



MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-15175-TDI/1292

**UMA METODOLOGIA PARA DETECÇÃO DE ATAQUES NO
TRÁFEGO DE REDES BASEADA EM REDES NEURAIS**

Lília de Sá Silva

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Antonio Montes Filho e José Demisio Simões da Silva, aprovada em 25 de maio de 2007.

INPE
São José dos Campos
2008

Publicado por:

esta página é responsabilidade do SID

Instituto Nacional de Pesquisas Espaciais (INPE)

Gabinete do Diretor – (GB)

Serviço de Informação e Documentação (SID)

Caixa Postal 515 – CEP 12.245-970

São José dos Campos – SP – Brasil

Tel.: (012) 3945-6911

Fax: (012) 3945-6919

E-mail: pubtc@sid.inpe.br

**Solicita-se intercâmbio
We ask for exchange**

Publicação Externa – É permitida sua reprodução para interessados.



MINISTÉRIO DA CIÊNCIA E TECNOLOGIA
INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

INPE-15175-TDI/1292

**UMA METODOLOGIA PARA DETECÇÃO DE ATAQUES NO
TRÁFEGO DE REDES BASEADA EM REDES NEURAIIS**

Lília de Sá Silva

Tese de Doutorado do Curso de Pós-Graduação em Computação Aplicada, orientada pelos Drs. Antonio Montes Filho e José Demisio Simões da Silva, aprovada em 25 de maio de 2007.

INPE
São José dos Campos
2008

681.3.019

Silva, L. S.

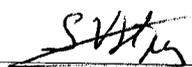
Uma metodologia para detecção de ataques no tráfego de redes baseada em redes neurais / Lília de Sá Silva. - São José dos Campos: INPE, 2007.

254 p. ; (INPE-15175-TDI/1292)

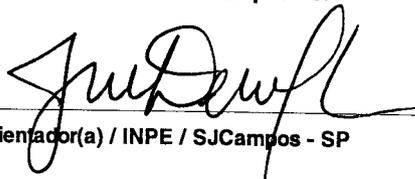
1. Detecção de intrusão. 2. Análise de tráfego de rede.
3. Detecção de intrusão por assinatura. 4. Detecção de intrusão por anomalia. I. Título.

Aprovado (a) pela Banca Examinadora
em cumprimento ao requisito exigido para
obtenção do Título de **Doutor(a)** em
Computação Aplicada

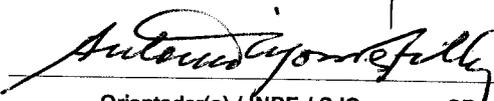
Dr. Stephan Stephany


Presidente / INPE / SJCampos - SP

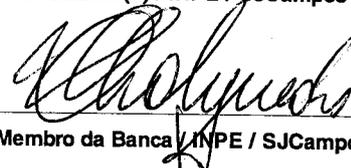
Dr. José Demisio Simões da Silva


Orientador(a) / INPE / SJCampos - SP

Dr. Antonio Montes Filho


Orientador(a) / INPE / SJCampos - SP

Dr. Ulisses Thadeu Vieira Guedes


Membro da Banca / INPE / SJCampos - SP

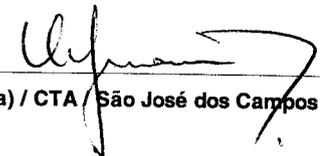
Dr. Adriano Mauro Cansian


Convidado(a) / UNESP/SJRP / São José do Rio Preto - SP

Dr. João Onofre Pereira Pinto


Convidado(a) / UFMS / Campo Grande - MS

Dr. Osvaldo Catsumi Imamura


Convidado(a) / CTA / São José dos Campos - SP

Aluno (a): **Lília de Sá Silva**

São José dos Campos, 25 de Maio de 2007

“Foi o tempo que perdeste com tua rosa que a fez tão importante para ti.”

ANTOINE DE SAINT-EXUPÉRY

Dedico este trabalho aos meus amados: Deus, pais, irmãos e amigos. Também aos prezados pesquisadores e profissionais das áreas de Segurança em Redes e Inteligência Artificial.

AGRADECIMENTOS

Aos prezados orientadores Dr. Antônio Montes e Dr. José Demísio Simões da Silva, agradeço pela confiança, condução dos trabalhos, correção de rumos, refinamentos e ajustes de idéias, enfim, por todo apoio recebido no desenvolvimento desta tese.

Aos membros da banca examinadora, agradeço pela aceitação e empenho na avaliação desta.

Aos colegas do INPE, professores do Curso de Computação Aplicada (CAP) e alunos da CAP, meus agradecimentos pelas experiências compartilhadas. Ao Luiz Gustavo C. Barbato, agradeço pelas informações nas disciplinas de Segurança de Redes; ao Carlos Henrique P. Chaves, pelas explicações sobre o sistema Recon; ao Luiz Otávio Duarte, pelo auxílio na criação dos primeiros *scripts* de monitoração de rede e ao André Luiz Grégio, pelo auxílio no uso do *tcpdump* e *Snort*.

À chefe da Divisão de Desenvolvimento de Sistemas de Solo do Instituto Nacional de Pesquisas Espaciais, Edenilse F. E. Orlandi, agradeço pela oportunidade e apoio. Aos colegas de trabalho da DSS, agradeço pelo incentivo e colaboração. Ao Leandro Toss Hoffman, pelo auxílio nos testes das redes neurais. Agradecimento ao Thiago Dias Mancilha, pelo apoio técnico nos trabalhos realizados no Laboratório de Redes da DSS.

Meus agradecimentos também à amiga Adriana C. Ferrari dos Santos, pelos esforços somados na implementação do sistema ANNIDA, e ao Pedro Inácio Hubscher, pelos esclarecimentos sobre técnicas estatísticas utilizadas neste trabalho.

Aos queridos pais, irmãos, amigos e familiares, agradeço pela atenção, carinho e amizade, pelo exemplo de dedicação ao trabalho e constante colaboração.

Ao bom Deus, que é a luz do meu caminho, agradeço por todas as graças recebidas.

RESUMO

A fim de precaver-se contra situações inesperadas e indesejadas e impedir a proliferação dos ataques continuamente lançados contra diferentes alvos na rede, são implantados mecanismos de proteção, tais como *firewalls*, antivírus, sistemas de autenticação, mecanismos de criptografia e sistemas de detecção de intrusão nos ambientes de rede por todo o mundo. Os sistemas de detecção de intrusão compõem uma parte essencial da infra-estrutura de segurança em camadas e tem por objetivo a análise de dados de auditoria de hosts ou dados do tráfego de rede em busca de eventos suspeitos ou ataques lançados contra redes ou sistemas. Diversas técnicas para reconhecimento de eventos de intrusão têm sido propostas e disponibilizadas em forma de ferramentas de domínio público ou soluções comerciais. Entretanto, observa-se a necessidade de uma metodologia de fácil aplicação que sirva de apoio aos analistas nas tarefas para detecção de ataques a redes. Portanto, esta tese propõe uma metodologia de apoio à detecção de ataques no tráfego de redes, baseada em redes neurais, provendo métodos, técnicas e ferramentas para modelagem e tratamento de dados, para geração de tráfego normal e anômalo para treinamento e testes de modelos de detecção e métodos para detecção de ataques no tráfego de rede baseados em redes neurais. Também são apresentadas informações sobre atualização de bases de assinaturas e de tráfego normal, bem como informações sobre análise de comportamento do tráfego. Os estudos de casos realizados comprovaram a factibilidade da metodologia proposta para detecção de ataques no tráfego HTTP, com base principal na aplicação de redes neurais para análise de dados de pacotes de rede.

A METHODOLOGY FOR ATTACK DETECTION IN THE NETWORK TRAFFIC BASED ON NEURAL NETWORKS

ABSTRACT

In order to be cautious against unexpected and undesired situations and to prevent the proliferation of attacks continuously launched to different targets in the network, protection mechanisms like firewalls, antivirus, authentication system, cryptography and intrusion detection systems are installed in network environments all over the world. Intrusion detection systems compose an essential part of the infrastructure of in-layer security and its objective is to analyze audit trails data of hosts or network traffic data in order to search suspected events or attacks against network or systems. Several techniques to recognize intrusion events have been proposed, from public domain tools to commercial solutions. However, a methodology of easy application to aid the analysts in the tasks for network attack detection is necessary. Thus, a neural network-based methodology to aid analysts in detecting attacks on the network traffic is proposed in this thesis. This methodology provides strategies, methods, techniques, and tools to model and treat data, to generate normal and anomalous traffic used for training and testing of detection models and methods for attack detection on the network traffic based on neural networks. Also, it provides information about signature and normal traffic databases updating, as well as information about computer network traffic behavior analysis. Studies of cases had disclosed the possibility of efficient use of the proposal methodology to detect attacks in computer network HTTP traffic, with emphasis in the application of neural networks to analyze network packet data.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

1	INTRODUÇÃO	29
1.1	Problemas Observados	37
1.2	Objetivo do Trabalho	39
1.3	Contribuições	39
2	ANOMALIAS NO TRÁFEGO DE REDE	41
2.1	Definição de Anomalias no Tráfego de Rede	41
2.2	Classificação de Anomalias na Rede	43
2.2.1	Anomalias de Operação da Rede	43
2.2.2	Anomalias ‘Flash Crowd’	43
2.2.3	Anomalias de Medição	44
2.2.4	Ataques	44
2.3	Classificação de Ataques a Redes	44
2.3.1	Ataques de Negação de Serviço	47
2.3.2	Ataques de Varredura	48
2.3.3	Ataques de Penetração	49
2.4	Ameaças e Vulnerabilidades em Aplicações Web	50
3	SISTEMAS DE DETECÇÃO DE INTRUSÃO	57
3.1	Classificação de IDS	57
3.2	Considerações sobre Avaliação de IDS	62
4	MINERAÇÃO DE DADOS DO TRÁFEGO DE REDE	67
4.1	Abordagens de <i>Data Mining</i>	68
4.1.1	Clustering	68
4.1.2	Classificação	69
4.1.3	Detecção de <i>Outliers</i>	70
4.1.4	Regras de Associação	70
4.2	Técnicas de <i>Data Mining</i> em Detecção de Intrusão	72
4.2.1	Rede Neural SOM	74
4.2.2	Rede Neural MLP	77
4.2.3	Rede Neural Hamming Net	80

4.2.4	Rede Neural LVQ	81
4.2.5	Técnicas de Detecção de Outliers	83
4.2.5.1	Abordagem LOF	85
4.2.5.2	Abordagem <i>LSC-MINE</i>	88
4.3	Estado da Arte em Detecção de Ataques no Tráfego de Rede	91
4.3.1	EMERALD	92
4.3.2	Exemplo de Aplicação de Lógica Fuzzy	95
4.3.3	FIDS	97
4.3.4	MINDS	100
4.3.5	Exemplo de Aplicação de Rede MLP	103
4.3.6	ACME!	104
4.3.7	Exemplo de Aplicação da Rede SOM	107
4.3.8	HIDE	108
4.3.9	Exemplo de Aplicação da Teoria <i>Rough Set</i>	111
4.3.10	SNORT	115
4.3.11	BRO	116
4.4	Visualização de Anomalias no Tráfego de Rede	119
5	UMA METODOLOGIA PARA DETECÇÃO DE ATAQUES NO TRÁFEGO DE REDES	121
5.1	Coleta de Dados do Tráfego	121
5.2	Reconstrução de Sessões do Tráfego	125
5.3	Seleção de Atributos do Tráfego	128
5.4	Armazenamento de Sessões do Tráfego	132
5.5	Preparação de dados para treinamento das redes neurais	134
5.5.1	Geração de Tráfego Normal	134
5.5.2	Geração de Tráfego de Ataque	134
5.6	Técnicas para Detecção de Ataques e Modelagem de Dados	136
5.6.1	Normalização dos Dados	138
5.6.2	Detecção de Ataques e Modelagem de Dados	139
5.6.3	Atualização da Base de Assinaturas e Tráfego Normal	140
5.6.4	Avaliação de Resultados	141
5.7	Considerações sobre Análise do Comportamento do Tráfego	142
6	APLICAÇÕES DESENVOLVIDAS	143
6.1	A Aplicação ANNIDA	143
6.1.1	Histórico	143
6.1.2	Características da Aplicação	144
6.1.3	Recursos de Desenvolvimento	148
6.1.4	Evolução no Desenvolvimento da Aplicação	148
6.1.5	Módulos da Aplicação	151
6.1.5.1	Módulo de Modelagem de Assinaturas	152
6.1.5.2	Módulo de Extração de <i>Payloads</i> e Dados	153
6.1.5.3	Módulo de Classificação Hamming Net	153
6.1.5.4	Módulo de Classificação LVQ	155

6.1.5.5	Mecanismo de Alerta	156
6.1.6	Considerações	157
6.2	A Aplicação RGCOM	159
6.3	O Sistema ADTRAF	164
6.3.1	Visão Geral do Sistema	164
6.3.2	Recursos de Desenvolvimento e Teste	165
6.3.3	Arquitetura do Sistema	167
6.3.4	Módulo de Captura de Pacotes	168
6.3.5	Processo de Reconstrução de Sessões	169
6.3.6	Processo de Extração de Atributos	169
6.3.7	Módulo de Detecção de Assinaturas	171
6.3.8	Módulo de Detecção de Anomalias	172
6.3.9	Módulo de Alerta	172
6.3.10	Módulo de Apoio à Análise Visual do Tráfego	172
6.3.11	Interface Gráfica	173
6.3.11.1	Janela Principal	173
6.3.11.2	Seleção de Métodos	173
6.3.11.3	Classificação Manual do Tráfego	174
6.3.11.4	Detecção de Assinaturas	177
6.3.11.5	Detecção de Anomalias	179
6.3.11.6	Manipulação da Base de Dados	180
7	ESTUDO DE CASOS	183
7.1	Geração de Tráfego Normal e Tráfego Anômalo	186
7.2	Descrição dos Ataques	189
7.3	Conjuntos de Dados Utilizados	192
7.4	Descrição dos Estudos de Casos	194
7.4.1	Análise Visual do Comportamento do Tráfego	194
7.4.2	Detecção de Assinaturas	201
7.4.2.1	Análise de dados do Conteúdo dos Pacotes	201
7.4.2.2	Análise de dados do Cabeçalho dos Pacotes	205
7.4.3	Detecção de Anomalias no Tráfego de Rede	212
7.4.3.1	Detecção de Anomalias utilizando as redes LVQ e MLP	214
7.4.3.2	Detecção de Anomalias utilizando as Técnicas LOF e LSC	219
8	CONCLUSÕES	223
	REFERÊNCIAS BIBLIOGRÁFICAS	229
	BIBLIOGRAFIA COMPLEMENTAR	245

LISTA DE FIGURAS

3.1 - Classificação de IDS	58
4.1 - Exemplo de Representação Geométrica dos Neurônios da SOM	75
4.2 - RNA de 4 camadas e fórmulas de ajuste de pesos. Fonte: Adaptado de MARTINS (2003) 77	
4.3 - Arquitetura da rede Hamming Net.....	80
4.4 - Arquitetura da Rede LVQ Simplificada.....	82
4.5 - Conjunto de dados bi-dimensional simples	87
4.6 - A arquitetura do monitor EMERALD.....	94
4.7 - Framework do FIDS	97
4.8 - Arquitetura do MINDS	100
4.9 - Regra de Associação descrevendo atividade de varredura na porta 139..	103
4.10 - Estrutura Geral do ACME. Fonte: CANSIAN (1997)	104
4.11 - Estrutura dos módulos do ACME! Fonte: CANSIAN (1997).....	105
4.12 - Entradas e Saídas do Hide.....	109
4.13 - Um agente de detecção de intrusão (IDA)	110
4.14 - arquitetura do IDS	112
4.15 - Arquitetura Básica do Sistema Snort. Fonte: Adaptado de CASWELL et al (2003)	115
4.16 - Arquitetura do sistema Bro	118
5.1 - Seqüência de atividades para coletar o tráfego de ataque. Fonte: Adaptado de FAGUNDES (2002).....	135
5.2 - Operação de junção dos detectores de assinaturas e anomalias. Fonte: Adaptado de TAPIADOR et. al (2004a).....	137
6.1 - Visão Geral da Aplicação ANNIDA.....	144
6.2 - Exemplo de regra Snort.....	145
6.3 - Exemplo de Dados do Payload em Pacotes de Rede TCP/IP	146
6.4 - Visão geral da Entrada, Exemplos e Saída do Classificador	147
6.5 - Detecção de Múltiplas <i>Strings</i> de uma Assinatura de Ataque.....	148
6.6 - Módulos do Sistema ANNIDA	151
6.7 - Arquitetura da rede neural Hamming Net.....	154
6.8 - Arquitetura da rede neural LVQ	155
6.9 - Saída da Aplicação com Informações sobre o Ataque.....	156
6.10 - Módulos da aplicação RGCOM	160
6.11 - Tela de Seleção de Atributos e Intervalo de Sessões.....	160

6.12 - Ilustração de um gráfico gerado pelo RGCOM	162
6.13 - Arquitetura Modular do Sistema ADTRAF.....	167
6.14 - Esquema de Captura de Pacotes	168
6.15 - Esquema da Reconstrução de Sessões do Tráfego de Rede	169
6.16 - Arquitetura da rede LVQ no sistema ADTRAF.....	171
6.17 - Janela Principal para Seleção do Tráfego a ser Analisado.....	173
6.18 - Janela para seleção de técnicas de análise do tráfego.....	174
6.19 - Janela para Escolha do Detector de Anomalias.....	174
6.20 - Janela para seleção de horário e tráfego a exibir	175
6.21 - Seleção de região do tráfego para classificação de sessões normais.....	176
6.22 - Seleção de Tabela para gerar gráfico	177
6.23 - Visualização Gráfica de Tráfego Classificado	177
6.24 - Janela com as opções de classificação do tráfego	178
6.25 - Seleção de opções para classificação LVQ	178
6.26 - Janela de alerta de ataques	179
6.27 - Janela para a Seleção do Método de Detecção de Anomalias.....	179
6.28 - Janela com Opções de Treinamento ou Classificação MLP	180
6.29 - Janela com Opções de Treinamento ou <i>Clustering</i> SOM	180
6.30 - Janela para seleção da localização da base de dados.....	181
6.31 - Janela para seleção de Tabelas a excluir do banco	181
7.1 - Ambiente para Captura de Pacotes da RedeLab.....	187
7.2 - Ambiente para Captura de Pacotes da RedeBeta	188
7.3 - Janela de 10 minutos de sessões do tráfego.....	195
7.4 - Janela de 5 horas de sessões do tráfego em horário de pico.....	198
7.5 - Janela de 8 horas de sessões do tráfego de rede	199
7.6 - Destaque de Sessões Anômalas fora da Região de Normalidade	200
7.7 - Detecção de Sessões Isoladas de Ataque U2R e R2L	207
7.8 - Detecção de Múltiplas Sessões de Ataque U2R e R2L	207
7.9 - Detecção de Sessões Mistas de Ataque DoS e Probe	208
7.10 - Detecção de Sessões Mistas de Ataque U2R e R2L.....	208
7.11 - Arquitetura da LVQ e MLP para detecção de anomalias	214

LISTA DE TABELAS

3.1 - Métricas padrão para avaliações de intrusões de conexão-única. Fonte: Adaptada de Lazarec et al. (2003)	64
4.1 - Descrição dos Atributos.....	113
4.2 - Exemplo de Tabela de Decisão.....	114
4.3 - Exemplo de Regras.....	115
5.1 - Atributos típicos utilizados em detecção de intrusão.....	129
5.2 - Exemplo de consulta a registros de sessões do tráfego na base	133
6.1 - Mudanças e Benefícios	149
6.2 - Atributos utilizados na composição de cada sessão de rede.....	170
7.1 - Fases da Metodologia Proposta.....	184
7.2 - Ataques do tipo R2L e U2R.....	190
7.3 - Ataques do tipo DoS e Probe	191
7.4 - Conjuntos de dados utilizados nos estudos de casos.....	193
7.5 - Resultados de Classificação e Desempenho das Redes Neurais	202
7.6 - Resultados dos testes de busca de strings de ataque.....	204
7.7 - Resultados dos testes de busca de múltiplas strings de ataque.....	204
7.8 - Reconhecimento de ataques com ADTRAF, Dragon e Snort	209
7.9 - Conjuntos de dados utilizados nos testes LVQ e MLP.....	216
7.10 - Resultados de Detecção de Anomalias utilizando LVQ e MLP	217
7.11 - Descrição dos conjuntos de teste.....	218
7.12 - Exemplos de Conjuntos de Dados Analisados.....	220
7.13 - Clustering do tráfego com LOF e variação de Minpts	221
7.14 - Clustering do tráfego com LSC e variação de Minpts	221
7.15 - Valores médios de Tempo de Processamento e Taxas.....	222

LISTA DE SIGLAS E ABREVIATURAS

ACME	- Advanced Counter Measures Environment
ADTRAF	- Attack Detection on the network TRAFfic system
ANNIDA	- Artificial Neural Network for Intrusion Detection Application
ASP.NET	- Ambiente de programação Web da Microsoft para construção de páginas Web dinâmicas, aplicações Web e serviços Web XML
CERT.br	- Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil
CSRF	- Cross Site Request Forgery
DDoS	- Distributed Denial of Service
DFA	- Deterministic Finite-state Automaton
DIDS	- Distributive Intrusion Detection System
DNS	- Domain Name Service
DoS	- Denial of Service
DSS	- Divisão de Desenvolvimento de Sistemas de Segmento Solo
EMERALD	- Event Monitoring Enabling Responses to Anômalous Live Disturbances
FIDS	- Fuzzy Intrusion Detection System
FPGA	- Field Programmable Gate Arrays
FTP	- File Transfer Protocol
GrIDS	- Graph-based Intrusion Detection System
HAMMING NET	- Rede neural de rápida classificação
HIDE	- Hierarchical Network Intrusion Detection System
HIDS	- Host Intrusion Detection System
HMM	- Hidden Markov Model
HTML	- HyperText Markup Language
HTTP	- HyperText Transfer Protocol
ICMP	- Internet Control Message Protocol
IDS	- Intrusion Detection System
IIS	- Internet Information Services

INPE	- Instituto Nacional de Pesquisas Espaciais
IP	- Internet Protocol
ISP	- Internet Service Provider
KDDCup99	- Knowledge Data Discovery Competition in 1999
LabRedes	- Laboratório de Pesquisa e Desenvolvimento em Redes da DSS
LOF	- Local Outlier Factor
LSC	- Local Sparsability Coefficient
LVQ	- Learning Vector Quantization
MINDS	- Minnesota Intrusion Detection System
MinPTS	- Número máximo de pontos vizinhos de um determinado ponto analisado pelo algoritmo de detecção de outliers
MLP	- MultiLayer Perceptron
NFR	- Network Flight Recorder intrusion detection system
NIDS	- Network Intrusion Detection System
NSM	- Network Security Monitor intrusion detection system
OWASP	- Open Web Application Security Project
PHP	- Linguagem de programação projetada para a produção de páginas Web dinâmicas
R2L	- Remote to Local
RBF	- Radial Basis Function
RECON	- Sistema de Reconstrução de Sessões TCP/IP
RedeBETA	- Ambiente de rede do Prédio Beta do INPE
RedeLAB	- Ambiente de rede do Laboratório de Redes da DSS
RGCOM	- Sistema de Representação Gráfica do COMportamento do tráfego de rede
RIDAS	- Rough Set Intelligent Data Analysis System
RNA	- Rede Neural Artificial
RST	- TCP flag (RESET)
SETBETA	- Conjunto de dados do tráfego da RedeBETA
SETLAB	- Conjunto de dados do tráfego da RedeLAB
SGBD	- Sistema de Gerenciamento de Banco de Dados
SMB	- Samba Server

SNMP	- Simple Network Management Protocol
SNNS	- Stuttgart Neural Network Simulator
SOM	- Self-Organizing Map
SQL	- Structured Query Language
SSL	- Secure Socket Layer communication protocol
SVM	- Support Vector Machine
TCP	- Transfer Control Protocol
TCP/IP	- Transfer Control Protocol / Internet Protocol
U2R	- User to Root
UDP	- User Datagram Protocol
URG	- TCP flag (URGENT)
URL	- Uniform Resource Locator
VLAN	- Virtual LAN (Local Area Network)
XSS	- Cross Site Scripting

1 INTRODUÇÃO

O uso da rede Internet expandiu-se rapidamente possibilitando a comunicação de dados, voz e imagem através de aplicações, tais como as de compartilhamento de dados e de comércio eletrônico. Disponibilizar determinadas informações pessoais de clientes através das aplicações Web (incluindo dados financeiros, endereço e identidade, dados estratégicos de empresas, incluindo planos futuros, aplicações financeiras, fornecedores, moldes ou listas de componentes utilizados na fabricação de produtos, que dão o diferencial para o negócio da empresa, entre outros), requer a implantação de políticas e mecanismos de segurança de sistema e da rede de informação, em alto nível, para manter a integridade da aplicação, a privacidade de seus usuários, a confidencialidade de seus dados e o funcionamento correto das estações servidoras nas quais as aplicações Web são executadas.

A segurança de aplicações Web não trata apenas de proteger o número de cartão de crédito do usuário através de criptografia, mas deve levar em consideração a forma como a aplicação obtém o número do cartão de crédito, o armazena em banco de dados e, posteriormente, o recupera desse banco (MARTINEZ, 2006).

O tema segurança de redes ganhou maior importância com o surgimento de fraudes de diversas naturezas, em muitas empresas, que provocaram grandes prejuízos financeiros. No Brasil, ainda existem inibidores do crescimento da comercialização e da adoção de software e outras práticas de segurança. O descaso por falta de políticas específicas para tratar do assunto na grande maioria das empresas faz com que os gastos nesta área sejam vistos como custos, em vez de investimento (MARTINEZ, 2006). Em alguns segmentos existe ainda interesse maior pela compra de aplicativos para áreas consideradas críticas para a empresa, tais como de gestão empresarial, cadeia

de suprimentos, inteligência de negócios e outras soluções, e a aquisição de software de segurança encontra-se em segundo plano.

Aos poucos este cenário está mudando. As empresas que possuem seus negócios apoiados por tecnologias têm buscado fazer investimentos em segurança; muitas são fornecedoras multinacionais, entre estas: Accenture, Cisco, HP, IBM, PriceWaterHouseCoopers e empresas nacionais, tais como CPM, DBA Engenharia, Politec, Stefanini, têm colocado em pauta o tema segurança de aplicações e redes. No mercado brasileiro, fornecedoras multinacionais de software de segurança, tais como CA, Check Point, ISS, McAfee, Sonic Wall, Symantec e Trend Micro, têm operado, juntamente com empresas nacionais, tais como Aker, Módulo e Scua (MARTINEZ, 2006).

Estatísticas de incidentes de segurança notificados ao CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil (CERT.BR, 2006) em 2006, comprovam a proliferação de ataques a redes, incluindo *worms*, varreduras de portas, ataques de negação de serviços e incidentes em aplicações e estações servidoras Web.

Estas informações aceleram as pesquisas e o mercado de software de segurança, tais como os firewalls, as aplicações de criptografia, as ferramentas de software conhecidas como 3A (Autenticação, Autorização e Administração de usuários) e os sistemas de detecção de intrusão (IDS – *Intrusion Detection System*).

Os *firewalls* e os métodos de autenticação não são suficientes para assegurar a segurança dos sistemas em rede. Os *firewalls* não podem evitar todos os ataques provenientes de fora da rede e são inúteis para defender os sistemas contra ataques internos. Sistemas de autenticação, por sua vez, não podem evitar que usuários legítimos realizem operações nocivas em um sistema computacional. Portanto, é necessário que uma camada complementar de proteção à rede, relatando e/ou prevenindo contra ações maliciosas e acessos

não autorizados, seja implementada através de tecnologias de detecção de intrusão.

As tecnologias de detecção de intrusão são utilizadas para viabilizar a observação de conjuntos de dados de rede, em busca de ameaças potenciais ou conhecidas no tráfego de rede (*Network Intrusion Detection System* - NIDS) e/ou nos dados de auditoria registrados nos hosts (*Host Intrusion Detection System* - HIDS).

Duas principais categorias de IDSs englobam as técnicas de detecção: detecção baseada em assinaturas e baseada em anomalias. O método de detecção por assinatura (ou por abuso) consiste na modelagem de padrões de ataques conhecidos (assinaturas) e na pesquisa destes padrões em um conjunto de dados. Caso ocorra o casamento de dados com um padrão de ataque, um alerta é gerado. A grande vantagem destas técnicas é seu alto grau de precisão na detecção de ataques conhecidos e suas variações. Uma limitação destas é a incapacidade de detectar novos ataques cujas instâncias ainda não tenham sido observadas.

No processo de detecção por anomalia o comportamento normal de processos de usuários, sistemas ou rede é modelado e desvios deste modelo representam anomalias ou suspeitas de ataque. Estas técnicas são capazes de identificar novos tipos de ataques, a partir de observação de desvios do comportamento normal do tráfego de rede e de uso de recursos por parte dos usuários e sistemas. Se estas técnicas apresentam grande poder de detectar novos ataques, possuem uma restrição por gerar alta taxa de alarmes falsos (falsos positivos), principalmente por reconhecer comportamentos do sistema ainda não previstos (ainda que legítimos) como anomalias e por sinalizá-los como potenciais intrusões.

Os sistemas de detecção de intrusão baseados em assinaturas utilizam assinaturas catalogadas, em geral, em bases extensas e requerem muito trabalho e conhecimento de especialistas para serem estabelecidas e

revisadas. O conhecimento sobre novos ataques é obtido através da análise pos-mortem de dados do tráfego de rede.

O uso crescente de ambientes de redes corporativas integradas na rede mundial, e a disponibilidade de ferramentas gratuitas para ataque às redes, têm implicado diretamente no surgimento de novas formas de ataques não disponíveis nos catálogos. Estas novas formas de ataques podem ser complexas ou simples, em função da experiência e dos objetivos dos atacantes. Idealmente, os sistemas de detecção de intrusão deveriam ser capazes de detectar novos ataques observando o comportamento do tráfego da rede, estabelecendo os padrões de comportamento normal do tráfego e sendo capazes de detectar as variações que ocorrem neste tráfego normal observado, ou seja, as anomalias no tráfego.

A busca por anomalias no tráfego de redes é uma tarefa extremamente complicada em função da necessidade de analisar muitas sessões do tráfego, procurando considerar todas as particularidades de uso da rede observada. Isso pode significar um número muito grande de sessões, com diferentes tipos de serviços que podem incluir transferências de grandes arquivos, uso excessivo de determinadas portas de serviço, entre outros. Além disso, a detecção por anomalia, conceitualmente caracterizada por basear-se em desvios do comportamento normal do ambiente de rede, como anteriormente mencionado, pode levar a um grande número de falsos positivos, na medida que padrões de serviços, comportamento de hosts ou usuários mudem no ambiente de rede considerado. A detecção de ataque por anomalia requer, portanto, a atualização constante da base de conhecimento de tráfego normal para evitar falsos positivos.

Existem várias abordagens de detecção por anomalia incluindo métodos determinísticos, baseados em conhecimento a priori do comportamento do tráfego do ambiente de rede; métodos estatísticos, que analisam o comportamento das variáveis observáveis no tráfego; métodos de detecção de *outliers*, que buscam por objetos ou pontos estranhos em relação aos demais

objetos existentes nos conjuntos de dados com base em informações sobre a vizinhança dos pontos e métodos de mineração de dados, que procuram extrair as relações entre as diversas variáveis e estabelecem padrões de tráfego normal e de tráfego anômalo. A mineração de dados, por sua vez, pode usar diferentes técnicas de manipulação de dados, supervisionadas ou não, para extrair conhecimento útil que permita uma melhoria nos processos de detecção.

A maioria das abordagens para concepção e desenvolvimento de IDSs reside na construção de sistemas especialistas baseados em regras e filtros ajustados sob medida, segundo a observação dos administradores de redes sobre o tráfego da rede sob sua responsabilidade. Estes IDSs caracterizam-se pelo número de regras que pode ser muito alto, dado o dinamismo do fluxo de informação nas redes de computadores em nível mundial. Além disso, as regras devem ser revisadas ao catalogar novas formas de ataques. Como os processos de reconhecimento das assinaturas trabalham com a correspondência exata das condições das regras existentes, variações nos padrões emergentes nos ambientes de redes, mesmo que pequenas, não podem ser reconhecidas.

Neste sentido, outras abordagens têm sido exploradas procurando tornar os sistemas mais flexíveis e computacionalmente mais velozes, podendo, em primeira instância, serem capazes de correlacionar padrões de ataques novos com aqueles existentes nos catálogos de ataques disponíveis na literatura. Para isso muitos pesquisadores têm buscado a concepção e desenvolvimento de IDSs baseados em redes neurais artificiais (RNA) e outras técnicas de inteligência artificial ou estatísticas, supervisionadas ou não-supervisionadas, na tentativa de superar as limitações dos métodos baseados em regras e melhorar a qualidade da análise dos dados (CANSIAN, 1997; BONIFACIO et al., 1998; CABRERA et al., 2000; ERTOZ et al., 2003a; RAPAKA et al., 2003; MUKKAMALA et al., 2002a; MUKKAMALA et al., 2002b; MUKKAMALA et al., 2003a; MUKKAMALA et al., 2003b; MUKKAMALA et al., 2003c; SHAH et al.,

2003; ZHANG et al., 2003; SILVA et al., 2004). Além disso, as abordagens baseadas em redes neurais buscam conceber métodos para o estabelecimento automático de padrões de assinatura de ataques ou de condições do tráfego na rede.

Os modelos baseados em redes neurais exploram várias hipóteses simultaneamente utilizando elementos computacionais denominados neurônios, interconectados através de pesos sinápticos (FAUSSET,1994; HAYKIN, 2001; MARTINS, 2003). O processamento paralelo destas estruturas pode reduzir o tempo de processamento dos pacotes em busca de anomalias e de padrões de ataques, permitindo a geração de respostas rápidas aos incidentes. Além disso, existem diversas aplicações em diferentes áreas em que os modelos de redes neurais são implementados em hardware programável (FPGA – *Field Programmable Gate Arrays*), sendo esta uma característica importante para a abordagem do problema de detecção de intrusão em tempo real.

Apesar da literatura reportar os sucessos dos vários métodos de detecção de ataques em situações conhecidas de tráfego, deve-se refletir sobre os resultados obtidos, quanto à aplicação de alguns desses métodos em situações de tráfego diferentes daquelas estudadas nos trabalhos relatados. Isto porque cada ambiente de rede tem suas peculiaridades e especificidades dependentes das aplicações e do número e diversidade de serviços disponíveis, máquinas e usuários em rede, que podem mudar com a dinâmica da demanda.

Diversas técnicas para reconhecimento de eventos de intrusão têm sido propostas e disponibilizadas em forma de ferramentas de domínio público ou soluções comerciais. Entretanto, observa-se a necessidade de uma metodologia de fácil aplicação para auxílio aos analistas na tarefa de detecção de ataques a redes e que permita a fácil adaptação das técnicas de detecção na análise de dados e situações de tráfego diferentes.

Portanto, nesta tese é proposta uma metodologia de apoio à detecção de ataques no tráfego de redes, englobando um conjunto de fases, métodos, técnicas e ferramentas.

As principais contribuições desta tese são os métodos de detecção de assinaturas e anomalias no tráfego de rede desenvolvidos, baseados em redes neurais.

No contexto deste trabalho, assinaturas são especificamente padrões de ataques conhecidos que podem ser encontrados nos dados de rede, e anomalias são quaisquer tipos de variações ou desvios encontrados nos dados de rede que destoam das características normais do tráfego, incluindo as assinaturas de ataques, descobertas não necessariamente através da apresentação de padrões normais de comportamento do tráfego para o treinamento de modelos conforme o conceito de detecção por anomalia.

O método de detecção de assinaturas no tráfego de rede utilizado neste trabalho consiste na análise *off-line* do conteúdo (*payloads*) dos pacotes de rede envolvendo o uso de redes neurais de rápida classificação do tipo Hamming Net (FAUSSET, 1994), que utiliza a distância de Hamming entre padrões binários ou bipolares como medida de similaridade, e LVQ (*Learning Vector Quantization* – Aprendizagem por Quantização Vetorial) (FAUSSET, 1994) que, através de um processo de competição e cálculo de distância Euclidiana entre padrões de entrada e exemplares (assinaturas), identifica o tipo de ataque detectado. Ambas as redes utilizam a informação de *payloads* proveniente das sessões de tráfego para buscar padrões de ataques catalogados no SNORT (SOURCEFIRE, 2007). O diferencial destas duas abordagens reside na modelagem e pré-processamento dos dados anteriormente ao processo de detecção, ponto este extremamente importante quando se considera a diversidade de tipos de informação que trafega por um ambiente de rede diariamente.

Outro método utilizado neste trabalho para a detecção de assinaturas no tráfego compreende a análise *off-line* de dados do cabeçalho dos pacotes de rede através da aplicação da rede neural LVQ, visto que em trabalho de pesquisa recente (SILVA et al., 2006) apresentou melhor resultado de classificação e melhor modelagem dos dados que a Hamming Net. Neste contexto, os registros de sessões de tráfego de rede obtidos do cabeçalho dos pacotes são utilizados como atributos de entrada e exemplares para a rede LVQ.

O método de detecção de anomalias no tráfego de rede desenvolvido consiste no uso combinado das redes neurais artificiais não supervisionadas, mapas auto-organizáveis (SOM – *Self Organizing Maps*), e supervisionadas Perceptron de Múltiplas Camadas (MLP – *Multi-Layer Perceptron*). Estas técnicas foram utilizadas com base em recentes estudos sobre a eficiência destes métodos no campo de detecção de intrusão (ABDI et al, 2003; GOTO et al., 2003; KAYACIK et al., 2003; LABIB et al., 2002; VIGNA et al., 1999; ZANERO, 2005a; ZANERO, 2005b; ZHANG et al.,2001; ZHANG et al., 2003). Outros métodos que foram utilizados para teste são os métodos estatísticos LOF (*Local Outlier Factor* – Fator de Diferença Local) (BREUNIG et al., 2000) e LSC (*Local Sparsability Coefficient* – Coeficiente de Espalhamento ou Dispersão Local) (AGYEMANG et al., 2004). O primeiro método, LOF, foi utilizado levando-se em consideração os resultados satisfatórios produzidos na detecção de anomalias pelo sistema MINDS (ERTOZ et al., 2003a) e o segundo, LSC, é uma técnica da mesma classe que a LOF (detecção de *outliers*) comprovadamente mais eficiente que a LOF quanto ao tempo de processamento de conjuntos de dados (AGYEMANG et al., 2004).

Resumindo, foram realizados estudos de aplicação das redes neurais SOM, LVQ e MLP e técnicas de detecção de outliers LOF e LSC para análise de dados de cabeçalho dos pacotes de rede em busca de anomalias no tráfego HTTP. As redes neurais Hamming Net e LVQ foram utilizadas neste trabalho para a identificação de assinaturas no conteúdo dos pacotes de rede. Com

base em resultados satisfatórios na detecção de assinaturas nos dados de *payload* dos pacotes, a rede LVQ também foi utilizada para detecção de assinaturas a partir de dados de cabeçalho dos pacotes de rede.

A metodologia foi aplicada em ambiente de rede do LabRedes-DSS (Laboratório de Pesquisa e Desenvolvimento em Redes da DSS), possibilitando a realização de estudos de casos a partir de dados simulados e dados reais de tráfego HTTP.

Os estudos de casos realizados mostraram a factibilidade da metodologia desenvolvida para detecção de ataques no tráfego HTTP em modo *off-line*, baseando-se na aplicação de redes neurais para a detecção de assinaturas e anomalias.

