soapAction="urn:aplicar" style="document"/>
37            <wsdl:input>
38                <soap:body use="literal"/>
39            </wsdl:input>
40            <wsdl:output>
41                <soap:body use="literal"/>
42            </wsdl:output>
43        </wsdl:operation>
44    </wsdl:binding>
45    <wsdl:binding name="WsReactionSOAP12Binding" type="
          axis2:WsReactionPortType">
46        <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"
              style="document"/>
47        <wsdl:operation name="aplicar">
48            <soap12:operation soapAction="urn:aplicar" style="document"/>
49            <wsdl:input>
50                <soap12:body use="literal"/>
51            </wsdl:input>
52            <wsdl:output>
53                <soap12:body use="literal"/>
54            </wsdl:output>
55        </wsdl:operation>
56    </wsdl:binding>
57    <wsdl:binding name="WsReactionHttpBinding" type="
          axis2:WsReactionPortType">
58        <http:binding verb="POST"/>
59        <wsdl:operation name="aplicar">
60            <http:operation location="WsReaction/aplicar"/>
61            <wsdl:input>
62                <mime:content type="text/xml" part="aplicar"/>
63            </wsdl:input>
64            <wsdl:output>
65                <mime:content type="text/xml" part="aplicar"/>
66            </wsdl:output>
67        </wsdl:operation>
68    </wsdl:binding>
69    <wsdl:service name="WsReaction">
70        <wsdl:port name="WsReactionSOAP11port_http" binding="
              axis2:WsReactionSOAP11Binding">
71            <soap:address location="http://localhost:8080/axis2/services/
                  WsReaction"/>
72        </wsdl:port>
```

```
73          <wsdl:port name="WsReactionSOAP12port_http" binding="
                axis2:WsReactionSOAP12Binding">
74              <soap12:address location="http://localhost:8080/axis2/services
                    /WsReaction"/>
75          </wsdl:port>
76          <wsdl:port name="WsReactionHttpport" binding="
                axis2:WsReactionHttpBinding">
77              <http:address location="http://localhost:8080/axis2/services/
                    WsReaction"/>
78          </wsdl:port>
79      </wsdl:service>
80  </wsdl:definitions>
```

Listing B.3: Código do WsReaction

B.4 Código BPEL

```
1 <!-- NidiaUm BPEL Process [Generated by the Eclipse BPEL Designer] -->
2 <!-- Date: Tue May 01 21:25:49 BRT 2012 -->
3 <bpel:process name="NidiaUm"
4           targetNamespace="http://localhost:8080/bpel/sample"
5           suppressJoinFailure="yes"
6           xmlns:tns="http://localhost:8080/bpel/sample"
7           xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
                executable"
8           >
9
10     <!-- Import the client WSDL -->
11   <bpel:import location="NidiaUmArtifacts.wsdl" namespace="http://
         localhost:8080/bpel/sample"
12            importType="http://schemas.xmlsoap.org/wsdl/" />
13
14     <!-- ================================================================
           -->
15     <!-- PARTNERLINKS
                                                                    -->
16     <!-- List of services participating in this BPEL process
                          -->
17     <!-- ================================================================
           -->
18     <bpel:partnerLinks>
19         <!-- The 'client' role represents the requester of this service.
              -->
20         <bpel:partnerLink name="client"
21                           partnerLinkType="tns:NidiaUm"
22                           myRole="NidiaUmProvider"
23                           />
24     </bpel:partnerLinks>
25
26     <!-- ================================================================
           -->
27     <!-- VARIABLES
                                                                      -->
28     <!-- List of messages and XML documents used within this BPEL process
           -->
29     <!-- ================================================================
           -->
30     <bpel:variables>
31         <!-- Reference to the message passed as input during initiation --
              >
32         <bpel:variable name="input"
33                        messageType="tns:NidiaUmRequestMessage"/>
```

```
34
35          <!--
36            Reference to the message that will be returned to the requester
37            -->
38          <bpel:variable name="output"
39                    messageType="tns:NidiaUmResponseMessage"/>
40      </bpel:variables>
41
42      <!-- ================================================================
            -->
43      <!-- ORCHESTRATION LOGIC
                                                                    -->
44      <!-- Set of activities coordinating the flow of messages across the
             -->
45      <!-- services integrated within this business process
                          -->
46      <!-- ================================================================
            -->
47      <bpel:sequence name="main">
48
49          <!-- Receive input from requester.
50               Note: This maps to operation defined in NidiaUm.wsdl
51              -->
52          <bpel:receive name="receiveInput" partnerLink="client"
53                   portType="tns:NidiaUm"
54                   operation="process" variable="input"
55                   createInstance="yes"/>
56
57          <bpel:flow name="Flow">
58              <bpel:sequence name="Sequence">
59                  <bpel:invoke name="Invoke_WsSniffer"></bpel:invoke>
60                  <bpel:invoke name="Invoke_WsFormater"></bpel:invoke>
61                  <bpel:invoke name="Invoke_WsAnalyser"></bpel:invoke>
62                  <bpel:invoke name="Invoke_WsSender"></bpel:invoke>
63              </bpel:sequence>                    </bpel:flow>
64
65          <!-- Generate reply to synchronous request -->
66          <bpel:reply name="replyOutput"
67                 partnerLink="client"
68                 portType="tns:NidiaUm"
69                 operation="process"
70                 variable="output"
71                 />
72      </bpel:sequence>
73  </bpel:process>
```

Listing B.4: Código BPEL

Apêndice C

Código BPEL dos testes realizados

C.1 Código da composição dos serviços WsSniffer e WsFormater