1.1 Problemas Observados

Através de pesquisas recentes, relacionadas à detecção de ataques a redes de computadores, foram observados os seguintes pontos:

- Diferentes características dificultam a análise automática, imediata e precisa, com mínima intervenção humana, de eventos de rede em busca de sinais de atividades ilegítimas, incluindo: o grande volume de dados que trafegam diariamente pela rede; a variedade de sistemas operacionais e hardware dos computadores interconectados em rede; a qualidade diversificada dos dados provenientes de pedidos e respostas a diferentes serviços de rede; a natureza e origem diversificadas das ameaças aos sistemas; e a existência de falhas e de vulnerabilidades nos sistemas operacionais e aplicações em rede;
- Quanto aos métodos de detecção de intrusão utilizados no desenvolvimento de IDSs, existem limitações técnicas. Os IDSs baseados em assinaturas são incapazes de detectar novos ataques ou variações de ataques conhecidos, apresentando alta taxa de falsos negativos, ou seja, não alertam sobre ataques ocorridos cujas

assinaturas não estejam modeladas em sua base. Por outro lado, os IDSs baseados em anomalia, embora eficientes na detecção de novos ataques, apresentam alta taxa de falsos positivos (alarmes falsos), alertando sobre ataques que na realidade não ocorreram. Qualquer novo comportamento na rede, como adição de hosts e usuários, e geração de diferentes eventos por falha na configuração de dispositivos de rede, é visto por estes sistemas como um desvio do padrão normal ou uma anomalia, mesmo não sendo um ataque;

- O crescente número de ataques requer uma atualização contínua da base de conhecimento dos sistemas de detecção baseados em assinaturas. Além disto, existe um número desconhecido de vulnerabilidades descobertas, mas não reveladas que não se encontram disponíveis para análise e inclusão na base de conhecimento. Muitos ataques são polimorfos e os atacantes exploram esta característica para evadir os sistemas detectores. A solução óbvia seria utilizar uma abordagem de detecção por anomalia, modelando o que é normal ao invés do que é anômalo. Isto é semelhante às concepções preliminares do que deveria ser um IDS (DENNING, 1987). Entretanto, enquanto varios sistemas de detecção por anomalia baseados em host tem sido propostos e implementados, tanto na literatura quanto na prática, a detecção por anomalia baseada em rede é ainda um campo aberto de pesquisa;
- Diversas técnicas para reconhecimento de eventos de intrusão têm sido propostas, desde soluções de domínio público a soluções comerciais. Entretanto, observa-se a necessidade de concepção de uma metodologia para apoio à detecção de ataques no tráfego de redes com base em estratégias, métodos, procedimentos, técnicas e ferramentas para modelagem e tratamento dos dados e preparação de ambiente de testes, descrição de métodos e soluções para detecção de ataques às redes e informações úteis para análise do comportamento do tráfego.

1.2 Objetivo do Trabalho

Este trabalho tem por objetivo desenvolver uma metodologia de apoio à detecção de ataques no tráfego de redes baseada em redes neurais.

São reportados estudos de casos que mostram a factibilidade da metodologia desenvolvida para detecção de ataques no tráfego HTTP. Dentre estes estudos, encontra-se a aplicação de redes neurais SOM, LVQ e Backpropagation e técnicas de detecção de outliers LOF e LSC para análise de comportamento do tráfego a partir dos dados de cabeçalho dos pacotes de rede em busca de anomalias no tráfego. Para a detecção de assinaturas no *payload* dos pacotes são utilizadas as redes neurais Hamming Net e LVQ.

1.3 Contribuições

As principais contribuições desta tese são:

- A aplicação de redes neurais de rápida classificação Hamming Net e LVQ para detecção de assinaturas no *payload* dos pacotes de rede;
- A aplicação da rede neural LVQ para detecção de padrões de comportamento do tráfego a partir de dados do cabeçalho (atributos de sessões) dos pacotes de rede;
- O uso combinado das redes neurais LVQ e MLP para detecção de anomalias no tráfego de rede, com *pré-clustering* de tráfego normal através da LVQ; e classificação do tráfego completo, incluindo tráfego normal clusterizado e tráfego anômalo, através da rede MLP.

Dentre as contribuições secundárias desta tese, destacam-se:

- Desenvolvimento da aplicação *Artificial Neural Network for Intrusion Detection Application* (ANNIDA), baseada no uso de redes neurais para detecção de assinaturas de ataques através da análise do conteúdo dos pacotes de redes de computadores;

- Desenvolvimento da ferramenta (Representação Gráfica do Comportamento do Tráfego de Rede (RGCOM) para auxílio na análise do tráfego de rede;
- Desenvolvimento do sistema *Attack Detection on the Network Traffic* (ADTRAF), baseada no uso de redes neurais e técnicas estatísticas para detecção de assinaturas e anomalias na rede, através de análise dos dados do cabeçalho dos pacotes de rede, com recurso simples de visualização gráfica do tráfego.

2 ANOMALIAS NO TRÁFEGO DE REDE

Neste capítulo serão apresentadas uma classificação para anomalias freqüentemente observadas no tráfego de rede, uma descrição das ameaças e principais vulnerabilidades das aplicações Web.

2.1 Definição de Anomalias no Tráfego de Rede

Anomalias no tráfego de rede são ações diferentes observadas no comportamento normal do tráfego previamente observado, que podem ser indicativos de ataques, abuso (ou mau uso) na rede, eventos de falha na rede, problemas de infra-estrutura na coleta de dados, entre outros. Portanto, nem toda anomalia na rede é um ataque, mas uma informação suspeita que deve ser analisada.

Sejam as anomalias de rede maliciosas ou não-intencionais, é importante analisá-las por dois motivos (LAKHINA et al., 2004):

- Anomalias podem criar congestão na rede e estressar a utilização de recursos de um roteador;
- Algumas anomalias podem não necessariamente gerar impacto na rede, mas causar grande impacto para um cliente ou um usuário final.

Quando diagnosticando anomalias na rede observa-se que suas formas e causas podem variar consideravelmente: de abusos à rede (ataques DoS - *Denial of Service*) a falhas de equipamentos (configurações incorretas do roteador, por exemplo).

Os operadores de rede devem detectar as anomalias tão logo estas ocorrem e classificá-las de modo a seleccionar a resposta apropriada. O principal desafio em detectar e classificar anomalias é que as anomalias podem se estender a um vasto domínio de eventos: de abuso a rede (por exemplo, ataques DoS, scans e worms) a falhas de equipamentos (por exemplo, interrupções), a comportamento não comum de usuários (por exemplo, alterações bruscas sob

demanda, *flash crowds*, alto volume de fluxos) e mesmo devido a novos eventos previamente não conhecidos. Um sistema de diagnóstico de anomalias geral deveria, portanto, ser capaz de detectar uma variedade de anomalias com estruturas diversas, distinguindo-as entre tipos diferentes de anomalias e anomalias de mesmo grupo.

Apesar da ampla literatura sobre caracterização do tráfego, anomalias no tráfego de rede permanecem pouco entendidas por vários motivos (LAKHINA et al., 2004). Primeiro, identificar anomalias requer uma sofisticada infraestrutura de monitoração. Segundo Lakhina (2004), muitos ISPs (*Internet Service Provider*) somente coletam medições simples do tráfego, ou seja, volumes de tráfego médios, usando o protocolo SNMP. Existem ISPs que coletam contagens de fluxos, mas o processamento dos dados coletados é uma tarefa exigente. Outro motivo que dificulta o entendimento de anomalias no tráfego é que os ISPs não possuem ferramentas rápidas o suficiente para processar medições de modo a detectar anomalias em tempo real. Estes são, em geral, informados dos principais eventos (worms ou ataques DoS) após a ocorrência destes, não são capazes de detectá-los quando estão em progresso. Um terceiro motivo é a natureza do tráfego de rede, que é multidimensional e ruidoso, o que torna difícil extrair informação significativa sobre anomalias seja qual for o tipo de estatística do tráfego.

Anomalias no tráfego de rede, incluindo ataques e outros eventos suspeitos, podem ser detectadas pelos IDSs a partir de informações do host, como acesso a arquivos, seqüência e quantidade de *system calls* (chamadas ao sistema), dados de processos de usuários e de sistema e dados coletados do tráfego de rede, que são informações obtidas dos cabeçalhos e *payload* dos pacotes de rede. Seqüestro de conexões TCP, varredura de portas, DNS *spoofing*, negação de serviço, dentre outros, são alguns exemplos de ataques dificilmente detectados por analisadores baseados em host. Para detectá-los são necessárias ferramentas que capturam e analisam os pacotes de rede, permitindo a busca de ataques direcionados a componentes de rede e,

inclusive, a determinadas máquinas, através da análise dos dados transportados nos pacotes.

2.2 Classificação de Anomalias na Rede

Através de análise visual, Barford (2001, 2002) classificou as anomalias no tráfego de rede em quatro principais categorias: anomalias de operação da rede, anomalias '*flash crowd*', anomalias de medição e ataques. Uma descrição de cada uma destas categorias é apresentada a seguir.

2.2.1 Anomalias de Operação da Rede

Incluem eventos de falhas na rede, tais como interrupção de funcionamento de dispositivos na rede, ou eventos que geram diferenças significativas no comportamento da rede, tais como a adição de novos equipamentos, configuração inadequada temporária de dispositivos e definição de limites de taxas. Por exemplo, um roteador pára de publicar uma das redes classe B de um site para os roteadores de borda do site. Anomalias desta categoria são distinguidas visualmente através de declives, alterações súbitas na taxa de bits seguidas de taxas de bits que são estáveis, em um nível diferente, em determinado período de tempo.

2.2.2 Anomalias '*Flash Crowd*'

Anomalias nesta categoria são, em geral, devido a uma nova versão do software ou interesse interno por acesso a um site Web devido a algum tipo de publicidade nacional. Distingue-se um comportamento *flash crowd* por um rápido crescimento nos fluxos de tráfego de um determinado tipo (por exemplo, fluxos FTP) ou para um destino bem conhecido com uma redução gradual com o tempo. Por exemplo, um acréscimo no tráfego de saída de uma estação servidora FTP de um site que fornece uma nova versão de uma distribuição Linux.

2.2.3 Anomalias de Medição

Uma anomalia que determina-se não ser devido a problemas de infra-estrutura da rede ou por uso abusivo da rede. Por exemplo, problemas com a infra-estrutura de coleta de dados, que incluem perda de dados do fluxo devido à sobrecarga do roteador ou transporte Netflow UDP não confiável para o coletor.

2.2.4 Ataques

Estas anomalias são conhecidas como abusos à rede e podem ser identificadas principalmente através de fluxos de dados do tráfego. Exemplos desta categoria são ataques de inundação de DoS (*Denial of Service*) e varreduras de portas.

Estes tipos de abusos são, em geral, observados com certa frequência na rede. Um exemplo é um fluxo muito grande de pacotes TCP de 40 bytes provenientes de um host que teve sua segurança comprometida e está sendo remotamente controlado por atacantes.

Ataques são distintos das anomalias *'flash crowd'* e das anomalias de operação da rede ou de medição por não poderem ser detectados através de medições de taxas de bits ou de pacotes. Entretanto, as medições de contagem de fluxos (conexões ou sessões) de rede claramente indicam estas atividades abusivas a partir de informações de pares de portas e endereços lógicos de origem e destino das conexões.

2.3 Classificação de Ataques a Redes

Ataques em redes de computadores compreendem um “conjunto de ações ilícitas que tentam comprometer a integridade, confidencialidade, ou disponibilidade de recursos na rede”, independente do sucesso ou não destas ações. Regras de privacidade podem ser quebradas devido a um ataque, comprometendo a confidencialidade da informação. Informações podem ser alteradas, modificando a integridade dos dados. E a infra-estrutura de rede

pode tornar-se indisponível e não confiável, afetando a disponibilidade do recurso.

Violações às propriedades de segurança de dados e sistemas computacionais em rede podem ser descritas como:

- Impedimento de acesso a recursos e sistemas em rede por um usuário, humano ou máquina, autorizado (violação de disponibilidade);
- Acesso aos dados sem autorização (implícita ou explícita) do proprietário da informação (violação de confidencialidade).
- Alteração ilegal do estado do sistema ou de dados residindo ou trafegando no sistema (violação de integridade).

A grande maioria dos ataques bem sucedidos ocorre devido a vulnerabilidades ou falhas potenciais existentes nos sistemas em rede, que podem estar relacionadas com uma configuração incorreta do sistema ou falha no desenvolvimento do software. Estas vulnerabilidades e erros nos sistemas podem ser explorados por meio de diversos tipos de ataques, desde tentativas simples de negação de serviços a ataques mais sofisticados, utilizando recursos distribuídos.

A maioria dos ataques é executada através de *scripts* os quais automatizam os processos de tentativa de conexões em varias portas, enviando pacotes com *payloads* fabricados ou com códigos para consultas a bases de dados de sistemas, entre outros.

Certos atacantes realizam a falsificação de endereço (*spoofing*) de origem dos pacotes. Deste modo, o atacante consegue esconder sua origem ou até mesmo se passar por outra máquina conseguindo um acesso privilegiado.

Conhecer os principais ataques, as vulnerabilidades mais exploradas e os objetivos dos atacantes, é fundamental para a criação de contramedidas.

Existem diferentes classificações para ataques. Estes podem ser classificados quanto à sua origem, ao seu alvo e quanto aos seus objetivos.

Quanto à origem, os ataques podem ser externos ou internos:

- Ataques externos: são lançados de fora da rede por um atacante que tenta acessar a rede para obter informações, divertir-se ou tornar indisponíveis determinados serviços da rede alvo;
- Ataques internos: são aqueles provenientes de usuários internos à rede que abusam de seus direitos e privilégios para realizar atividades não autorizadas e para obter acesso não autorizado a recursos e sistemas em rede.

Uma classificação de ataques com base em alvo pode ser descrita como:

- Ataques à rede: visam impedir os usuários de utilizar recursos de rede ou torna os serviços de rede indisponíveis. Podem também monitorar o tráfego de rede para analisá-lo e coletar informação conveniente;
- Ataques a sistemas: o propósito destes ataques é comprometer o sistema, bem como modificar ou remover arquivos críticos, tais como arquivos de senhas e de configuração do sistema. Neste contexto, encontram-se os ataques que visam à modificação de páginas web para depreciação ou ridicularização da imagem de empresas.

Quanto a seus objetivos, os ataques mais freqüentes a redes de computadores podem ser classificados como DoS, Probing, U2R (User to Root) e R2L (Remote to Local). Estes ataques podem ser lançados no local, com a conta de usuário autorizado do sistema alvo, ou lançados remotamente, através de uma conexão de rede, sem qualquer conta de usuário ou acesso privilegiado ao sistema alvo, mas utilizando apenas o acesso público concedido pelo sistema alvo.

Uma estratégia típica utilizada pelos atacantes é disparar um ataque utilizando uma conta de usuário sem privilégio para ganhar acesso inicial ao sistema. Uma vez obtido o acesso ao sistema, o atacante utiliza conta de usuário autorizado para tentar elevar seus privilégios e obter controle completo do alvo.

A classificação de ataques apresentada a seguir tem por base informações dos artigos (ERTOZ, 2003a; MUKKAMALA, 2002a; MUKKAMALA, 2003b).

2.3.1 . Ataques de Negação de Serviço

Os ataques DoS tentam reduzir o desempenho ou interromper o funcionamento de sistemas e serviços de rede. O principal objetivo deste tipo de ataque é tornar inoperante um serviço ou interromper a atividade de uma estação servidora ou equipamento ligado em rede. Exemplos de ataques desta categoria são: Apache2, Back, Land, Mail bomb, SYN Flood, Ping of death, Process table, Smurf, Syslogd, Teardrop e Udpstorm.

Algumas estratégias utilizadas nos ataques DoS, segundo (TAROUCO et al., 2003), incluem:

- Inundar uma rede visando impedir que usuários legítimos façam uso dela;
- Impedir ou romper a conexão entre duas máquinas visando impedir o acesso a um serviço;
- Impedir o acesso de um determinado serviço ou site;
- Impedir ou negar um serviço a um sistema ou pessoa específicos.

Existem três tipos principais de ataques de negação de serviço:

- Exploração de falhas: exploram vulnerabilidades no software do sistema alvo causando falhas em seu processamento ou extinguindo seus recursos. Um exemplo deste tipo de exploração pode ser constatado no ataque '*WinNuke*'.

- *Flooding*: enviam a um sistema mais informação do que ele é capaz de manipular. Mesmo que a capacidade de processamento do sistema não seja totalmente tomada, o atacante pode ser capaz de monopolizar a conexão de rede do alvo, bloqueando assim qualquer tipo de uso deste recurso.
- Ataques de negação de serviço distribuído (*Distributed DoS - DDoS*): são inundações de ataques DoS onde os atacantes utilizam vários computadores para lançar o ataque e mais rapidamente sobrecarregar um determinado sistema alvo. Neste tipo de ataque é realizada uma sobrecarga ou inundação de pacotes contra um determinado serviço, host ou rede, gerando muitas vezes uma quantidade de dados global maior que a rede ou host pode suportar, tornando a rede ou serviços instáveis e conseqüentemente prejudicando o seu desempenho. Estes ataques possuem uma estrutura previamente montada, onde diversas máquinas lançam um ataque sobre uma vítima. Esses ataques fazem uso de várias “máquinas zumbis” para que o número de requisições de conexão ao servidor seja muito maior. São ataques mais eficientes e complexos, mais difíceis de se detectar.

Segundo estatísticas do CERT/BR (CERT.BR, 2006), vários ataques DoS e DDoS são registrados diariamente, e envolvem principalmente novos vermes e ferramentas para DDoS. Alguns desses vermes incluem comandos e estrutura de controle que permite ao intruso dinamicamente modificar o comportamento do verme após ele infectar a vítima. Em alguns casos esse controle é realizado sem que o atacante precise saber quem são os sistemas que foram infectados.

2.3.2 Ataques de Varredura

Ataques desta natureza realizam varredura de uma rede ou de sistemas-alvo, através do envio de diferentes tipos de pacotes de rede, em busca de vulnerabilidades e informações de interesse para planejamento de ataques. As respostas recebidas destes sistemas são utilizadas pelo atacante para

aprender sobre as características da rede e dos sistemas, incluindo a topologia da rede, os hosts ativos e suas respectivas configurações de software, incluindo sistema operacional, software do servidor e versões de aplicativos. Ataques de sondagem são denominados ataques “Probing” e não penetram ou comprometem os sistemas. Estes ataques possuem várias denominações, dependendo das atividades que executam: network ou port scanners, port ou network mappers ou vulnerability scanners. Alguns exemplos de ataques desta categoria são: Ipsweep, Mscan, Nmap, Saint, Satan.

2.3.3 Ataques de Penetração

Os ataques de penetração, conhecidos como ataques R2L e U2R, realizam aquisição ou alteração não autorizada dos privilégios, recursos ou dados do sistema, violando as propriedades de integridade e controle dos recursos e dados. Com estes ataques, pode-se ganhar controle de um sistema ao explorar uma variedade de falhas de software.

Um ataque R2L (Remote to Local) ocorre quando um usuário remotamente realiza um acesso não autorizado a uma máquina e consegue privilégios de usuário local. Neste tipo de ataque, são enviados pacotes para uma máquina na rede na qual o atacante não tem conta e, em seguida, são exploradas algumas vulnerabilidades desta máquina que permitem a obtenção de acesso local como se fosse um usuário daquela máquina. Exemplos deste tipo de ataque incluem ataques de dicionários, Ftp_write, Guest, Imap, Named, Phf, Sendmail, Xlock, Xsnoop e Named .

Um ataque U2R (*User to Root*) ocorre quando um atacante inicia a exploração do host com uma conta de usuário normal do sistema e consegue explorar vulnerabilidades deste para ganhar acesso de root ao sistema. Exemplos destes ataques incluem buffer overflows, Eject, Ffbconfig, Fdformat, LoadModule, Perfl, Ps e Xterm.

2.4 Ameaças e Vulnerabilidades em Aplicações Web

Como o foco deste trabalho é a detecção de anomalias no tráfego HTTP de rede, nesta seção são descritas as ameaças e vulnerabilidades mais frequentes em aplicações Web. Os ataques utilizados para avaliação dos detectores de anomalias explorados neste trabalho, bem como as aplicações Web que estes ataques comprometem serão apresentados no capítulo 7, seção 7.1.

Por estarem disponíveis para acesso na rede mundial de computadores, as aplicações Web caracterizam-se como o principal alvo de diversas tentativas de ataques.

Aplicações Web podem ser arquivos HTML estáticos ou páginas Web complexas, dinâmicas e baseadas em banco de dados. Ao se referir às aplicações Web deve-se ter em mente as seguintes plataformas: servidor web, servidor de aplicação e banco de dados.

O servidor web fornece as páginas Web para o navegador do usuário, exemplos mais comuns são IIS e Apache. Servidor de aplicação é um componente que manipula, interpreta e apresenta dados ao usuário. Este pode ser parte do servidor Web, por exemplo, PHP no Apache ou ASP.NET no IIS, ou pode ser uma estação servidora fisicamente separada, como o mecanismo de servlet Tomcat (SHEMA, 2003). No banco de dados são armazenados os dados necessários para a aplicação. Na maioria das vezes, o servidor de aplicação atua como intermediário entre o usuário e o banco de dados, formatando os dados para armazenamento correto, por exemplo. É importante ressaltar que as plataformas Web (IIS, Oracle, Apache, MySQL e PHP, citando apenas algumas) possuem suas próprias vulnerabilidades.

Além de vulnerabilidades existentes nas plataformas operacionais e banco de dados sobre os quais as aplicações Web são executadas, uma outra categoria de ataques visa vulnerabilidades da própria aplicação. Dentre estas, enquadram-se os erros de programação da páginas Web, expondo detalhes do

cartão de crédito de usuários, permitindo a execução de consultas a bancos de dados arbitrárias ou possibilitando acesso de linha de comando remoto ao servidor (SHEMA, 2003).

Os ataques à aplicação, tais como injeção de SQL ou seqüestro de sessão, são mais difíceis de serem automatizados, mas as vulnerabilidades mais comuns podem ser codificadas de modo que algumas linhas de código maliciosas possam verificar a presença da vulnerabilidade, como no caso das verificações básicas de validação de entrada.

Ataques de *buffer overflow* são usados para corromper a pilha de execução de uma aplicação web. Enviando entradas maliciosas a uma aplicação web, um atacante pode fazer com que a aplicação execute códigos arbitrários, permitindo que este obtenha domínio sobre a máquina. Falhas de *buffer overflow* podem apresentar-se no servidor de aplicação ou no servidor web que forneçam aspectos dinâmicos e estáticos do site, ou em aplicações web.

Enfim, as ameaças a aplicações Web são sustentadas por um conjunto diversificado de ferramentas e informações disponíveis.

O projeto *Open Web Application Security Project (OWAS) Top Ten* (OWASP, 2006) apresenta uma lista com as vulnerabilidades mais comuns encontradas nas aplicações Web, incluindo a descrição de cada vulnerabilidade, a fim de educar desenvolvedores, projetistas, arquitetos e organizações sobre as consequências dos ataques, a forma de detectar as vulnerabilidades das aplicações e como protegê-las.

As dez principais vulnerabilidades em aplicações Web listadas em 2007 pelo OWASP são resumidamente apresentadas a seguir:

a) Cross Site Scripting (XSS)

Cross site scripting, mais conhecida como XSS, é o mais nocivo problema de segurança em aplicações web. Falhas XSS ocorrem quando uma aplicação captura os dados fornecidos por um usuário e os

envia a um *web browser* sem primeiro validar ou codificar aquele conteúdo. XSS permite aos atacantes executar *scripts* no browser da vítima para seqüestrar sessões de usuários, desFigurar web sites, inserir conteúdo hostil, conduzir ataques de *phishing* e controlar o *browser* do usuário utilizando *scripts malware*. O script malicioso é geralmente codificado em JavaScript, mas qualquer linguagem de script suportada pelo *browser* da vítima é um alvo potencial para este ataque.

b) Injeção de Falhas

Injeção de falhas, particularmente injeção de SQL (*SQL Injection*), são comuns em aplicações web. A injeção ocorre quando dados fornecidos pelo usuário são enviados a um interpretador como parte de um comando ou consulta ao banco. Os dados hostis do atacante trapaceiam o interpretador levando-o a executar comandos que não entende ou a alterar dados. Injeção de falhas permitem aos atacantes criar, ler, atualizar ou remover dados arbitrários disponíveis na aplicação. No pior cenário, estas falhas permitem ao atacante comprometer completamente a aplicação.

c) Execução de Arquivo Malicioso

Vulnerabilidades de execução de arquivos maliciosos são encontradas em muitas aplicações. Desenvolvedores em geral utilizarão ou concatenarão entrada potencialmente hostil com funções de *streams* ou de arquivos, ou impropriamente confiarão nos arquivos de entrada. Em muitas plataformas, as estruturas permitem o uso de referências a objetos externos, tais como URLs ou referências a sistema de arquivos. Quando os dados são insuficientemente verificados, isto pode fazer com que conteúdo hostil ou remoto arbitrário seja incluído, processado ou invocado pelo servidor web, permitindo a execução de código remoto, instalação remota de root kits e comprometimento completo dos sistemas.

Este ataque é particularmente predominante no PHP, por default, PHP 4.0.4 e posteriores e 5.x são vulneráveis a inclusão de arquivos remotos. Muito cuidado deve ser tomado na implementação de qualquer função de arquivo ou de *stream* para garantir que a entrada fornecida pelos usuários não influencie no nome dos arquivos.

d) Referência a Objeto Direto Inseguro

Uma referência a um objeto direto ocorre quando um desenvolvedor expõe uma referência a um objeto de implementação interno, tal como um arquivo, um diretório, um registro do banco de dados, ou chave, bem como uma URL ou parâmetro de formulário. A menos que uma verificação de controle de acesso esteja disponível, um atacante pode manipular aquelas referências para acessar outros objetos sem ter autorização para isto.

e) *Cross Site Request Forgery* (CSRF)

Cross site request forgery (falsificação de solicitação) não é um ataque novo, mas é simples e devastador. Este ataque força o *browser* de uma vítima logada no sistema a enviar uma solicitação a uma aplicação web vulnerável, a qual então força o browser da vítima a realizar uma ação hostil em benefício do atacante. Esta vulnerabilidade é muito comum, como qualquer aplicação web que autoriza requisições baseadas somente em credenciais que são automaticamente submetidas pelo browser vulnerável. Infelizmente, hoje em dia, a maioria das aplicações web utiliza somente credenciais automaticamente submetidas, tais como sessões de cookies, credenciais de autenticação básica, endereços IP de origem, certificados SSL, credenciais do domínio Windows, entre outras.

f) Manipulação Imprópria de Erros e Vazamento de Informação

Aplicações podem não intencionalmente vazarem informação sobre a sua configuração, seu funcionamento interno ou violar a privacidade através de uma variedade de problemas de aplicação. Aplicações podem também vazarem informação sobre seu estado interno através de quanto tempo levam para processar certas operações ou através de diferentes respostas a diferentes entradas, tais como exibir o mesmo texto de erro com diferentes números de erros. As aplicações web em geral apresentam esta vulnerabilidade através de mensagens de erros de depuração ou mensagens detalhadas.

As aplicações em geral geram mensagens de erro e as exibem aos usuários. Muitas vezes, as mensagens de erro são muito úteis para os atacantes, visto que revelam detalhes ou informações úteis para a exploração de uma vulnerabilidade. Há vários exemplos disto: manipulação de erros detalhada, induzindo um erro a exibir grande quantidade de informação, tais como traços de pilha, comandos SQL não bem sucedidos ou outras informações de depuração; funções que produzem diferentes resultados baseados em diferentes entradas. Por exemplo, o fornecimento do mesmo *username*, mas diferentes *passwords*, para uma função de *login*, não deveria produzir o mesmo texto para os usuários e senhas ruins. Entretanto, muitos sistemas produzem diferentes códigos de erro.

g) Quebra de Autenticação e Gerenciamento de Sessão

Autenticação e gerenciamento de sessão são operações críticas para a segurança de aplicações web. Incluem-se nesta classificação as falhas ao proteger credenciais e tokens de sessão. Tokens podem ser cookies, valores de formulário HTML, parâmetros de URL, ID de sessão ou qualquer outro valor passado entre aplicação e o navegador. Falhas nesta área podem conduzir a seqüestro de contas administrativas ou de

usuários, chaves, enfraquecimento do controle de contas e de autorizações e provocar violações de privacidade.

h) Armazenamento Criptográfico Inseguro

A proteção de dados sensíveis através de criptografia tem se tornado uma parte essencial da maioria das aplicações web. As aplicações web raramente utilizam, de modo adequado, funções de criptografia para proteger dados e credenciais de usuários. Aplicações que utilizam criptografia geralmente possuem criptografia mal projetada, com cifras inadequadas ou produzem erros graves ao usar cifras fortes. Estas falhas podem levar a revelação de dados sensíveis e a violações de conformidade. Os atacantes utilizam dados fracamente protegidos para furta-mentos de identidades e realizar outros crimes, tais como fraude de cartão de crédito.

As falhas criptográficas mais comuns são: não criptografia de dados sensíveis; uso continuado de algoritmos comprovadamente fracos (MD5, SHA-1, RC3, RC4, etc...); codificação difícil de chaves e armazenamento de chaves em repositórios desprotegidos.

i) Comunicações Inseguras

As aplicações geralmente falham ao criptografar o tráfego de rede quando é necessário proteger as comunicações sensíveis. Criptografia (geralmente SSL) deve ser usada para todas as conexões autenticadas, especialmente para as páginas web acessíveis pela Internet. De outra forma, a aplicação deixará exposta uma autenticação ou token de sessão. Além disto, criptografia deveria ser utilizada quando dados sensíveis, tais como cartão de crédito ou informação de exames médicos, forem transmitidos.

j) Falhas de Restrição de Acesso a URL

Em geral, a única proteção para áreas sensíveis de uma aplicação é não apresentar os links para as páginas ou URLs a usuários não autorizados. Entretanto, os atacantes podem ser capazes de encontrar e acessar estas páginas, invocar funções e visualizar dados. Segurança por obscuridade não é suficiente para proteger funções sensíveis e dados em uma aplicação. Verificações de controle de acesso devem ser realizadas antes que uma requisição para uma função sensível seja concedida, garantindo que o usuário é autorizado a acessar aquela função.

O método primário de ataque a este tipo de vulnerabilidade é chamado "*forced browsing*", que atua utilizando técnicas de força bruta e adivinhação de links em busca de páginas desprotegidas.

3 SISTEMAS DE DETECÇÃO DE INTRUSÃO

Detecção de intrusão a redes de computadores é o processo de monitoração de eventos em sistemas computacionais em rede ou no tráfego de rede e a verificação destes em busca de traços de intrusões, ataques ou mau uso, enfim, de ações que violam a política de segurança da organização, de modo que os requisitos de disponibilidade, confidencialidade e integridade de sistemas de informação críticos em rede sejam mantidos.

Sistemas de detecção de intrusão (IDS – Intrusion Detection System) são ferramentas de software ou *appliance* utilizadas em conjunto com outros mecanismos de segurança tais como *firewalls*, antivírus e mecanismos de criptografia para reforçar a segurança de um ambiente de rede, relatando eventos suspeitos ou impedindo que ações maliciosas tenham êxito e se propaguem pela rede.

Neste capítulo será apresentada uma classificação de sistemas de detecção de intrusão, descrevendo os métodos e dados que podem ser utilizados na construção destes e uma descrição de possível localização do sensor de IDS na rede. Uma apresentação de dados e atributos selecionados para o desenvolvimento de IDSs será provida neste capítulo. Comentários sobre avaliação de IDS serão apresentados e, finalmente, uma ilustração de ferramentas que permitem a visualização do tráfego de rede a fim de facilitar a detecção de anomalias será abordada.

3.1 Classificação de IDS

Os sistemas de detecção de intrusão podem ser classificados segundo o método de detecção utilizado, a arquitetura do sistema, a frequência de uso e o comportamento deste após a detecção de eventos suspeitos ou intrusivos, conforme ilustrado na Figura 3.1.

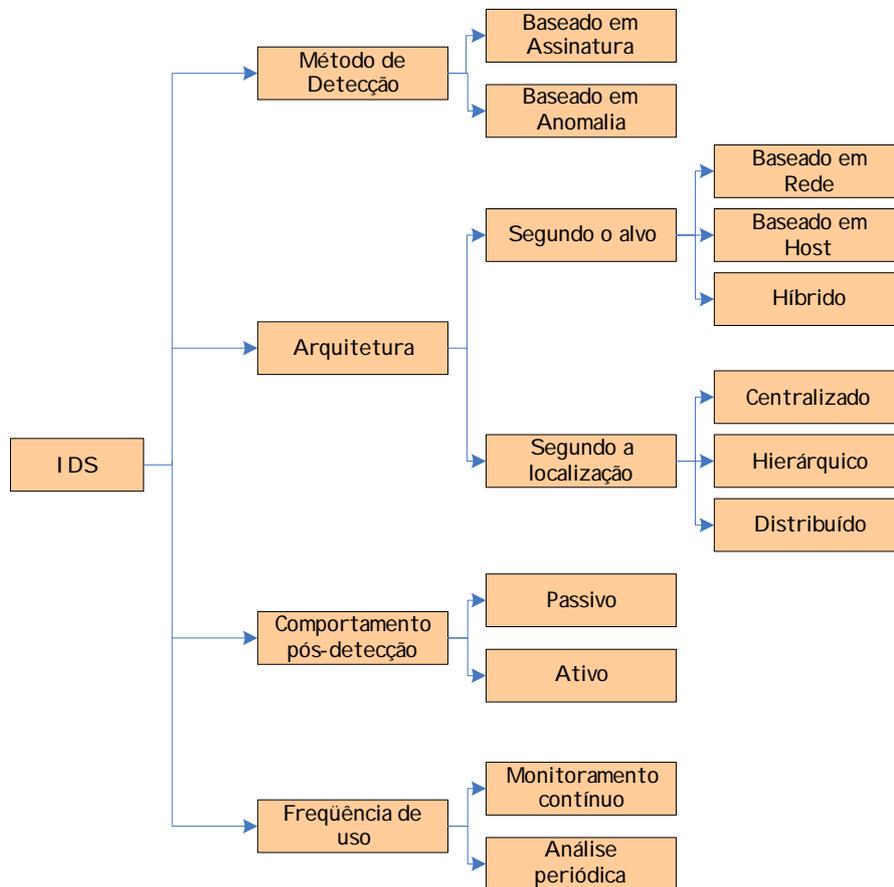


Figura 3.1 - Classificação de IDS

Quanto à frequência de uso, o IDS pode estar continuamente analisando os eventos de rede (análise em tempo real) ou ser configurado para análise periódica dos eventos de rede (análise off-line ou pos-mortem).

Um IDS pode ter sido projetado para realizar, por exemplo, um bloqueio de conexões provenientes de uma origem quando da identificação de determinado tipo de pacote. Este tipo de IDS que executa esta e outras medidas reativas após a detecção de intrusos é denominado ativo. IDSs passivos são aqueles que examinam as informações da rede ou sistemas e alertam sobre os ataques e ameaças emergentes, não reagindo aos ataques.

Segundo a localização, o IDS pode ter arquitetura centralizada, hierárquica ou distribuída. Um IDS de arquitetura centralizada possui seus módulos de detecção instalados em um único host e apresenta a vantagem de facilidade no desenvolvimento, instalação e configuração. O IDS de arquitetura distribuída possui seus módulos de detecção distribuídos em várias máquinas pela rede, que se comunicam de modo cooperativo, através de troca de mensagens, para garantir uma redundância inerente. Grande vantagem desta arquitetura, além de maior robustez, é a facilidade de crescimento modular, possibilitando agregar novos mecanismos ao sistema de acordo com a necessidade, a possível distribuição de tarefas, retirando de um único ponto o custo e a responsabilidade de todo o processamento, e a maior abrangência de detecção, com módulos espalhados pelos mais diferentes pontos do sistema.

Uma arquitetura hierárquica para IDS consiste na distribuição parcial dos componentes do IDS de modo que os módulos de detecção distribuídos subordinados se comuniquem segundo a coordenação de gerentes centralizados, ou seja, os módulos obedecem a uma estrutura hierárquica, com a interação entre os módulos do sistema sendo regida por fortes relações de subordinação. Esta abordagem surgiu para resolver alguns problemas referentes à implementação de sistemas completamente distribuídos. Algumas vantagens próprias dos sistemas distribuídos continuam existindo enquanto a solução de problemas, como a detecção de falhas nos módulos do sistema, é facilitada através da estrutura hierárquica. Em contrapartida, aumentam as chances de ataques ao sistema, principalmente pela criação de pontos únicos de falha, representados pelos módulos mais altos da cadeia de hierarquia. Se um desses módulos falhar, todo o sistema pode tornar-se indisponível, contrapondo-se à principal motivação no uso de sistemas distribuídos.

Um IDS pode ser classificado pelo tipo de alvo a ser monitorado, como baseado em rede, baseado em host ou híbrido. Uma das vantagens de um IDS em relação aos métodos tradicionais de análise de incidentes é a possibilidade de correlacionar diferentes tipos de dados. Coletar informações

locais como os *logs* e o estado dos processos em execução, ou capturar na rede pacotes destinados à determinada porta, são exemplos de ações distintas com um mesmo fim: gerar dados que representem indícios de uma intrusão. Estas atividades de coleta de dados, no entanto, sejam elas feitas no host ou na rede, determinam o tipo do IDS, baseado em host, em rede ou híbrido.

IDSs baseados em host (*Host Intrusion Detection System* - HIDS) utilizam dados coletados na própria máquina, arquivos de log ou registros de auditoria para monitorar os sistemas, permitindo a determinação exata de quais usuários e processos estão realizando operações maliciosas no sistema, o que garante boa precisão na detecção. Este tipo de IDS é capaz de monitorar acessos a sistemas e alterações em arquivos de sistemas críticos, modificações nos privilégios dos usuários, processos, programas que estão sendo executados, uso da CPU e memória, entre outros eventos. Examinam estas informações em busca de padrões de ataques ou de desvios do comportamento normal de hosts ou usuários. Estes podem acessar diretamente recursos do sistema, como discos, memória, processos, estado do SO, etc, informações de contabilização do SO, *logs* - sejam do SO ou de algum aplicativo -, e trilhas de auditoria.

IDSs baseados em rede (*Network Intrusion Detection System* - NIDS) realizam a monitoração do sistema através da captura e análise de cabeçalhos e conteúdo de pacotes de rede, os quais podem ser comparados com padrões de ataques conhecidos ou assinaturas previamente armazenadas em regras, arquivos ou bancos de dados, ou com padrões normais do tráfego, para verificação de algum desvio do comportamento normal da rede. As fontes de informação usadas por um IDS de rede variam desde dados de gerenciamento, obtidos através de agentes SNMP, por exemplo, até pacotes de rede carregando protocolos de mais alto nível (HHTTP, SMTP, SMB, etc). A análise de todas essas informações oferece bons subsídios aos sistemas de detecção de intrusão, agregando precisão e desempenho às suas tarefas. Grande parte das ferramentas atuais de detecção de intrusão explora o melhor das

arquiteturas baseadas em host e rede, adotando soluções híbridas. Em busca de equilíbrio entre desempenho, simplicidade, abrangência e robustez, algumas implementações recentes provêm coleta de diferentes dados de hosts e do tráfego de rede, e interagem mecanismos centralizados com distribuídos.

Quanto ao método utilizado para detectar ataques, o IDS pode ser baseado em assinatura (abuso ou baseado em conhecimento) ou em anomalia (baseado em comportamento). Anomalias são desvios do comportamento normal de uso e assinaturas, por outro lado, são padrões conhecidos de ataques.

O método de detecção por assinatura, como também é conhecido, consiste no casamento de padrões de dados contra bases de dados contendo padrões de ataques conhecidos referentes a atividades hostis ocorridas no passado. Sistemas deste tipo são altamente eficientes na identificação de ataques e vulnerabilidades conhecidos, mas fracos na identificação de novas ameaças aos sistemas.

O método de detecção por anomalia, por outro lado, busca dados raros ou não usuais em um conjunto de dados, aplicando medidas estatísticas ou de inteligência artificial para comparar atividades correntes de usuários, rede ou sistemas contra o conhecimento histórico armazenado sobre estes elementos. Problemas comuns com os sistemas baseados em anomalias são que requerem treinamento extensivo de dados históricos, através de algoritmos de aprendizagem artificial, e tendem a ser computacionalmente mais caros, porque devem realizar várias medições com muita frequência, possui a sobrecarga de manter informação das ações realizadas no passado e ser atualizados a cada novo comportamento dos sistemas, usuários ou rede. Isto resulta em armazenamento de grande quantidade de dados e satisfatórios recursos de CPU para processá-los. A atualização de várias métricas de perfil do sistema deve ser feita sob medida rede a rede, sistema a sistema e, às vezes, usuário a usuário, devido ao fato dos padrões de comportamento e de uso do sistema variarem muito.

O crescente número de ataques requer uma atualização contínua da base de conhecimento dos sistemas de detecção baseados em assinaturas. Além disto, existe também um número desconhecido de vulnerabilidades descobertas mas não reveladas que não se encontram disponíveis para análise e inclusão na base de conhecimento. A maioria dos ataques é polimorfa e os atacantes exploram este polimorfismo para evadir os sistemas detectores.

A solução óbvia seria utilizar uma abordagem de detecção por anomalia, modelando o que é normal ao invés do que é anômalo. Entretanto, enquanto vários sistemas de detecção por anomalia baseados em host tem sido propostos e implementados, tanto na literatura quanto na prática, a detecção por anomalia baseada em rede é ainda um campo aberto de pesquisa.

Atualmente, tem sido utilizadas técnicas de *data mining*, tais como aprendizagem de máquina e detecção de *outliers* para a detecção de anomalias no tráfego de rede. O comportamento do tráfego normal é modelado por um método sistemático. Os traços de desvio significativo entre as atividades de rede monitoradas e o modelo construído indicam possíveis anomalias.

Enquanto as assinaturas de ataques são, em geral, mais simples de processar e localizar, os padrões de anomalia auxiliam mais frequentemente na identificação de eventos suspeitos na rede. Alguns IDSs combinam as técnicas de detecção por assinatura e por anomalia; estes são conhecidos como sistemas híbridos.

3.2 Considerações sobre Avaliação de IDS

Avaliar sistemas de detecção de intrusão é uma tarefa complexa devido a vários motivos:

- É difícil obter dados de alta qualidade para realizar a avaliação devido às questões de privacidade e competitividade, muitas organizações não desejam compartilhar seus dados com outras instituições;

- Mesmo se dados reais estejam disponíveis, rotular as conexões de rede como normal ou intrusivas requer muito tempo por parte dos especialistas humanos;
- A mudança constante do tráfego de rede pode não somente introduzir novos tipos de intrusões, mas pode também alterar os aspectos de comportamento normal, tornando a construção de marcas comparativas ou de referência muito difícil;
- Quando medindo a desempenho de um IDS, há a necessidade de medir não somente a taxa de detecção (ou seja, quantos ataques foram detectados corretamente), mas também a taxa de alarmes falsos (ou seja, quantas conexões normais foram incorretamente detectadas como ataques) bem como o custo da má (incorreta) classificação;
- A complexidade da avaliação de um IDS torna-se também complexa pelo fato de que alguns ataques (por exemplo, negação de serviço e varredura) podem usar centenas de pacotes ou conexões de rede, enquanto outros tipos de ataques tais como U2R (*user to root*) e R2L (*remote to local*) geralmente usam somente uma ou algumas poucas conexões.

As métricas padrão que têm sido utilizadas para avaliar intrusões na rede em geral correspondem à taxa de detecção bem com a taxa de alarmes falsos, como mostra a Tabela 3.1 abaixo (LAZAREVIC et al., 2003). Dois tipos de alarmes podem ser gerados pelo IDS: falso-negativo ou falso-positivo (alarme falso). Falso-negativos são ocorrências de eventos intrusivos sinalizados pelo IDS como normais, geralmente ocorrem nos sistemas baseados em assinaturas. Falso-positivos (ou alarmes falsos) são ocorrências de eventos normais sinalizados pelo IDS como intrusivos, gerados com frequência pelos IDSs baseados em anomalias.

A taxa de detecção é calculada como a razão entre o número de ataques corretamente detectados e o número total de ataques. A taxa de alarmes falsos

(falso-positivos) é calculada como a razão entre o número de conexões normais que são incorretamente classificadas como ataques (alarmes falsos) e o número total de conexões normais.

Tabela 3.1 - Métricas padrão para avaliações de intrusões de conexão-única

Métricas padrão		Rótulo da conexão predito pelo algoritmo	
		Normal	Ataque
Rótulo real da conexão	Normal	Verdadeiro negativo	Alarme falso (falso positivo)
	Ataque	Falso negativo	Ataques corretamente detectados

Fonte: Adaptada de Lazarec et al. (2003)

Qualquer IDS, seja operando em modo on-line ou offline, resolve um problema de classificação (MAHONEY et al., 2002), que é distinguir entre atividade legítima e um ataque.

A taxa de falsos positivos é importante no que se refere à aceitação do sistema pelo usuário. Frente aos diversos alarmes falsos, os usuários aprendem a ignorar as advertências do sistema. Deste modo, positivos verdadeiros são ignorados e não levados a sério. Portanto, um classificador bem sucedido tem que exibir especificidade excepcional.

Redes de computadores geram grandes quantidades de dados durante sua operação normal e as atividades constituintes de um ataque formam uma parte muito pequena do tráfego e dos dados de auditoria. Esta parte pequena de instâncias de ataques nos dados de treinamento redobram a dificuldade da tarefa de aprendizagem dos métodos inteligentes.

Sistemas baseados em detecção por anomalia são mais flexíveis, ou seja, generalizam bem para novos cenários de ataques. Infelizmente, devido a grande diversidade das atividades de comunicação que normalmente ocorre em um ambiente de rede, estes modelos em geral produzem uma grande taxa de alarmes falsos. Os desenvolvedores têm limitado o escopo dos modelos de

detecção por anomalia a um usuário ou uma aplicação somente, onde a complexidade é possível de ser trabalhada.

Sistemas de reconhecimento de padrões ou detectores por assinatura são melhores na eliminação de alarmes falsos, porque estes comparam o tráfego corrente observado a um repositório de modelos de ataques conhecidos. Em contrapartida, estes sofrem de uma taxa de detecção mais baixa com relação a novas atividades hostis.

4 MINERAÇÃO DE DADOS DO TRÁFEGO DE REDE

De modo a viabilizar a análise de grandes conjuntos de dados do tráfego de rede, técnicas de mineração de dados (*data mining*) têm sido utilizadas como componentes-chave nos sistemas de detecção de intrusão atuais. *Data Mining* é um processo automático ou semi-automático de descoberta de padrões em dados (FAYYAD et al., 1996a; FAYYAD et al., 1996b). Envolve a aplicação de algoritmos sobre grandes e heterogêneos conjuntos de dados multidimensionais em busca de conhecimentos implícitos e úteis. (GOLDSCHMIDT et al., 2005). Técnicas de *data mining* podem ser aplicadas para a obtenção de conhecimento útil dos dados coletados pelos IDSs, auxiliando os analistas na detecção de novas vulnerabilidades e intrusões (LEE et al., 1998; BLOEDORN et al., 2001; DOKAS et al., 2003), na descoberta de padrões previamente conhecidos de comportamento intrusivo e prover suporte à decisão no gerenciamento de intrusões.

Padrões de comportamento normal de rede, usuários ou sistemas (anomalias) e padrões de eventos intrusivos (assinaturas) podem ser encontrados utilizando-se técnicas de *data mining*. Para aplicar tais técnicas em detecção de intrusão, primeiramente, deve-se preprocessar os dados coletados e convertê-los em um formato conveniente para o processo de mineração. Em seguida, os dados formatados serão utilizados na construção de um modelo de classificação ou *clustering* (agrupamento) .

Dentre as técnicas de *data mining* do campo de Inteligência Computacional, utilizadas em detecção de intrusão destacam-se: Redes Neurais Artificiais; Regras de Indução; Algoritmos Genéticos; Lógica Nebulosa e Teoria dos Conjuntos Aproximativos (*Rough Sets*).

No campo da Estatística, as técnicas de *data mining* que têm sido utilizadas em detecção de intrusão são: abordagens de detecção de *outliers* e cadeia de *Markov*, entre outras.

A mineração de dados deve ser um processo eficiente, pois trata de grandes quantidades de dados e usa algoritmos de complexidade computacional elevada. Existem basicamente três formas de acelerar esse processo: reduzindo a quantidade de dados, otimizando algoritmos e utilizando técnicas de processamento paralelo e/ou distribuído. Além da questão do desempenho, deve-se preocupar com a qualidade e a forma com que os dados encontram-se armazenados, ou seja, se contêm inconsistências, valores ausentes, ou necessitam de algum tipo de transformação. Para solucionar estes problemas grande quantidade de tempo é necessária durante o processo de descoberta.

4.1 Abordagens de *Data Mining*

Diferentes abordagens de *data mining*, tais como *clustering* (agrupamento), classificação, detecção de *outliers* e regras de associação são freqüentemente utilizadas (RAHMAN et al., 2006) para analisar os dados de rede em busca de conhecimento relacionado à intrusão. Estas abordagens são descritas a seguir e foram adaptadas de (POLITI, 2005; RAHMAN et al., 2006).

4.1.1 Clustering

Clustering é um mecanismo de aprendizagem de máquina não supervisionado empregado para encontrar padrões em dados não rotulados de várias dimensões, atribuindo dados a grupos. Algoritmos de *clustering* podem agrupar novas instâncias de dados em grupos similares os quais podem ser usados para ampliar o desempenho dos classificadores existentes. *Clusters* de alta qualidade podem também auxiliar os especialistas humanos na categorização ou rotulação de dados. Técnicas de *clustering* podem ser classificadas como: *pair-wise clustering* e *central clustering*. *Pair-wise clustering* é uma técnica de *clustering* baseada em similaridade que unifica instâncias de dados semelhantes com base em uma medida de distância. Por outro lado, *central clustering*, também chamada *clustering* baseada em modelo ou baseada em centróide, modela cada *cluster* a partir do seu centróide (RAHMAN et al., 2006). Em termos de complexidade em tempo de execução, algoritmos de

clustering baseados em centróide são mais eficientes que algoritmos de *clustering* baseados em similaridade. O processo de *clustering* possibilita descobrir intrusões complexas ocorridas sobre períodos maiores de tempo e

4.1.2 Classificação

Classificação é uma técnica semelhante a *clustering*, em que os registros de dados são particionados em segmentos distintos denominados classes. Mas, diferente de *clustering*, a análise baseada em classificação requer que um especialista humano conheça a priori como as classes são definidas. É necessário que cada registro no banco de dados usado para construir o classificador já possua um valor para o atributo utilizado na definição das classes. Como cada registro possui um valor para o atributo de definição de classes, e como o usuário decide sobre qual atributo utilizar, a técnica de classificação é muito menos exploratória que *clustering*. O objetivo de um classificador não é explorar os dados para descobrir segmentos de interesse, mas decidir como os novos registros devem ser classificados. Os algoritmos de classificação podem ser classificados em três tipos (RAHMAN et al., 2006):

- Extensões de discriminação linear (por exemplo, *multilayer perceptron*, discriminação logística);
- Árvore de decisão;
- Métodos baseados em regras.

Em comparação a *clustering*, a técnica de classificação é menos empregada no domínio de detecção de intrusão. O principal motivo é a grande quantidade de dados que precisa ser coletada para classificação. Para construir os traços e formar os grupos normais e anormais, é necessária a análise de uma quantidade significativa de dados para assegurar a formação correta dos grupos. Através do uso de dados coletados como modelos empíricos, a taxa de alarme falso neste caso é significativamente mais baixa quando comparada a *clustering*. A abordagem de classificação pode ser útil para ambos os métodos

de detecção por assinatura e por anomalia, mas é mais utilizada para detecção por assinatura.

4.1.3 Detecção de *Outliers*

A técnica de detecção de *outliers* é aplicada em várias tarefas, tais como sanitização de dados, detecção de fraudes e detecção de intrusos. A existência de *outliers* em uma base de dados indica que existem indivíduos ou grupos que possuem comportamento muito diferente da maioria dos indivíduos desta base. Em outras palavras, um *outlier* pode ser definido como um ponto de dado que é muito diferente do restante dos dados de um conjunto, com base em alguma métrica (ERTOZ, 2003b). Na maioria das vezes, os *outliers* são removidos para melhorar a precisão dos estimadores.

A maioria dos algoritmos de detecção baseados em anomalias requer um conjunto de dados essencialmente normais para treinar o modelo, e implicitamente supõe-se que as anomalias possam ser tratadas como padrões antes não vistos ou *outliers*, razão pela qual a técnica de detecção de *outliers* seja tão útil no processo de detecção de intrusos baseada em anomalia. Abordagens de detecção de *outliers* podem ser úteis para detectar quaisquer ataques não conhecidos. Este é o principal motivo pelo qual estas técnicas sejam tão popularmente utilizadas na construção de sistemas de detecção de intrusão (RAHMAN et al., 2006). As técnicas de detecção de *outliers* utilizadas com mais frequência são: Distância de *Mahalanobis*, *Partitioning Around Medias* (PAM) e algoritmo de Bay para detecção de *outliers* baseada em distância.

4.1.4 Regras de Associação

Regras de Associação são utilizadas para descobrir n -tuplas de elementos, itens ou atributos que ocorrem com determinada frequência dentro de um conjunto de dados, e para prever as próximas combinações de atributos que podem aparecer no conjunto (WITTEN, 2000). Uma Regra de Associação caracteriza o quanto a presença de um conjunto de itens nos registros de uma

base de dados implica na presença de algum outro conjunto distinto de itens nos mesmos registros (AGRAWAL et al., 1994). Deste modo, o objetivo das Regras de Associação é encontrar tendências que possam ser usadas para entender e explorar padrões de comportamento dos dados, obtendo as correlações entre os atributos de tabelas do banco de dados. Por exemplo, observando os dados de vendas de um supermercado, sabe-se que 80% dos clientes que compram o produto Q também adquirem, na mesma ocasião, o produto W. Nessa regra, 80% corresponde à sua confiabilidade.

Cada regra de associação provê informação do tipo “*if-then*” e possui dois lados: o lado esquerdo (o antecedente) e o lado direito (o conseqüente), definidos por conjuntos distintos de itens. Uma regra de associação considera cada par atributo/valor como um item. Um conjunto de itens é denominado *itemset*.

Uma aplicação típica para regras de associação é a análise de transações de compras. A partir de uma base onde estão registrados os itens adquiridos por cada cliente, uma estratégia de mineração de regras de associação poderia gerar o seguinte exemplo (GONÇALVES, 2005): {cerveja} → {salaminho}, a qual indica que se o cliente compra {cerveja}, com um determinado grau de certeza, compra também {salaminho}. Este grau de certeza costuma ser definido através da utilização de índices estatísticos como o fator de suporte e o fator de confiança. O **suporte** de um conjunto de itens Z, $Sup(Z)$, representa a porcentagem de transações da base de dados que contêm os itens de Z. O **suporte** de uma regra de associação $A \rightarrow B$, $Sup(A \rightarrow B)$, representa a porcentagem de transações da base de dados que contêm os itens do itemset A (antecedente da regra) e os itens de B (conseqüente da regra), indicando a relevância da mesma. A **confiança** desta regra, denotada por $Conf(A \rightarrow B)$, representa, dentre as transações que possuem os itens de A, a porcentagem de transações que também possuem os itens de B, indicando a validade da regra. Por exemplo, se a $confiança(A \rightarrow B) = 0,5$ (50%), A e B ocorrem juntos em 50% das vezes em que A ocorre.

Um exemplo de *itemset* freqüente é a uma combinação de valores de atributos que aparece com certa freqüência em um conjunto de dados do tráfego de rede. O algoritmo percorre a base de dados em busca de *itemsets* que tendem a aparecer com freqüência nos dados de rede.

Os passos básicos para incorporar regras de associação em detecção de intrusão são encontrados em (RAHMAN et al., 2006).

Regras de associação e freqüência de episódios em série são dois métodos de *data mining* que têm sido utilizados no campo de detecção de intrusão, permitindo a extração automatizada de padrões normais de dados, os quais são utilizados para treinamento dos modelos de detecção. Estes métodos têm sido propostos para recuperar dados de auditoria, seleção de características ou atributos e análise *off-line* para detecção de anomalias.

Enquanto a regra de associação é usada para encontrar a correlação entre diferentes *item sets* em uma transação, a técnica de freqüência de episódios busca por um padrão que ocorre repetidas vezes na seqüência de eventos. A vantagem do uso destes dois métodos é que ambos os padrões, padrões entre diferentes atributos e padrões entre eventos seqüenciais, podem ser explorados.

Neste trabalho foram utilizadas as abordagens de *clustering*, classificação e detecção de outliers para detecção de padrões anômalos (desvios de comportamento normal do tráfego) e assinaturas de ataques conhecidos.

4.2 Técnicas de Data Mining em Detecção de Intrusão

Diferentes técnicas de *data mining* têm sido exploradas para a detecção por anomalia e assinatura no tráfego de rede.

Neste capítulo as anomalias são consideradas eventos diferentes do modelo de comportamento normal do tráfego, incluindo ações legítimas, embora

previamente não observadas, ou novos ataques. Assinaturas, por outro lado, correspondem a padrões de ataques conhecidos.

O processo de detecção por anomalia, por definição, requer um conjunto de dados normais para treinar o modelo, e, então, implicitamente as anomalias são tratadas como padrões antes não observados. A maioria das pesquisas em detecção de anomalias supervisionada tenta construir um modelo com base nos dados normais e então verificar o quanto os novos dados se enquadram neste modelo. Algumas abordagens supervisionadas empregam modelos de predição obtidos através do treinamento de árvores de decisão sobre dados normais, enquanto outros usam redes neurais para obter o modelo normal e detectar novos ataques. As redes neurais do tipo *MultiLayer Perceptron* (MLP) e *Radial Basis Function* (RBF) bem como as Máquinas de Vetor de Suporte (SVM) têm sido utilizadas (CANSIAN, 1997; BONIFACIO et al., 1998; LEE et al., 2001; LINGXUAN et al., 2001; AMBWANI, 2003; BRIDGES et al., 2003; JIANG et al., 2003; MUKKAMALA, 2002a; MUKKAMALA, 2003a; MORADI et al., 2004; TAPIADOR et al., 2004a; TAPIADOR et al., 2004b) para treinamento supervisionado dos modelos de detecção por anomalia em redes.

O conjunto de dados normais para treinar o módulo de detecção do IDS pode ser obtido através da coleta de amostras do tráfego normal da rede a ser examinada, em diferentes intervalos de tempo e dias da semana. Outra forma de gerar o tráfego normal a ser apresentado ao detector é a sanitização de conjuntos de dados de tráfego histórico através de análise manual por um especialista ou com o auxílio de um classificador automatizado.

Uma outra abordagem (não-supervisionada) para obter anomalias no tráfego de rede é ajustar os parâmetros necessários de um detector que “por si só” é capaz de aprender sobre o comportamento normal do tráfego e identificar anomalias. Técnicas não-supervisionadas para detecção de intrusão em redes podem ser implementadas através de algoritmos de clustering, tais como redes neurais SOM, utilizadas em trabalhos como (ABDI et al., 2003; LABIB et al., 2002; CHAVES et al., 2005a; CHAVES, 2005b, KAYACIK et al., 2003;

ZANERO, 2005a; ZANERO, 2005b) algoritmos de detecção de *outliers*, incluindo abordagem *Nearest Neighbor*, *distância de Mahalanobis*, abordagem *Local Outlier Factor* (DOKAS et al., 2002; ERTOZ et al., 2003a; ERTOZ et al., 2003b; ERTOZ et al., 2003c; LAZAREVIC et al., 2003) e algoritmo *fuzzy clustering* (SHAH et al., 2003).

Outras pesquisas em que regras fuzzy são empregadas para suporte à decisão também têm sido conduzidas para a detecção de ataques no tráfego de redes (DICKERSON, 2000; DICKERSON, 2001; LUO et al., 2000; SIRAJ et al., 2001; ORFILA et al., 2003; HOFMANN et al., 2003; OUYANG et al., 2002; PETROVSKIY et al., 2003; QUANMIN et al., 2001; SOURCEFIRE, 2007; TSAI et al., 2003).

Portanto, para a detecção de tráfego anômalo em conjuntos de dados de rede, podem ser utilizadas técnicas de *data mining* inteligentes ou estatísticas, supervisionadas ou não supervisionadas.

Neste trabalho, as técnicas de *data mining* exploradas para a identificação de tráfego anômalo foram as redes neurais SOM, LVQ e MLP e os algoritmos de detecção de outliers LOF e LSC. Para identificação de padrões de ataques conhecidos no tráfego de rede foram utilizadas as redes neurais Hamming Net e LVQ. Conceitos básicos destas abordagens são apresentados a seguir.

4.2.1 Rede Neural SOM

A rede SOM (*Self Organizing Map*), também conhecida como “Mapa de Kohonen” (KOHONEN, 1989), pertence à classe de redes neurais não-supervisionadas que se baseiam no processo de aprendizagem competitiva, onde somente um neurônio de saída ou grupo local de neurônios fornece uma resposta ativa a um sinal de entrada corrente. O nível de ativação indica a similaridade entre o vetor de dados de entrada e o vetor de pesos do neurônio. Uma forma usual de expressar a similaridade é através da distância euclidiana entre esses vetores. Uma vez que a distância entre o vetor de pesos de um determinado neurônio e o vetor de dados de entrada é a mínima para todos os

neurônios da rede, esse neurônio juntamente com um conjunto pré-definido de neurônios vizinhos terá seus pesos automaticamente reajustados pelo algoritmo de aprendizagem da rede. A vizinhança de cada neurônio pode ser definida de acordo com a forma geométrica usada para representar os neurônios da rede. A Figura 4.1 ilustra dois exemplos de representação propostos por Kohonen em 1989: um na forma de um *array* retangular e o outro na forma de um *array* hexagonal.

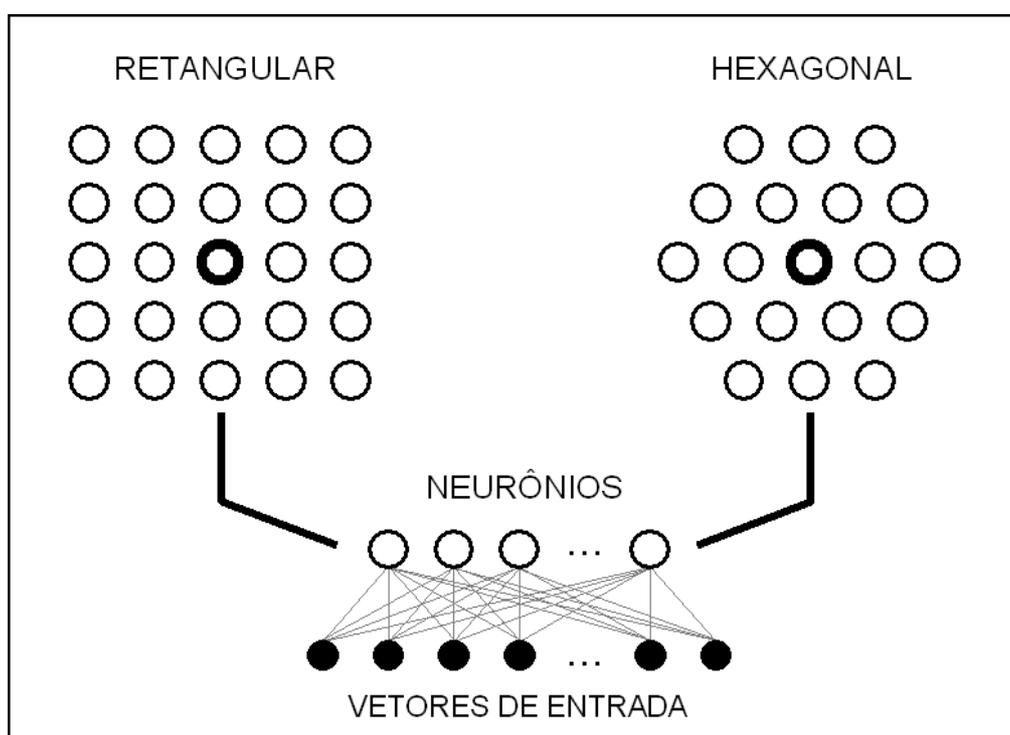


Figura 4.1 - Exemplo de Representação Geométrica dos Neurônios da SOM

Um dos princípios em que se baseiam os modelos auto-organizáveis é o de que padrões que compartilham características comuns devem ser agrupados, com cada grupo de padrões representando uma e apenas uma classe (embora uma mesma classe possa ser representada por mais de um agrupamento).

A rede SOM é utilizada em duas fases: na primeira ocorre o treinamento da rede para organizar os dados, de forma que os mais parecidos fiquem próximos entre si. Para isso, quando um padrão de entrada \mathbf{p} é apresentado, a rede procura a unidade mais parecida com \mathbf{p} . Assim, a rede constrói um mapa topológico, onde os nós que estão topologicamente próximos respondem de

forma semelhante a padrões de entrada semelhantes. A segunda fase é a de classificação, onde a rede SOM utiliza o mapa organizado para identificar a classe mais próxima à entrada.

O algoritmo de aprendizagem da rede SOM é apresentado a seguir:

1) Selecionar um padrão de treinamento $X = (x_1, x_2, \dots, x_n)$ e fornecê-lo como entrada à rede;

2) Calcular as distâncias d_i entre o vetor de entrada e o vetor de pesos de cada neurônio j da rede:

$$d_i = \sum_{j=1}^N (x_j(t) - w_{i,j}(t))^2 \quad (4.1)$$

onde $x_j(t)$ é a j -ésima entrada em uma dada iteração e $w_{i,j}(t)$ é o peso do neurônio j da camada de entrada conectado ao neurônio i da camada de saída;

3) Selecionar um neurônio i^* com a distância mínima entre todos os outros neurônios, e ajustar o vetor de pesos de i^* e de seus vizinhos através da seguinte regra:

$$w_{i,j}(t+1) = w_{i,j}(t) + \alpha(t) * (x_j(t) - w_{i,j}(t)) \quad (4.2)$$

$$\text{para } i \in N_i^*, j = 1, 2, \dots, N$$

onde N_i^* é o conjunto que contém i^* e seus vizinhos, e $\alpha(t)$ é a taxa de aprendizagem que é, em geral, menor do que 1. O procedimento continua até que o ajuste dos pesos não seja mais significativo.

No final do processo de aprendizagem cada neurônio ou grupo de neurônios vizinhos representará um padrão distinto dentro do conjunto de padrões fornecidos como entrada para a rede.

4.2.2 Rede Neural MLP

Os *Multi-Layer Perceptrons* (MLP) ou Perceptrons de Múltiplas Camadas têm sido aplicados com sucesso para resolver diversos problemas complexos, através de seu treinamento de forma supervisionada, com um algoritmo muito popular conhecido como retropropagação do erro ou *BackPropagation* (FAUSSET, 1994; HAYKIN, 2001).

O algoritmo de *backpropagation* é uma generalização do algoritmo do método dos mínimos quadrados, que utiliza técnicas de gradiente descendente iterativo para minimizar uma função de custo igual à diferença média quadrada entre a saída desejada e a saída real da RNA. O processo de treinamento (MARTINS, 2003) é apresentado através de uma rede neural de 4 camadas na Figura 4.2.

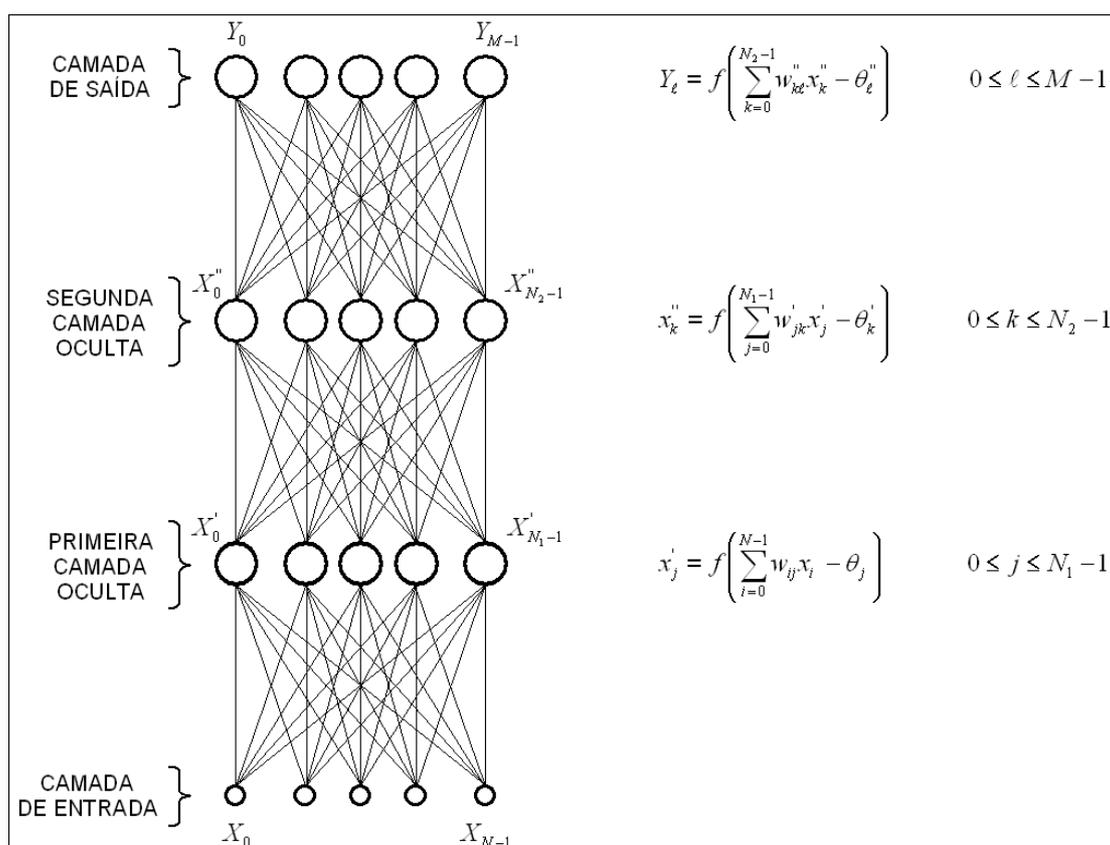


Figura 4.2 - RNA de 4 camadas e fórmulas de ajuste de pesos
Fonte: Adaptado de MARTINS (2003).

Esta rede utiliza N valores contínuos como as suas entradas, M valores como saídas e duas camadas de unidades ocultas. Usa uma função não-linear a cada unidade de processamento e a regra de decisão escolhida é: “selecione a classe correspondente à unidade de processamento com a maior saída”. Nas fórmulas, x_j e x_k correspondem às saídas das unidades de processamento da primeira e segunda camada oculta; e θ_i , θ'_k e θ''_l são os “offsets” internos das unidades de processamento das camadas ocultas e de saída, w_{ij} denota os pesos das conexões entre a camada de entrada e a primeira camada oculta, w_{ij} , e w'_{ij} refere-se aos pesos entre as duas camadas ocultas e aos pesos entre a última camada oculta e a camada de saída.

No processo de treinamento, inicialmente as unidades de processamento da RNA são aleatoriamente inicializadas com diferentes limiares internos e as conexões entre estas são inicializadas com pesos de valores pequenos (são comuns valores entre -1.0 e 1.0). Em seguida, os dados de treinamento são apresentados à RNA e, a cada ciclo de treinamento, os pesos são ajustados através de uma informação complementar que indica a correta classe de saída, até que a função de custo seja reduzida a um valor aceitável. A parte principal do algoritmo de *backpropagation* é a maneira interativa pela qual os erros utilizados para adaptar os pesos são propagados para trás, isto é, a partir da camada de saída para as camadas anteriores.

No processo de treinamento utilizando o algoritmo de *backpropagation* são realizados os seguintes passos:

- 1- Ajustar todos os pesos e todos os “offsets” das unidades de processamento a partir de valores aleatórios pequenos;
- 2- Apresentar nova entrada como um vetor (x_0, x_1, x_N) de valores contínuos e especificar a saída desejada como um vetor (d_0, d_1, d_N) . Cada novo vetor de entrada de um conjunto de treinamento pode ser apresentado ciclicamente até os pesos se estabilizarem;

3- Calcular as saídas reais da rede (y_0, y_i, y_N) utilizando as fórmulas contidas na Figura 4.2;

4- Adaptar os pesos, através de um algoritmo recursivo iniciando nas unidades de processamento de saída, atuando para trás no sentido da primeira camada e ajustando os pesos através da fórmula:

$$w_{ij}(t+1) = w_{ij}(t) + \eta \delta_j x_i' \quad (4.3)$$

onde w_{ij} é o peso do elemento de processamento oculto j no tempo t ; x_i pode ser tanto uma unidade de processamento de saída, quanto uma unidade de entrada, denota um fator de ganho (velocidade da aprendizagem), e d_j equivale a um fator de erro para a unidade de processamento j . Se j for uma unidade de saída, então:

$$\delta_j = y_j(1 - y_j)(d_j - y_j) \quad (4.4)$$

onde d_j denota a saída desejada e y_j é a saída real da rede; se a unidade j for uma unidade oculta então:

$$\delta_j = x_j'(1 - x_j') \sum_k \delta_k W_{jk} \quad (4.5)$$

onde k denota todas as unidades acima das unidades j ; os limiares delta das unidades internas são ajustados de forma semelhante. A convergência algumas vezes pode ser mais rápida se um fator de momento for adicionado e se os pesos forem alterados de forma mais suave, através da equação:

$$W_{ij}(t+1) = W_{ij}(t) + \eta \delta_j x_i' + \alpha (W_{ij}(t) + W_{ij}(t-1)) \quad (4.6)$$

onde $0 < \alpha < 1$;

5 – Repetir o processamento, retornando ao passo 2.

4.2.3 Rede Neural Hamming Net

Hamming Net é uma rede classificadora de máxima verossimilhança (FAUSSET,1994) que pode ser usada para determinar quais dos vários vetores exemplares é mais similar a um vetor de entrada (*uma n -tupla*). As entradas e exemplares nesta rede devem ser inseridos em formato bipolar. Os vetores exemplares determinam os pesos da rede. A medida da similaridade entre o vetor de entrada e os vetores exemplares armazenados é n (número de componentes nos vetores) menos a distância de Hamming entre os vetores. A distância de Hamming entre os vetores é o número de componentes nos quais os vetores se diferem. A Figura 4.3 ilustra a arquitetura desta rede.

Configurando os pesos para ter o valor equivalente à metade dos valores do vetor exemplar e configurando o bias para $n/2$, a rede encontrará a unidade com o exemplar mais próximo da entrada.

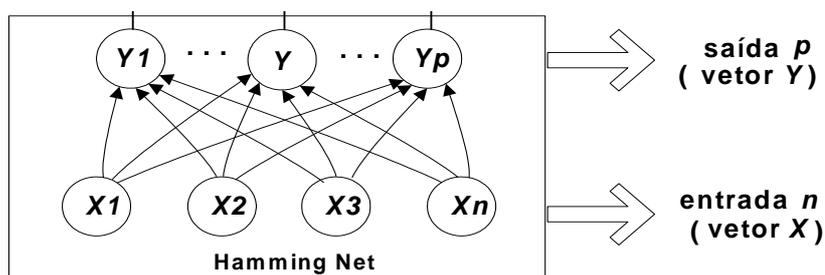


Figura 4.3 - Arquitetura da rede Hamming Net

Nesta rede, p diferentes classes ou exemplares são previamente introduzidas na Hamming Net (vetor Y) que requer p neurônios de saída. Após o processamento, a saída da Hamming Net corresponderá ao valor correspondente à classe mais semelhante ao padrão de entrada, em termos de distância de Hamming.

A rede Hamming Net possui não requer treinamento exaustivo para aprendizagem dos padrões de interesse. Além disto, é capaz de ser facilmente atualizada na presença de novos padrões desconhecidos com a inserção de novos neurônios na rede. Outro aspecto desta rede é que sempre provê um

casamento entre um padrão de entrada e um padrão exemplar, mesmo que seus valores sejam aproximados. Na análise de comportamento de tráfego utilizando dados de cabeçalho dos pacotes, esta característica é útil por permitir a detecção de variações de ataques ou de novos ataques.

4.2.4 Rede Neural LVQ

A rede LVQ (*Learning Vector Quantization*) pertence à família de algoritmos de Quantificação Vetorial Adaptativa, originalmente criada por *Kohonen* (HAYKIN, 2001), que podem ser utilizados para a classificação e reconhecimento de padrões, a partir de um conjunto de amostras de treinamento. As redes LVQ são precursoras das redes SOM e podem ser vistas como um tipo especial de RNA.

Uma rede LVQ em sua forma simplificada, como ilustrado na Figura 4.4, consiste basicamente de duas camadas: uma camada de entrada e uma camada de saída. A camada de saída corresponde a um conjunto de vetores de referência (exemplares), as coordenadas dos quais são os pesos das conexões dos neurônios de entrada a cada neurônio de saída. Conseqüentemente, pode-se dizer também que cada neurônio de saída corresponde a um vetor de referência (FAUSSET, 1994).

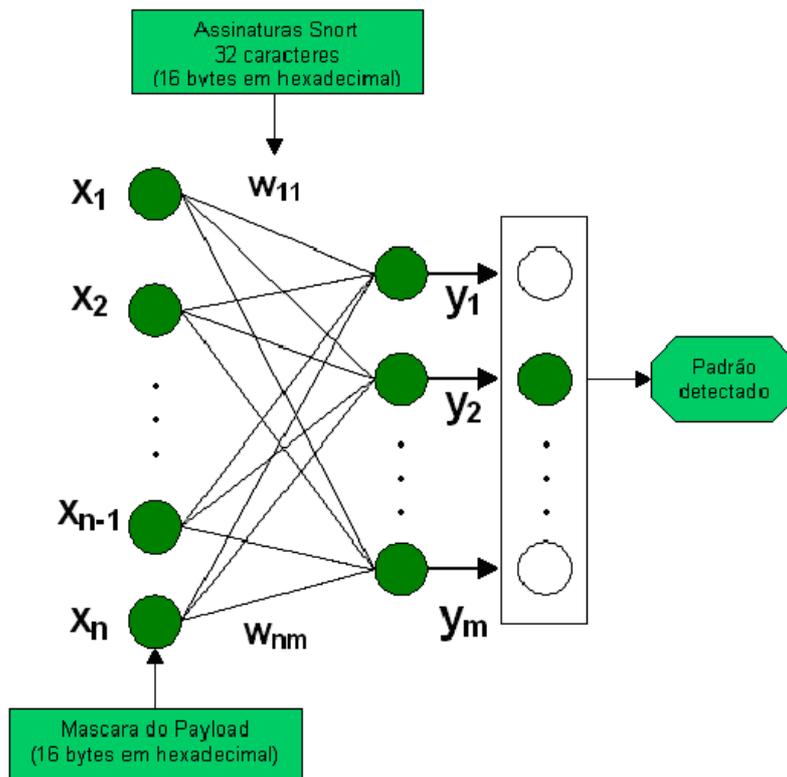


Figura 4.4 – Arquitetura da Rede LVQ Simplificada

O método de aprendizagem da LVQ é chamado aprendizagem por competição, porque funciona da seguinte forma: para cada padrão de treinamento o vetor de referência que é mais próximo deste deve ser determinado. O neurônio de saída correspondente é chamado de neurônio vencedor. Os pesos das conexões a este neurônio (o vencedor *leva-tudo*) são então adaptados. A direção da adaptação depende se a classe do padrão de treinamento e a classe atribuída ao vetor de referência coincidem ou não. Se coincidirem, o vetor de referência é movido mais próximo do padrão de treinamento, senão, é movido para longe deste. Este movimento do vetor de referência é controlado por um parâmetro chamado taxa de aprendizagem $\alpha(t)$. Em geral, a taxa de aprendizagem é decrementada ao longo do tempo, de modo que as alterações iniciais sejam maiores que as mudanças feitas em épocas passadas no processo de treinamento. O treinamento pode ser encerrado quando as posições dos vetores de referência não se alteram mais de modo significativo.