```
1 <!-- NidiaUm BPEL Process [Generated by the Eclipse BPEL Designer] -->
2 <!-- Date: Tue May 01 21:25:49 BRT 2012 -->
3 <bpel:process name="NidiaUm"
4           targetNamespace="http://localhost:8080/bpel/sample"
5           suppressJoinFailure="yes"
6           xmlns:tns="http://localhost:8080/bpel/sample"
7           xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
                executable"
8           >
9
10     <!-- Import the client WSDL -->
11   <bpel:import location="NidiaUmArtifacts.wsdl" namespace="http://
         localhost:8080/bpel/sample"
12            importType="http://schemas.xmlsoap.org/wsdl/" />
13
14     <!-- ================================================================
           -->
15     <!-- PARTNERLINKS
                                                                    -->
16     <!-- List of services participating in this BPEL process
                       -->
17     <!-- ================================================================
           -->
18     <bpel:partnerLinks>
19         <!-- The 'client' role represents the requester of this service.
              -->
20         <bpel:partnerLink name="client"
21                       partnerLinkType="tns:NidiaUm"
22                       myRole="NidiaUmProvider"
```

```
23                      />
24      </bpel:partnerLinks>
25
26      <!-- ================================================================
            -->
27      <!-- VARIABLES
                                                                        -->
28      <!-- List of messages and XML documents used within this BPEL process
            -->
29      <!-- ================================================================
            -->
30      <bpel:variables>
31          <!-- Reference to the message passed as input during initiation --
               >
32          <bpel:variable name="input"
33                    messageType="tns:NidiaUmRequestMessage"/>
34
35          <!--
36            Reference to the message that will be returned to the requester
37            -->
38          <bpel:variable name="output"
39                    messageType="tns:NidiaUmResponseMessage"/>
40      </bpel:variables>
41
42      <!-- ================================================================
            -->
43      <!-- ORCHESTRATION LOGIC
                                                              -->
44      <!-- Set of activities coordinating the flow of messages across the
             -->
45      <!-- services integrated within this business process
                           -->
46      <!-- ================================================================
            -->
47      <bpel:sequence name="main">
48
49          <!-- Receive input from requester.
50               Note: This maps to operation defined in NidiaUm.wsdl
51              -->
52          <bpel:receive name="receiveInput" partnerLink="client"
53                    portType="tns:NidiaUm"
54                    operation="process" variable="input"
55                    createInstance="yes"/>
56
57          <bpel:flow name="Flow">
58              <bpel:sequence name="Sequence">
59                  <bpel:invoke name="Invoke_WsSniffer"></bpel:invoke>
60                  <bpel:invoke name="Invoke_WsFormater"></bpel:invoke>
```

```
61                  <bpel:invoke name="Invoke_WsAnalyser"></bpel:invoke>
62                  <bpel:invoke name="Invoke_WsSender"></bpel:invoke>
63              </bpel:sequence>                        </bpel:flow>
64
65          <!-- Generate reply to synchronous request -->
66          <bpel:reply name="replyOutput"
67                  partnerLink="client"
68                  portType="tns:NidiaUm"
69                  operation="process"
70                  variable="output"
71                  />
72      </bpel:sequence>
73  </bpel:process>
```

Listing C.1: Código BPEL

C.2 Código do teste WsAnalyser

```
1 <!-- WS_Analyser BPEL Process [Generated by the Eclipse BPEL Designer] -->
2 <!-- Date: Tue Jul 24 16:54:16 BRT 2012 -->
3 <bpel:process name="WS_Analyser"
4          targetNamespace="http://helloworld.localhost"
5          suppressJoinFailure="yes"
6          xmlns:tns="http://helloworld.localhost"
7          xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
               executable"
8          xmlns:ns="http://Nidia/" xmlns:xsd="http://Nidia/xsd">
9
10     <!-- Import the client WSDL -->
11     <bpel:import namespace="http://Nidia/" location="WsAnalyser.wsdl"
           importType="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
12     <bpel:import location="WS_AnalyserArtifacts.wsdl" namespace="http://
           helloworld.localhost"
13           importType="http://schemas.xmlsoap.org/wsdl/" />
14
15     <!-- ================================================================
           -->
16     <!-- PARTNERLINKS
                                                                    -->
17     <!-- List of services participating in this BPEL process
                          -->
18     <!-- ================================================================
           -->
19     <bpel:partnerLinks>
20         <!-- The 'client' role represents the requester of this service.
               -->
21         <bpel:partnerLink name="client"
22                      partnerLinkType="tns:WS_Analyser"
23                      myRole="WS_AnalyserProvider"
24                      />
25         <bpel:partnerLink name="WSA" partnerLinkType="tns:WS_Analyser"
               partnerRole="WS_AnalyserProvider"></bpel:partnerLink>
26         <bpel:partnerLink name="WSA" partnerLinkType="tns:WSA1"
               partnerRole="WSA2"></bpel:partnerLink>
27         <bpel:partnerLink name="Analise_NEWPL" partnerLinkType="
               tns:Analise_NEWPLT" partnerRole="Analise_NEWPLRole"></
               bpel:partnerLink>
28     </bpel:partnerLinks>
29
30     <!-- ================================================================
           -->
31     <!-- VARIABLES
                                                                    -->
```

```
32      <!-- List of messages and XML documents used within this BPEL process
            -->
33      <!-- ====================================================================
            -->
34      <bpel:variables>
35          <!-- Reference to the message passed as input during initiation --
               >
36          <bpel:variable name="input"
37                    messageType="tns:WS_AnalyserRequestMessage"/>
38
39          <!--
40            Reference to the message that will be returned to the requester
41            -->
42          <bpel:variable name="output"
43                    messageType="tns:WS_AnalyserResponseMessage"/>
44          <bpel:variable name="Analise_NEWPLResponse" messageType="
               ns:isBlackListedSnortPackageResponse"></bpel:variable>
45          <bpel:variable name="Analise_NEWPLRequest" messageType="
               ns:isBlackListedSnortPackageRequest"></bpel:variable>
46          <bpel:variable name="Variable"></bpel:variable>
47      </bpel:variables>
48
49      <!-- ====================================================================
            -->
50      <!-- ORCHESTRATION LOGIC
                                                              -->
51      <!-- Set of activities coordinating the flow of messages across the
             -->
52      <!-- services integrated within this business process
                           -->
53      <!-- ====================================================================
            -->
54      <bpel:sequence name="main">
55
56          <!-- Receive input from requester.
57               Note: This maps to operation defined in WS_Analyser.wsdl
58              -->
59          <bpel:receive name="receiveInput" partnerLink="client"
60                    createInstance="yes" operation="process" portType="
                       tns:WS_Analyser" variable="input"/>
61          <bpel:assign validate="no" name="AssignInputWsAnalyser">
62
63
64
65
66
67
68              <bpel:copy>
```