Um resumo do algoritmo de uma LVQ com aprendizagem, utilizando a regra de aprendizagem competitiva padrão, é descrito da seguinte forma:

- Escolher o número de clusters M ;
- Inicializar o vetor de pesos conectados a cada nó de saída w_1, w_2, \dots, w_m .
Um método simples de fazer isto é randomicamente escolhendo os M vetores de pesos a partir dos dados de entrada;
- Repetir o procedimento até que o critério de parada seja satisfeito:
 - Randomicamente obter uma entrada x ;
 - Determinar o nó vencedor definindo o vetor de pesos que satisfaz a equação:

$$|w_{*k} - x| \leq |w_{*i} - x| \quad (4.7)$$

para todo i . Observar se os pesos estão normalizados é equivalente a maximizar $w_{*i} \cdot x$;

- Atualizar somente os pesos do neurônio vencedor utilizando a equação:

$$w_{*k}(new) = w_{*k}(old) + \alpha(t) \cdot (x - w_{*k}(old)) \quad (4.8)$$

4.2.5 Técnicas de Detecção de Outliers

Uma anomalia pode ser considerada um *outlier*, que equivale a um ponto de dado que apresenta características diferentes em relação aos demais dados em um conjunto, com base em alguma métrica.

O conceito de *outliers* tem sido extensivamente pesquisado pela comunidade estatística. Nestas técnicas, os pontos de dados são modelados usando uma distribuição estocástica, e os pontos são rotulados como '*outliers*' dependendo de seu relacionamento com este modelo. Entretanto, com a dimensionalidade

crescente, torna-se muito difícil e impreciso estimar as distribuições multidimensionais dos pontos de dados, conforme descrito em (BREUNIG et al., 2000).

As técnicas de mineração de *outliers* podem ser baseadas em distribuição, baseadas em profundidade ou baseadas em distância.

Técnicas de mineração de *outliers* baseadas em distribuição de dados são encontradas com maior frequência no meio estatístico, onde as distribuições estatísticas padrão, tais como distribuição *Normal*, *Poisson*, *Student t*, entre outras, são usadas para distribuição dos pontos de dados (BREUNIG et al., 2000). Métodos baseados em distribuição requerem conhecimento prévio da distribuição estatística dos dados e são mais utilizados para dados de variável única.

Nas técnicas baseadas em profundidade, os objetos são organizados em camadas com a idéia de que nas camadas rasas existe maior probabilidade de conter dados *outliers* do que nas mais profundas. Métodos baseados em profundidade trabalham bem hipoteticamente para dados de alta dimensão, mas, na prática, não funcionam para mais de 3 dimensões.

Knorr et al. (1998) propuseram o conceito de *outlier* baseado em distância para solucionar os problemas apresentados nas técnicas de mineração de *outliers* anteriores. Um *outlier* baseado em distância é descrito como uma 'fração de objetos dados com distância maior que a distância mínima do *outlier*'. A escolha da distância mínima do *outlier* é deixada a critério do analista. Os algoritmos baseados em índice e *loops* aninhados propostos para minerar *outlier* baseados em distância tem uma complexidade de tempo de $O(kN^2)$ que é linear no número de dimensões, mas quadrática em termos de tamanho dos dados.

Dentre os algoritmos mais comuns de detecção de *outliers* citam-se: distância ao k-ésimo vizinho mais próximo, abordagem *Nearest Neighbor* (NN),

abordagem baseada em distância de *Mahalanobis* e abordagem LOF também denominados *outliers* locais baseados em densidade.

A seguir são descritos os dois algoritmos estatísticos baseados em detecção de *outliers* utilizados neste trabalho para detecção por anomalia.

4.2.5.1 Abordagem LOF

A idéia principal deste método é atribuir a cada amostra de dado um grau de ser *outlier* (estranho ou bem diferente dos demais). Este grau é chamado *Local outlier Factor* (LOF) ou Fator de Diferença Local de uma amostra de dado. Em (LAZAREVIC et al., 2003), um *outlier* é visto como uma observação com o mais alto fator LOF dentro de uma dada vizinhança. O fator LOF é uma medida da distribuição ou densidade de objetos ao redor de um ponto particular. Um fator LOF alto indica que os vizinhos estão muito longes, por outro lado, um fator LOF baixo indica vizinhos mais próximos.

O algoritmo para processar os valores LOF para todos os objetos de dados segue os seguintes passos:

1. Para cada objeto O calcular a distância k (a distância ao K -ésimo vizinho mais próximo) e a vizinhança da distância k (todos os objetos na esfera de distância k);
2. Calcular a distância de alcance (*reachability distance*) para cada objeto de dado O com relação ao objeto de dado p como:

$$reach-dist(O,p)=Max\{k-distance(p), d(O,p)\} \quad (4.9)$$

onde $d(O,p)$ é a distância do objeto O ao objeto p .

3. Calcular a densidade de alcance do objeto O como o inverso da distância de alcance médio baseado no valor *MinPts* (número mínimo de objetos de dados) mais próximos vizinhos do objeto O .

4. Calcular o LOF do objeto O como a média das proporções da densidade de alcance local do exemplo O e a densidade de alcance dos vizinhos $Minpts$ mais próximos de O .

Uma descrição de cada fase do algoritmo LOF encontrada em (LAZAREVIC et al., 2003) é apresentada a seguir:

Passo 1: Calcular a k-distância de p

O cálculo da k -distância de p tem por objetivo determinar os vizinhos de p . Em termos simples, a k -distância de p é a distância máxima do objeto p quando cada objeto no conjunto de dados é considerado ter no mínimo k -vizinhos.

Passo 2: Encontrar a K-Vizinhança de p

A vizinhança k -distante de p denotada por $N_k(p)$ contém todo objeto com distância não maior que (maior ou igual a) $kdistance(p)$. O motivo do cálculo da vizinhança k -distante é encontrar os vizinhos mais próximos de cada objeto.

Passo 3: Calcular a distância de alcance (reachability distance) de p

A distância de alcance de um objeto p com relação a um objeto o é a $distance(p,o)$ ou $kdistance(o)$, o valor que for maior ($reachdistk(p,o) = \max\{kdistance(o), distance(p,o)\}$). O objetivo é garantir que todos os objetos dentro de uma vizinhança são homogêneos. Além disto, LOF estabiliza quando os objetos dentro de uma vizinhança são uniformes mesmo se $MinPts(k)$ varia. As flutuações nas distâncias de alcance podem ser controladas pela escolha de valores maiores para k .

Passo 4: Calcular a densidade de alcance local (local reachability density) de p

A densidade de alcance local de um objeto p , denotada por $lrd_k(p)$, é o inverso da média das distâncias de alcance dos vizinhos k -distantes de p . É um meio de comparar as distâncias de alcance. Para calculá-la, utilizar a equação 4.10.

$$lrd_k(p) = \frac{1}{\frac{\sum_{o \in N_k(p)} reachdist_k(p, o)}{|N_k(p)|}} \quad (4.10)$$

Passo 5: Cálculo do LOF (Local Outlier Factor) de p

O fator de outlier local (LOF) é a razão que determina se um objeto é ou não um *outlier* com relação a sua vizinhança. O valor LOF de um objeto p denotado por $LOF_k(p)$ é a média das razões entre a densidade de alcance local de p e as densidades dos k -vizinhos mais próximos de p .

Para ilustrar as vantagens da abordagem LOF, Lazareck et al (2003) descrevem o exemplo a seguir. Considere um conjunto de dados bi-dimensional simples dado na Figura 4.5.

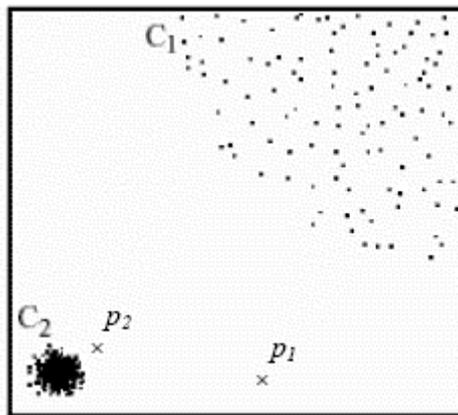


Figura 4.5 - Conjunto de dados bi-dimensional simples

Observa-se que há um número muito maior de objetos no *cluster* C1 que no *cluster* C2 e que a densidade do *cluster* C2 é consideravelmente maior que a densidade do *cluster* C1. Devido à baixa densidade do *cluster* C1, nota-se que, para cada objeto q dentro do *cluster* C1, a distância entre o objeto q e seu vizinho mais próximo é maior que a distância entre o objeto p_2 e o vizinho mais próximo do *cluster* C1, e o objeto p_2 não será considerado um *outlier*.

Entretanto, o objeto p_1 pode ser detectado como *outlier* usando somente as distâncias ao vizinho mais próximo. Por outro lado, LOF é capaz de capturar

ambos *outliers* (p1 e p2) devido ao fato de que ele considera a densidade ao redor dos objetos.

4.2.5.2 Abordagem *LSC-MINE*

Conforme observado por Agyemang et al. (2004) a maior limitação do algoritmo LOF reside no cálculo das distâncias de alcance definidas como $reachdistk(p,o) = \max\{kdistance(o),distance(p,o)\}$. O cálculo de distância de alcance de um objeto p envolve o cálculo das distâncias de todos os objetos dentro da vizinhança de p e cada um é comparado com a k-distância do objeto vizinho, o que torna esta técnica computacionalmente muito cara quando o conjunto de MinPts é grande. Outra limitação é que a LOF deve ser calculada para cada objeto antes que os poucos *outliers* sejam detectados, o que não é bom, uma vez que os *outliers* constituem somente uma pequena fração de todo o conjunto de dados.

No trabalho de Agyemang et al (2004) é proposto o algoritmo *LSC-Mine*, denominado *Local Sparsability Coefficient* (LSC) ou Coeficiente de Espalhamento Local, para mineração de *outliers* baseada em distâncias de objetos de seus vizinhos mais próximos sem precisar calcular as distâncias de alcance local (*Local Reachability Distance*) e as densidades de alcance local (*Local Reachability Density*) entre estes. Além disto, objetos de dados que não são possíveis candidatos a *outlier* são descartados logo que identificados, tornando o algoritmo *LSC-Mine* computacionalmente menos caro e mais eficiente comparado com outras técnicas de detecção de *outliers* descritas na literatura.

LSC-Mine apresenta melhor tempo de resposta que a LOF por evitar o cálculo de distâncias de alcance e densidades de alcance local e porque os objetos que não são possíveis candidatos a *outlier* são descartados quando identificados. Este descarte reduz drasticamente o número de cálculos dos Coeficientes de Espalhamento Local (*Local Sparsity Coefficient*) utilizados no *LSC-Mine*. O conjunto candidato a *outlier* é então usado como entrada para

cálculo deste coeficiente. Resultados experimentais (AGYEMANG, 2004) comprovaram que a LSC provê melhores resultados que a LOF com relação ao tempo de resposta para quaisquer tamanhos de dados do conjunto de testes utilizado e *MinPts*.

O algoritmo *LSC-Mine* é sucintamente descrito a seguir:

- 1) Calcular a *K*-distância de cada objeto;
- 2) Encontrar a vizinhança *K*-distante de cada objeto;
- 3) Calcular a Razão de Espalhamento Local (*LSR – Local Sparsity Ratio*) de cada objeto;
- 4) Calcular o Fator de Descarte (*Pf – Pruning Factor*) de cada objeto;
- 5) Obter o conjunto candidato a *outlier*;
- 6) Calcular o Coeficiente de Espalhamento Local (*LSC-Local Sparsity Coefficient*) utilizando o conjunto candidato;
- 7) Classificar como *outliers* aqueles objetos que possuem maior valor LSC.

Conforme consta em (AGYEMANG, 2004) as definições para cálculo dos valores das etapas 3 a 6 acima são:

Definição 4.2.5.2.1: A razão de espalhamento local de um objeto p denotado por $lsr_k(p)$ é definida como a razão entre a cardinalidade da vizinhança k -distante de p e a soma de todas as distâncias reais daquela vizinhança, como apresentado na equação 4.11, onde $distofN_k(P)$ consiste nas distâncias reais de objetos na vizinhança k -distante de p .

$$lsr_k(p) = \frac{|N_k(p)|}{\sum_{o \in N_k(p)} distofN_k(P)} \quad (4.11)$$

A razão de espalhamento local mede a concentração de objetos ao redor do objeto p . Objetos com baixa taxa de espalhamento local tem alto potencial de ser *outlier* e vice-versa. A declaração final de *outliers* depende do coeficiente de espalhamento local (LSC) ao invés da razão (LSR). O cálculo do *fator de pruning* baseia-se na hipótese de que a razão de espalhamento local de um objeto p em um conjunto de dados não deve ser menor que uma razão similar calculada a partir do conjunto de dados inteiro se aquele objeto não for um outlier. Isto gera um limite superior para qualquer objeto que é um candidato a outlier.

Definição 4.2.5.2.2: O fator de *pruning* (Pf) é a razão entre a soma das distâncias absolutas da vizinhança e a soma total de todas as distâncias reais da vizinhança. O cálculo deste fator é realizado através da equação:

$$Pf = \frac{\sum |N_k(p)|}{\sum_{o \in N_k(p)} distofN_k(p)} \quad (4.12)$$

Uma vez obtido o fator de *pruning*, qualquer objeto com uma razão de espalhamento local (LSR) maior que Pf (*Pruning Factor*) é removido visto que este não é um potencial candidato a *outlier*. O processo de descarte pode remover mais da metade dos dados, deste modo, ressaltando o desempenho da *LSC-Mine*.

Definição 4.2.5.2.3: O coeficiente de espalhamento local de p denotado por $LSC_k(p)$ é a razão média entre a razão de espalhamento local de p e a razão de espalhamento local de seus k -vizinhos mais próximos. O coeficiente de espalhamento local é calculado através da equação:

$$LSC_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lsr_k(o)}{lsr_k(p)}}{|N_k(p)|} \quad (4.13)$$

Um alto coeficiente de espalhamento local (LSC) indica que a vizinhança ao redor de um objeto não é povoada e, portanto, existe um alto potencial deste

objeto ser um outlier, enquanto que um baixo coeficiente de espalhamento local (LSC) indica uma vizinhança bem povoada e, portanto, o objeto apresenta um baixo potencial de ser um *outlier*.

4.3 Estado da Arte em Detecção de Ataques no Tráfego de Rede

Muitos trabalhos recentes tratam de detecção por anomalia baseada em hosts, para detectar eventos suspeitos em seqüências de *system calls* e freqüências de comandos de usuários. Estes sistemas modelam o comportamento normal de usuários ou sistemas, através de redes neurais LVQ e regras especialistas, rede neural *Radial Basis Function* (RBF) (JIANG et al., 2003; RAPAKA et al., 2003), HMM (*Hidden Markov Model* ou processo de *Markov*) e regras *fuzzy*. Outros trabalhos envolvem redes ART2 (*Adaptative Ressonance Theory*) aplicadas, a combinação de HMM, estatística, regras (LUO et al., 2000) e uso de redes SOM.

Os sistemas estritamente baseados em rede incluem NSM (HEBERLEIN et al., 1990), Bro (PAXSON, 1999a; PAXSON, 1999b; PAXSON, 2003), NFR e NetStat. NSM é um sistema projetado para monitorar o tráfego entre hosts em uma LAN. Bro funciona como um monitor de rede passivo de alta velocidade que filtra o tráfego para determinadas aplicações. NFR foi desenvolvido como uma ferramenta flexível para dados de rede cujos atributos incluem sua própria linguagem para criar filtros que são compilados dentro da ferramenta. Netstat é um IDS de rede que fornece personalização dos coletores de eventos de rede.

Sistemas que monitoram ambos hosts e redes incluem Emerald (PORRAS et al., 1997; NEUMANN et al., 1999), Grids e Dids. Emerald, foi concebido para detectar intrusões em amplas redes distribuídas. É um sistema hierárquico capaz de responder a ameaças sobre alvos locais e coordenar seus monitores para formar uma hierarquia de análise para ameaças à rede como um todo. Grids acumula os resultados de host e rede com base em componentes, os quais são exibidos em um gráfico. Gráficos permitem facilmente visualizar ataques que poderiam comprometer a rede. O IDS Dids é uma extensão do

NSM e utiliza dados de auditoria de sistemas em hosts e tráfego da LAN para detectar intrusões.

Pesquisas em detecção por anomalia baseada em rede são mais raras do que aquelas que envolvem detecção por anomalia em *hosts*, possivelmente devido ao grande volume de dados de qualidade diversificada a ser trabalhado nas pesquisas, mas encontra-se em crescimento no meio científico. Lógica Fuzzy, Algoritmos Genéticos, teoria dos conjuntos aproximativos (*Rough Set*), Redes Neurais e Máquinas de Vetor de Suporte são técnicas de Inteligência Artificial que estão sendo mais recentemente aplicadas na construção de IDSs baseados em anomalia no tráfego de rede.

A seguir, são apresentados alguns sistemas relevantes encontrados na literatura que são capazes de detectar ataques utilizando dados do tráfego de redes. Alguns destes sistemas realizam análise complementar a partir de dados de auditoria de hosts e de sistemas.

4.3.1 EMERALD

EMERALD (*Event Monitoring Enabling Responses to Anômalous Live Disturbances*), desenvolvido por Porras and Newmann (1997), é uma ferramenta distribuída e escalável, cuja finalidade é identificar atividades mal intencionadas em redes de grande porte. Implementa os métodos de detecção por anomalia e por assinatura. É o sistema mais recente na linha de IDSs desenvolvidos pelo *Stanford Research Institute* (SRI).

Utiliza o conceito de organização hierárquica, com distribuição de tarefas e modularidade, para alcançar uma solução escalar. Utiliza monitores de sondagem e resposta distribuídos, ajustáveis de forma independente, que são empregados em várias camadas abstratas na rede. Os monitores do EMERALD constituem um conjunto de unidades interoperáveis de análise e resposta que provêm proteção localizada de alvos específicos em uma rede.

Os monitores provêm análise dinâmica, localizada e em tempo real de infraestrutura (por exemplo, roteadores, gateways ou firewalls) e serviços (subsistemas privilegiados com interfaces de rede e serviços tais como FTP, SMPT, HTTP). Também interagem passivamente, lendo os logs de atividades ou pacotes de rede, ou ativamente, via sondagem que adiciona a coleta de eventos normais. Produzem resultados analíticos e os dissemina assincronamente aos outros monitores clientes do EMERALD. Possuem interface bem definida para compartilhar e receber eventos de dados e resultados e realizam detecção por anomalia baseada em perfil estatístico e análise de assinaturas.

Os monitores do EMERALD são: *service monitor*, *domain monitors* e *enterprise-level monitoring*. Cada *service monitor*, o componente da camada mais baixa, tem a função de inspecionar a operação de um serviço de rede. *Domain monitors* coletam dados do monitor de serviços e correlaciona-os para formar um quadro coerente da situação de segurança em um domínio limitado, tal como um único departamento. Neste nível de abstração, estes são bem posicionados para alertar tarefas e registrá-las. *Enterprise-level monitoring* usa os resultados das análises de domínio para formar uma visibilidade global da rede. Tal visibilidade é útil para identificar problemas propagados e extensos, por exemplo vírus ou infecções por vermes.

O monitor foi projetado para ocupar pouco espaço, ser bem rápido e genérico de modo a ser empregado em qualquer camada no esquema hierárquico de análise do EMERALD. O monitor opera como um detector de intrusão descentralizado, que consiste de três unidades computacionais básicas de análise, conforme apresentado na Figura 4.6: um processador baseado em assinatura (*Signature Engines*), um processador de perfil estatístico (*Profiler Engines*), e uma unidade de medida defensiva denominada *Resolver*.

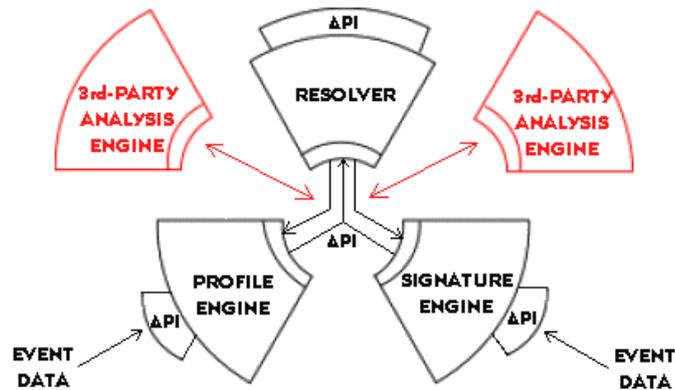


Figura 4.6 - A arquitetura do monitor EMERALD

Profiler Engines realiza detecção de anomalias, baseada em perfil estatístico dado um fluxo de eventos generalizado de um alvo de análise. *Signature Engines* requer gerenciamento de estado mínimo e emprega um esquema de codificação de regras para prover um modelo de análise de assinaturas distribuído.

O subsistema de análise estatística de assinaturas emprega uma variação do P-BEST (*Production-Based Expert System Tool*). Utiliza um conjunto de regras para detectar atividades indesejadas e conhecidas ocorrendo no alvo em análise. A busca por estas atividades ocorre através de quatro tipos de medidas estatísticas: categóricas (por exemplo, tipos discretos), contínuas (por exemplo, quantidades numéricas), intensidade do tráfego (por exemplo, volume sobre o tempo) e distribuição de eventos (por exemplo, uma meta-medida obtida de outras medidas).

Resolver é o coordenador do sistema de notificação externa dos monitores e implementa a política de resposta.

Os monitores são computacionalmente independentes, provendo, portanto, um grau de paralelismo na cobertura das análises, enquanto auxilia na distribuição da carga computacional e utilização de espaço em disco. Através do uso de monitores locais nos alvos em monitoração, o EMERALD ajuda a reduzir os possíveis atrasos na análise e resposta que possam ocorrer ao longo da rede.

Também insere uma abordagem de análise hierárquica, onde as análises locais são compartilhadas e correlacionadas em camadas de abstração mais altas. O processo de análise é iniciado a partir da camada de interface de rede, de domínios administrativos individuais. Os monitores são empregados por todo o domínio e em cada um dos domínios, para analisar a operação dos serviços de rede e outros componentes externamente acessíveis. Cada monitor contém um conjunto específico de manipuladores de respostas, que são chamados quando detectada uma possível assinatura. Estes monitores da camada de serviço (*Service Layer*) também disseminam suas análises para outros monitores do EMERALD, que realizam a correlação para o domínio.

O EMERALD implementa uma análise para correlacionar os relatórios de atividades produzidos através do conjunto de domínios monitorados. Este sistema representa um grande avanço, em relação às pesquisas anteriores e o desenvolvimento de detecção por abuso (assinatura) e anomalia para tratar o monitoramento de grandes sistemas distribuídos e redes. Devido à capacidade de análise em tempo real de modo distribuído e aplicada onde for mais eficaz, em diferentes camadas de abstração, o EMERALD apresenta vantagens significativas sobre as abordagens centralizadas, em termos da capacidade de detecção e resposta a eventos, detectando não apenas ataques locais, mas também ataques coordenados como negação de serviço distribuído ou padrões de ataque repetidos contra vários domínios.

4.3.2 Exemplo de Aplicação de Lógica Fuzzy

A Lógica *Fuzzy* (ou Lógica Nebulosa ou Teoria de Conjuntos *Fuzzy*) é baseada na noção de que objetos pertencem a conjuntos e, diferente da teoria de conjuntos tradicional, os limites entre os conjuntos são graduais ou *fuzzy* (SANDRI et al., 1999). Segundo RAHMAN et al (2006), a teoria de conjuntos *fuzzy* adota a noção de grau de pertinência a um conjunto (um número real entre 0 e 1, inclusive) para tratar de imprecisão ou incerteza.

O real benefício da Lógica *Fuzzy* é prover um modo de desenvolver uma solução aproximada com custo efetivo para problemas complexos explorando a tolerância de imprecisão. Também permite tratar o gargalo na aquisição do conhecimento associado à construção dos sistemas especialistas. Quando construindo uma base de regras *fuzzy*, o engenheiro do conhecimento pode permitir que o especialista humano descreva o problema em termos semânticos, tal como “muito rápido”, “pouco rápido” e “não rápido”.

Muitas características quantitativas, ambas comuns e categóricas, estão envolvidas no campo de detecção de intrusão e podem potencialmente ser vistas como variáveis *fuzzy*. Por exemplo, o tempo de uso da CPU e a duração da conexão são dois exemplos de medidas ordinárias. Um exemplo é o número de serviços TCP/UDP diferentes inicializados pelo mesmo *host* de origem. O segundo motivo é que segurança envolve incertezas. Dada uma medida quantitativa, um domínio de valores ou um intervalo pode ser usado para denotar um valor normal. Então, quaisquer valores obtidos fora do intervalo serão considerados anômalos para o mesmo grau, não considerando suas diferentes distâncias do intervalo. O mesmo aplica-se para valores dentro do intervalo, ou seja, todos serão vistos como normal para o mesmo grau. Infelizmente, isto provoca uma separação abrupta entre normalidade e anormalidade. A introdução de nebulosidade às características quantitativas ajuda a suavizar esta separação abrupta.

A proposta do trabalho realizado por Luo e Bridgest (LUO et al., 2000) integra lógica *fuzzy* com regras de associação e episódios de frequência para ampliar o desempenho de um IDS, através de aproveitamento e aperfeiçoamento de técnicas desenvolvidas por diferentes autores, envolvendo:

- extensão do trabalho de Lee, Stolfo e Mok nas áreas de uso de regras de associação e frequência de episódios para detecção de anomalias, introduzindo lógica *fuzzy*;

- Adição de um passo de normalização ao procedimento de minerar as regras de associação *fuzzy* do trabalho de Kuok, Fu e Wong, de modo a evitar que uma instância de dados venha a contribuir mais que as outras;
- Modificação do procedimento de Mannila e Toivonen, para minerar frequência de episódios, de modo a extrair padrões para medidas estatísticas temporais em nível mais alto que o nível de dados;
- Criação de uma função de avaliação de similaridade que é contínua e monotônica para a aplicação de regras de associação *fuzzy* e frequência de episódios *fuzzy* em detecção de anomalias.

4.3.3 FIDS

O sistema FIDS (*Fuzzy Intrusion Detection System*) é um sistema baseado em regras *fuzzy*, cuja finalidade é detectar tráfego de rede intrusivo e informar o administrador do sistema sobre os ataques (TILLAPART et al., 2002). A arquitetura do FIDS é apresentada na Figura 4.7.

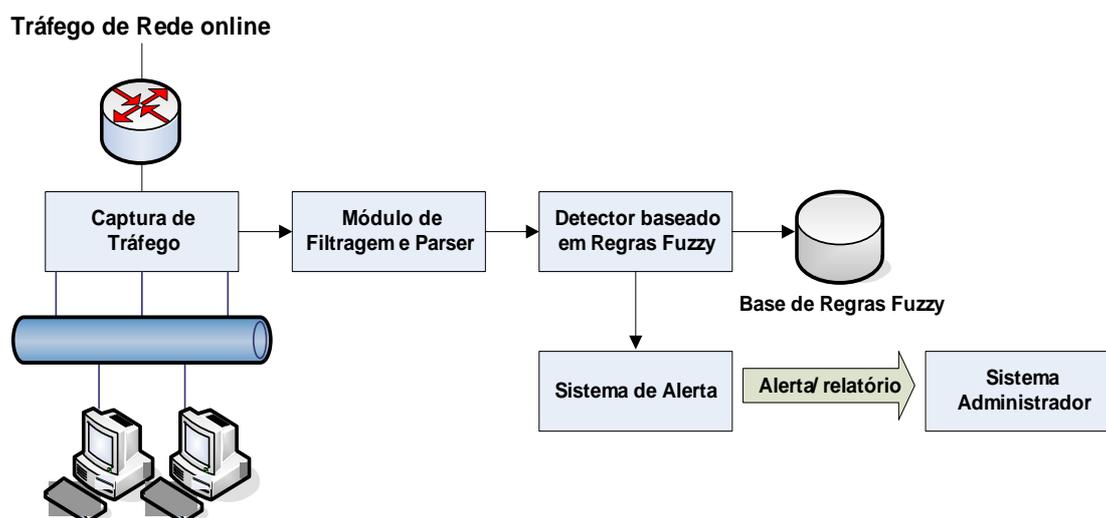


Figura 4.7 - Framework do FIDS

Ao invés de utilizar valor *crisp* ou exato (detecção baseada em limiar) para distinguir entre tráfego de rede normal e anormal é utilizada nesta pesquisa

sistema baseado em regras *fuzzy*. Conseqüentemente, determinada quantidade de tráfego anormal, que está entre normal e ataque, pode ser considerada como anormal (com baixo grau de ser um ataque).

A arquitetura FIDS compreende três principais componentes: módulo de filtragem e parsing - FPM (Filter and Parser Module), detector baseado em regras fuzzy - FDs (*Fuzzy Rule Based Detectors*) e sistema de alerta - WS (*Warning System*).

FPM filtra os pacotes capturados e coletados de acordo com as assinaturas de ataques pré-definidas, tais como:

Assinatura Syn-flood:

flag = S, dst_host = victim (same), dst_service = vulnerable port (same)

Assinatura Ping-of-death:

src_host = victim (same), fragment_identification = same

Port Scanning Signature:

(flag = S, src_host = attacking machine, dst_service = vulnerable port) =>

(flag = R, dest_host = attacking machine, src_service = dst_vulnerable port)

Conseqüentemente, para quaisquer pacotes que casam com assinaturas de ataques pré-definidas, o FPM contabiliza a freqüência de ocorrências dentro de cada segundo e envia estes números aos FDs correspondentes. O módulo FD analisa o grau de severidade de ataque do tráfego filtrado. Se ataques são detectados, o sistema de alerta WS exibe informação dos ataques detectados e cria um relatório de ataque ao administrador.

Os componentes FDs são o núcleo do FIDS e constituem sete detectores: Syn-flood detector, Udp-flood detector, Ping-of-death detector, E-mail bomb detector, FTP password guessing detector, Telnet password guessing detector,

Port scanning detector. Cada detector é usado para identificar diferentes tipos de ataques.

Em geral, estes detectores compreendem dois conjuntos de regras fuzzy, LEVEL BOX e DETECTOR BOX. Mais detalhes sobre o processamento destes é encontrado em (TILLAPART et al., 2002). Para configurar as regras fuzzy do LEVEL BOX são utilizadas regras heurísticas configuradas de acordo com o seguinte conhecimento especialista:

1. Se a frequência do tráfego é *baixa* então o nível é “0.”
2. Se a frequência do tráfego é *média* então o nível é “1.”
3. Se a frequência do tráfego é *alta* então o nível é “2.”
4. Se a frequência do tráfego é *muito alta* então o nível é “3.”
5. Se a frequência do tráfego é *extremamente alta* então o nível é “4.”

Para configurar as regras *fuzzy* do DETECTOR BOX, as regras heurísticas são ativadas, baseadas no conhecimento de especialistas. Estas regras são baseadas em duas variáveis, o número do nível de tráfego no segundo corrente e a quantidade de tráfego durante os segundos anteriores. Um exemplo de regras heurísticas do Detector box é dado por:

- Se o nível corrente do tráfego é *muito alto* e os níveis dos tráfegos passados são também *muito altos* então este evento é considerado *ataque severo*.
- Se o nível corrente do tráfego é *muito alto* e os níveis dos tráfegos passados são *baixos* então este evento é considerado *anômalo*.

O uso do sistema baseado em regras permite ao FIDS ser capaz de detectar tráfego intrusivo de modo mais flexível que aqueles que utilizam detecção baseada em limiar, pois, ao invés de utilizar limite brusco na decisão entre tráfego normal e intrusivo, o FIDS considera ambos nível de tráfego corrente e

o número acumulativo ponderado, número de tráfego intrusivo durante os segundos passados.

Conforme TILLAPART et al. (2002). as regras de detecção do FIDS são flexíveis e podem ser aplicadas a outros ambientes de rede. As funções de pertinência das variáveis deste sistema são ajustadas com base nos dados normais e anômalos da rede específica, portanto, para aplicar este framework em outra rede, conjuntos de dados normais e anômalos do novo ambiente de rede devem ser coletados, minerados e estudados para determinar o padrão do tráfego.

4.3.4 MINDS

A Universidade de Minnesota tem desenvolvido um sistema denominado *Minnesota Intrusion Detection System* (MINDS) que utiliza a técnica de detecção de *outliers* para identificar intrusões em rede (DOKAS et al., 2002; ERTOZ et al., 2003a; ERTOZ et al., 2003b; ERTOZ et al., 2003c; LAZAREVIC et al., 2003). A Figura 4.8 ilustra a arquitetura deste sistema.

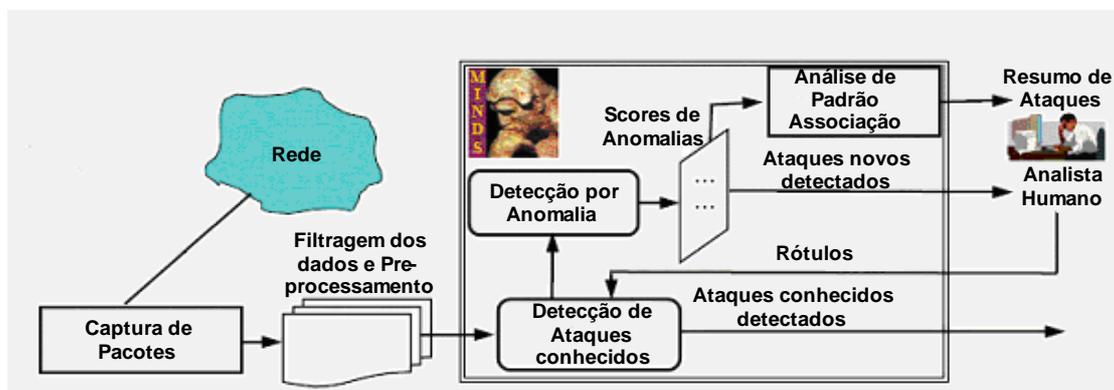


Figura 4.8 - Arquitetura do MINDS

Os módulos que compõem este projeto são: filtragem, pré-processamento e módulo de detecção de ataques conhecidos; detector de varreduras; algoritmos de detecção de anomalias e sumarização de ataques utilizando análise de padrões de associação.

A seguir são descritas as etapas de processamento do MINDS:

Etapa 1: Nesta etapa, ferramentas de monitoração de rede, tais como o *Netflow* (2004), são usadas para coletar os dados do tráfego de rede dos roteadores CISCO ou são obtidos dados *tcpdump* brutos do tráfego de rede através da ferramenta *tcptrace*. O sistema é executado em modo *batch* para que o analista possa ter tempo de analisar o tráfego. A execução do sistema em um microcomputador simples, para uma janela de tráfego de 10 min (aproximadamente 2 milhões de conexões) dura menos de 3 minutos.

Etapa 2: Os dados são filtrados para remover conexões irrelevantes para análise (tráfego de fontes confiáveis ou comportamento de rede não usual ou anômalo que é sabidamente livre de intrusão) e pré-processados para coletar características básicas (tais como: endereços IP origem e destino, portas origem e destino, protocolo, flags, número de bytes e número de pacotes) e para extrair características derivadas (tais como: número de fluxos para endereços IP com destino único e provenientes da mesma origem dentro da rede nos últimos T segundos) (ERTOZ et al., 2003c). Os dados criados são introduzidos no MINDS.

Etapa 3: Nesta etapa, conexões de rede para os quais se tem assinaturas disponíveis são detectadas através do Módulo de Detecção de Ataques Conhecidos. As conexões restantes, que não possuem assinaturas de ataques, são inseridas no módulo de detecção de anomalias. São utilizados diferentes algoritmos de classificação desenvolvidos especialmente para aprender a partir de classes raras, ou seja, sistemas baseados em regras (PN rule, CREDOS), métodos *ensemble-based* (Rare-Boost, SMOTEBoost), e modificação do algoritmo de classificação baseado em associação (ERTOZ et al., 2003a).

Etapa 4: Esta etapa consiste na detecção de anomalias através da técnica de detecção de *outliers* baseado em densidade. Este método atribui a cada ponto de dado um grau equivalente a “ser estranho”, que é denominado *Local Outlier Factor* (LOF). O fator LOF é local no sentido de que apenas a vizinhança restrita a cada ponto é considerada. Para cada ponto de dado, a densidade de sua vizinhança é calculada. O valor LOF do ponto de dado específico p

representa a média da razão entre a densidade de seus vizinhos mais próximos e a densidade do próprio ponto.

A tarefa final deste módulo é atribuir um *score* indicando o quão anômala é a conexão, comparada ao tráfego normal da rede.

Diferentes algoritmos de detecção de *outliers* bem como algoritmo de Máquina de Vetor de Suporte (SVM), modificada para treinamento não supervisionado foram utilizados para testes deste módulo (LAZAREVIC et al., 2003).

Etapa 5: As conexões que são classificadas com alto grau de anomalia são observadas pelos analistas de segurança de rede da Universidade de Minnesota a fim de determinar se são intrusões de fato, novo comportamento normal ou apenas alarmes falsos.

Etapa 6: As conexões classificadas como alto grau de anomalia são posteriormente analisadas e resumidas pelo Módulo de Análise do Padrão de Associação utilizando a técnica de associação de padrões. Regras de associação produzem uma caracterização concisa das anomalias detectadas e são úteis na criação de novas assinaturas e modelos para novos ataques. Após analisar os resumos criados, o analista decide se estes resumos são úteis na criação de novas regras para uso no módulo de detecção de ataques conhecidos.

Um exemplo de regra de associação é apresentado na Figura 4.9, onde 10 conexões são resumidas em um *item set* freqüente. Os parâmetros da conexão que não provêm informação significativa são eliminados, reduzindo assim a sobrecarga de dados ao analista de segurança (ERTOZ et al., 2003c).

IP origem	Tempo inicial	IP destino	Porta dest.	No. Bytes
x.y.z.95	11.07.20	A.B.C.223	139	192
x.y.z.95	11.13.56	A.B.C.219	139	195
x.y.z.95	11.14.29	A.B.C.217	139	180
x.y.z.95	11.14.30	A.B.C.255	139	199
x.y.z.95	11.14.32	A.B.C.254	139	186
x.y.z.95	11.14.35	A.B.C.253	139	177
x.y.z.95	11.14.36	A.B.C.252	139	172
x.y.z.95	11.14.38	A.B.C.251	139	192
x.y.z.95	11.14.41	A.B.C.250	139	195
x.y.z.95	11.14.44	A.B.C.249	139	163

Sumarização

{SrcIP=X.Y.Z.95, DestPort=139}

Figura 4.9 - Regra de Associação descrevendo atividade de varredura na porta 139

4.3.5 Exemplo de Aplicação de Rede MLP

Silva et al (2004) desenvolveram uma aplicação para detecção de ataques a redes, através de redes neurais. Primeiramente, utilizaram a toolbox de redes neurais do software matlab para treinamento e testes de algumas redes. Após alguns testes, decidiram pelo uso da rede MLP com 2 camadas ocultas e 10 neurônios em cada camada oculta. Utilizou-se a função tangente hiperbólica para ativação dos neurônios com valores no intervalo entre -1 e 1. Utilizaram nos testes 41 atributos fornecidos pelo KDDCup1999 (LEE, 1998) e quatro redes neurais para os testes.

Os atributos básicos considerados no trabalho de Silva et al (2004), são os mesmos considerados por Mukkamala (2002a) incluindo: duração da conexão (em segundos), tipo de protocolo de transporte usado na comunicação (TCP, UDP...), tipo de serviço (http, FTP, telnet...), número de bytes enviados da origem para destino, número de bytes enviados do destino para origem, status da conexão (normal ou erro), *land* (1 se a conexão é de/para o mesmo host, 0 caso contrário), número de fragmentos errados e número de pacotes com flag URG ativada. Os principais tipos de ataques considerados nos testes foram DoS e Probing.

Para os cenários avaliados foi possível comprovar (SILVA et al., 2004) que a capacidade de generalização das redes neurais permite a identificação de padrões de ataques desconhecidos. A utilização de um conjunto de redes

neurais, com várias redes, cada uma especializada em detectar uma classe de ataques ou até mesmo especializada em categorias específicas das variáveis de entrada, pode aumentar a precisão da resposta do sistema, simultaneamente diminuindo a quantidade de falsos-positivo.

4.3.6 ACME!

O ACME! é um IDS baseado em abuso caracterizado pela operação de um sistema de captura de pacotes em conjunto com uma rede neural (CANSIAN, 1997; BONIFACIO et al., 1998; ROSSI et al., 2006). A rede neural verifica padrões de comportamento (previamente codificados em assinaturas) e devolve um percentual mostrando o qual similar um padrão de entrada é de algum ataque conhecido. A partir desse ponto o administrador analisa se o padrão de entrada é ou não um ataque.

Uma base de dados de ataques é freqüentemente atualizada no ACME! e os ataques são simulados para obtenção de novas assinaturas.

O modelo de detecção de intrusão ACME! é formado por um sistema de captura e tratamento de pacotes, um sistema de rede neural, e um gerenciador de comunicações e interface com o usuário, conforme mostra a Figura 4.10.

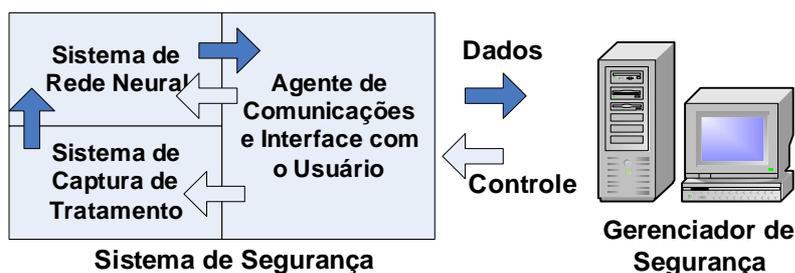


Figura 4.10 - Estrutura Geral do ACME!
Fonte: CANSIAN (1997)

O sistema de captura e tratamento de pacotes é organizado em módulos, que tratam o fluxo de pacotes e que têm sua atuação concluída ao fornecer o vetor de estímulo para a rede neural. O nível mais baixo apenas captura um fluxo de

dados na rede e passa os pacotes ordenados para o módulo de conexão, sob controle do módulo de pré-seleção, conforme mostra a Figura 4.11.

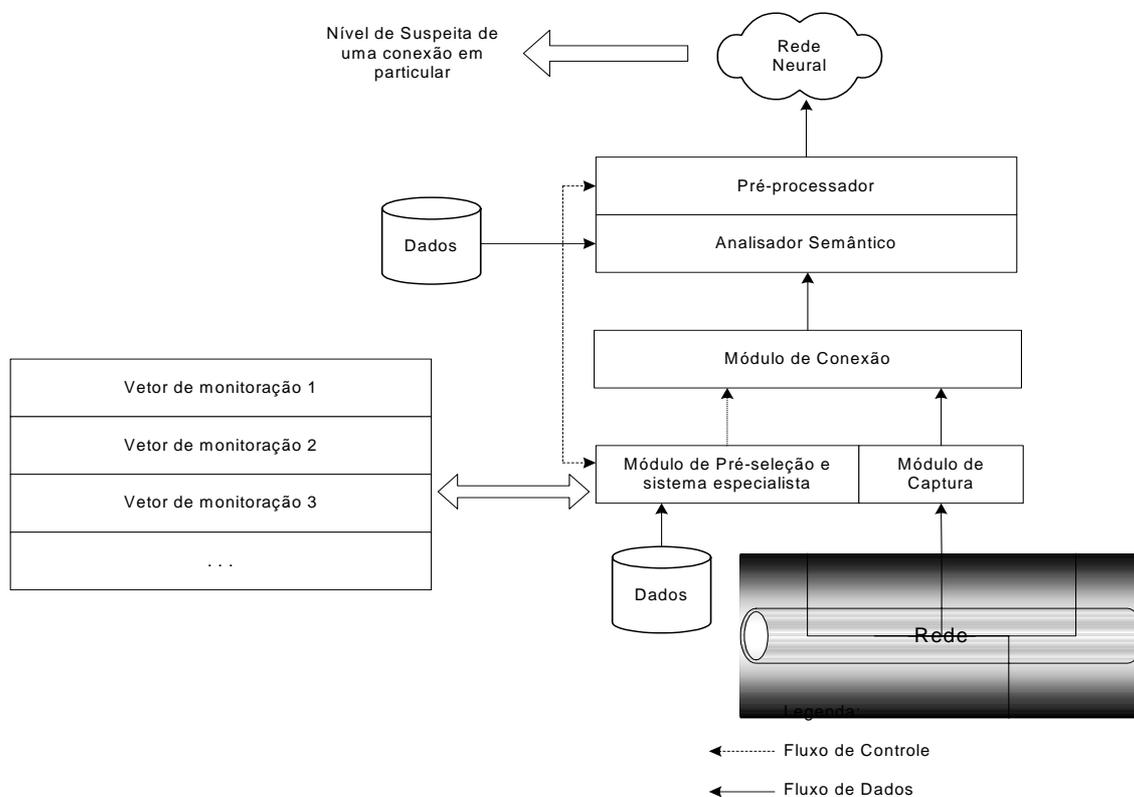


Figura 4.11 - Estrutura dos módulos do ACME!.
Fonte: CANSIAN (1997)

Os módulos existentes no ACME! são (CANSIAN, 1997; BONIFACIO et al., 1998) :

- Módulo de Captura de Pacotes (CAP): a função básica deste módulo é identificar e coletar os pacotes, enviando-os para os módulos de pré-seleção e sistema especialista e para o módulo de conexão. É utilizada a biblioteca de funções *libpcap* (*Packet Capture Library*) para a comunicação com o dispositivo de rede.
- Módulo de Pré-Seleção e Sistema Especialista (PSSE): este módulo decide a partir de quando uma conexão é considerada suspeita. Ele é o responsável por mais um nível de filtragem, pois o Módulo de Conexão só

começa a registrar as atividades de máquinas ou domínios considerados suspeitos que são indicados pelo PSSE.

- **Módulo de Conexão (CON):** a finalidade deste módulo é a criação e manutenção de vetores de conexão. Vetor de conexão é um arquivo contendo a reconstrução do fluxo de dados. A partir destes dados, o analisador semântico faz a busca por trechos suspeitos que, combinados, caracterizem um ataque. A manutenção desse vetor é feita em tempo real.
- **Analisador Semântico e Pré-Processador (ASPP):** a partir do vetor de conexão, é criado o vetor de estímulo, que é a codificação binária do vetor de conexão. Com os dados condensados pelo módulo de conexão, o sistema é capaz interpretar e selecionar aquilo que for relevante. Nesta fase, ele realiza uma tradução, ou ainda, uma codificação de linguagens, possibilitando a conexão ao módulo de rede neural. O ASPP varre todos os vetores de conexão através de um autômato de pilha, analisando os dois sentidos de fluxo, com o auxílio de uma base de dados constituída por duas Tabelas de conversão. A primeira Tabela associa cada seqüência de caracteres relevante a um número inteiro, enquanto a segunda associa cada número inteiro válido a um código binário reconhecível pela rede neural.
- **Módulo de Rede Neural (MRN):** a interface com a rede neural é o elemento principal deste módulo, o qual recebe todo o conjunto de bits que compõe o vetor de estímulo e, baseado no treinamento ao qual a rede foi submetida, retorna uma porcentagem que indica o quanto a sessão é suspeita. A rede neural é treinada com o aplicativo simulador de redes neurais, denominado *Stuttgart Neural Network Simulator* (SNNS) Este simulador reduz sensivelmente o tempo necessário para a criação do sistema da rede neural. Para treinamento, a rede é exposta a um conjunto grande e variado de vetores de estímulo, os quais representam sessões suspeitas, intrusivas e ilegítimas.

4.3.7 Exemplo de Aplicação da Rede SOM

Chaves (2005a, 2005b) desenvolveu um sistema de detecção de *backdoors* e canais dissimulados baseado na análise das sessões TCP/IP do tráfego de uma rede. A metodologia consiste em três fases: a reconstrução das sessões TCP/IP, a análise e classificação, e a geração do resultado.

O princípio básico para detecção de um *backdoor* é encontrar características que indiquem a atividade de interesse, dentre estas as assinaturas nos dados, o tamanho e a taxa de transmissão dos pacotes e o intervalo de tempo entre os pacotes de uma dada sessão (2005b).

Neste trabalho foi utilizado um mapa SOM bidimensional, onde cada neurônio é representado como um vetor de nove dimensões, formado pelos valores normalizados de atributos das sessões.

Para treinamento da SOM um tráfego legítimo (sem nenhum pacote ou sessão maliciosa) foi utilizado.

O treinamento da rede SOM requer a inicialização de valores aleatórios entre 0 e 1. Em seguida, para cada vetor (instância) das sessões, um algoritmo competitivo é utilizado para encontrar o nó vencedor na rede (aquele que possui menor distância euclidiana em relação a instância avaliada). Ao ser encontrado, o nó vencedor e seus vizinhos dentro de um certo raio ou vizinhança atualizam seus pesos (através de uma função de aprendizagem) para representar a classe do padrão de entrada. Após treinamento, a SOM pode ser utilizada para o reconhecimento de padrões (fase de ativação), ou seja, para detecção de sessões anômalas.

Na fase de ativação do SOM, para cada sessão extraída pelo Recon, é criado um vetor (instância) contendo as nove características utilizadas. Esse vetor é normalizado e busca-se pelo nó vencedor (com menor distância euclidiana) .

A sessão é classificada como normal, se estiver muito próxima do nó vencedor, e anômala, se sua distância ao nó vencedor for maior que um limiar pré-

definido. Neste trabalho foi utilizado o valor 2 para o limiar. Este limiar foi definido através de tentativa e erro, até que o número de falso-positivos e de falso-negativos fosse mínimo.

Após a detecção por anomalia, realiza-se detecção por abuso, onde o conteúdo das sessões classificadas como anômalas são analisados em busca de assinaturas de ataques conhecidos. O mecanismo de detecção utiliza regras baseadas nas utilizadas pelo sistema de detecção de intrusão *Snort*.

Neste trabalho, a utilização de mapas auto-organizáveis (SOMs) como mecanismo de detecção por anomalia mostrou-se eficiente para detecção de *backdoors* e canais dissimulados.

4.3.8 HIDE

Hierarchical Network Intrusion Detection System (HIDE) é um sistema de detecção por anomalia que utiliza pre-processamento estatístico e classificação por redes neurais para detectar ataques em redes. É multicamadas, distribuído e hierárquico, e monitora vários parâmetros do tráfego de rede simultaneamente. Opera de modo automático, adaptativo e proativo e pode ser aplicado em varias tecnologias de rede, incluindo redes cabeadas e *wireless* (ZHANG et al., 2001). O método utiliza modelos estatísticos e classificadores multivariados para detectar condições anômalas de rede.

Os tipos de dados de entrada e saída do HIDE são mostrados na Figura 4.12 a seguir.

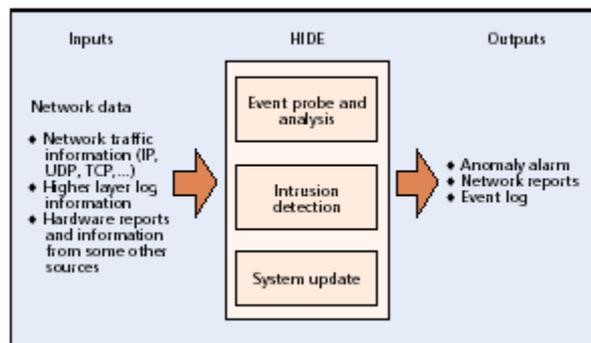


Figura 4.12 - Entradas e Saídas do Hide

Este sistema obtém dados do tráfego de rede, logs de sistemas e relatórios de hardware, estatisticamente processa e analisa as informações, detecta padrões de atividades anormais com base em modelos de referência, os quais correspondem a atividades esperadas de usuários, para cada parâmetro individualmente ou em grupos combinados utilizando classificação através de redes neurais, gera alarmes e notificações de eventos e atualiza os perfis do sistema com base nos padrões de rede recentemente observados e na saída do sistema.

Cada camada do HIDE contém vários agentes de detecção de intrusão denominados Intrusion Detection Agents (IDAs) que monitoram as atividades de um host ou de uma rede. Diferentes camadas correspondem a diferentes escopos de rede que são monitorados pelos agentes que neles atuam.

Cada IDA contém os seguintes componentes: sensor (*probe*), pre-processador de eventos (*event processor*), processador estatístico (*statistical processor*), classificador neural (*neural network classifier*) e o pos-processador (*post processor*), conforme apresentado na Figura 4.13.

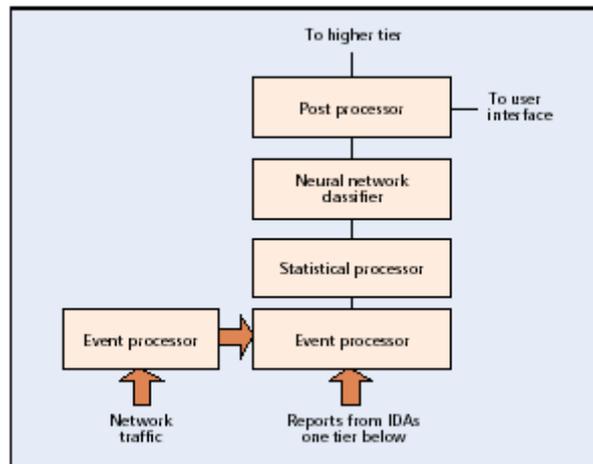


Figura 4.13 - Um agente de detecção de intrusão (IDA)

Uma breve descrição da finalidade dos componentes do IDA é apresentada a seguir:

- *Probe*: coleta o tráfego de rede de um host ou da rede, abstrai o tráfego em um conjunto de variáveis estatísticas para refletir o status da rede e periodicamente gera notificações ao *event preprocessor*.
- *Event pre-processor*: recebe notificações de ambos *probe* e IDAs de camadas mais baixas, e converte a informação em um formato requerido pelo modelo estatístico.
- *Statistical Processor*: mantém um modelo de referência das atividades típicas de rede, compara os relatórios do *event preprocessor* aos modelos de referência, e forma um vetor de estímulo para apresentar à rede neural classificadora.
- *Neural Network Classifier*: analisa o vetor de estímulo do modelo estatístico para decidir se o tráfego da rede é normal ou não.
- *Post Processor*: gera notificações aos agentes das camadas mais altas. Ao mesmo tempo, é capaz de mostrar os resultados através de uma interface com o usuário.

Redes neurais classificadoras são utilizadas no HIDE. O vetor de estímulo, com valores de similaridades calculados, gerado pelo processador estatístico é introduzido no classificador neural para uma decisão. São utilizadas redes neurais pequenas, com o mínimo de entradas, poucos neurônios computacionais e menor número de pesos correspondentes (ZHANG et al., 2001).

O grupo de desenvolvimento do HIDE estudou o desempenho de classificação de cinco tipos de redes neurais: uma *Perceptron*, uma *Perceptron-Backpropagation-Hybrid*, (PBH) que é a superposição de um perceptron e uma pequena rede *Backpropagation*, uma *Backpropagation* (BP), *Radial-Based Function* (RBF) e Fuzzy-ARTMAP. Os resultados experimentais mostraram que as redes BP e PBH realizaram melhor classificação. Sendo assim, passaram a usar uma backpropagation com dois neurônios ocultos nos desenvolvimentos posteriores, que se mostrou muito eficiente e apresentou muito boa capacidade de classificação.

4.3.9 Exemplo de Aplicação da Teoria *Rough Set*

Como a maioria dos IDSs atuais é construída através da codificação manual do conhecimento de especialistas e o tempo de atualização da base de conhecimento é muito grande, o trabalho desenvolvido por Li et al (2004) apresenta uma alternativa para solucionar este problema. A proposta deste trabalho em desenvolvimento é a construção de um modelo de detecção capaz de identificar intrusões conhecidas, suas variações e novos ataques de natureza desconhecida. O método baseia-se na Teoria *Rough Set* (Teoria dos Conjuntos Aproximativos) e é capaz de extrair um conjunto de regras de detecção a partir dos valores de atributos dos pacotes de rede. Após obter uma Tabela de decisão a partir do pre-processamento dos dados brutos dos pacotes, são aplicados processo de redução baseado em RST e algoritmos de geração de regras úteis para detecção de intrusão. Além disto, um algoritmo de aquisição de conhecimento incremental baseado em árvore de regras e conjuntos aproximativos é apresentado como solução para atualização do

conjunto de regras quando novos ataques são detectados. A idéia no uso deste método é o uso de conjuntos de dados de treinamento de menor tamanho e menos esforço na coleta dos dados de treinamento.

Os seguintes módulos compõem o sistema: Sensor, Decodificador de Protocolos, Módulo de Geração de Regras, Módulo de Detecção de Intrusão, Módulo de Alarme e Resposta e Administrador, como mostra a Figura 4.14.

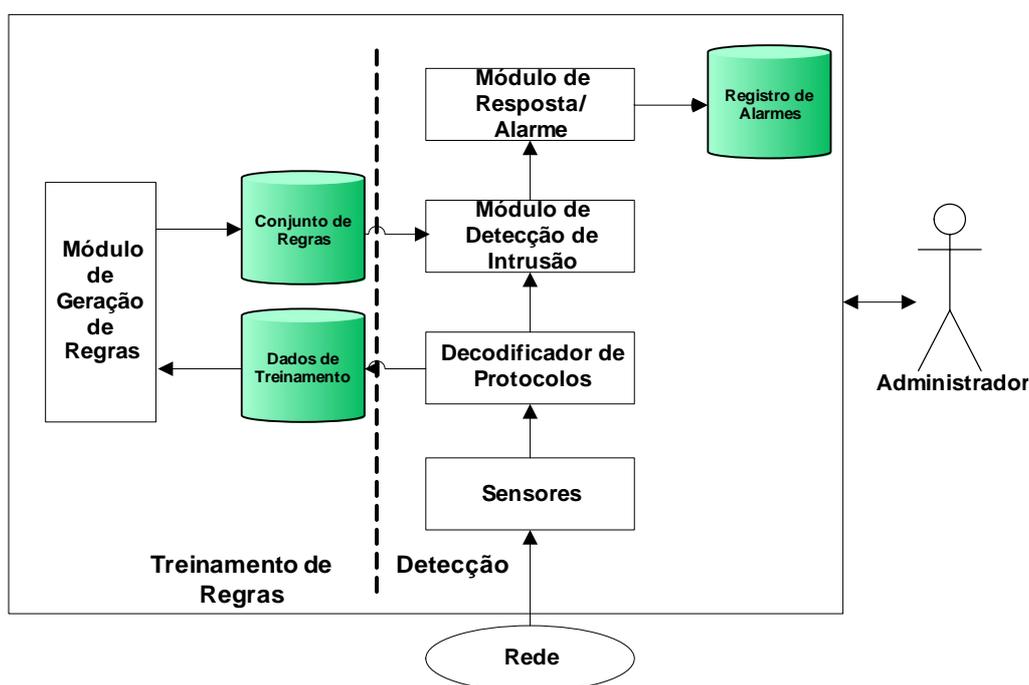


Figura 4.14 - arquitetura do IDS

Os sensores coletam os dados do tráfego de rede. O decodificador de protocolos remonta dos dados brutos coletados pelos sensores de acordo com seu protocolo e converte os dados brutos e dados de conexões em formato que o módulo de geração de regras entende. O módulo de geração de regras utiliza RST para gerar regras a partir dos dados de treinamento. As regras são modeladas em formato de árvore de regras e esta árvore é armazenada no banco de dados de regras. O módulo de detecção de intrusão examina as árvores de regras e compara os dados que chegam com as regras e transfere os resultados para o módulo de alarme/resposta. O módulo de resposta notifica o

administrador os resultados de detecção através de alertas, entradas em arquivos de log ou ferramenta de visualização.

Os atributos básicos utilizados no desenvolvimento deste sistema são: endereço IP de origem e destino, portas de origem e destino, data/hora, protocolo de transporte e duração do tráfego, entre outros.

Os atributos básicos não são, em geral, suficientes para identificar atividades maliciosas ou anômalas. Por exemplo, em um ataque de negação de serviço registros de conexão isolados ou individuais não são maliciosos por si próprios, mas chegam em grandes quantidades de modo a sobrecarregar a rede de destino. Portanto, também foram utilizados atributos derivados, tais como: porcentagem de conexões para o mesmo serviço, número de conexões iguais à conexão atual na janela de tempo de N segundos, entre outros. Alguns exemplos dos atributos utilizados são exibidos na Tabela 4.1.

Tabela 4.1- Descrição dos Atributos

Atributos	Descrição
A1	tipo de protocolo, ex: TCP, UDP, ICMP
A2	serviço de rede do destino, ex:HTTP, FTP,etc
A3	duração da conexão
A4	flag no cabeçalho TCP, ex: URG, ACK, SYN, PSH, RST, FIN
A5	número de porta de origem
A6	número de porta de destino
A7	número de conexões feitas pela mesma origem nos últimos N segundos
A8	número de conexões feitas pelo mesmo destino nos últimos N segundos
A9	porcentagem de conexões para o mesmo serviço da mesma origem quando A7 é diferente de zero
...	...
A47	porcentagem de conexões para o mesmo serviço da mesmo destino quando A8 é diferente de zero
D	normal ou determinado tipo de ataque

Um exemplo de Tabela de decisão gerada é apresentada na Tabela 4.2 a seguir. O atributo de decisão, contido na Tabela decisão, é um rotulo de registro, representando sessão normal (valor 0) ou determinado tipo de ataque (valor maior que 0).

Tabela 4.2- Exemplo de Tabela de Decisão

A1	A2	A3	...	A47	D
6	80	1.00		0.00	0
6	0	0.00		0.00	3
17	0	0.00		0.04	0
6	0	1.00		1.00	0
5	23	1.00		1.00	0
7	0	1.00		0.00	19
4	0	0.00		0.00	4
5	21	0.00		1.00	0
2	0	1.00		0.00	12

As metodologias de RST podem ser vistas como um procedimento de redução ou a simplificação de uma Tabela de decisão. Neste trabalho foi adotado o sistema *Rough Set Intelligent Data Analysis System*, (RIDAS) que possui 12 algoritmos de discretização, 10 algoritmos de redução de atributos e 5 algoritmos de redução de valor.

A maioria dos atributos foi discretizada através do algoritmo Naive e todos os valores de atributos foram divididos em vários intervalos de números contínuos. Alguns atributos, tais como tipo de protocolo, número de porta de origem e porta de destino, não são convenientes para discretização, porque seus valores já se encontram discretizados e possuem um significado particular. Portanto, estes atributos são discretizados manualmente, conforme seu significado, por exemplo, o número da porta de destino é discretizada em 2 intervalos [1,1025] e [1025, 65536]. Portas de 1 a 1024 são portas privilegiadas, e algumas conexões maliciosas a estas portas são possíveis ataques.

Em seguida, um algoritmo de redução de atributos (LI et al., 2004) é adotado e alguns atributos, tais como IP origem, IP destino e número de conexões feitas pela mesma origem são removidas da Tabela de decisão.

Finalmente, todas as regras são geradas utilizando um algoritmo de redução de valores. Algumas regras são exemplificadas na Tabela 4.3. $A1_0$ significa que o atributo A1 está no primeiro intervalo de valores discretizados.

Tabela 4.3 - Exemplo de Regras

Regra	Explicação
A1 ₀ e A2 ₀ então D3	se protocolo=IP e IP_origem=IP_destino então é um Ataque "land"
A1 ₆ e A4 ₁ e A6 ₄ e A8 ₀ então D12	se protocolo=TCP e tcpflag=FIN e porta_destino=[1..1024] e número_conexoes para o mesmo destino =[180.5, 254.5] então é um ataque "TCP FIN scan"

4.3.10 SNORT

Snort é um IDS baseado em rede de código livre que realiza casamento de padrões de ataques, portanto, trata-se de um detector de assinaturas. A arquitetura do Snort consiste de 3 partes principais (CASWELL et al., 2003): o decodificador de pacotes (pré-processador), o módulo de detecção e o subsistema de alerta e registro, como apresentado na Figura 4.15.

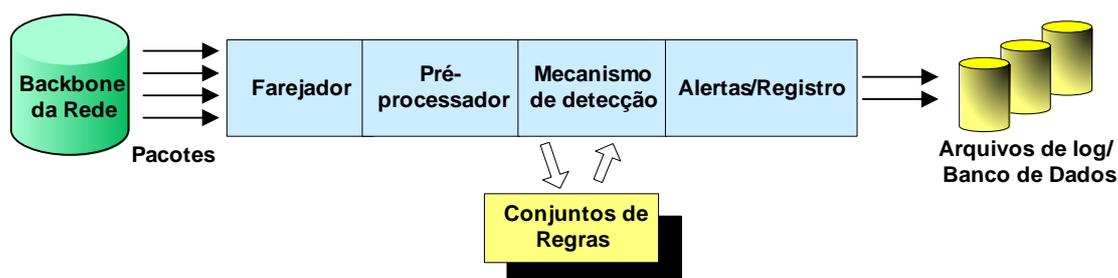


Figura 4.15 - Arquitetura Básica do Sistema Snort
Fonte: Adaptado de CASWELL et al (2003)

O decodificador de pacotes interage com a pilha de protocolos das camadas 2 a 4, principalmente assinalando pacotes para análise subsequente. O mecanismo de detecção inspeciona o pacote em dois estágios. Primeiro, casa o cabeçalho do pacote com uma seqüência de padrões, contendo informação de terminais IP (IP de origem e destino, portas de origem e destino). Cada padrão tem uma cadeia de regras associadas. A cadeia de regras associada com o primeiro par de IPs terminais que casa com o cabeçalho do pacote é ativada. As regras em cada cadeia casa com as seguintes características do pacote: flags TCP, tamanhos do payload e conteúdo corrente. As regras são pesquisadas em ordem e o primeiro casamento dispara a ação especificada na regra. Estas ações são conduzidas pelo subsistema de registro e alerta e compreendem principalmente a gravação de informações sobre o pacote

malicioso (a um nível configurável de granularidade) em um arquivo de log e envio de uma mensagem instantânea para um conjunto de estações de trabalho. A base de conhecimento do Snort necessária para reconhecer os ataques está disponível de modo on-line (SOURCEFIRE, 2007) em forma de regras.

4.3.11 BRO

Bro é um IDS que inspeciona o tráfego de rede para detecção de assinaturas de ataques (PAXSON, 1999a; PAXSON, 1999b; PAXSON, 2003). Assinatura equivale à seqüência bruta de bytes, ou seja, *strings* no *payload* dos pacotes de rede. O Bro tenta casar as *strings* nos pacotes com as assinaturas existentes. Possui um processador de assinaturas flexível, aproveitando os recursos de outros NIDSs e utilizando assinaturas contextuais para reduzir o número de falsos positivos.

A maioria dos NIDS somente trabalha com *strings* fixas como padrão de busca, o Bro utiliza expressões regulares, que apresentam as seguintes vantagens: são mais flexíveis; possuem muita expressividade por prover contexto sintático adicionais e por utilizar classes de caracteres, união e elementos opcionais são muito úteis para especificar assinaturas de ataques; são casadas mais eficientemente; isto é feito compilando as expressões em *Deterministic Finite-state Automaton* (DFAs) cujos estados terminais indicam se um casamento ocorreu. Um DFA determinístico é uma máquina de estado finito determinístico onde o próximo estado é unicamente determinado por um único evento de entrada.

As assinaturas possuem propriedades atrativas tais como: é fácil explicar o que o módulo de casamento de padrões (*pattern matcher*) está procurando e porque, e que tipo de cobertura total ele provê; devido a sua simplicidade, as assinaturas podem ser fáceis de compartilhar e de serem acumuladas em grandes bibliotecas de ataques; para algumas assinaturas, o casamento pode ser bem forte: um casamento indica com alta fidelidade que um ataque ocorreu.

Porém o uso de assinaturas apresenta limitações (PAXSON, 2003): especialmente quando utilizando assinaturas fortes (*tight signatures*), o *pattern matcher* não é capaz de detectar ataques diferentes daqueles para os quais existem assinaturas explícitas; novos ataques serão perdidos, em geral; existem assinaturas fracas (*loose signatures*) que aumentam o problema de falsos-positivos: alertas que de fato não refletem um ataque real.

Bro apresenta uma solução para resolver problemas de falsos-positivos: uso de contexto adicional, incluindo: detalhes adicionais relacionados à atividade exata e sua semântica, de modo a eliminar falsos-positivos provenientes das assinaturas frouxas ou informação adicional de como o sistema atacado respondeu ao ataque: que geralmente indica se o ataque foi bem sucedido.

A idéia é a implementação de um *pattern matcher* com idéia similar ao de um processador de assinaturas tradicional, porém usando um contexto adicional que provê expressões regulares completas ao invés de *strings* fixas e dá ao processador de assinaturas uma noção do estado completo da conexão, permitindo correlacionar múltiplos casamentos interdependentes em ambas as direções de uma sessão de usuário.

Uma linguagem personalizada foi criada para a definição de assinaturas, permitindo integrar os conceitos utilizados: expressões regulares e contexto.

Como as assinaturas do Snort são fáceis de entender, gratuitas e freqüentemente atualizadas, os desenvolvedores do Bro as utilizam, convertendo-as na linguagem particular de assinaturas desenvolvida, utilizando um *script* escrito em Python que retorna assinaturas na sintaxe do Bro.

A arquitetura do sistema Bro é apresentada na Figura 4.16.

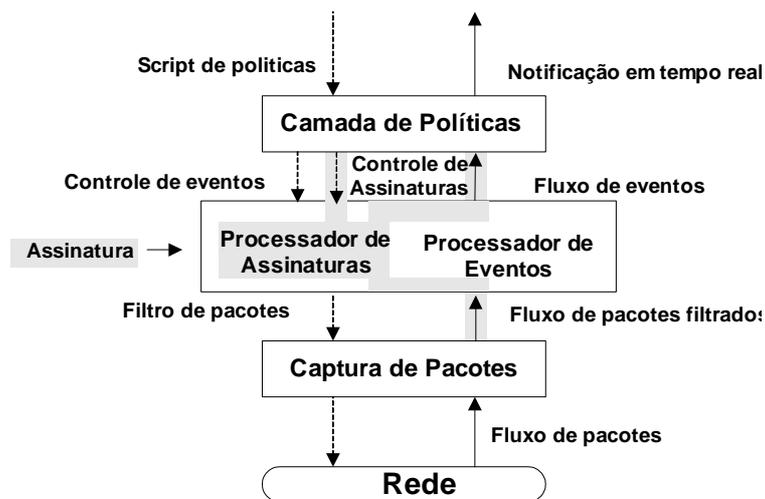


Figura 4.16 - Arquitetura do sistema Bro

O módulo de captura de pacotes utiliza a biblioteca *libpcap*. A chave para filtragem de pacotes no Bro é selecionar quais pacotes manter e quais descartar.

O fluxo de pacotes filtrado na etapa anterior é enviado para a próxima camada do sistema, denominada “Processador de Eventos”. Esta camada tem por finalidade reduzir o fluxo de pacotes filtrados a um fluxo de eventos de rede de mais alto nível. Realiza varias verificações da integridade dos pacotes para garantir que seus cabeçalhos estão bem formados, incluindo a verificação do *checksum* do cabeçalho IP. Se ocorrer uma falha, o Bro gera um evento indicando o problema e descarta o pacote. Neste ponto, o sistema remonta os fragmentos IP tal que os datagramas IP possam ser completamente analisados.

O processador de assinaturas (*Signature Engine*) situa-se ao lado do processador de eventos de análise de protocolo (*Event Engine*) no projeto do sistema Bro. As assinaturas são casadas contra os fluxos de pacotes através de um componente localizado na camada intermediária do Bro. Quando ocorre um casamento, este componente insere um novo evento dentro do fluxo de eventos. A camada de políticas pode então decidir como reagir a este evento. Além disto, pode-se transferir informação da camada de políticas para o

processador de assinaturas de volta para controlar sua operação. Uma assinatura pode especificar um *script* a ser chamado quando uma assinatura particular for casada.

4.4 Visualização de Anomalias no Tráfego de Rede

Determinadas práticas atuais para identificação e diagnóstico de anomalias no tráfego de rede consistem na visualização do tráfego, a partir de diferentes perspectivas, e identificação de anomalias a partir do conhecimento de especialistas.

A visualização de informações do tráfego possibilita a codificação de grandes quantidades de dados complexos inter-relacionados em uma representação gráfica, permitindo que estes dados sejam mais facilmente quantificados, manipulados e compreendidos pelo usuário humano.

Várias ferramentas têm sido utilizadas para coletar dados do tráfego de rede, algumas livremente disponibilizadas e mais populares, tais como: *MRTG* e *Cricket*. Estas ferramentas realizam medições periodicamente coletando valores de SNMP, tais como a interface ou contadores de portas de roteadores e switches, e têm sido eficientes, visto que disponibilizam dados ao usuário final de forma útil e simplificada. Outras aplicações, tais como *FlowScan* (PLONKA, 2000), *FlowAnalyzer* da Cisco e *AutoFocus* também são utilizadas para análise e classificação do tráfego de rede, com base em informações de uso da rede e protocolos. Algumas destas ferramentas provêm recursos de alerta em tempo real, mas a maioria das análises é feita de modo *off-line* (pos-mortem).

Uma robusta análise em tempo real é necessária para detectar e identificar anomalias de modo que as ações de mitigação possam ser tomadas o mais breve. Algumas ferramentas utilizam dados de volume do tráfego, tais como contagem de bytes e de pacotes. Quando os links estão congestionados, é possível observá-los completamente utilizados, e nem sempre se pode obter

informações posteriores sobre as alterações no tráfego de rede. Ataques sofisticados de atuação lenta, bem como ataques de substituições de strings não variam o volume do tráfego de modo significativo, portanto não podem ser detectados pelas ferramentas específicas de análise do volume do tráfego. KIM et al (2004, 2005a, 2005b, 2005c) realizaram estudos sobre o comportamento do tráfego de rede e desenvolveram o sistema *NetViewer*, analisando os padrões do tráfego através de técnicas de processamento e compressão de imagens e análise de vídeo em tempo real. Esta abordagem monitora passivamente os cabeçalhos dos pacotes do tráfego de rede em intervalos regulares e gera imagem destes dados.

IDGraphs (REN et al., 2006) é um sistema de visualização interativo para detecção de intrusão que utiliza histogramas para apresentar os fluxos de dados com o tempo na horizontal e o número de conexões não-sucedidas no eixo vertical. O sistema detecta e analisa vários ataques e anomalias, incluindo varredura de portas, worms, inundações TCP SYN furtivas e outros ataques distribuídos.

Uma técnica gráfica (ONUT et al., 2004; ONUT et al., 2007) para detecção de anomalias foi desenvolvida para visualização do tráfego de rede, a qual alerta sobre as anomalias que chegam à rede, utilizando informações da camada de rede.

O trabalho de ZACHARY et al (2005) apresenta um mecanismo de visualização capaz de detectar varreduras de portas interativamente, baseando-se em três níveis semânticos: uma vista de mais alto nível que mostra o conjunto de dados todo em baixa resolução, uma vista de nível médio que mostra todas as portas em um dado tempo e uma vista de baixo nível que mostra uma porta individual sobre todas as métricas para o domínio de tempo completo da base de dados.

Outro exemplo de ferramenta relacionada à visualização do tráfego de rede é descrita em (BEARAVOLU et al., 2003).

5 UMA METODOLOGIA PARA DETECÇÃO DE ATAQUES NO TRÁFEGO DE REDES

Neste capítulo é descrita a metodologia desenvolvida composta por um conjunto de fases, métodos, técnicas e ferramentas para apoio à detecção de ataques no tráfego de redes.

A metodologia foi aplicada na rede do LabRedes-DSS do INPE na preparação do ambiente de detecção de ataques descrito em (SILVA et al, 2005c; SILVA et al., 2006), possibilitando a realização de estudos de casos a partir de dados simulados e dados reais de tráfego HTTP de rede. Foram adotadas técnicas de redes neurais e estatísticas para a detecção de ataques.

Compõem esta metodologia sete principais fases, incluindo:

- Coleta de dados do tráfego;
- Reconstrução de sessões do tráfego;
- Seleção de atributos do tráfego;
- Armazenamento de sessões do tráfego;
- Preparação de dados para treinamento das redes neurais;
- Técnicas para detecção de ataques e modelagem de dados;
- Considerações sobre análise do comportamento do tráfego.

A principal contribuição desta tese é o desenvolvimento de técnicas e métodos de detecção de assinaturas e anomalias no tráfego de redes, através da aplicação de redes neurais artificiais.

5.1 Coleta de Dados do Tráfego

A primeira fase refere-se à coleta de dados do tráfego de rede. Para isto, uma estação de trabalho é configurada para a captura de pacotes e armazenamento de dados. Esta máquina denominada Sensor do IDS deve ter memória em disco suficiente para armazenar grandes volumes de dados do tráfego.

O Sensor do IDS pode ser posicionado em diferentes pontos da rede. Em geral, dependendo do objetivo da análise a ser realizada, adota-se uma das seguintes estratégias para posicionamento do sensor:

- um sensor fora dos limites de proteção do *firewall*;
- um sensor dentro dos limites de proteção do *firewall*;
- dois sensores, um dentro e outro fora dos limites de proteção do *firewall*.

Quando um sensor é posicionado fora dos limites protegidos pelo *firewall*, todo o tráfego direcionado à rede monitorada proveniente da Internet é coletado pelo sensor (CHAVES, 2002). Esta configuração permite a observação de ataques dirigidos ao *firewall* e também aos recursos internos na rede. Uma limitação desta técnica é que ataques internos não são observados com o sensor nesta posição e a quantidade de dados a ser tratada é muito grande. Além disto, se os atacantes descobrirem o sensor, podem atacá-lo, porque existe menor possibilidade de suas atividades serem auditadas. Embora alguns ataques não possam ser detectados por um sensor fora do *firewall*, esta é a melhor localização do sensor para detectar ataques (NORTHCUTT et al., 2002). Um benefício apresentado por esta técnica é que os analistas podem ver os tipos de ataques aos quais sua rede e o *firewall* estão expostos.

Quando o sensor é posicionado dentro dos limites de proteção do *firewall*, o sensor observa o tráfego externo permitido pelo *firewall* em direção à rede monitorada e o tráfego desta rede em direção às redes externas passando pelo *firewall*. Vantagens desta técnica são: menor quantidade de dados a ser examinada e possibilidade de observação de ataques internos. Sensores dentro dos limites de proteção do *firewall* são menos vulneráveis que aqueles posicionados fora dos limites do *firewall*. Se o sensor está dentro dos limites de proteção do *firewall* e recebe menos ruído, pode gerar menos falsos positivos (NORTHCUTT et al., 2002).