```
69                <bpel:from><bpel:literal><ns:isBlackListedSnortPackage
                      xmlns:ns="http://Nidia/xsd" xmlns:xsi="http://www.w3.
                      org/2001/XMLSchema-instance">
70    <ns:XmlPacket>
71      <ns:dst_ip>ns:dst_ip</ns:dst_ip>
72      <ns:dst_port>0</ns:dst_port>
73      <ns:sec>0</ns:sec>
74      <ns:sec_port>0</ns:sec_port>
75      <ns:src_ip>ns:src_ip</ns:src_ip>
76      <ns:src_port>0</ns:src_port>
77      <ns:usec>0</ns:usec>
78    </ns:XmlPacket>
79  </ns:isBlackListedSnortPackage>
80  </bpel:literal></bpel:from>
81                <bpel:to variable="Analise_NEWPLRequest" part="parameters"
                      ></bpel:to>
82            </bpel:copy>
83            <bpel:copy>
84                <bpel:from part="payload" variable="input">
85                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                          .0:sublang:xpath1.0"><![CDATA[tns:ip]]></bpel:query
                          >
86                </bpel:from>
87                <bpel:to part="parameters" variable="Analise_NEWPLRequest"
                      >
88                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                          .0:sublang:xpath1.0"><![CDATA[xsd:XmlPacket/
                          xsd:dst_ip]]></bpel:query>
89                </bpel:to>
90            </bpel:copy>
91            <bpel:copy>
92                <bpel:from part="payload" variable="input">
93                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                          .0:sublang:xpath1.0"><![CDATA[tns:port]]></
                          bpel:query>
94                </bpel:from>
95                <bpel:to part="parameters" variable="Analise_NEWPLRequest"
                      >
96                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                          .0:sublang:xpath1.0"><![CDATA[xsd:XmlPacket/
                          xsd:dst_port]]></bpel:query>
97                </bpel:to>
98            </bpel:copy>
99        </bpel:assign>
100       <bpel:invoke name="InvokeWS_Analyser" partnerLink="Analise_NEWPL"
              operation="isBlackListedSnortPackage" portType="
              ns:WsAnalyserPortType" inputVariable="Analise_NEWPLRequest"
              outputVariable="Analise_NEWPLResponse"></bpel:invoke>
```

```
101
102         <!-- Generate reply to synchronous request -->
103         <bpel:assign validate="no" name="AssignOutputWsAnalyser">
104
105
106             <bpel:copy>
107                 <bpel:from><bpel:literal><tns:WS_AnalyserResponse
                        xmlns:tns="http://helloworld.localhost" xmlns:xsi="
                        http://www.w3.org/2001/XMLSchema-instance">
108   <tns:result>tns:result</tns:result>
109 </tns:WS_AnalyserResponse>
110 </bpel:literal></bpel:from>
111                 <bpel:to variable="output" part="payload"></bpel:to>
112             </bpel:copy>
113             <bpel:copy>
114                 <bpel:from part="parameters" variable="
                        Analise_NEWPLResponse">
115                     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0"><![CDATA[xsd:return]]></
                            bpel:query>
116                 </bpel:from>
117                 <bpel:to part="payload" variable="output">
118                     <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0"><![CDATA[tns:result]]></
                            bpel:query>
119                 </bpel:to>
120             </bpel:copy>
121         </bpel:assign>
122         <bpel:reply name="replyOutput"
123                 partnerLink="client"
124                 portType="tns:WS_Analyser"
125                 operation="process"
126                 variable="output"
127                 />
128     </bpel:sequence>
129 </bpel:process>
```

Listing C.2: Código BPEL WsAnalyser

C.3 Código do teste WsReaction

```
1 <!-- BPEL_escritafw BPEL Process [Generated by the Eclipse BPEL Designer]
      -->
2 <!-- Date: Tue Sep 04 16:20:50 BRT 2012 -->
3 <bpel:process name="BPEL_escritafw"
4           targetNamespace="http://localhost/"
5           suppressJoinFailure="yes"
6           xmlns:tns="http://localhost/"
7           xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
                executable"
8           >
9
10      <!-- Import the client WSDL -->
11      <bpel:import namespace="http://nidia/" location="../../
            BPEL_Escritafwnew/bpelContent/Escrever.wsdl" importType="http://
            schemas.xmlsoap.org/wsdl/"></bpel:import>
12      <bpel:import location="BPEL_escritafwArtifacts.wsdl" namespace="http:
            //localhost/"
13              importType="http://schemas.xmlsoap.org/wsdl/" />
14
15      <!-- ================================================================
            -->
16      <!-- PARTNERLINKS
                                                                    -->
17      <!-- List of services participating in this BPEL process
                        -->
18      <!-- ================================================================
            -->
19      <bpel:partnerLinks>
20          <!-- The 'client' role represents the requester of this service.
                -->
21          <bpel:partnerLink name="client"
22                      partnerLinkType="tns:BPEL_escritafw"
23                      myRole="BPEL_escritafwProvider"
24                      />
25          <bpel:partnerLink name="EscreverPL" partnerLinkType="
                tns:EscreverPLT" partnerRole="EscreverPLRole"></
                bpel:partnerLink>
26      </bpel:partnerLinks>
27
28      <!-- ================================================================
            -->
29      <!-- VARIABLES
                                                                    -->
30      <!-- List of messages and XML documents used within this BPEL process
            -->
```

```
31      <!-- ================================================================
            -->
32      <bpel:variables>
33          <!-- Reference to the message passed as input during initiation --
                >
34          <bpel:variable name="input"
35                    messageType="tns:BPEL_escritafwRequestMessage"/>
36
37          <!--
38            Reference to the message that will be returned to the requester
39            -->
40          <bpel:variable name="output"
41                    messageType="tns:BPEL_escritafwResponseMessage"/>
42      </bpel:variables>
43
44      <!-- ================================================================
            -->
45      <!-- ORCHESTRATION LOGIC
                                                                  -->
46      <!-- Set of activities coordinating the flow of messages across the
             -->
47      <!-- services integrated within this business process
                          -->
48      <!-- ================================================================
            -->
49      <bpel:sequence name="main">
50
51          <!-- Receive input from requester.
52               Note: This maps to operation defined in BPEL_escritafw.wsdl
53              -->
54          <bpel:receive name="receiveInput" partnerLink="client"
55                   portType="tns:BPEL_escritafw"
56                   operation="process" variable="input"
57                   createInstance="yes"/>
58          <bpel:assign validate="no" name="AssignInputWsReaction">
59              <bpel:copy>
60                  <bpel:from><bpel:literal><tns:BPEL_escritafwRequest
                        xmlns:tns="http://localhost/" xmlns:xsi="http://www.w3.
                        org/2001/XMLSchema-instance">
61    <tns:input>tns:input</tns:input>
62  </tns:BPEL_escritafwRequest>
63  </bpel:literal></bpel:from>
64                  <bpel:to variable="input" part="payload"></bpel:to>
65              </bpel:copy>
66              <bpel:copy>
67                  <bpel:from part="payload" variable="input">
68                      <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0"><![CDATA[tns:input]]></
```

```
                    bpel:query>
69                </bpel:from>
70                <bpel:to part="payload" variable="input">
71                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                        .0:sublang:xpath1.0"><![CDATA[tns:input]]></
                        bpel:query>
72                </bpel:to>
73            </bpel:copy>
74        </bpel:assign>
75        <bpel:invoke name="InvokeWsReaction" partnerLink="EscreverPL"
            operation="process" portType="tns:BPEL_escritafw" inputVariable
            ="input" outputVariable="output"></bpel:invoke>
76        <bpel:assign validate="no" name="AssignOutputWsReaction">
77            <bpel:copy>
78                <bpel:from><bpel:literal><tns:BPEL_escritafwResponse
                    xmlns:tns="http://localhost/" xmlns:xsi="http://www.w3.
                    org/2001/XMLSchema-instance">
79  <tns:result>tns:result</tns:result>
80 </tns:BPEL_escritafwResponse>
81 </bpel:literal></bpel:from>
82                <bpel:to variable="output" part="payload"></bpel:to>
83            </bpel:copy>
84            <bpel:copy>
85                <bpel:from part="payload" variable="output">
86                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                        .0:sublang:xpath1.0"><![CDATA[tns:result]]></
                        bpel:query>
87                </bpel:from>
88                <bpel:to part="payload" variable="output">
89                    <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                        .0:sublang:xpath1.0"><![CDATA[tns:result]]></
                        bpel:query>
90                </bpel:to>
91            </bpel:copy>
92        </bpel:assign>
93
94        <!-- Generate reply to synchronous request -->
95        <bpel:reply name="replyOutput"
96                partnerLink="client"
97                portType="tns:BPEL_escritafw"
98                operation="process"
99                variable="output"
100               />
101    </bpel:sequence>
102 </bpel:process>
```

Listing C.3: Código BPEL WsReaction

C.4 Código do teste WsAnalyser Consulta IP

```
1 <!-- WS_Busca BPEL Process [Generated by the Eclipse BPEL Designer] -->
2 <!-- Date: Sun Dec 09 18:02:57 BRST 2012 -->
3 <bpel:process name="WS_Busca"
4           targetNamespace="http://localhost:7070/"
5           suppressJoinFailure="yes"
6           xmlns:tns="http://localhost:7070/"
7           xmlns:bpel="http://docs.oasis-open.org/wsbpel/2.0/process/
                executable"
8           xmlns:ns="http://nidia/" xmlns:xsd="http://nidia/xsd">
9
10      <!-- Import the client WSDL -->
11      <bpel:import namespace="http://nidia/" location="Main.wsdl" importType
            ="http://schemas.xmlsoap.org/wsdl/"></bpel:import>
12      <bpel:import location="WS_BuscaArtifacts.wsdl" namespace="http://
            localhost:7070/"
13            importType="http://schemas.xmlsoap.org/wsdl/" />
14
15      <!-- ================================================================
            -->
16      <!-- PARTNERLINKS
                                                                    -->
17      <!-- List of services participating in this BPEL process
                          -->
18      <!-- ================================================================
            -->
19      <bpel:partnerLinks>
20          <!-- The 'client' role represents the requester of this service.
                -->
21          <bpel:partnerLink name="client"
22                       partnerLinkType="tns:WS_Busca"
23                       myRole="WS_BuscaProvider"
24                       />
25          <bpel:partnerLink name="BuscaPL" partnerLinkType="tns:BuscaPLT"
                partnerRole="BuscaPLRole"></bpel:partnerLink>
26      </bpel:partnerLinks>
27
28      <!-- ================================================================
            -->
29      <!-- VARIABLES
                                                                    -->
30      <!-- List of messages and XML documents used within this BPEL process
            -->
31      <!-- ================================================================
            -->
32      <bpel:variables>
```