O uso de dois sensores, um dentro e outro fora dos limites de proteção do *firewall*, permite usufruir das características das duas técnicas anteriormente

descritas (CHAVES, 2002), além de permitir a validação das regras de controle do *firewall*. Esta técnica apresenta as seguintes vantagens (NORTHCUTT et al., 2002): não é necessário adivinhar se um ataque ultrapassou os limites de proteção do *firewall*; possibilita a detecção tanto de ataques internos quanto externos; e possibilita a detecção de sistemas mal configurados, que não podem ultrapassar os limites de proteção do firewall, auxiliando o administrador de sistemas.

Uma estratégia para a captura de dados do tráfego é o uso de ferramentas de monitoração de rede, tais como *Netflow*¹, que coleta dados do tráfego de roteadores Cisco, a ferramenta Unix *tcpdump*² (JACOBSEN et al., 1997) ou similares, por exemplo, *tcptrace*³ ou *Ethereal*⁴, para obter os dados brutos do tráfego de rede.

A seguir, é apresentado um exemplo de uso da ferramenta *tcpdump* utilizada para capturar todo o tráfego passando pela interface eth0 de rede e armazenar os dados em formato binário no arquivo file.dump.

```
tcpdump -nv -i eth0 -w /logs/file.dump
```

Se o objetivo é monitorar sessões de tráfego HTTP na porta 80, por exemplo, pode-se filtrar os dados da seguinte maneira:

```
tcpdump -nv -i eth0 (tcp and port 80) -w /logs/file.dump
```

Para capturar apenas tráfego IP e não incluir pacotes RIP de roteamento, recomenda-se o uso do comando apresentado a seguir:

1 http://www.cisco.com/warp/public/cc/pd/iosw/ioft/neflct/tech/napps_wp.htm

2 <http://www.tcpdump.org/>

3 <http://jarok.cs.ohiou.edu/software/tcptrace/>

4 <http://www.ethereal.com/>

```
tcpdump -nv -i eth0 (ip and not udp port 520) -w /logs/file.dump
```

Uma rede TCP/IP pode prover diferentes serviços, tais como: correio eletrônico, acesso a aplicações Web, transferência de arquivos, compartilhamento de dados e recursos em rede, tradução de nomes de domínios, conexão remota, entre outros. Para detectar intrusos na rede, deve-se ter conhecimento de quais tipos de serviços são fornecidos nesta rede, ter informação da quantidade de usuários e obter informações de acesso aos serviços. Estes dados são importantes para análise do comportamento da rede.

Se o objetivo é analisar o comportamento de um serviço de rede específico, deve-se filtrar o tráfego armazenado em disco utilizando opções do *tcpdump* para este fim. Por exemplo, para análise de tráfego HTTP, o seguinte comando pode ser utilizado:

```
tcpdump -nv -r /logs/file.dump tcp and port 80
```

No caso de uso de redes com VLAN e análise de serviço HTTP, deve-se também utilizar filtros para a leitura dos pacotes capturados com tag de VLAN e porta 80:

```
tcpdump -nv -r /logs/file.dump (tcp and port 80) or (vlan <no.>  
and tcp port 80)
```

O *tcpdump* é uma ferramenta que utiliza a biblioteca de captura de pacotes *libpcap* (*Packet Capture Library*) para a comunicação com o dispositivo de rede, a qual executa as seguintes tarefas: especificar o dispositivo de rede onde a coleta deve ser efetuada; criar uma sessão de captura e associá-la a um descritor da biblioteca; compilar uma expressão de filtragem para o formato que pode ser entendido pela biblioteca; aplicar a expressão de filtragem já compilada à sessão de captura previamente criada; iniciar o processo de obtenção de pacotes da rede. O dispositivo de rede é colocado em "modo promíscuo", o qual permite à máquina ter acesso aos pacotes que não sejam destinados a ela.

Um crescente número de ferramentas de segurança pode ser utilizado para monitorar uma rede. O *tcpdump* não provê uma saída refinada, mas fornece uma boa quantidade de detalhes que permitem avaliações sobre a atividade do tráfego (NORTHCUTT et al., 2002).

Como os pacotes devem ser coletados continuamente, recomenda-se o rotacionamento dos logs, para processar o sistema em modo *batch* e armazenar os dados em disco em intervalos de tempo regulares.

A escolha do intervalo ou janela de tempo depende do propósito da detecção de intrusão. Um intervalo de tempo mais curto permite a geração de alarmes mais próximos de tempo real, mas requer maior capacidade de processamento e de armazenamento de dados. Uma janela de tempo maior produzirá um retardo na detecção, mas permite economizar recursos de CPU e memória.

A presente metodologia não se aplica a ataques que são executados em janelas de tempo muito grandes.

5.2 Reconstrução de Sessões do Tráfego

O conjunto de centenas de sessões de um segmento de rede, em diferentes intervalos de tempo, representa o tráfego desta rede.

Uma sessão de rede TCP/IP corresponde a qualquer seqüência de pacotes, que caracterize a troca de informações entre dois endereços IP, durante um determinado intervalo de tempo, relacionada a um determinado serviço de rede, que tenha informação de início, meio e fim, mesmo que toda comunicação esteja contida em um único pacote (CHAVES, 2002). Sessões de diferentes aplicações possuem comportamentos próprios.

A reconstrução das sessões do tráfego corresponde à remontagem dos pacotes que participaram de uma comunicação entre dois hosts. Ataques nem sempre podem ser detectados através de medições de taxas de bits ou de pacotes. Determinadas atividades abusivas, tais como ataques de negação de

serviço e varredura de portas e serviços, são detectadas através da análise de dados das sessões de rede.

Ferramentas alternativas para a reconstrução de sessões são o software *tcptrace*, utilizado em (ERTOZ et al., 2003c) e o sistema Recon, utilizado em (CHAVES, 2005b).

O Sistema de Reconstrução de sessões TCP/IP (Recon), desenvolvido por Chaves (2002) e atualizado por Chaves (2005b), permite a reconstrução e rastreio do estado das sessões TCP/IP, baseando-se em um modelo gerado a partir de dados extraídos dos cabeçalhos e conteúdo dos pacotes da pilha de protocolos TCP/IP. Este sistema tem por escopo analisar apenas os pacotes dos protocolos *Ethernet*, IP, ICMP, TCP e UDP. À medida que o tráfego é lido de um arquivo em formato *tcpdump*, o Recon reconstrói as sessões ICMP, TCP e UDP entre os pares de endereços IP. Este sistema organiza as sessões TCP/IP de modo que as informações de cada sessão podem ser facilmente obtidas, mostrando-se uma ferramenta interessante de base para um sistema de detecção de intrusão.

Uma sessão TCP no Recon tem como identificadores as duas portas <portA,portB> contidas nas mensagens TCP dos pacotes que a compõem, onde a primeira está associada ao menor endereço IP dos pacotes, e a segunda está associada ao maior endereço IP. Outra informação, que determina se um pacote pertence ou não a uma sessão TCP, é o seu estado.

Os estados da sessão obtidos através do sistema Recon são:

- TCP_NO_STATE (0): nenhum estado definido
- TCP_SYN_STATE(1): primeiro estado (SYN) no '*Three Way Handshake*'
- TCP_HALF_OPEN_STATE(2): segundo estado (SYN/ACK) no '*Three Way Handshake*'
- TCP_OPEN_STATE(3): terceiro estado (ACK) no '*Three Way Handshake*'
- TCP_FIN1_A_STATE(4): FIN1 enviado pelo menor IP (IPA)

- TCP_FIN1_B_STATE(5): FIN1 enviado pelo maior IP (IPB)
- TCP_ACK_FIN1_A_STATE(6): ACK (FIN1) - ack fin1 enviado pelo menor IP;
- TCP_ACK_FIN1_B_STATE(7): ACK (FIN1) - ack fin1 enviado pelo maior IP;
- TCP_FIN2_A_STATE(8): FIN2 - fin2 enviado pelo menor IP;
- TCP_FIN2_B_STATE(9): FIN2 - fin2 enviado pelo maior IP;
- TCP_FIN2_A_SIM_STATE(10): FIN2-Sim - simultaneous fin2 enviado pelo menor IP;
- TCP_FIN2_B_SIM_STATE(11): FIN2-Sim - simultaneous fin2 enviado pelo maior IP;
- TCP_ACK_FIN1_A_SIM_STATE(12): ACK (FIN1-Sim) - ack simultaneous fin1 enviado pelo menor IP;
- TCP_ACK_FIN1_B_SIM_STATE(13): ACK (FIN1-Sim) - ack simultaneous fin1 enviado pelo maior IP;
- TCP_ACK_FIN2_A_SIM_STATE(14): ACK (FIN2-Sim) - ack simultaneous fin2 enviado pelo menor IP;
- TCP_ACK_FIN2_B_SIM_STATE(15): ACK (FIN2-Sim) - ack simultaneous fin2 enviado pelo maior IP;
- TCP_FIN2_A_CLOSED_STATE(16): ACK (FIN2) - ack fin2 enviado pelo menor IP e encerra a sessão;
- TCP_FIN2_B_CLOSED_STATE(17): ACK (FIN2) - ack fin2 enviado pelo maior IP e encerra a sessão;
- TCP_ACK_FIN1_A_SIM_CLOSED_STATE(18): ACK (FIN1-Sim) – ack simultaneous fin1 enviado pelo menor IP e encerra a sessão;
- TCP_ACK_FIN1_B_SIM_CLOSED_STATE(19): ACK (FIN1-Sim) – ack simultaneous fin1 enviado pelo maior IP e encerra a sessão;
- TCP_ACK_FIN2_A_SIM_CLOSED_STATE(20): ACK (FIN2-Sim) – ack simultaneous fin2 enviado pelo menor IP e encerra a sessão;
- TCP_ACK_FIN2_B_SIM_CLOSED_STATE(21): ACK (FIN2-Sim) – ack simultaneous fin2 enviado pelo maior IP e encerra a sessão;
- TCP_RESET_CLOSED_STATE(22): encerra a sessão com um RST.

Para a reconstrução das sessões do tráfego capturado e gravação destas em base de dados, utiliza-se o seguinte comando do sistema Recon:

```
recon -H -r file.dump
```

Para a reconstrução das sessões de ataque, utiliza-se a opção “-J”, como segue:

```
recon -J -r file.dump
```

5.3 Seleção de Atributos do Tráfego

Uma sessão de rede pode ser unicamente caracterizada pela combinação de diferentes características ou atributos. Os atributos de sessão contribuem para a detecção de diferentes tipos de ataques. Alguns tipos de ataques são detectados através de informações localizadas na parte de conteúdo (*payload*) dos pacotes, enquanto outros podem ser descobertos a partir da observação dos campos no cabeçalho dos pacotes. Determinados campos do cabeçalho dos pacotes são utilizados de forma indevida pelos atacantes, por exemplo, pode constar no cabeçalho que a porta de serviço utilizada é a 80, quando na verdade outro serviço foi utilizado nesta porta, ao invés do HTTP. Neste caso, para identificar estes tipos de ataques, é necessária uma análise complementar do conteúdo do pacote.

Um atributo de sessão pode ser primitivo, ou seja, um valor extraído diretamente de um campo do cabeçalho do pacote ou um atributo derivado, construído a partir do processamento de atributos primitivos. Compreendem os atributos primitivos aqueles extraídos do cabeçalho IP: endereço IP de origem e de destino, protocolo de transporte utilizado, tamanho total do pacote em *bytes*; e aqueles extraídos do cabeçalho TCP: portas de origem e de destino e *flags*. Os atributos derivados são obtidos do processamento dos atributos primitivos e correspondem à informação semanticamente mais forte para a representação do tráfego. Como exemplos de atributos derivados tem-se a

duração da sessão, que corresponde à diferença entre o tempo de captura do último e do primeiro pacote da sessão.

Do grande número de atributos que pode ser monitorado para o propósito de detecção de intrusão, é importante definir quais são realmente úteis, menos significativos e quais são mais úteis para análise. A questão é relevante porque a eliminação de atributos (chamado de redução da trilha de auditoria) pode melhorar a precisão no processo de detecção e acelerar o processamento, assim melhorando a desempenho do IDS como um todo (MUKKAMALA e SUNG, 2003c; MUKKAMALA e SUNG, 2003d). Bons atributos deveriam prover informação útil, se o tráfego de rede é normal ou não, gerando alta taxa de detecção com baixa taxa de alarme falso.

No caso de uso dos atributos como entrada para classificadores inteligentes, a recomendação é não selecionar muitos atributos, pois podem invalidar os resultados do classificador.

Na maioria dos IDSs existentes, os atributos são escolhidos manualmente, com base na análise de tipos de ataques existentes, análise estatística e visualização de dados. Mas esforços têm sido dedicados à avaliação de estratégias de seleção de atributos automatizadas para detecção de intrusão. Alguns dos atributos mais comumente encontrados na literatura são listados na Tabela 5.1 a seguir.

Tabela 5.1-Atributos típicos utilizados em detecção de intrusão

serviço	Serviço acessado (por porta): http, ftp, telnet
duração	Duração da conexão
Src_ip	Endereço IP do iniciador da conexão
Dst_ip	Endereço IP do host
Src_bytes	Número de bytes enviados pelo iniciador
dst_bytes	Número de bytes enviados pelo host
protocolo	TCP, UDP, ICMP, ...
num_conn	Número de conexões abertas
tcp_flags	Flags TCP (SYN, ACK, RST, ...)

Uma linha de trabalho conduzida por W. Lee e colaboradores (LEE e STOLFO, 1999) tenta abordar a construção de atributos e *data mining* sistematicamente. O principal foco está na aprendizagem baseada em regras.

Nos trabalhos conduzidos por Lazarec et al. (2003) os dados são previamente filtrados para remover sessões não interessantes para análise (tráfego de fontes confiáveis ou comportamento de rede não usual ou anômalo que é sabidamente livre de intrusão) e pré-processados para coletar atributos derivados e atributos característicos.

Devido à falta de um modelo analítico, pode-se apenas pesquisar e determinar a importância relativa das variáveis de entrada (atributos) dos IDSs através de métodos empíricos (MUKKAMALA et al., 2003c). Uma análise completa requisitaria a verificação de todas as possibilidades, ou seja, tomando dois atributos por vez para analisar sua dependência ou correlação, em seguida, tomar 3 atributos por vez e assim por diante. Isto, entretanto, é inviável (requer 2^n experimentos!).

Esforços de seleção de atributos relacionados a métodos de Aprendizagem de máquina são realizados por Mukkamala et al (2003c). Estes autores utilizam a abordagem denominada *wrapper*, onde os atributos são removidos um por vez e, para cada conjunto restante de atributos, é avaliado o desempenho dos modelos de detecção de intrusão utilizando SVMs (*Support Vector Machines*). Embora a validade dos resultados possa ser questionada, este artigo (MUKKAMALA et al., 2003c) representa um esforço bem vindo de aplicação de técnicas de aprendizagem de máquina automatizáveis no domínio de detecção de intrusão.

Os atributos de sessões escolhidos para uso no presente trabalho correspondem aos nove atributos utilizados com êxito por Chaves (2005b), descritos a seguir:

- **Tamanho médio dos pacotes recebidos pelo cliente:** valor calculado através de uma média aritmética da quantidade de dados (em bytes) da carga útil dos pacotes recebidos pelo cliente em uma sessão;
- **Tamanho médio dos pacotes recebidos pelo servidor:** valor calculado através de uma média aritmética da quantidade de dados (em bytes) da carga útil dos pacotes recebidos pelo servidor em uma sessão;
- **Número de pacotes recebidos pelo cliente:** quantidade de pacotes recebidos pelo cliente em uma sessão;
- **Número de pacotes recebidos pelo servidor:** quantidade de pacotes recebidos pelo servidor em uma sessão;
- **Porcentagem de pacotes pequenos:** valor calculado através da soma do número de pacotes com uma quantidade de dados inferior a um dado limiar (130 bytes), seguido da divisão da soma pelo total de pacotes de uma sessão;
- **Direção do tráfego:** valor inicializado com 0 (zero). Para cada pacote recebido pelo servidor na sessão, o valor é subtraído de uma unidade, e para cada pacote recebido pelo cliente, o valor é incrementado de uma unidade;
- **Total de dados recebidos pelo cliente:** valor calculado através da soma da quantidade de dados (em bytes) de todos os pacotes recebidos pelo cliente em uma sessão;
- **Total de dados recebidos pelo servidor:** valor calculado através da soma da quantidade de dados (em bytes) de todos os pacotes recebidos pelo servidor em uma sessão;
- **Duração da sessão:** valor (em segundos) calculado pela diferença entre o momento em que o último pacote da sessão foi capturado e o momento em que o primeiro pacote foi capturado.

Além dos nove atributos acima mencionados, atributos complementares foram considerados neste trabalho, os quais são descritos a seguir:

- **Estado da sessão:** valor inteiro entre 0 e 22, apresentados na seção 5.1, que corresponde ao estado atual da sessão;
- **Instante de captura do pacote:** horário em que o primeiro pacote da sessão foi capturado;
- **Endereços IP:** menor (IP_A) e maior (IP_B) endereços IP participantes da sessão;
- **Portas:** portas de comunicação utilizadas pelo menor (port_A) e maior (port_B) endereços IP participantes da sessão.

Estes quatro atributos são armazenados na base para permitir uma melhor compreensão da origem e tipo dos ataques após estes terem sido identificados. Por exemplo, qual é a origem e o destino do ataque e qual é o estado das sessões para diferentes tipos de ataques. Porém, os métodos de detecção implementados neste trabalho, aqueles que utilizam atributos de sessões do tráfego de rede, não processam estes quatro atributos em busca de anomalias, apenas os nove atributos anteriormente descritos.

5.4 Armazenamento de Sessões do Tráfego

Após reconstrução das sessões por meio dos dados contidos nos pacotes que participaram da comunicação entre pares de hosts, e a seleção dos atributos de sessões a serem analisados, a próxima fase consiste em armazenar os atributos das sessões do tráfego para análise.

Além de remontar as sessões e gravar dados em memória temporária, o Recon foi modificado neste trabalho para gravar os atributos escolhidos em base de dados para uso pelos métodos de detecção.

O sistema de gerenciamento de bases de dados de domínio público MySQL é uma alternativa para armazenamento de dados e foi utilizado neste trabalho, permitindo rápida criação de bases de dados estruturadas, com recursos úteis de gerenciamento, incluindo backups, importação e exportação de dados.

Na construção das Tabelas do banco devem-se considerar os tipos de dados (inteiro, ponto flutuante, booleano, varchar, char) dos atributos de sessões a serem inseridos nas Tabelas. Também se pode optar pela criação de um campo adicional usado para armazenar o valor de classificação (*decision*) para a sessão do tráfego: 0 (para sessão normal) ou 1 (para sessão anômala). Com o sistema ADTRAF é possível uma classificação manual das sessões do tráfego com conhecimento prévio sobre seu status.

Um exemplo de Tabela de sessões armazenadas com o campo de classificação (*decision*) é ilustrado na Tabela 5.2 a seguir, obtido através da seguinte consulta à base de dados:

```
select psize_cl, psize_sv, pnum_cl, pnum_sv, smallpkt, data_dir, brecv_cl, brecv_sv, duration, decision from C09012007 where codigo > 375;
```

Tabela 5.2 - Exemplo de consulta a registros de sessões do tráfego na base

psize_cl	psize_sv	pnum_cl	pnum_sv	smallpkt	data_dir	brecv_cl	brecv_sv	duration	decision
738	99,5	12	10	54,55	2	8856	995	16,241	0
93,6	82,5	5	6	81,82	-1	468	495	4,041	0
424,2	109,89	10	9	63,16	1	4242	989	67,304	0
208,14	139,43	7	7	71,43	0	1457	976	67,304	0
924,15	110,61	60	41	43,56	19	55449	4535	133,979	0
150,54	214,5	13	14	62,96	-1	1957	3003	127,439	0
0	0	1	5	83,33	-4	0	0	0,007	1
0	0	1	4	100	-3	0	0	6,964	1
0	0	1	6	85,71	-4	0	0	1,01	1
0	0	1	5	83,33	-4	0	0	0,006	1

A base de dados, além das sessões do tráfego a ser monitorado, é utilizada também para armazenamento das assinaturas de ataques.

5.5 Preparação de dados para treinamento das redes neurais

Um ambiente de testes é necessário para a criação de conjuntos de dados com tráfego de ataque simulado e tráfego normal a serem utilizados para treinamento, testes e ajustes de modelos de detecção de ataques antes de utilizá-los em rede de produção.

As estações servidoras (Estações Vítima), com sistemas operacionais e aplicações de rede semelhantes aos existentes na rede de produção a ser monitorada, devem ser instaladas no ambiente.

Nesta etapa, os dados para avaliação *pos-mortem* encontram-se na Estação de Análise.

5.5.1 Geração de Tráfego Normal

Uma forma de geração de tráfego normal sintético é o acesso normal aos serviços de rede disponíveis, a captura dos pacotes e a reconstrução das sessões de tráfego normal geradas.

A captura de tráfego normal da rede de produção a ser monitorada também é importante. Recomenda-se o uso de uma ferramenta, tal como o Snort, para a sanitização dos dados de modo a separar o tráfego de ataque do tráfego normal desta rede para fins de testes dos métodos de detecção.

5.5.2 Geração de Tráfego de Ataque

Uma recomendação importante nesta fase é obter a maior quantidade possível de ataques aos serviços de rede instalados no ambiente. Para isto, um estudo dos programas de ataques deve ser feito para a seleção destes. Os ataques

podem ser obtidos na Internet em *sites* como Milworm¹, Security Focus², Live-CD Backtraq³, FRSIRT⁴ e Packetstorm⁵.

Para a geração do tráfego de ataque a estratégia é, em laboratório, instalar e configurar os serviços de rede simulando o ambiente de rede real a ser monitorado, atacar os serviços instalados e coletar as sessões do tráfego de ataque geradas, envolvendo a captura dos pacotes de rede e a reconstrução das sessões.

As atividades de coleta do tráfego de ataque seguem o fluxo apresentado na Figura 5.1.

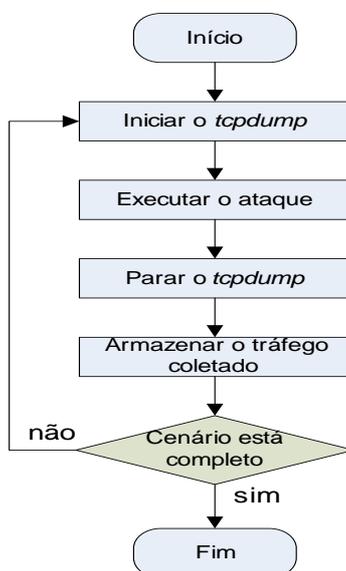


Figura 5.1 - Seqüência de atividades para coletar o tráfego de ataque.

Fonte: Adaptado de FAGUNDES (2002)

¹ <http://www.milw0rm.com>

² <http://www.securityfocus.com>

³ <http://www.remote-exploit.org>

⁴ <http://www.frsirt.com>

⁵ <http://www.packetstorm.com>

Antes de reproduzir um determinado ataque o *tcpdump* deve ser iniciado, só então o ataque é executado. Logo que o ataque for concluído, deve-se interromper o serviço *tcpdump* e o arquivo gerado por este *sniffer* é armazenado. Essa seqüência de atividades deve ser realizada até que o último ataque do cenário definido tenha sido reproduzido (FAGUNDES, 2002).

5.6 Técnicas para Detecção de Ataques e Modelagem de Dados

Através de métodos baseados em assinaturas ou anomalias, e uso de técnicas inteligentes ou estatísticas, torna-se possível o processo de detecção de ataques nos dados do tráfego de rede.

Os IDSs baseados em assinaturas requerem a especificação prévia de padrões de ataques para classificar o tráfego. Esta técnica é incapaz de identificar novos ataques e gera alarmes falsos quando as assinaturas casam com atividades verdadeiramente não intrusivas. Os eventos de ataques também podem sofrer variações de modo que as assinaturas antigas não possam mais ser encontradas no conjunto e, além disto, a busca por assinaturas no conteúdo dos dados é inútil se estes estiverem criptografados (CHAVES, 2005b).

Por outro lado, a estratégia de apenas utilizar mecanismos de detecção baseados em anomalias para o propósito de detecção de intrusão não deveria ser vista como uma solução geral para o problema de identificar ataques em uma infra-estrutura de rede. Pelo contrário, seu uso em conjunto com técnicas baseadas em assinaturas é fortemente recomendado (TAPIADOR et al., 2004a).

Considerando a afirmação acima, a combinação de processos de detecção de assinaturas e detecção de anomalias é recomendada nesta tese para a busca de ataques no tráfego de rede. A estratégia de implementação da seqüência: primeiro realiza-se detecção de assinaturas e, em seguida, detecção de anomalias, é uma melhor prática, visto que os métodos de detecção de

assinaturas, em geral, são mais rápidos que os de detecção de anomalias. Isto porque os detectores de assinaturas utilizam uma base de padrões de ataques que é menor que a base de dados legítimos a ser introduzida para treinamento supervisionado ou não supervisionado de modelos baseados em anomalias. Com isto, ataques conhecidos podem ser mais rapidamente identificados.

Em um ambiente real, a operação do detector de ataques pode seguir uma arquitetura lógica como a apresentada na Figura 5.2. Para o caso de detecção de intrusos na camada de aplicação, ambas as técnicas compartilham a mesma infra-estrutura para a captura do tráfego de rede e remontagem dos pacotes. A maioria dos IDSs baseados em rede (por exemplo, Snort, Bro ou NetSTAT) implementa este processo.

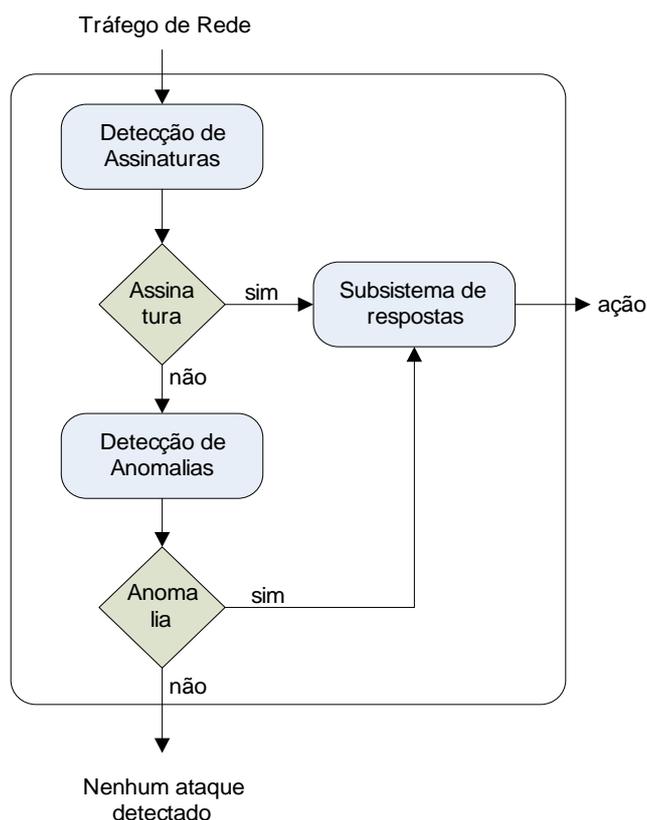


Figura 5.2 - Operação de junção dos detectores de assinaturas e anomalias
Fonte: Adaptado de TAPIADOR et. al (2004a)

No caso de análise baseada em anomalias supervisionada, a idéia é armazenar dados do tráfego histórico normal para futura comparação com

dados do tráfego corrente em busca de eventuais anomalias. Observando o tráfego atual e correlacionando-o a seus estágios anteriores pode-se verificar se seu comportamento variou ou tem-se apresentado dentro da normalidade. Este tipo de análise requer o uso de técnicas estatísticas ou técnicas inteligentes supervisionadas.

A detecção de anomalias em que o algoritmo não precisa ser previamente treinado com dados normais pode ser realizada através de técnicas estatísticas de detecção de *outliers*, por exemplo, ou técnicas inteligentes não supervisionadas, realizando *clustering* de dados.

O processo de análise de payload dos pacotes e análise do comportamento do tráfego através de dados do cabeçalho dos pacotes (atributos de sessões), através de técnicas baseadas em assinaturas, seguido do processo de detecção de anomalias sobre atributos de sessões, melhora a busca por ataques no tráfego de rede.

5.6.1 Normalização dos Dados

Os atributos que compõem as sessões de rede estão em ordem de grandeza e unidades de medida diferentes, portanto, deve-se realizar uma normalização dos dados para valores entre [0,1] ou [-1, 1] antes da apresentação destes aos algoritmos de detecção.

Este processo pode ser conduzido de diversas maneiras; neste trabalho foi utilizada a técnica de divisão de valores dos atributos pelo maior valor, através da seguinte equação:

$$n_i = (v_i / MaxValue) \quad (5.1)$$

onde v_i é o valor real do atributo, $MaxValue$ é o maior valor de todos os atributos de mesmo tipo e n_i é o valor do atributo normalizado.

5.6.2 Detecção de Ataques e Modelagem de Dados

Nesta tese redes neurais foram aplicadas como técnicas para detecção de ataques, constituindo a principal contribuição do trabalho. As redes neurais escolhidas incluem: a Hamming Net e LVQ, para detecção de assinaturas no *payload* dos pacotes de rede; a rede neural LVQ para detecção de padrões de comportamento do tráfego, a partir de dados do cabeçalho (atributos de sessões) dos pacotes de rede; e as redes neurais LVQ e MLP para detecção de anomalias no tráfego de rede, com a pré-clusterização do tráfego normal usando a LVQ e a classificação do tráfego completo, incluindo tráfego normal clusterizado e tráfego anômalo, usando a rede MLP.

Para a detecção de assinaturas no tráfego de rede, podem ser utilizados dados do cabeçalho (atributos de sessões) ou de *payload* (*strings* de dados) dos pacotes. Para ambos os tipos de dados, uma rede neural de rápida classificação - a rede LVQ (*Learning Vector Quantization* – Aprendizagem por Quantização Vetorial), foi utilizada neste trabalho para classificação de padrões de ataques.

A rede LVQ, através de um processo de competição e cálculo de distância Euclidiana entre padrões de entrada e exemplares (assinaturas), é capaz de identificar os ataques que casam com exemplares existentes na base.

Os dados introduzidos nas redes neurais, em geral, devem ser modelados para que tenham um formato único para que a rede possa processá-los de modo correto.

No caso de análise de atributos de sessões, os registros de sessões de tráfego de rede obtidos do cabeçalho dos pacotes são utilizados como dados de entrada e exemplares para a rede LVQ. Os valores de nove atributos de cada sessão são inseridos na rede, tanto as entradas como os exemplares, do modo como se encontram na base, ou seja, como números decimais.

Para a análise de *payload* de pacotes, as *strings* de dados de entrada e exemplares são pré-processadas de modo que aquelas em formato ASCII sejam convertidas para o formato hexadecimal. Estando todas as *strings* em formato único hexadecimal, estas são processadas pelo algoritmo '*nhash*' e convertidas cada uma para uma chave única de 32 caracteres em hexadecimal que é a entrada para a LVQ. Para entrada na rede neural Hamming Net, a saída resultante do algoritmo *nhash* (32 caracteres em hexa) é convertida para 128 bits em bipolar. Cada chave é armazenada na base MySQL em formato hexadecimal.

Para o processo de detecção de anomalias no tráfego de rede, uma combinação das redes neurais não supervisionada - Mapas Auto-organizável (SOM – *Self Organizing Maps*) e supervisionadas LVQ e Perceptron de Múltiplas Camadas (MLP) foi utilizada neste trabalho. Outras técnicas utilizadas neste estudo são os algoritmos de detecção de *outliers* LOF e LSC.

Dados de atributos de sessões do tráfego de rede, em formato decimal, são previamente normalizados, utilizando-se o método de normalização pelo valor máximo apresentado pela equação 5.1 na seção 5.6.1, e estes são introduzidos nas redes neurais SOM, LVQ e MLP e nos algoritmos LOF e LSC para classificação.

5.6.3 Atualização da Base de Assinaturas e Tráfego Normal

Enquanto um mecanismo de detecção baseado em assinaturas requer constante atualização da base de assinaturas de modo a detectar os novos ataques conhecidos, um sistema baseado em anomalias requer retreinamento periódico do modelo, pois o comportamento normal da rede pode mudar.

Em geral, a atualização da base de dados de assinaturas de ataques conhecidos, a partir de dados do *payload* dos pacotes, é feita através da inclusão contínua de novas assinaturas disponibilizadas no site do Snort (SOURCEFIRE, 2007), mas esta também pode ser atualizada a partir de

strings de ataques obtidas com o lançamento de novos ataques em ambiente de teste.

Para a atualização da base de assinaturas de ataques conhecidos, a partir de dados de cabeçalho dos pacotes (atributos de sessões), as sessões de novos ataques lançados contra serviços e aplicações no ambiente de teste devem ser reconstruídas e seus atributos armazenados. Outra forma de atualização da base de assinaturas de atributos de sessões é, de tempos em tempos, coletar sessões do tráfego da rede monitorada, cujo comportamento é dinâmico, e passar estas sessões por um detector de assinaturas conhecido e corretamente configurado, por exemplo, o Snort ou Bro, para a separação de padrões conhecidos de padrões de tráfego normal. As novas assinaturas serão adicionadas à base de assinaturas existentes e o tráfego normal, que é o tráfego restante, comporá uma nova base.

Um exemplo de comando para sanitização de dados utilizando o sistema Snort é apresentado a seguir:

```
snort -c /etc/snort.conf -r /diretorio/arquivo.dump tcp -l /diretorio/log_snort/
```

Neste comando, *arquivo.dump* é o arquivo a ser filtrado e *log_snort* é o diretório onde serão registrados os logs de alerta do Snort.

A base de tráfego normal para entrada nas ferramentas de detecção de ataques pode ser proveniente de dados de rede de produção, sanitizados pelo Snort ou outro IDS e observados por especialistas, ou pode ser tráfego normal sintético, coletado através de acessos normais a serviços de rede em ambiente de teste.

5.6.4 Avaliação de Resultados

Uma avaliação de resultados dos algoritmos de detecção de ataques sobre atributos de sessões pode ser realizada através da contagem de sessões de tráfego normal e tráfego de ataque, classificadas por estes algoritmos e a comparação destes valores com os valores reais de sessões normais e

anômalas utilizados no treinamento dos algoritmos inteligentes. Um exemplo é a utilização dos comandos `SELECT COUNT(*) FROM <nome_Tabela> WHERE decisao=0;` para contagem de sessões classificadas como normal e `SELECT COUNT(*) FROM <nome_Tabela> WHERE decisao=1;` para contagem de sessões classificadas como anômalas na base de dados.

A avaliação de resultados dos algoritmos de detecção de assinaturas no *payload* dos pacotes consiste em comparar os resultados das técnicas aplicadas (Snort, Bro, ANNIDA ou outro sistema) com a base de assinaturas existente.

Após avaliação de resultados e ajustes dos modelos para correta detecção de ataques, as ferramentas de detecção podem ser utilizadas em ambiente de rede de produção.

5.7 Considerações sobre Análise do Comportamento do Tráfego

O tráfego de uma rede se comporta de modo dinâmico, ou seja, a rede pode ser expandida ou reduzida, através da inserção ou remoção de usuários, hosts, serviços de rede e acessos aos serviços. Em diferentes horários e dias da semana a rede também se comporta de modo diferente.

Dois tipos de análise podem ser conduzidos para acompanhamento do comportamento do tráfego de rede em busca e anomalias:

- observação do tráfego em diferentes períodos do dia, dias da semana e dias do mês;
- análise da quantidade de sessões de rede anômalas.

Deve-se levar em consideração que sessões de ataques apresentam comportamento homogêneo. Ataques DoS e de varredura aparecem em *bursts* (rajadas) ou em sessões múltiplas, enquanto outras formas de ataques (U2R e R2L) aparecem, em geral, em sessão única (TAPIADOR, 2004b).

6 APLICAÇÕES DESENVOLVIDAS

6.1 A Aplicação ANNIDA

6.1.1 Histórico

O início das pesquisas e desenvolvimento relacionados a esta tese ocorreu em 2004, com a aplicação de redes neurais artificiais para detecção de assinaturas no tráfego de rede.

Inicialmente, a rede neural Hamming Net foi aplicada como algoritmo de classificação para detecção de assinaturas em *strings* isoladas. Neste trabalho (SILVA et al., 2004), foram utilizados dados simulados, semelhantes aos dados em formato ASCII encontrados no *payload* de pacotes de rede, contendo *strings* de conteúdo normal e *strings* de conteúdo malicioso.

Em seguida, a aplicação passou a ser chamada *Artificial Neural Network for Intrusion Detection Application* (ANNIDA) e foi modificada para a busca de assinaturas em múltiplas *strings* de conteúdo malicioso (SILVA et al., 2005a; SILVA et al., 2005b). Ainda, nesta época, foram utilizados dados simulados para análise, mas ampliou-se o poder de precisão da aplicação com a detecção de ataques representados por mais de uma *string* de conteúdo malicioso.

Uma outra rede neural de rápida classificação, a *Learning Vector Quantization* (LVQ), foi utilizada posteriormente (SILVA et al., 2006). A rede LVQ produziu taxas de detecção de *strings* de conteúdo malicioso semelhantes às taxas de detecção geradas pela Hamming Net, porém a LVQ apresentou menor tempo de processamento em relação a Hamming Net. Deste modo, a rede neural LVQ passou a ser considerada o processador principal de assinaturas na aplicação ANNIDA.

Em seguida, com o amadurecimento da pesquisa, realizou-se uma melhoria significativa da aplicação em termos de projeto e desenvolvimento (SILVA et al., 2007), o que permitiu a detecção de assinaturas em dados reais de *payload*

de pacotes de rede, em formato ASCII e hexadecimal, através do uso da rede LVQ. Nesta fase, ANNIDA foi reconhecida como uma nova contribuição no domínio de aplicação de rede neural de rápida classificação para o reconhecimento de padrões de ataques no conteúdo de pacotes do tráfego real de rede.

6.1.2 Características da Aplicação

ANNIDA é uma ferramenta de detecção de intrusão baseada em rede e em assinaturas. Esta aplicação permite o uso de uma das redes neurais: Hamming Net ou LVQ. Como anteriormente mencionado, a rede LVQ apresenta menor tempo de processamento na detecção de assinaturas.

Basicamente, os processos que compõem esta aplicação são: captura de pacotes, modelagem de dados, detecção de assinaturas e alerta, como ilustrado na Figura 6.1. O dispositivo de rede é colocado em "modo promíscuo" para que a máquina de captura tenha acesso a todos os pacotes trafegando pela rede, não somente aos pacotes destinados a ela. A biblioteca de funções *libpcap* (*Packet Capture Library*) é utilizada para a comunicação com o dispositivo de rede.

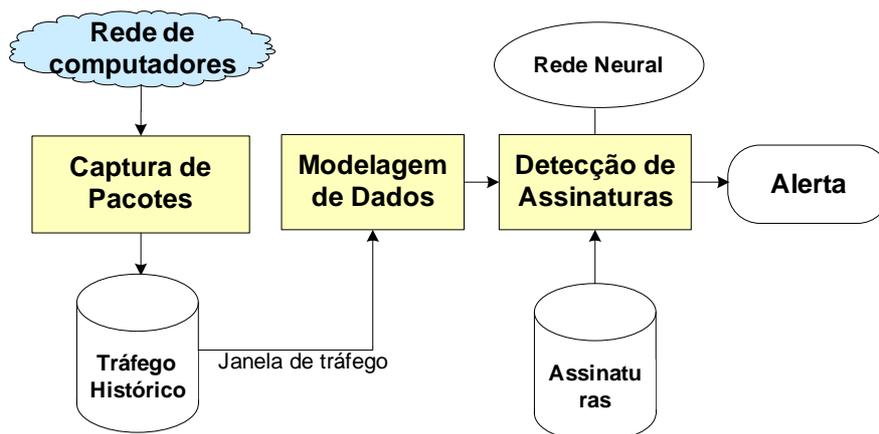


Figura 6.1 - Visão Geral da Aplicação ANNIDA

Na implementação dos métodos de detecção são empregadas redes neurais e utilizadas duas bases de dados: uma para armazenamento de tráfego histórico e outra para armazenamento de assinaturas de ataques conhecidos.