```
33          <!-- Reference to the message passed as input during initiation --
                >
34          <bpel:variable name="input"
35                    messageType="tns:WS_BuscaRequestMessage"/>
36
37          <!--
38            Reference to the message that will be returned to the requester
39            -->
40          <bpel:variable name="output"
41                    messageType="tns:WS_BuscaResponseMessage"/>
42          <bpel:variable name="BuscaPLResponse" messageType="
                ns:buscaResponse"></bpel:variable>
43          <bpel:variable name="BuscaPLRequest" messageType="ns:buscaRequest"
                ></bpel:variable>
44      </bpel:variables>
45
46      <!-- ================================================================
            -->
47      <!-- ORCHESTRATION LOGIC
                                                                    -->
48      <!-- Set of activities coordinating the flow of messages across the
             -->
49      <!-- services integrated within this business process
                           -->
50      <!-- ================================================================
            -->
51      <bpel:sequence name="main">
52
53          <!-- Receive input from requester.
54               Note: This maps to operation defined in WS_Busca.wsdl
55               -->
56          <bpel:receive name="receiveInput" partnerLink="client"
57                    portType="tns:WS_Busca"
58                    operation="process" variable="input"
59                    createInstance="yes"/>
60          <bpel:assign validate="no" name="AssignInputConsultaIP">
61              <bpel:copy>
62                  <bpel:from><bpel:literal><ns:busca xmlns:ns="http://nidia/
                        xsd" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
                        instance">
63    <ns:ip>ns:ip</ns:ip>
64  </ns:busca>
65  </bpel:literal></bpel:from>
66                  <bpel:to variable="BuscaPLRequest" part="parameters"></
                        bpel:to>
67              </bpel:copy>
68              <bpel:copy>
69                  <bpel:from part="payload" variable="input">
```

```
70                      <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0">
71                          <![CDATA[tns:input]]>
72                      </bpel:query>
73                  </bpel:from>
74                  <bpel:to part="parameters" variable="BuscaPLRequest">
75                      <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0"><![CDATA[xsd:ip]]></bpel:query
                            >
76                  </bpel:to>
77              </bpel:copy>
78          </bpel:assign>
79          <bpel:invoke name="Invoke_ConsultaIP" partnerLink="BuscaPL"
               operation="busca" portType="ns:MainPortType" inputVariable="
               BuscaPLRequest" outputVariable="BuscaPLResponse"></bpel:invoke>
80          <bpel:assign validate="no" name="AssignOutputConsultaIP">
81              <bpel:copy>
82                  <bpel:from><bpel:literal><tns:WS_BuscaResponse xmlns:tns="
                        http://localhost:7070/" xmlns:xsi="http://www.w3.org
                        /2001/XMLSchema-instance">
83   <tns:result>tns:result</tns:result>
84 </tns:WS_BuscaResponse>
85 </bpel:literal></bpel:from>
86                  <bpel:to variable="output" part="payload"></bpel:to>
87              </bpel:copy>
88              <bpel:copy>
89                  <bpel:from part="payload" variable="output">
90                      <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0"><![CDATA[tns:result]]></
                            bpel:query>
91                  </bpel:from>
92                  <bpel:to part="payload" variable="output">
93                      <bpel:query queryLanguage="urn:oasis:names:tc:wsbpel:2
                            .0:sublang:xpath1.0"><![CDATA[tns:result]]></
                            bpel:query>
94                  </bpel:to>
95              </bpel:copy>
96          </bpel:assign>
97
98          <!-- Generate reply to synchronous request -->
99          <bpel:reply name="replyOutput"
100                 partnerLink="client"
101                 portType="tns:WS_Busca"
102                 operation="process"
103                 variable="output"
104                 />
105     </bpel:sequence>
106 </bpel:process>
```

Listing C.4: Código BPEL Busca IP

Apêndice D

Código BPEL dos testes realizados

D.1 Log de acesso a porta 22 - IP 101.78.202.18

```
1 Sep 16 19:11:03 bacuri sshd[31370]: User root from 101.78.202.18 not
    allowed because not listed in AllowUsers
2 Sep 16 19:11:03 bacuri sshd[31370]: Failed password for invalid user root
    from 101.78.202.18 port 53669 ssh2
3 Sep 16 19:11:04 bacuri sshd[20748]: Received disconnect from 101.78.202.18
    : 11: Bye Bye
4 Sep 16 19:11:08 bacuri sshd[342]: User root from 101.78.202.18 not allowed
     because not listed in AllowUsers
5 Sep 16 19:11:08 bacuri sshd[342]: Failed password for invalid user root
    from 101.78.202.18 port 53812 ssh2
6 Sep 16 19:11:08 bacuri sshd[19365]: Received disconnect from 101.78.202.18
    : 11: Bye Bye
7 Sep 16 19:11:13 bacuri sshd[13765]: User root from 101.78.202.18 not
    allowed because not listed in AllowUsers
8 Sep 16 19:11:13 bacuri sshd[13765]: Failed password for invalid user root
    from 101.78.202.18 port 53982 ssh2
9 Sep 16 19:11:13 bacuri sshd[6884]: Received disconnect from 101.78.202.18:
     11: Bye Bye
10 Sep 16 19:11:17 bacuri sshd[11854]: User root from 101.78.202.18 not
    allowed because not listed in AllowUsers
11 Sep 16 19:11:17 bacuri sshd[11854]: Failed password for invalid user root
    from 101.78.202.18 port 54154 ssh2
12 Sep 16 19:11:18 bacuri sshd[28562]: Received disconnect from 101.78.202.18
    : 11: Bye Bye
13 Sep 16 19:11:22 bacuri sshd[8887]: User root from 101.78.202.18 not
    allowed because not listed in AllowUsers
14 Sep 16 19:11:22 bacuri sshd[8887]: Failed password for invalid user root
    from 101.78.202.18 port 54327 ssh2
15 Sep 16 19:11:23 bacuri sshd[1721]: Received disconnect from 101.78.202.18:
     11: Bye Bye
```

```
16 Sep 16 19:11:27 bacuri sshd[17264]: User root from 101.78.202.18 not
     allowed because not listed in AllowUsers
17 Sep 16 19:11:27 bacuri sshd[17264]: Failed password for invalid user root
     from 101.78.202.18 port 54483 ssh2
18 Sep 16 19:11:27 bacuri sshd[6090]: Received disconnect from 101.78.202.18:
      11: Bye Bye
19 Sep 16 19:11:31 bacuri sshd[10568]: User root from 101.78.202.18 not
     allowed because not listed in AllowUsers
20 Sep 16 19:11:31 bacuri sshd[10568]: Failed password for invalid user root
     from 101.78.202.18 port 54615 ssh2
21 Sep 16 19:11:32 bacuri sshd[32570]: Received disconnect from 101.78.202.18
     : 11: Bye Bye
22 Sep 16 19:11:36 bacuri sshd[12945]: User root from 101.78.202.18 not
     allowed because not listed in AllowUsers
23 Sep 16 19:11:36 bacuri sshd[12945]: Failed password for invalid user root
     from 101.78.202.18 port 54794 ssh2
24 Sep 16 19:11:37 bacuri sshd[16031]: Received disconnect from 101.78.202.18
     : 11: Bye Bye
25 Sep 16 19:11:41 bacuri sshd[10527]: User root from 101.78.202.18 not
     allowed because not listed in AllowUsers
26 Sep 16 19:11:41 bacuri sshd[10527]: Failed password for invalid user root
     from 101.78.202.18 port 54959 ssh2
27 Sep 16 19:11:41 bacuri sshd[29343]: Received disconnect from 101.78.202.18
     : 11: Bye Bye
28 Sep 16 19:11:46 bacuri sshd[1944]: User root from 101.78.202.18 not
     allowed because not listed in AllowUsers
29 Sep 16 19:11:46 bacuri sshd[1944]: Failed password for invalid user root
     from 101.78.202.18 port 55128 ssh2
30 Sep 16 19:11:46 bacuri sshd[7166]: Received disconnect from 101.78.202.18:
      11: Bye Bye
```

Listing D.1: Log de acesso do IP 101

D.2 Log de acesso a porta 22 - IP 222.221.2.210

```
1  1 Sep 18 02:44:57 bacuri sshd[28214]: Did not receive identification
       string from 222.221.2.210
2  2 Sep 18 02:51:51 bacuri sshd[2011]: User root from 222.221.2.210 not
       allowed because not listed in AllowUsers
3  3 Sep 18 02:51:51 bacuri sshd[2011]: Failed password for invalid user
       root from 222.221.2.210 port 15085 ssh2
4  4 Sep 18 02:51:51 bacuri sshd[25370]: Received disconnect from
       222.221.2.210: 11: Bye Bye
5  5 Sep 18 02:51:56 bacuri sshd[26645]: User root from 222.221.2.210 not
       allowed because not listed in AllowUsers
6  6 Sep 18 02:51:56 bacuri sshd[26645]: Failed password for invalid user
       root from 222.221.2.210 port 16767 ssh2
7  7 Sep 18 02:51:56 bacuri sshd[3560]: Received disconnect from
       222.221.2.210: 11: Bye Bye
8  8 Sep 18 02:52:00 bacuri sshd[2459]: Failed password for admin from
       222.221.2.210 port 14177 ssh2
9  9 Sep 18 02:52:01 bacuri sshd[9751]: Received disconnect from
       222.221.2.210: 11: Bye Bye
10 10 Sep 18 02:52:05 bacuri sshd[17466]: Failed password for admin from
       222.221.2.210 port 16034 ssh2
11 11 Sep 18 02:52:06 bacuri sshd[23391]: Received disconnect from
       222.221.2.210: 11: Bye Bye
12 12 Sep 18 02:52:10 bacuri sshd[10500]: Invalid user test from
       222.221.2.210
13 13 Sep 18 02:52:10 bacuri sshd[10500]: Failed password for invalid user
       test from 222.221.2.210 port 16224 ssh2
14 14 Sep 18 02:52:11 bacuri sshd[17050]: Received disconnect from
       222.221.2.210: 11: Bye Bye
15 15 Sep 18 02:52:16 bacuri sshd[20302]: Invalid user test from
       222.221.2.210
16 16 Sep 18 02:52:16 bacuri sshd[20302]: Failed password for invalid user
       test from 222.221.2.210 port 14819 ssh2
17 17 Sep 18 02:52:17 bacuri sshd[19637]: Received disconnect from
       222.221.2.210: 11: Bye Bye
18 18 Sep 18 02:52:21 bacuri sshd[8722]: Invalid user user from
       222.221.2.210
19 19 Sep 18 02:52:21 bacuri sshd[8722]: Failed password for invalid user
       user from 222.221.2.210 port 13832 ssh2
20 20 Sep 18 02:52:22 bacuri sshd[5403]: Received disconnect from
       222.221.2.210: 11: Bye Bye
21 21 Sep 18 02:52:26 bacuri sshd[2469]: Invalid user user from
       222.221.2.210
22 22 Sep 18 02:52:26 bacuri sshd[2469]: Failed password for invalid user
       user from 222.221.2.210 port 14598 ssh2
23 23 Sep 18 02:52:26 bacuri sshd[27235]: Received disconnect from
       222.221.2.210: 11: Bye Bye
```