Esta aplicação realiza análise *off-line* do conteúdo (*payload*) dos pacotes de rede em busca de assinaturas compostas por uma ou múltiplas *strings* de dados de conteúdo malicioso. Assinaturas são padrões de ataques conhecidos representados por conjuntos de caracteres ou *strings* em formato ASCII, hexadecimal ou por uma combinação de ambos os formatos. As assinaturas utilizadas neste trabalho foram obtidas através do pré-processamento das regras do IDS Snort (CASWELL et al., 2003).

O conteúdo de uma assinatura na regra Snort é *<content: "string;">*. A regra pode incluir dados em formato ASCII, binário e hexadecimal. Um exemplo de uma regra Snort é apresentado na Figura 6.2 a seguir.

```
alert tcp any any -> any any (content: "|0101 FFFF|/etc/passwd|E234|" ;  
msg: "Procurando conteúdo misturado!" ;)
```

Figura 6.2 – Exemplo de regra Snort

O mecanismo de detecção de assinaturas do Snort implementa o algoritmo denominado *Boyer-Moore* (MOORE, 2002) para identificar *strings* de conteúdo malicioso nos pacotes capturados. Este algoritmo foi inventado em 1975 e ainda hoje é um dos mais eficientes para pesquisa e correspondência de padrões. Além disto, o Snort utiliza uma estratégia de saída rápida: quando encontra um pacote correspondente a uma regra, alerta sobre esta ocorrência e não verifica este pacote novamente com relação a quaisquer outras regras (CASWELL et al., 2003). Esta estratégia de saída rápida também é utilizada na implementação da ANNIDA.

Uma explicação mais detalhada sobre as regras do Snort e de seus componentes que foram utilizados na construção das assinaturas do ANNIDA é encontrada em (SILVA et al., 2004, SILVA et al., 2005a).

Na Figura 6.3 é ilustrado um exemplo de *payload* de pacote de rede contendo uma assinatura de ataque formada por duas *strings* de conteúdo malicioso: “00 FA 00 FF” e “/bin/sh”. O objetivo da aplicação é reconhecer estas *strings* como traços de um mesmo ataque.

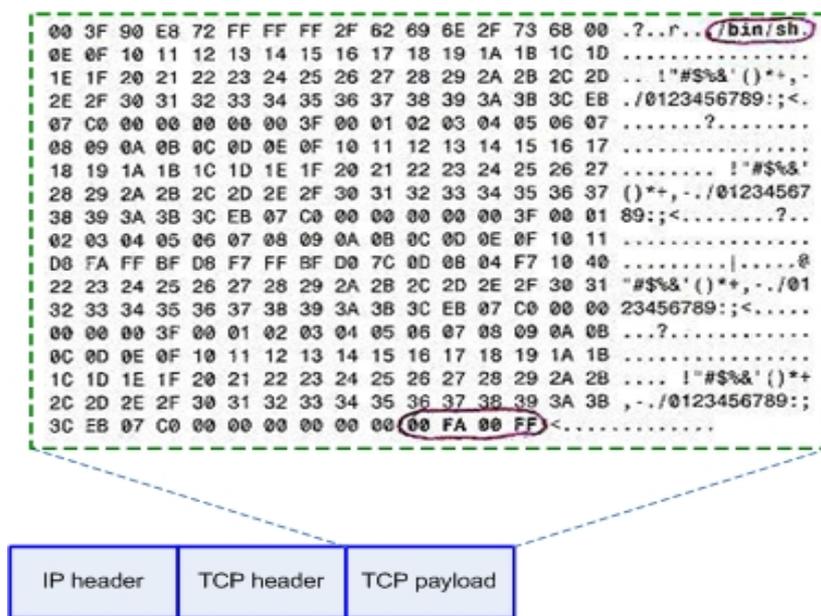


Figura 6.3 - Exemplo de Dados do Payload em Pacotes de Rede TCP/IP

Até o momento, foram explorados nesta aplicação os métodos de detecção baseados nas redes neurais *Hamming Net* e LVQ (FAUSSET, 1994) para a classificação supervisionada de padrões de ataques. Ambas as redes pertencem ao grupo de redes neurais artificiais de rápida classificação e possuem arquitetura semelhante, compostas por apenas uma camada de entrada e uma camada de saída. Dados de entrada, dados exemplares, pesos dos neurônios, taxa de aprendizagem e grau de similaridade entre dados de entrada e dados de saída são informações necessárias na construção destas redes.

Os dados de entrada correspondem aos dados da máscara do *payload* (pequenas janelas fixas e deslizantes de dados, com deslocamento de um carácter) de pacotes de rede a serem classificados em cada passo do processo

de detecção. Os dados exemplares são as *strings* de assinaturas que são utilizadas como padrões de busca.

Dados reais de carga útil dos pacotes de rede são previamente capturados e introduzidos (vetor de entrada X) no módulo de detecção de assinaturas para verificação, como apresentado na Figura 6.4. Nesta Figura, um conteúdo de pacote de rede contendo a assinatura “%20-display%20” é exibido. Os pesos são calculados com base nos valores dos exemplares ou assinaturas de ataques extraídas da base Snort, tais como “tracertoute%20” e “%20-display%20”. Nesta ilustração, o algoritmo de classificação detecta a *string* “%20-display%20” no *payload* examinado e gera uma mensagem de alerta.

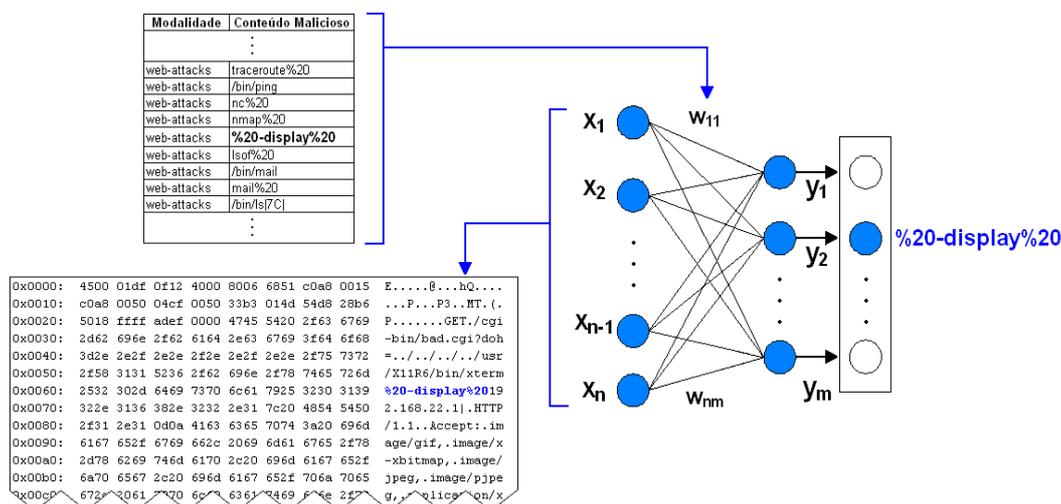


Figura 6.4 - Visão geral da Entrada, Exemplos e Saída do Classificador

O módulo de detecção desta aplicação provê a chamada recursiva da rede neural para a detecção de múltiplas *strings* de conteúdo malicioso, como mostra a Figura 6.5. A estratégia utilizada é a seguinte: a cada busca de uma *string* de uma única assinatura de ataque, tenha esta uma ou múltiplas *strings*, a rede neural é processada.

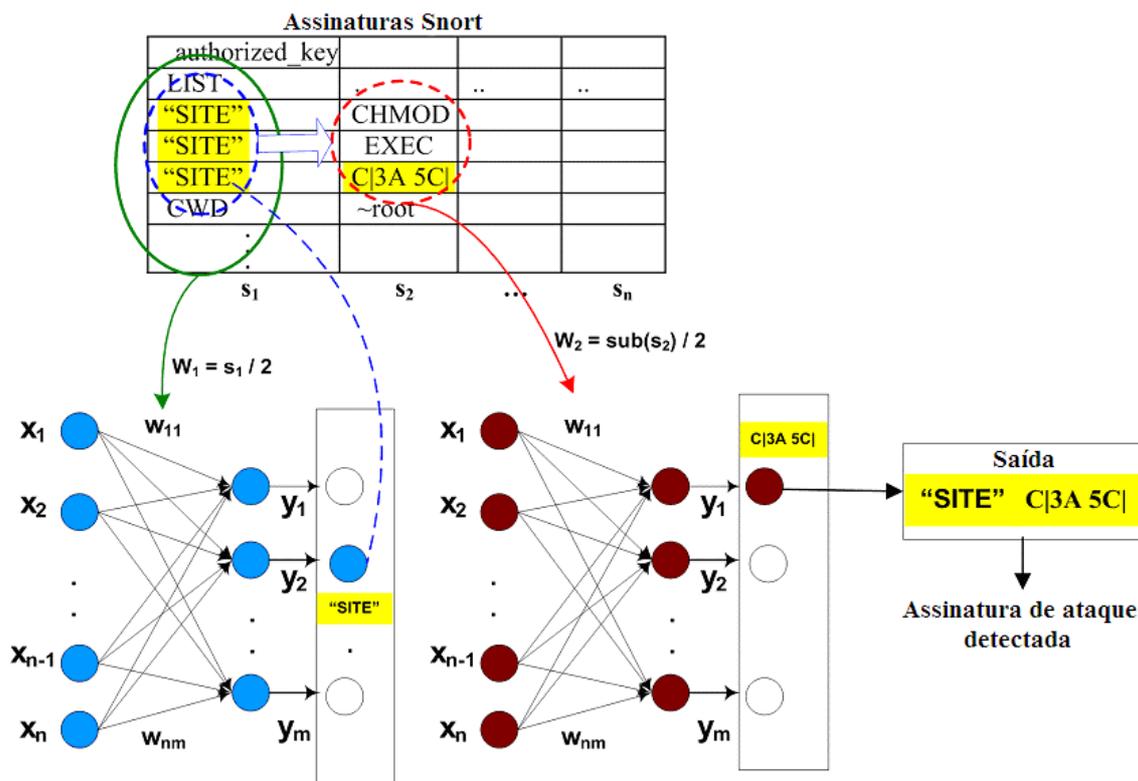


Figura 6.5 - Detecção de Múltiplas *Strings* de uma Assinatura de Ataque

6.1.3 Recursos de Desenvolvimento

As versões preliminares 2004 e 2005 da aplicação ANNIDA foram implementadas em linguagem C e compiladas com o compilador DevCpp, em ambiente operacional Windows. O código da versão 2006 desta aplicação está escrito em C e compilado em GCC na plataforma *Linux Slackware* 10.1 com *kernel* versão 2.4.29. Esta mudança de plataforma operacional permitiu o uso da base MySQL para tratamento e consulta dos dados da aplicação. Uma estação de trabalho com processador AMD Athlon de 64 bits a 2.2 GHz, 1 GB de memória RAM e HD de 160 GB tem sido utilizada para desenvolvimento e teste desta aplicação.

6.1.4 Evolução no Desenvolvimento da Aplicação

Com o intuito de aperfeiçoar e expandir a aplicação ANNIDA, em busca de melhor desempenho, precisão, robustez e escalabilidade da aplicação, mudanças significativas foram efetuadas em 2006, incluindo: alteração na

forma de armazenamento e recuperação de dados de entrada para a aplicação, alteração das estruturas de dados, migração do código para a plataforma operacional Linux, melhor modelagem dos dados para entrada na rede neural, refinamento do código, criação de filtros e, principalmente, entrada de dados reais de pacotes de rede para classificação.

Na Tabela 6.1 são mostrados os itens modificados na aplicação, bem como os benefícios obtidos.

Tabela 6.1- Mudanças e Benefícios

Itens	ANNIDA 2005	ANNIDA 2006	Benefícios
Rede Neural	Hamming Net	LVQ	Menor tempo de processamento
Dados de Entrada	Dados simulados (conteúdo normal + conteúdo ilegítimo)	Dados de tráfego real (conteúdo normal + conteúdo ilegítimo)	Precisão, ajuste à realidade
Repositório de dados	Arquivos texto	Tabelas em base MySQL	Melhor desempenho, escalabilidade, rapidez no armazenamento e acesso, consistência dos dados
Tipo dados de entrada	ASCII	Hexadecimal	Maior precisão
Tipo dados exemplares	ASCII	Hexadecimal	Maior precisão
Formato da Entrada	14 bits (bipolar)	128 bits (32 caracteres em hexadecimal)	Maior precisão (colisão de entradas reduzida)
Total de assinaturas	Conjunto de assinaturas Snort (3000)	Classes de assinaturas Snort (variável)	Menor tempo de processamento
Leitura de <i>strings</i> de uma assinatura	Número fixo de <i>strings</i> (9 strings em 2005)	Número variável de <i>strings</i>	Escalabilidade, maior precisão
Estrutura do Programa	Não Modular	Modular	Flexibilidade de uso; inclusão de novas abordagens de detecção

Os dados de entrada e os dados exemplares foram convertidos de arquivos do tipo texto para registros em Tabelas da base de dados MySQL. Esta alteração teve por objetivo melhorar o desempenho da aplicação, prover busca mais rápida em arquivo indexado; maior escalabilidade, permitindo a entrada de maior volume de dados, além de prover consistência dos dados e flexibilidade

na consulta aos dados, permitindo o uso de funções do próprio banco na análise de comparação dos resultados contra os padrões existentes.

Os dados de entrada (dados de *payload* de pacotes do tráfego real de rede) foram tratados para armazenamento em formato hexadecimal para estar compatível com o tipo de dados das assinaturas Snort. As assinaturas Snort são obtidas em uma combinação de formato hexadecimal e ASCII, em ambos os formatos ou em cada um destes separadamente. Na fase de casamento de dados do *payload* com as assinaturas, ambos dados do *payload* a ser analisado e assinaturas devem estar no mesmo formato, no caso, em hexadecimal e não ASCII, pois nem todo caracter hexadecimal possui um caracter ASCII correspondente para conversão.

Os dados de entrada são modelados considerando a camada de entrada com número fixo de neurônios na rede neural. Por isto, os dados são modelados, na versão atual da aplicação, para serem introduzidos com o mesmo tamanho, com a quantidade de 14 bits. Para ambos os casos, os dados em ASCII do *payload* e das assinaturas são convertidos em chaves de tamanho único através do algoritmo de hashing '*Folding-Shift*' (MORAES, 2001). A partir dos testes com este tipo de técnica observou-se o problema de colisão de chaves, onde diferentes dados de entrada eram casados com a mesma assinatura. Para resolver este problema, o algoritmo de *hashing* foi modificado para '*nhash*' (2006), gerando chaves únicas em 128 bits.

Conjuntos parciais de assinaturas de ataque, contendo somente aquelas assinaturas relacionadas com um determinado serviço de rede analisado (exemplo utilizado: *web-attacks.rules*) ou a uma certa categoria de ataque (conforme estão agrupadas as assinaturas nas regras Snort) foram considerados na nova versão da aplicação, a fim de agilizar o processo de busca de ataques. Neste caso, considerou-se a entrada de dados de *payload* referentes a determinado tipo de serviço de rede analisado (por exemplo, HTTP), requerendo uma pré-filtragem dos dados capturados por porta 80.

Como uma assinatura de ataque real é composta por uma ou mais *strings* de conteúdo malicioso, denominadas, neste trabalho, *strings associadas*, a aplicação, cujo início teve por meta a busca de uma única string, foi modificada para a leitura de número variável de *strings*, conforme os registros de cada assinatura na base de dados.

Finalmente, o código da aplicação foi estruturado em módulos independentes e configuráveis, de modo a permitir a escolha dos módulos necessários, e quando necessário para uso. Esta mudança permite a inclusão de novas funcionalidades, de diferentes mecanismos de detecção e facilita a realização de alterações no código.

6.1.5 Módulos da Aplicação

A aplicação ANNIDA é composta por quatro módulos, ilustrados na Figura 6.6, compreendendo: “Módulo de Modelagem de Assinaturas”, “Módulo de Extração de *Payloads* e Dados”, “Módulo de Classificação Hamming Net” e “Módulo de Classificação LVQ”. É possível selecionar o módulo desejado (Hamming Net ou LVQ) para processar a detecção de assinaturas.

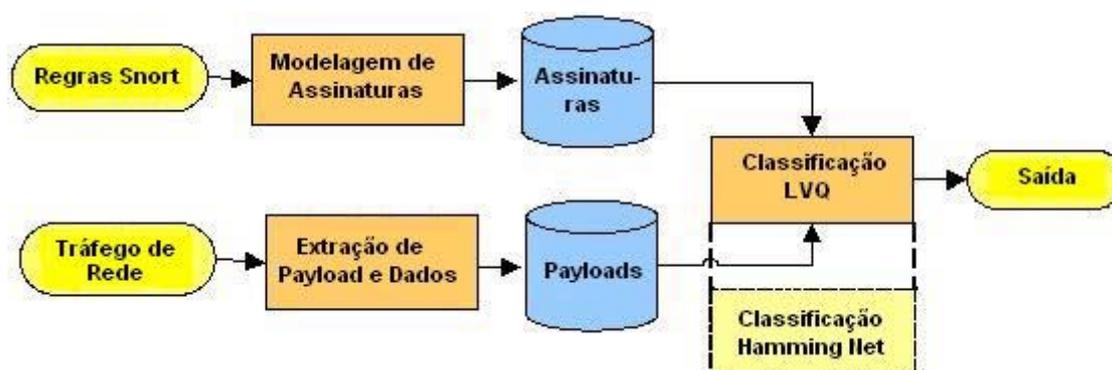


Figura 6.6 - Módulos do Sistema ANNIDA

O Módulo de Modelagem de Assinaturas tem por finalidade preparar os padrões de ataques para serem corretamente introduzidos como exemplares nas redes neurais dos módulos de Classificação LVQ ou Hamming Net.

O Módulo de Extração de *Payloads* e Dados faz a leitura dos dados brutos do *payload* dos pacotes de rede armazenados em arquivos, previamente

capturados através da ferramenta *tcpdump*, e registra dados de interesse dos pacotes para comparação futura.

O Módulo de Classificação Hamming Net é responsável por detectar assinaturas de ataques no conjunto de dados de entrada, utilizando a rede neural Hamming Net.

O Módulo de Classificação LVQ realiza a detecção de assinaturas de ataques no conjunto de dados de entrada, utilizando a rede neural LVQ.

6.1.5.1 Módulo de Modelagem de Assinaturas

Neste módulo, *strings* isoladas ou múltiplas *strings* de assinaturas são obtidas a partir de arquivos de regras Snort em formato ASCII e hexadecimal.

As *strings* de dados de assinaturas são pré-processadas de modo que aquelas em formato ASCII sejam convertidas para o formato hexadecimal. Estando todas as *strings* de uma assinatura em formato único hexadecimal, cada uma é processada pelo algoritmo '*nhash*' e convertida para uma chave única de 32 caracteres em hexadecimal que é a entrada para a rede neural LVQ. Para entrada na rede neural Hamming Net, é utilizada a saída *nhash* de 32 caracteres em hexadecimal, convertida para 128 bits em bipolar. Cada chave de 32 caracteres em hexa é armazenada na base MySQL.

Neste módulo, inicialmente, seleciona-se o arquivo Snort referente ao tipo de ataque a ser considerado na aplicação para fins de classificação, utilizando, como exemplo, o seguinte comando:

gera_assinaturas web-attacks

Este comando dispara a geração de assinaturas de ataques a web a serem considerados como conjunto de exemplares (padrão de classificação da rede neural), armazenados na base de dados.

6.1.5.2 Módulo de Extração de *Payloads* e Dados

Neste módulo, os dados brutos de carga útil dos pacotes de rede são processados e armazenados em Tabelas do banco de dados como hexadecimal.

Também são armazenados dados de interesse, tais como: endereços IP de origem e destino, tipo de protocolo, portas de origem e destino, tamanho do pacote – incluindo cabeçalho e dados, tamanho do cabeçalho, tamanho do *payload* e tipo de serviço para futura comparação das respostas dos classificadores.

Neste módulo, o arquivo de log capturado com os dados reais do tráfego de rede é varrido e o conteúdo real do *payload* é adicionado também em uma base de dados, utilizando o comando:

gera_payload 22_12_2006.dump

Tendo na base os dados de assinaturas e de *payload*, no formato conhecido pelo sistema, é possível acionar os métodos de classificação de ataques. Para isso, deve-se selecionar qual o tipo de abordagem de classificação será utilizada: Hamming Net ou LVQ. Para cada um destes métodos, os dados de entrada e exemplares são introduzidos em bipolar (no caso Hamming Net) e hexadecimal (no caso LVQ).

6.1.5.3 Módulo de Classificação Hamming Net

Basicamente, a finalidade da *Hamming Net* consiste em identificar a classe à qual uma dada entrada pertence, tomando como referência um conjunto de padrões previamente introduzidos na rede, denominados “exemplares”. Em cooperação com a rede neural MAXNET (FAUSSET, 1994), a *Hamming Net* encontra o valor, no conjunto de exemplares, mais próximo de uma entrada a ser classificada.

A arquitetura da rede *Hamming Net* utilizada na ANNIDA é composta por 128 neurônios de entrada e Y neurônios de saída, onde Y corresponde ao total de assinaturas inseridas na rede como padrões de busca (assinaturas de

ataques). A rede Hamming Net possui um conjunto de pesos fixos w , calculados com base nos valores d das assinaturas de ataques, onde $w=d/2$.

A Figura 6.7 ilustra a arquitetura desta rede.

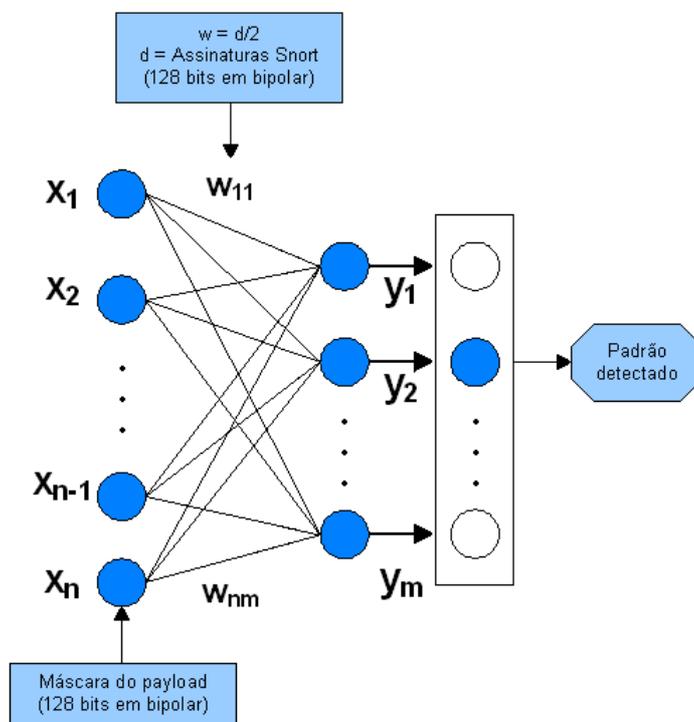


Figura 6.7 - Arquitetura da rede neural Hamming Net

Cada entrada da *Hamming Net* é um conjunto de 128 bits correspondente aos valores de uma máscara do *payload*. Por máscara do *payload*, entende-se um agrupamento de dados em formato hexadecimal, contido em uma janela que se desloca pelo *payload* (deslocamento de 1 *character*), cujo tamanho varia de uma busca para outra de acordo com o tamanho de todas as assinaturas em hexadecimal contidas na base de dados. Para entrada na rede, a máscara do *payload* no formato hexadecimal também é convertida para bipolar.

Cada máscara deslocada no *payload* é apresentada a Hamming Net para classificação e, em seguida, são calculados os valores dos neurônios de saída, com base nos valores da entrada e dos pesos da rede. Ocorre-se casamento da máscara de *payload* com uma assinatura, um alerta é gerado e este pacote não é mais verificado com relação a quaisquer outras assinaturas.

A medida da similaridade entre o vetor de entrada e os vetores exemplares armazenados é calculada com base no número de componentes nos vetores comparados menos à distância de Hamming entre estes vetores. A distância de Hamming é o número de componentes que se diferem nos dois vetores comparados (FAUSSET, 1994). Se resultar em 100% de similaridade, a aplicação alerta que um ataque foi detectado e o próximo pacote de rede é verificado contra os padrões.

6.1.5.4 Módulo de Classificação LVQ

A arquitetura da rede LVQ é composta por 32 neurônios de entrada e Y neurônios de saída, onde Y corresponde ao total de assinaturas inseridas na rede como assinaturas e no cálculo dos pesos (w), cujos valores são inseridos a partir dos dados exemplares (d) e não são reduzidos à metade, como no caso da Hamming Net. Portanto, nesta rede, $w=d$.

Cada entrada da LVQ corresponde, portanto, a uma máscara de *payload* representada por 32 caracteres em hexadecimal (16 bytes), com valores de caracteres no domínio de 0 a 9 e de A a F, conforme ilustrado na Figura 6.8.

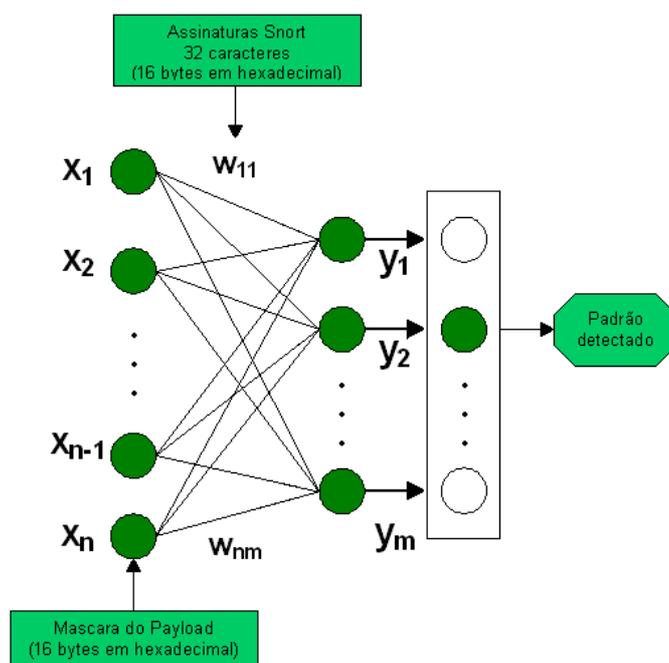


Figura 6.8 - Arquitetura da rede neural LVQ

A ativação da LVQ consiste no cálculo da distância Euclidiana entre cada vetor de entrada e todos os exemplares armazenados nos pesos da rede. A menor distância calculada indica o neurônio de saída vencedor. Determinado o vencedor, é iniciado o processo de comparação dos valores, verificando assim se a assinatura representada pelo neurônio vencedor é exatamente igual a máscara do *payload* analisada. Se ocorrer casamento perfeito entre assinatura e *payload*, um alerta é gerado e este pacote não é mais verificado com relação a quaisquer outras assinaturas.

6.1.5.5 Mecanismo de Alerta

A aplicação ANNIDA possui um mecanismo de alerta que é disparado quando uma assinatura de ataque é identificada pelo módulo de detecção. Uma tela semelhante à da Figura 6.9 é apresentada, informando, além do padrão de ataque encontrado, os dados: IP origem (atacante), IP destino (alvo), tipo de protocolo de rede, portas de origem e de destino e tamanho do *payload*. Estas informações auxiliam na tomada de decisão para evitar que os ataques registrados pela aplicação voltem a ocorrer.

```
<----- ATAQUE IDENTIFICADO ----->
ORIGEM DO ATAQUE --> 192.168.0.21          DESTINO DO ATAQUE --> 192.168.0.80
TIPO DE PROTOCOLO --> TCP
PORTA DE ORIGEM --> 1225                  PORTA DE DESTINO --> 80
TAMANHO DO PAYLOAD DO PACOTE --> 438 bytes
ASSINATURA DO ATAQUE --> wget%20
<----- ATAQUE IDENTIFICADO ----->

<----- ATAQUE IDENTIFICADO ----->
ORIGEM DO ATAQUE --> 192.168.0.21          DESTINO DO ATAQUE --> 192.168.0.80
TIPO DE PROTOCOLO --> TCP
PORTA DE ORIGEM --> 1231                  PORTA DE DESTINO --> 80
TAMANHO DO PAYLOAD DO PACOTE --> 439 bytes
ASSINATURA DO ATAQUE --> /usr/X11R6/bin/xterm
<----- ATAQUE IDENTIFICADO ----->
```

Figura 6.9 - Saída da Aplicação com Informações sobre o Ataque

6.1.6 Considerações

As redes LVQ e Hamming Net foram utilizadas para detecção de assinaturas no tráfego de rede por serem abordagens supervisionadas e conhecidas como redes de rápida classificação. Também, estas redes possuem arquiteturas simplificadas, composta por uma camada de entrada e uma de saída, com poucos neurônios na camada de entrada, sendo utilizados 128 neurônios de entrada para a Hamming Net e 32 neurônios de entrada para a LVQ. Ambas as redes utilizam a técnica de aprendizagem competitiva para classificação, onde a entrada é comparada com os exemplares (assinaturas) através de cálculo de distância e sempre um neurônio de saída (associado a um exemplar) é o vencedor, ou seja, sempre realiza um casamento entre o vetor de entrada e o exemplar mais semelhante. A rede Hamming Net utiliza o cálculo da distância de Hamming e da LVQ utiliza o cálculo da distância Euclidiana.

A Hamming Net e a LVQ enquadram-se no grupo de redes neurais de rápida classificação, pelo fato de não sofrerem atualização constante de pesos, os quais são fixos. Embora uma implementação da LVQ considere a atualização de pesos, neste trabalho foi utilizada a técnica de inserção de pesos para ambas as redes. A técnica de inserção de pesos consiste em calcular os pesos da rede com base nos valores dos exemplares. Este procedimento torna mais rápida a atualização de novos padrões de ataques detectados na base de assinaturas. Sem a etapa de treinamento da rede, os padrões exemplares são previamente introduzidos na rede e diretamente casados contra cada nova entrada. Provendo agilidade na classificação dos dados, a resposta do analista, em caso de incidente, pode ser mais rápida.

Nos testes realizados, ambas as redes Hamming Net e LVQ apresentaram excelente precisão (100% de taxa de acerto) na detecção de padrões de ataques conhecidos, nos dados reais dos pacotes de rede. Isto porque ambas as redes utilizam método de classificação semelhante, baseado em competição, onde o neurônio exemplar mais semelhante à entrada é o vencedor.

Comparado com a Hamming Net, a LVQ produziu classificação mais rápida, sendo o tempo gasto para detecção de assinaturas reduzido da ordem de minutos para segundos. O processamento da LVQ é mais rápido do que o da Hamming Net porque possui menor número de neurônios de entrada (32 neurônios na LVQ contra 128 neurônios na Hamming Net) e devido à modelagem mais simplificada dos dados de entrada. Para a LVQ o formato da entrada é hexadecimal, como lida da base, e para a Hamming Net, a entrada deve ser convertida de hexadecimal para formato bipolar.

Com esta pesquisa, constatou-se que a rede neural LVQ é uma técnica supervisionada viável para a detecção de padrões de ataques nos dados de carga útil dos pacotes de rede e é um campo de pesquisa a ser mais explorado para fins de testes de desempenho e eficiência em relação a outros métodos existentes de busca de padrões.

Uma limitação da aplicação ANNIDA é a incapacidade de detectar novos padrões de ataques diferentes daqueles previamente cadastrados na base de exemplares, visto que as “*strings*” contidas no *payload* possuem representação única em hexadecimal e a ANNIDA utiliza estratégia de 100% de similaridade entre entrada e exemplar para realizar o casamento.

Como trabalho futuro, o método de detecção de assinaturas da aplicação ANNIDA será aperfeiçoado e acoplado ao sistema ADTRAF, apresentado na seção 6.3.

6.2 A Aplicação RGCOM

Para facilitar o trabalho do analista na verificação de grandes volumes de dados é essencial o uso de ferramentas que representem graficamente o tráfego de rede de modo simplificado, claro e preciso. As técnicas de visualização devem facilitar o entendimento do comportamento do tráfego no tempo, reduzindo o tempo de observação e ampliando a precisão dos resultados da análise.

Nesta linha de raciocínio, foi desenvolvida a aplicação RGCOM (Representação Gráfica do Comportamento do Tráfego de Rede) (MANCILHA et al., 2006), que apresenta graficamente dados do tráfego de rede de modo off-line, conforme a data de captura do tráfego e o intervalo de tempo especificados pelo usuário.

Os dados apresentados pelo RGCOM são atributos do tráfego de rede, extraídos do cabeçalho dos pacotes, incluindo: tamanho médio dos pacotes recebidos pelo cliente (em *bytes*); tamanho médio dos pacotes recebidos pelo servidor (em *bytes*); número de pacotes recebidos pelo cliente; número de pacotes recebidos pelo servidor; porcentagem de pacotes pequenos (com menos de 130 *bytes*); direção do tráfego; total de dados recebidos pelo cliente (em *bytes*); total de dados recebidos pelo servidor (em *bytes*) e duração da sessão (em segundos).

Esta aplicação, implementada em Java, utiliza quatro módulos principais, como ilustrado na Figura 6.10: Módulo de Captura de Dados, Módulo de Reconstrução de Sessões e Armazenamento de Atributos, Módulo de Seleção de Atributos e Módulo de Impressão do Gráfico.

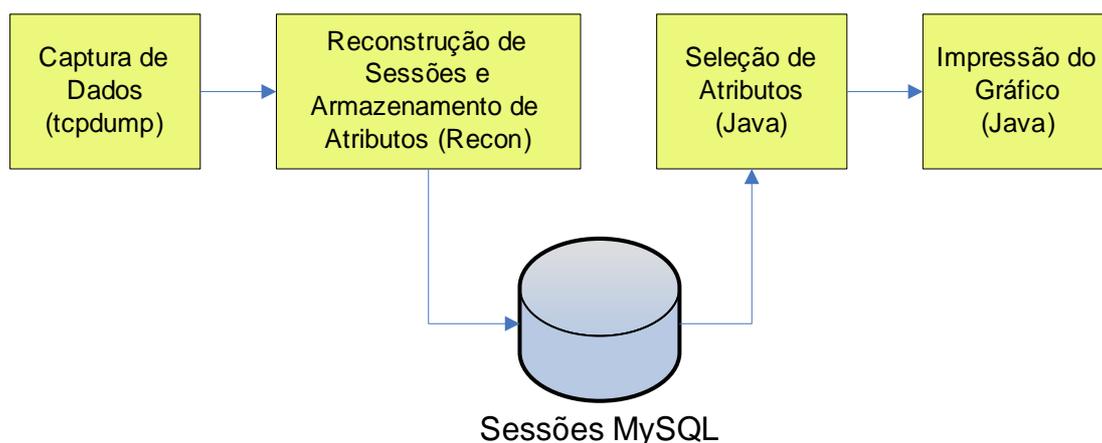


Figura 6.10 - Módulos da aplicação RGCOM

Para a captura de dados é utilizado o software *tcpdump* e para a reconstrução de sessões e gravação de dados utiliza-se o sistema Recon.

Foram implementados, no RGCOM, os módulos de seleção de atributos e de impressão do gráfico. Através do módulo de seleção de atributos o usuário pode escolher os atributos desejados para análise do comportamento da rede. A Figura 6.11 ilustra uma tela da aplicação, onde é possível selecionar os atributos desejados e o intervalo de sessões a que pertencem estes atributos.

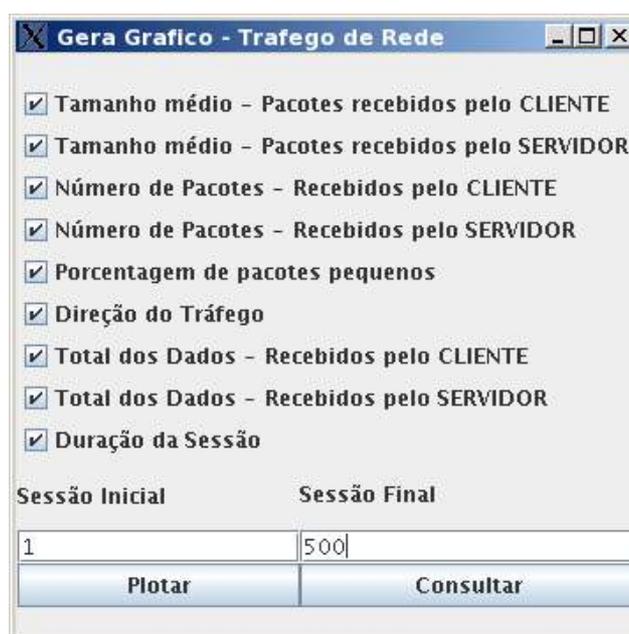


Figura 6.11 - Tela de Seleção de Atributos e Intervalo de Sessões

Quando informado o intervalo de sessões é feita uma busca na base de dados dos valores de máximo e mínimo dos atributos pertencentes ao intervalo selecionado que participarão dos cálculos para posicionamento dos pontos no gráfico.

Os pontos correspondentes aos valores dos atributos são impressos em eixos paralelos igualmente espaçados no gráfico, denominados 'Coordenadas Paralelas'. Em cada um destes eixos, que reproduzem uma dimensão dos dados, residem valores de um mesmo atributo.

Os valores dos atributos, que se diferenciam em escala e unidade de medida, são normalizados através do Teorema de Tales (WIKIPEDIA, 2007) de modo que exista uma proporcionalidade entre os pontos de atributos exibidos na tela, sendo apresentados dentro de um mesmo espaço amostral.

Interligando os pontos correspondentes de atributos nos eixos, o sistema apresenta linhas, onde cada uma representa uma sessão do tráfego. Ao conjunto de linhas de sessões impressas na tela, ou seja, a cada intervalo de tráfego, todas as linhas são desenhadas, pode-se observar o comportamento do tráfego selecionado em determinado período para análise.

Um exemplo de gráfico gerado pelo RGCOM é exibido na Figura 6.12. Este exemplo ilustra a representação gráfica do comportamento do tráfego HTTP de uma rede que descreve uma atividade normal de acesso a páginas Web em ambiente de teste controlado, através de 500 sessões normais do tráfego.