```
24 24 Sep 18 02:52:30 bacuri sshd[15315]: Invalid user user1 from
       222.221.2.210
25 25 Sep 18 02:52:30 bacuri sshd[15315]: Failed password for invalid user
       user1 from 222.221.2.210 port 16329 ssh2
26 26 Sep 18 02:52:31 bacuri sshd[30795]: Received disconnect from
       222.221.2.210: 11: Bye Bye
27 27 Sep 18 02:52:38 bacuri sshd[15231]: Invalid user user1 from
       222.221.2.210
28 28 Sep 18 02:52:38 bacuri sshd[15231]: Failed password for invalid user
       user1 from 222.221.2.210 port 12397 ssh2
29 29 Sep 18 02:52:39 bacuri sshd[157]: Received disconnect from
       222.221.2.210: 11: Bye Bye
30 30 Sep 18 02:52:43 bacuri sshd[16883]: Invalid user user1 from
       222.221.2.210
31 31 Sep 18 02:52:43 bacuri sshd[16883]: Failed password for invalid user
       user1 from 222.221.2.210 port 15559 ssh2
32 32 Sep 18 02:52:44 bacuri sshd[14313]: Received disconnect from
       222.221.2.210: 11: Bye Bye
33 33 Sep 18 02:52:48 bacuri sshd[31949]: Invalid user user from
       222.221.2.210
34 34 Sep 18 02:52:48 bacuri sshd[31949]: Failed password for invalid user
       user from 222.221.2.210 port 13773 ssh2
35 35 Sep 18 02:52:48 bacuri sshd[24378]: Received disconnect from
       222.221.2.210: 11: Bye Bye
36 36 Sep 18 02:52:53 bacuri sshd[3971]: Invalid user user1 from
       222.221.2.210
37 37 Sep 18 02:52:53 bacuri sshd[3971]: Failed password for invalid user
       user1 from 222.221.2.210 port 15891 ssh2
38 38 Sep 18 02:52:53 bacuri sshd[22212]: Received disconnect from
       222.221.2.210: 11: Bye Bye
39 39 Sep 18 02:53:00 bacuri sshd[21881]: Invalid user mail from
       222.221.2.210
40 40 Sep 18 02:53:00 bacuri sshd[21881]: Failed password for invalid user
       mail from 222.221.2.210 port 15737 ssh2
41 41 Sep 18 02:53:01 bacuri sshd[6867]: Received disconnect from
       222.221.2.210: 11: Bye Bye
42 42 Sep 18 02:53:05 bacuri sshd[24125]: Invalid user mail from
       222.221.2.210
43 43 Sep 18 02:53:05 bacuri sshd[24125]: Failed password for invalid user
       mail from 222.221.2.210 port 15223 ssh2
44 44 Sep 18 02:53:06 bacuri sshd[3769]: Received disconnect from
       222.221.2.210: 11: Bye Bye
45 45 Sep 18 02:53:10 bacuri sshd[7365]: Invalid user mail from
       222.221.2.210
46 46 Sep 18 02:53:10 bacuri sshd[7365]: Failed password for invalid user
       mail from 222.221.2.210 port 15754 ssh2
47 47 Sep 18 02:53:10 bacuri sshd[32506]: Received disconnect from
       222.221.2.210: 11: Bye Bye
```

```
48 48 Sep 18 02:53:15 bacuri sshd[15096]: User root from 222.221.2.210 not
       allowed because not listed in AllowUsers
49 49 Sep 18 02:53:15 bacuri sshd[15096]: Failed password for invalid user
       root from 222.221.2.210 port 16262 ssh2
50 50 Sep 18 02:53:15 bacuri sshd[25312]: Received disconnect from
       222.221.2.210: 11: Bye Bye
51 51 Sep 18 02:53:19 bacuri sshd[9144]: User root from 222.221.2.210 not
       allowed because not listed in AllowUsers
52 52 Sep 18 02:53:19 bacuri sshd[9144]: Failed password for invalid user
       root from 222.221.2.210 port 14049 ssh2
53 53 Sep 18 02:53:20 bacuri sshd[411]: Received disconnect from
       222.221.2.210: 11: Bye Bye
54 54 Sep 18 02:53:24 bacuri sshd[28465]: Invalid user apple from
       222.221.2.210
```

Listing D.2: Log de acesso do IP 222

Referências Bibliográficas

[1] CertBR. **Estatísticas dos Incidentes Reportados ao CERT.br**. Disponível em: <http://www.cert.br/stats/incidentes/>. Acesso em: 02 jul. 2012. x, 2

[2] Paulo Licio de. NAKAMURA, Emilio Tissato; GEUS. ***Segurança de Rede em Ambientes Cooperativos***. Editora Novatec,488p, 4a. edição, 2007, 2007. x, 7, 8, 10, 75

[3] Fernando Augusto Pestana Júnior. **Proposta de Atualização Automática dos Sistemas de Detecção de Intrusão por meio de Web Services**. 2005. 166f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luís, MA, 2005. x, 12, 13, 16, 20

[4] Falkner de Area Leão Moraes. **Segurança e Confiabilidade em IDS Baseado em Agentes**. 2009. 115f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão, São Luis, MA, 2009. x, 4, 14, 15, 52

[5] Jorge Abílio Abinader e Rafael Dueire Lins. ***Web Services em Java***. Editora Brasport, 192p, 1a edição, 2006, 2006. x, 16, 17, 18, 19, 20, 21

[6] M. L. Carvalho SILVA. **Modelo de IDS Remoto Baseado na Tecnologia de Agentes, Web Services e MDA**. 2006. 142f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luís, MA, 2006. x, 19, 20, 47

[7] Julio Cesar Damasceno. **Introdução à Composição de Serviços Web**, 2012. Disponível em: <http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/minicurso/mc8.pdf>. Acesso em 20 de jan. 2012. x, 19, 20, 22, 27

[8] José Luís Vaz Ribeiro. **Orquestração e Composição de Serviços Web Usando BPEL**. 2008. 87f. Dissertação (Departamento. Eletrônica, Telecomunicações e Informática) - Universidade de Aveiro. Aveiro, Portugal, 2008. x, xii, 19, 21, 22, 23, 25, 38, 39

[9] Fabio Perez MARZULLO. ***SOA na prática***. Editora Novatec,390p, 1a edição,2009, 2009. x, 24, 25

[10] Henrique Shoiti Fugita. **MAPOS: Métodos de Análise e Projetos Orientados a Serviços**. 2009. 175f. Escola Politécnica da Universidade de São Paulo, 2009. x, xi, 24, 25, 26, 27, 29, 30, 32, 40, 41, 42

[11] Michael Bell. **Modelação Orientada ao Serviço - SOA Análise, Designer e Arquitetura de Serviços**. AltaBooks, 2008. x, xi, 28, 29, 30, 31, 32, 33, 34, 35, 40, 43, 44, 49, 57, 60

[12] Nicolai M. JOSUTTIS. ***SOA na Prática A arte da Modelagem de sistemas distribuídos***. Editora Oreill'y, 265p, 1a edição,2008, 2008. x, 24, 26, 34, 35, 36, 37

[13] James P. Anderson. **Computer Security Threat Monitoring and Surveillance**, February 1980. Disponível em:<http://csrc.nist.gov/publications/history/ande80.pdf>. Acesso em: 12 dez. 2011. 6

[14] Lauffer. **Introdução a Sistemas de Detecção de Intrusão**, Novembro 2008. Disponível em:<http://www.gta.ufrj.br/grad/03_1/sdi/sdi-1.htm>. 6

[15] Dorothy E. Denning. **An Intrusion-Detection Model**, 1987. Disponível em: <http://www.cs.georgetown.edu/ denning/infosec/ids-model.rtf>. Acesso em 20 de jun. de 2012. 6

[16] Lindonete Gonalves Siqueira. **Tolerância a Falhas para o NIDIA: um Sistema de Detecção de Intrusão Baseado em Agentes Inteligentes**. Master's thesis, Universidade Federal do Maranhão, 2006. 9, 14

[17] Cisco Catalyst 6500 Series Switches. **Catalyst Switched Port Analyzer (SPAN) Configuration Example**. Disponível em:<http://www.cisco.com/image/gif/paws/10570/41.pdf>. Acesso em 20 de jul. de 2012. 10

[18] Igor Vinícius Mussoi de Lima. **Uma abordagem Simplificada de Detecção de Intrusão baseada em Redes Neurais Artificiais**. 2005. 109f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal de Santa Catarina. Florianópolis, SC, MA, 2005. 12

[19] CIDF Group. **The Common Intrusion Detection Framework (CIDF)**, 10 2011. Disponível em:<http://gost.isi.edu/cidf/ >. Acesso em 20 de set. de 2012. 12, 14, 16

[20] LEMOS LIMA. Christiane F. **Agentes Inteligentes para Detecção de Intrusos em Redes de Computadores**. 2002. 176f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luis, MA, 2002. 14, 15, 16, 47, 50

[21] Aline Lopes Silva. **Modelo de IDS para Usuários de Dispositivos Móveis**. 2008. 117f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luis, MA, 2008. 15

[22] Rômulo Alves Dias. **Um Modelo de Atualização Automática do Mecanismo de Detecção de Ataques de Rede para Sistemas de Detecção de Intrusão**. 2003. 117f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão, 2003. 16

[23] Antonio. A PIRES OLIVEIRA. **SAMARA Sociedade de agentes para a monitorao de ataques e Respostas automatizadas**. 2005. 115f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luis, MA, 2005. 16

[24] Emanoel Costa Claudino Silva. **Gerenciamento e Integração das Bases de Dados de Sistemas de Detecção de Intrusões**. 2006. 151f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luis, MA, 2006. 16

[25] OLIVEIRA Emerson J.S. **Comunicação Segura e Confiável para Sistemas Multi-agentes Adaptando Especificações XML**. 2006. 114f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal do Maranhão. São Luís, MA, 2006. 16

[26] Paulo Henrique. ***Coreografia de serviço web-Uma Abordagem para integração de Serviços Web***. 2007. 76f. Dissertação (Mestrado. em Engenharia de Eletricidade) - Universidade Federal de Santa Catarina. Florianópolis, SC, 2007. 16, 17, 21

[27] Sergio Manuel Serra da CRUZ. **Serviço Web - Uma breve Introdução** (parte i), jan 2010. Disponível em:<http://www.nce.ufrj.br/conceito/artigos/2005/01p1-1.htm>. Acesso em: 22 nov. 2011. 16, 18

[28] Eric Jendrock. **The Java EE 6 Tutorial**, July 2011. Disponível em:<http://docs.oracle.com/javaee/6/tutorial/doc/javaeetutorial6.pdf>. Acesso em 12 de dez. 2011. 16

[29] Antonio Carlos Benedete Junior. **Roteiro para a definição de uma arquitetura SOA utilizando BPM**. Master's thesis, 2007. 68f. Dissertação (MBA em Tecnologia da Informação) - Escola Politécnica da Universidade de São Paulo. São Paulo, SP, 2007., 2007. 24

[30] Mark D. HANSEN. ***SOA Using Java Web Services***. Editora Prentice Hall,574p, 1a edição,2007, 2007. 25

[31] Thomas ERL. ***SOA Princípios de desing de serviços***. Editora Pearson Prentice Hall,320p, 1a edição,2009, 2009. 26, 40, 42

[32] Oracle. **Um revolução em agilidade: Integração de negócios através de uma arquitetura orientada a serviços**, 2008. Disponível em :<http://www.oracle.com/technetwork/pt/middleware/integracao-negocios-atraves-de-soa-432516-ptb.pdf/>. Acesso em 16 maio. 2012. 28, 37

[33] TIExames. **Fundamentos do BMP(Bussines Process Management**), 2012. Disponível: em:<http://www.tiexames.com.br/curso-OCEB-BPM-Expert-Fundamental.php?gclid=CM7dtvCIiK4CFQ2R7QodVXcu5g>. Acessado em 14 fev. de 2012. 34

[34] BPEL Designer Project, 2012. Disponível em: <http://www.eclipse.org/bpel/>. Acesso em 06 de jun. 2011. 37

[35] Apache ODE. **Apache ODE**, 2012. Disponível em: <http://ode.apache.org/getting-ode.html>. Acesso em 01 de jun. 2012. 37

[36] **Bonita Soft**, 2012. Disponível em:<http://www.bonitasoft.com/>. Acesso em 30 maio de 2012. 37

[37] SANS About the SANS Institute. **SANS About the SANS Institute**, 2012. Disponível em: <http://www.sans.org/about/sans.php>. Acesso em 10 de set. 2012. 74