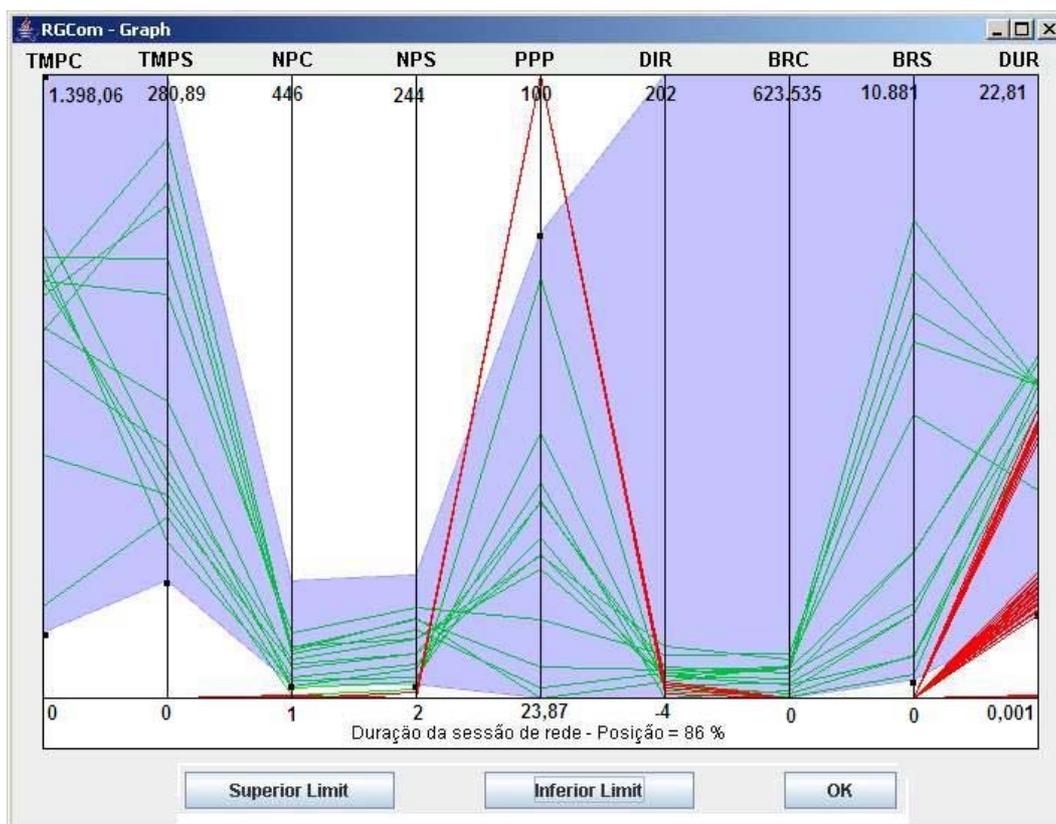


Figura 6.12 - Ilustração de um gráfico gerado pelo RGCOM¹

Nesta ilustração, observa-se a existência de regiões de maior concentração de dados de sessões, indicando os traços de comportamento normal da rede, através dos nove atributos selecionados. Em algumas regiões do gráfico o fluxo

¹ TMPC - tamanho médio dos pacotes recebidos pelo cliente
 TMPS - tamanho médio dos pacotes recebidos pelo servidor
 NPC - número de pacotes recebidos pelo cliente
 NPS - número de pacotes recebidos pelo servidor
 DIR - direção do tráfego
 PPP - porcentagem de pacotes pequenos
 BRC - total de dados de *payload (bytes)* recebidos pelo cliente
 BRS - total de dados de *payload (bytes)* recebidos pelo servidor
 DUR - duração da sessão

das linhas é mais intenso, demonstrando a existência de sessões de comportamento semelhante.

Uma das vantagens desta ferramenta é a possibilidade de visualização do tráfego em modo gráfico, apresentando um número reduzido de dados a observar, sendo cada sessão do tráfego representada por uma tupla de 9 atributos e cada linha do gráfico representando uma sessão.

A ferramenta RGCOM foi aperfeiçoada para permitir a visualização do tráfego em diferentes intervalos de tempo e foi inserida como um novo módulo no sistema ADTRAF (descrito na seção 6.3). Na versão atual da aplicação, ao invés de selecionar um intervalo de sessões, o usuário deve selecionar um intervalo de tempo, e as sessões ocorridas no período serão exibidas como linhas na tela.

Como trabalho futuro, pretende-se dar continuidade ao desenvolvimento desta aplicação, com os seguintes objetivos:

- realizar uma análise mais criteriosa de contribuição dos atributos, para definir a combinação destes ou de outros atributos que melhor representem o comportamento de diferentes tipos de tráfego de rede;
- prover exibição dinâmica das linhas de sessões, com o esmaecimento das linhas ao longo do tempo, mantendo um histórico temporário das sessões recentemente ocorridas para comparação com sessões correntes, de modo a facilitar a identificação de sessões suspeitas;
- permitir a seleção de uma linha e automaticamente exibir informações relevantes da sessão correspondente, tais como *timestamp*, IP de origem e IP de destino, porta de origem e de destino, de modo a identificar os *hosts* de origem dos ataques, o serviço de rede comprometido e o horário de ocorrência do evento. Esta técnica é conhecida como “*brushing*”.

6.3 O Sistema ADTRAF

Nesta seção é apresentado o sistema ADTRAF (*“Attack Detection on the network TRAFfic” – Detecção de Ataques no Tráfego de Redes*), construído com base na metodologia concebida nesta tese, incluindo uma visão geral do sistema, os recursos de desenvolvimento utilizados, sua arquitetura, módulos e interface gráfica.

6.3.1 Visão Geral do Sistema

O ADTRAF é um sistema de detecção de anomalias e assinaturas, baseado em redes, que realiza a análise *offline* (ou *pos-mortem*) das sessões TCP/IP contidas no tráfego de rede.

Este sistema possui uma arquitetura centralizada, em que o sensor do IDS coleta os dados de um ponto específico da rede e apresenta um comportamento passivo, visto que apenas alerta a ocorrência de eventos anômalos e não reage ao serem identificados tais eventos.

Dados extraídos dos cabeçalhos IP e TCP dos pacotes de rede são coletados, armazenados em arquivos, remontados em sessões e filtrados para a obtenção de atributos, os quais são utilizados como entrada para os módulos de detecção do sistema. Estes atributos do tráfego de rede são examinados em busca de anomalias e assinaturas (padrões de ataques).

O sistema foi desenvolvido com o propósito de realizar análises de sessões HTTP, mais especificamente das que utilizam a porta 80 da estação servidora Web, e detecção de ataques que envolvem uma única ou algumas sessões, mas podendo ser estendido para análise de outros protocolos de aplicação e para detecção de ataques que envolvem múltiplas sessões.

Para avaliação do sistema, foram realizados estudos de caso envolvendo ataques que deixam traços nos pacotes de sessões HTTP, na porta 80,

incluindo os ataques do tipo U2R e R2L, mais especificamente, os ataques de Buffer Overflow, SQL Injection e ataques a CGI/PHP.

O sistema ADTRAF foi construído com base na aplicação de técnicas de data mining não supervisionada para *clustering* (agrupamento) de dados e, supervisionadas, para classificação de dados. A técnica de *clustering* é implementada pela rede SOM (Mapa auto-organizado) para detecção de anomalias no tráfego. A técnica de classificação utiliza uma rede perceptron de múltiplas camadas - MLP para detecção de anomalias no tráfego e a rede LVQ é utilizada para a identificação de padrões de ataques no tráfego analisado. As técnicas supervisionadas requerem a apresentação dos padrões de ataque previamente rotulados. A técnica de *clustering* não requer a inserção de padrões de ataques para aprender o comportamento da rede, mas se encarrega de mapear os *clusters* ou agrupamentos de sessões com características semelhantes.

O sistema ADTRAF possui sete módulos, uma interface gráfica e uma base de dados com tabelas que armazenam dados do tráfego analisado, padrões de ataques e resultados de classificação dos detectores.

6.3.2 Recursos de Desenvolvimento e Teste

Este sistema foi desenvolvido em Java, é executado em plataforma operacional Linux Slackware e utiliza base de dados MySQL.

A linguagem Java foi utilizada neste trabalho, por fornecer recursos de interesse, tais como, programação orientada a objeto, boa comunicação com banco de dados e suporte gráfico satisfatório. Também fornece bibliotecas para a comunicação com redes e permite a criação de aplicativos independentes de hardware e sistema operacional. Deste modo, o programa pode ser compilado em qualquer sistema operacional que tenha o Java, e seu arquivo binário resultante pode ser executado em qualquer sistema operacional.

MySQL é um sistema de gerenciamento de banco de dados (SGBD) relacional de código-fonte aberto, que permite a criação rápida de bancos de dados bem estruturados. Pode ser instalado em diferentes sistemas operacionais e provê recursos úteis de gerenciamento, tais como mecanismos de *backup*, importação e exportação de dados.

Para o desenvolvimento do sistema ADTRAF foram utilizadas estações clientes e servidoras. Em um computador foi implantada a base de dados MySQL para armazenamento de dados de entrada, padrões de ataques, dados de saída e para consulta a estes dados. Com os dados desta base são validadas as rotinas dos mecanismos de detecção. Também são construídos os gráficos e relatórios apresentados pelo sistema.

Os dados introduzidos para análise do sistema foram sinteticamente produzidos na rede do Laboratório de Redes da DSS (RedeLab) e foram obtidos da rede do Prédio Beta (RedeBeta), ambas são redes internas do INPE.

Para a captura de pacotes de rede foram configurados dois sensores de IDS:

- um sensor posicionado entre o *gateway/servicoWeb* e a RedeLab. Nesta máquina foram instalados os softwares *tcpdump* para a coleta de pacotes e o sistema *Recon* para a reconstrução das sessões do tráfego.
- outro sensor instalado para a captura de pacotes de rede da RedeBeta. Nesta máquina foram inseridos, além do *tcpdump* e do *Recon*, *scripts* para gravação de dados do tráfego da RedeBeta em disco em janelas de 10 minutos e para a cópia automática de dados para uma máquina remota em outro prédio em intervalos de uma hora. O procedimento de backup remoto dos dados deste sensor foi abandonado após a aquisição de uma máquina com mais recursos de memória e espaço em disco e com a disponibilidade de uma leitora/gravadora de DVD, para backup dos dados.

Para os testes do sistema ADTRAF com dados de testes de tráfego HTTP produzidos na RedeLab, foi instalada e configurada uma estação servidora *Web Apache*, contendo a base MySQL e o programa CGI-PHP que foi utilizado na construção de páginas estáticas e dinâmicas residentes neste servidor. Esta máquina também funciona como *gateway* para os testes de acesso a páginas externas.

Os resultados do sistema foram analisados através de tabelas, gráficos e consultas à base de dados.

6.3.3 Arquitetura do Sistema

A arquitetura do sistema de detecção de anomalias desenvolvido é composta de sete principais módulos, quatro destes apresentados na Figura 6.13, incluindo: “Módulo de Captura de Pacotes”, “Módulo de Reconstrução de Sessões”, “Módulo de Extração de Atributos”, “Módulo de Detecção de Assinaturas”, “Módulo de Detecção de Anomalias”, “Módulo de Alerta de Ataques” e “Módulo de Representação Gráfica do Tráfego”.

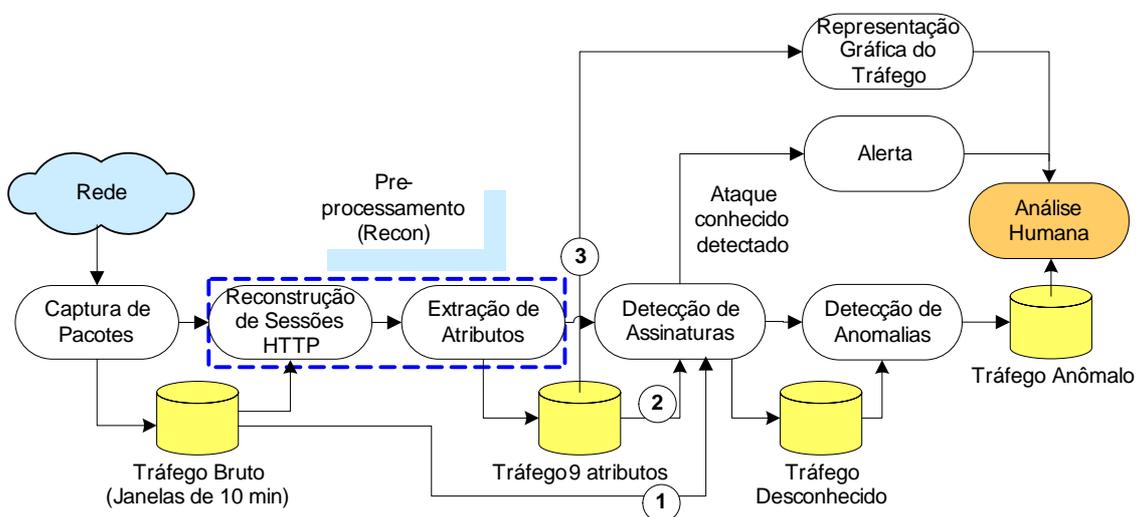


Figura 6.13 - Arquitetura Modular do Sistema ADTRAF

Com o sistema ADTRAF é possível detectar assinaturas a partir de dados do tráfego de rede bruto em janelas de 10 minutos, fazendo uma análise do *payload* de cada pacote de rede (identificação 1 – análise de conteúdo).

Também é possível realizar a detecção de assinaturas a partir de dados do cabeçalho dos pacotes, utilizando o tráfego reduzido para sessões de 9 atributos (identificação 2 – análise de cabeçalho), e em seguida, é possível realizar a detecção de anomalias sobre o tráfego desconhecido, ou seja, que passou pelo detector de assinaturas como tráfego normal. E o sistema ainda permite a análise visual do comportamento do tráfego de rede (identificação 3 – análise visual).

Os módulos de “Reconstrução de Sessões” e “Extração de Atributos” compõem o sistema Recon que foi atualizado neste trabalho para armazenamento de atributos em base MySQL para uso pelo sistema ADTRAF.

Além do módulo de “Captura de Pacotes”, técnicas de *data mining* (especialmente, classificação e *clustering*) foram empregadas na construção dos módulos “Detecção de Assinaturas” e “Detecção de Anomalias”.

O “Módulo de Representação Gráfica do Tráfego” no sistema ADTRAF corresponde à ferramenta RGCOM que foi acoplada ao sistema para auxiliar na análise visual do comportamento do tráfego de rede, permitindo a identificação de sessões suspeitas.

6.3.4 Módulo de Captura de Pacotes

Um computador na rede denominado sensor foi configurado para a captura de pacotes, utilizando a biblioteca *libpcap* através da ferramenta *tcpdump* e *scripts* para coleta de dados e armazenamento em intervalos regulares. A Figura 6.14 ilustra este procedimento.

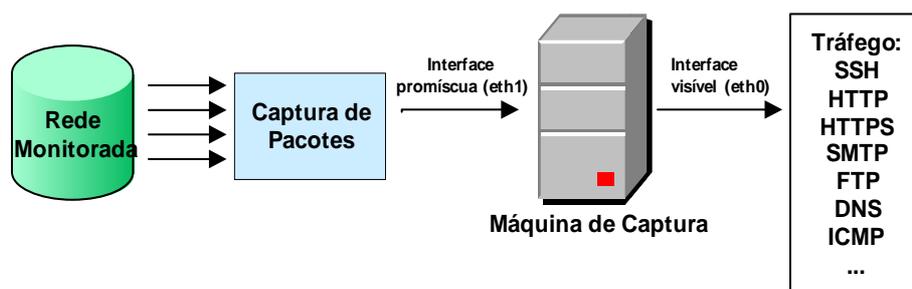


Figura 6.14 – Esquema de Captura de Pacotes

O sensor para captura de pacotes do sistema ADTRAF foi posicionado entre o gateway e as máquinas da rede interna monitorada.

6.3.5 Processo de Reconstrução de Sessões

Dados de entrada em arquivos no formato *tcpdump* contendo o tráfego de rede são introduzidos no Recon e processados para remontagem de sessões e extração de atributos. Como saída, o Recon gera um conjunto de dados referentes às sessões, armazenados em uma estrutura de árvore binária balanceada que permite acesso facilitado, com bom desempenho. Esta aplicação está preparada para capturar dados do cabeçalho dos pacotes TCP, UDP e ICMP (em nível de Transporte), IP (em nível de Rede) e Ethernet (em nível de Enlace).

6.3.6 Processo de Extração de Atributos

Cada sessão de dados utilizada nos estudos de casos desta tese é composta de 9 atributos que refletem o comportamento de uma sessão do tráfego de rede, por exemplo, porcentagem de pacotes pequenos, número de bytes recebidos pelo cliente, entre outros. Estes atributos são construídos a partir de atributos primitivos das sessões do tráfego, tais como, tipo de protocolos, quantidade de bytes provenientes de uma origem, etc. Uma ilustração do processo de reconstrução de sessões é apresentada na Figura 6.15.

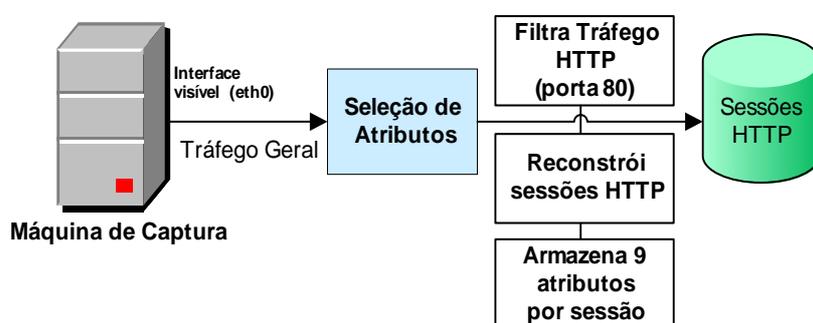


Figura 6.15 - Esquema da Reconstrução de Sessões do Tráfego de Rede

Neste módulo, os dados brutos do tráfego de rede são introduzidos no Recon para extração de pacotes referentes às sessões HTTP, reconstrução das sessões e gravação dos dados em disco.

Todos os atributos utilizados na implementação do sistema ADTRAF são apresentados na Tabela 6.2.

Tabela 6.2 - Atributos utilizados na composição de cada sessão de rede

Sigla	Descrição do Atributo	Unidade
TMPC	Tamanho médio dos pacotes recebidos pelo cliente	bytes
TMPS	Tamanho médio dos pacotes recebidos pelo servidor	bytes
NPRC	Número de pacotes recebidos pelo cliente	-
NPRS	Número de pacotes recebidos pelo servidor	-
DIR	Direção do tráfego (valor inicial = 0) (cliente recebe pacote (+1), servidor recebe (-1))	-
PPP	Porcentagem de pacotes pequenos (irregulares) *	%
TDRC	Total de dados (de payload) recebidos pelo cliente	bytes
TDRS	Total de dados (de payload) recebidos pelo servidor	bytes
DUR	Duração da sessão	segundos
STA	Estado da Sessão (valores inteiros de 0 a 22)	-
TIM	Horário de captura do primeiro pacote da sessão	HH:MM:ss:mmmm
IPA	Menor endereço IP da sessão	-
IPB	Maior endereço IP da sessão	-
PA	Porta do menor endereço IP da sessão	-
PB	Porta do maior endereço IP da sessão	-

Na análise do tráfego HTTP realizada nesta tese é considerado endereço IP de uma estação servidora, o IP precedido da porta 80 na sessão registrada. Para cada pacote da sessão direcionado ao servidor, a direção do tráfego, que inicialmente vale zero, é decrementada de 1. Para cada pacote da sessão direcionado ao cliente, a direção é acrescida de 1. O parâmetro de direção do tráfego (*data_dir*) tem por objetivo mostrar o quão balanceada é a sessão. No caso de sessões onde ocorre o *download* de arquivos, por exemplo, o número de pacotes recebido pelo cliente tende a ser maior que o número de pacotes recebidos pelo servidor.

6.3.7 Módulo de Detecção de Assinaturas

Neste módulo utiliza-se a rede neural LVQ, cuja arquitetura contém nove neurônios na camada de entrada e o número de neurônios equivale à quantidade de sessões de ataques apresentada à rede como padrões de ataque (vetor de exemplares) na camada de saída. A Figura 6.16 ilustra a arquitetura desta rede.

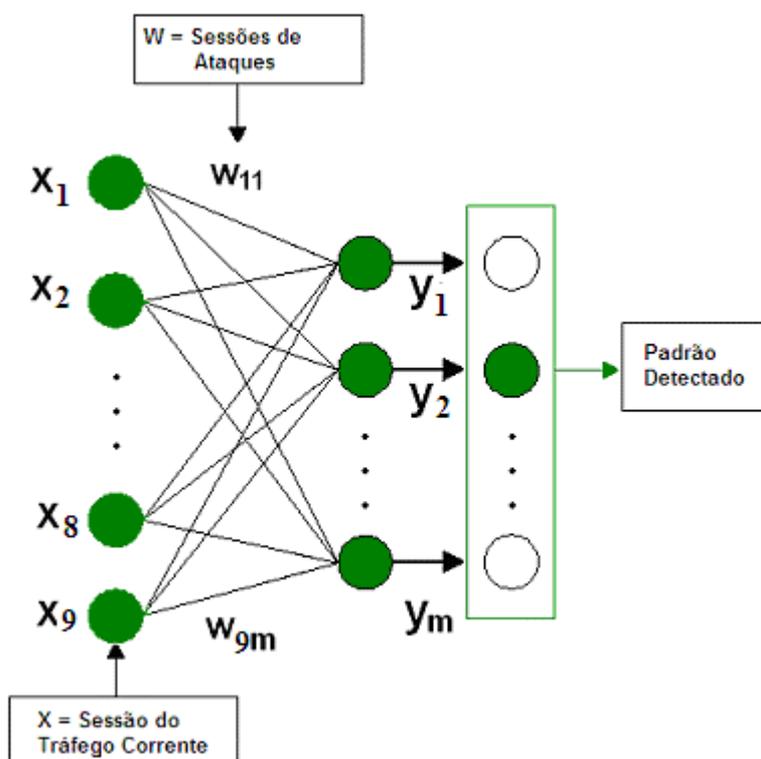


Figura 6.16 - Arquitetura da rede LVQ no sistema ADTRAF

Cada entrada é um vetor com valores dos nove atributos considerados, representando um registro de sessão do tráfego analisado.

O vetor de exemplares da rede LVQ é o conjunto de registros de sessões de ataques previamente coletadas no ambiente controlado de lançamento de ataques. Estes exemplares são inseridos na rede como os pesos que representam o conhecimento envolvido na tarefa de reconhecimento. Assim, os pesos equivalem aos exemplares de ataques disponíveis.

Quando submetido a uma avaliação de tráfego, o algoritmo LVQ calcula a distância Euclidiana entre o vetor de entrada e os vetores de exemplares. A menor distância Euclidiana obtida indica o neurônio de saída vencedor, correspondente à assinatura de ataque que casa com o padrão de entrada segundo um grau de similaridade (100%) entre os valores de atributos de sessões de entrada e dos exemplares.

Neste módulo o campo “decisão” no registro de cada sessão do tráfego analisada na base de dados é preenchido com o nome do ataque detectado ou com a identificação ‘*Unknown*’, indicando que a sessão não é de ataque, mas desconhecida para este módulo. Esta saída é exibida na tela através do módulo de Alerta.

6.3.8 Módulo de Detecção de Anomalias

Neste módulo foram implementados os métodos de detecção usando as redes SOM e MLP. As sessões do tráfego classificadas pelo método de detecção de assinaturas como “*Unknown*” são introduzidos na rede neural para detecção de anomalias e a saída deste módulo é a atualização do campo “decisão” das sessões do tráfego na base de dados com o número de identificação 0 (sessão normal) ou 1 (sessão anômala).

6.3.9 Módulo de Alerta

O sistema fornece um módulo de geração de alertas, que exhibe uma lista de ataques, identificados no conjunto de dados de tráfego analisado, com as seguintes informações: identificação do ataque, par de endereços IP associados à sessão e hora de captura do primeiro pacote da sessão de ataque, como ilustrado na seção 6.3.11.4.

6.3.10 Módulo de Apoio à Análise Visual do Tráfego

O Módulo de Apoio à Análise Visual do Tráfego foi construído a partir do acoplamento da ferramenta RGCOM ao sistema ADTRAF. Este módulo auxilia o analista de rede na análise do comportamento do tráfego através de

representação gráfica dos atributos das sessões do tráfego monitorado, conforme mostrado na seção 6.3.11.3.

6.3.11 Interface Gráfica

A interface gráfica do sistema ADTRAF foi construída utilizando-se recursos das bibliotecas e classes Java para a construção de telas, campos, botões e desenhos no sistema. A seguir são apresentadas algumas janelas do sistema.

6.3.11.1 Janela Principal

A janela principal do sistema é apresentada na Figura 6.17 e contém menus para seleção de arquivos (menu File) e para tratamento de dados (menu Database). Nesta janela está localizado o campo para seleção do tráfego a ser analisado.

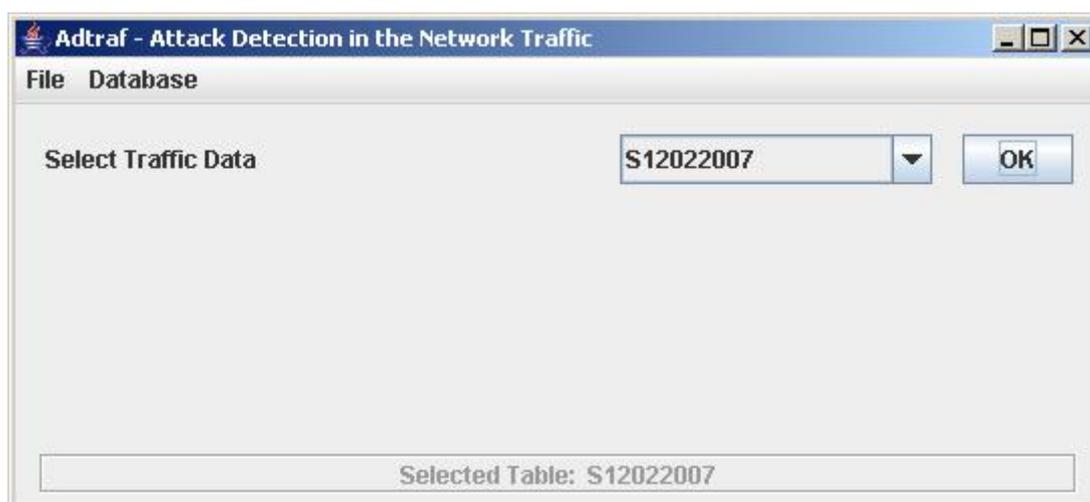


Figura 6.17 - Janela Principal para Seleção do Tráfego a ser Analisado

6.3.11.2 Seleção de Métodos

A partir do menu File e opção "Signature Detection", pode-se selecionar o método LVQ para detecção de assinaturas, como ilustrado na Figura 6.18.

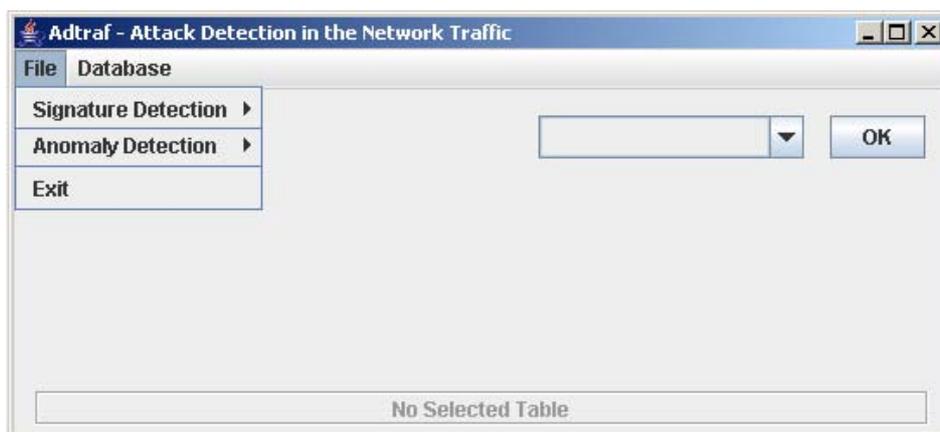


Figura 6.18 - Janela para seleção de técnicas de análise do tráfego

Também a partir do menu File é possível selecionar um dos métodos de detecção de anomalias, através da opção “*Anomaly Detection*”. Uma das técnicas MLP, SOM, LOF ou LSC pode ser selecionada.

6.3.11.3 Classificação Manual do Tráfego

Para auxiliar a análise dos resultados dos métodos de detecção, o sistema ADTRAF oferece uma ferramenta para classificação manual do tráfego, para comparação futura das sessões manualmente classificadas como normais ou anômalas com os resultados obtidos.

Através da janela apresentada na Figura 6.19 é possível escolher o item de menu “*Manual Selection of Normal Behavior*” para a classificação das sessões.

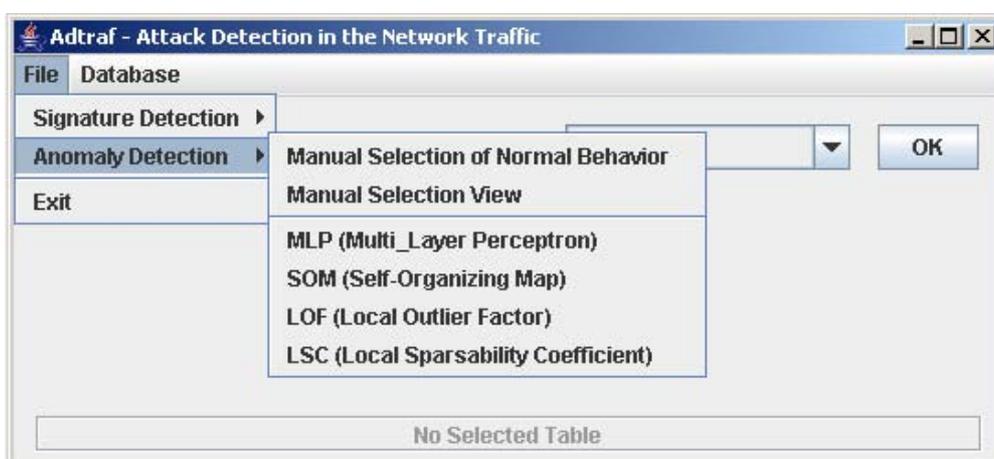


Figura 6.19 - Janela para Escolha do Detector de Anomalias

Ao selecionar a opção “*Manual Selection of Normal Behavior*” a janela da Figura 6.20 é exibida, permitindo a seleção da tabela de tráfego e do intervalo de tempo desejado para classificação manual.

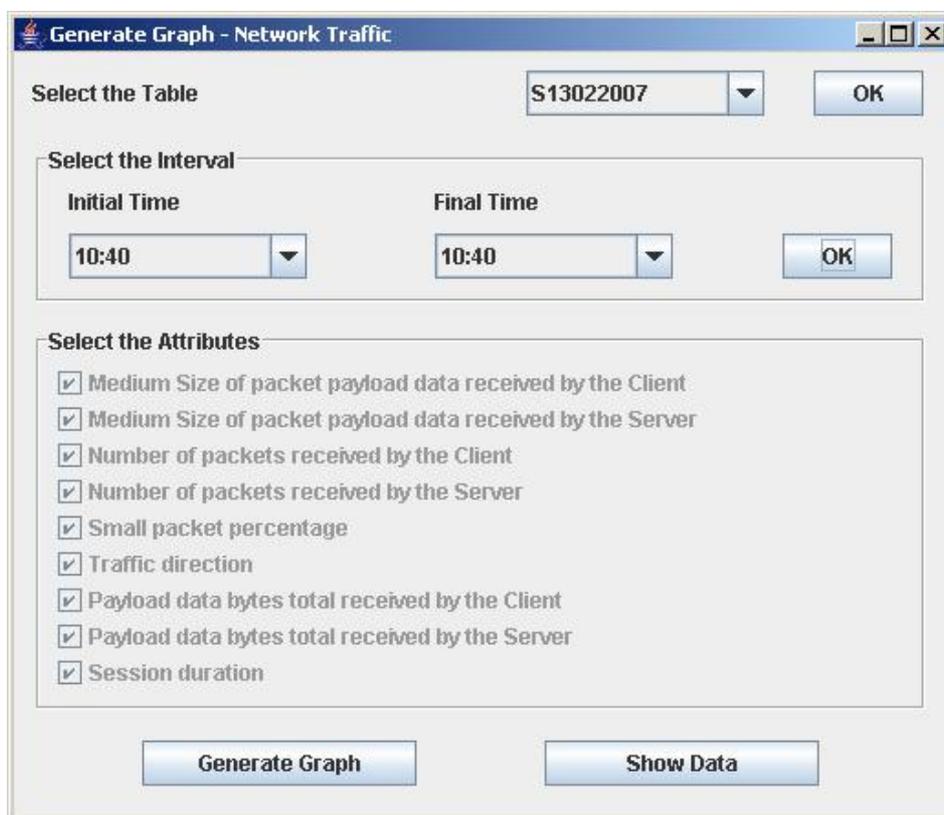


Figura 6.20 - Janela para seleção de horário e tráfego a exibir

Após seleção da tabela de tráfego a ser classificada e do intervalo de tempo desejado, ao clicar no botão “*Show Data*” é exibida a quantidade de registros da tabela selecionada.

Ao escolher a opção “*Generate Graph*” os atributos das sessões do tráfego são desenhados na tela. A ferramenta permite selecionar uma região, onde todas as sessões contidas serão marcadas como tráfego normal, como mostra a Figura 6.21.

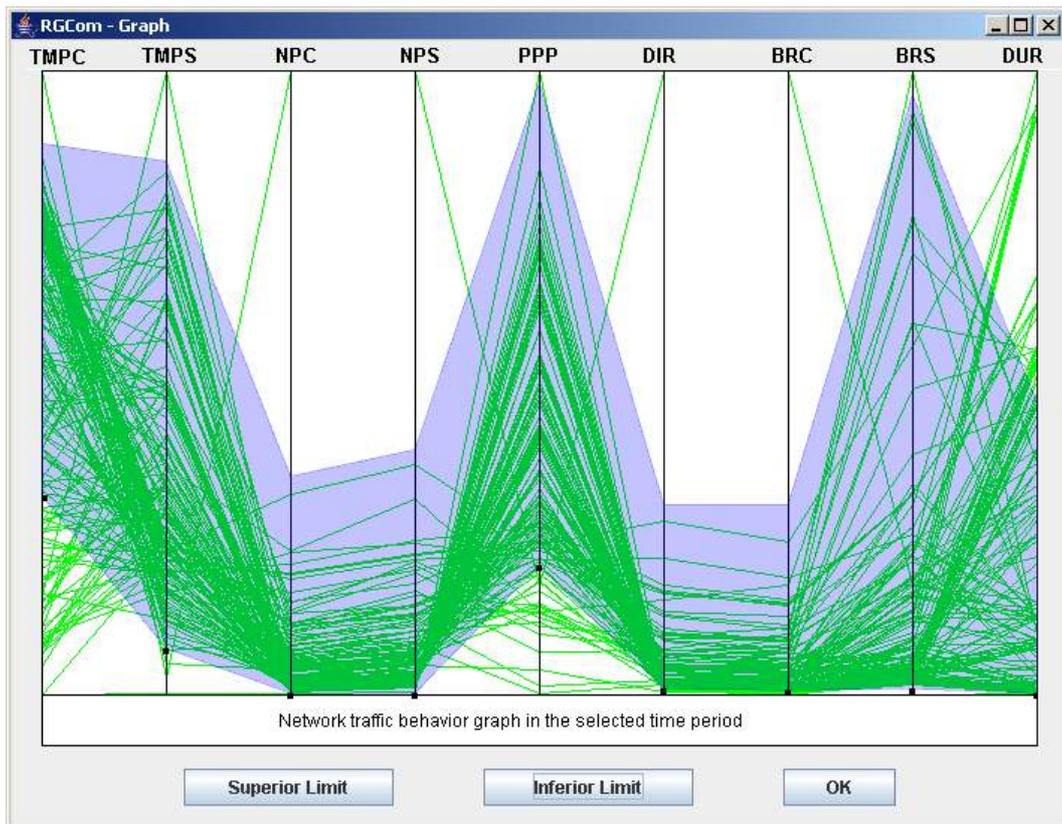


Figura 6.21 - Seleção de região do tráfego para classificação de sessões normais

Ao invés de selecionar a região que o analista considera de comportamento normal do tráfego, é possível clicar nos botões “*Superior Limit*”, “*Inferior Limit*” e “*OK*” para que toda a região no gráfico seja marcada como tráfego normal. Para as sessões dentro desta região, o sistema sinaliza o respectivo atributo de decisão na Tabela MySQL com o valor 0 (zero), indicando ser tráfego normal e sessões fora da região (anômalas) terão o atributo de decisão com valor 1 (um).

Após a classificação manual pelo analista ou automática, através das ferramentas de detecção, de sessões normais e anômalas, o sistema é capaz de desenhar o gráfico com cores diferentes para sessões normais e anômalas, a partir do seleção do botão “*View Manual Selection*” do menu “*File*”. Uma tela solicitando a tabela do tráfego a ser apresentado graficamente, conforme Figura 6.22, é exibida.

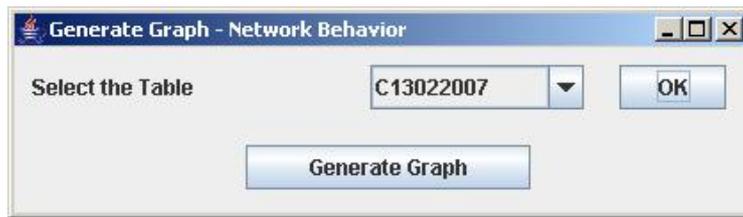


Figura 6.22 - Seleção de Tabela para gerar gráfico

Um tráfego classificado manual ou automaticamente gera uma tela semelhante a da Figura 6.23. O resultado apresentado nesta Figura exibe status das sessões: Normais em verde (valor 0) e Anômalas em azul (valor 1). Este gráfico auxilia o especialista a realizar uma verificação do comportamento do tráfego. Este recurso encontra-se em desenvolvimento.

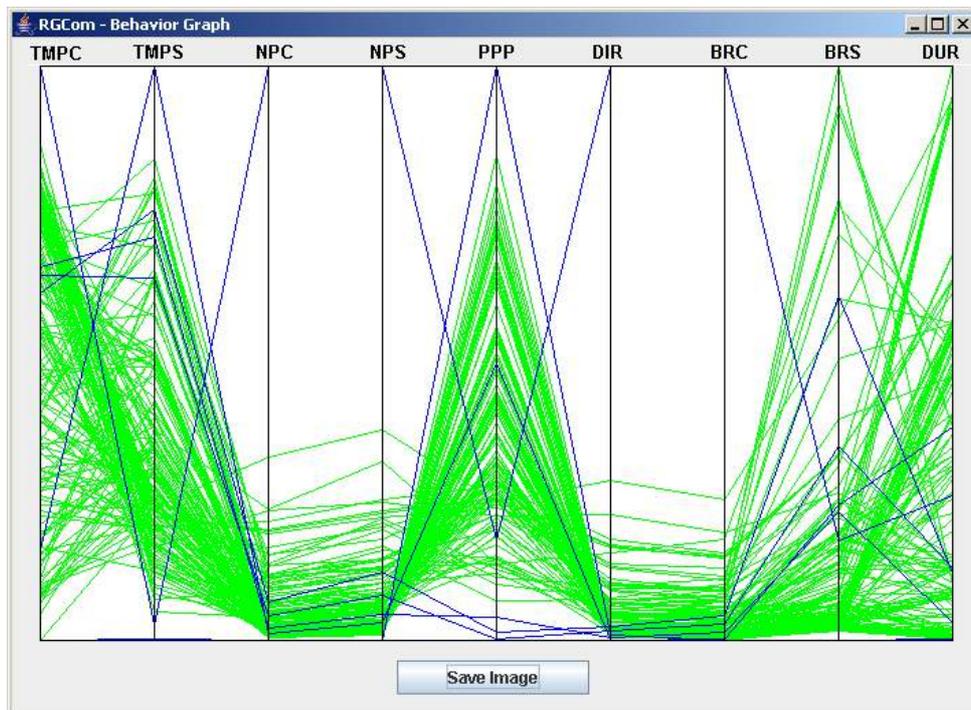


Figura 6.23 - Visualização Gráfica de Tráfego Classificado

6.3.11.4 Detecção de Assinaturas

Com a tela da Figura 6.24 é possível selecionar a ferramenta para detecção de assinaturas no tráfego desejado. As assinaturas neste contexto são os padrões de comportamento do tráfego detectados a partir de dados do cabeçalho (atributos de sessões) dos pacotes de rede e utilizando a rede neural LVQ.

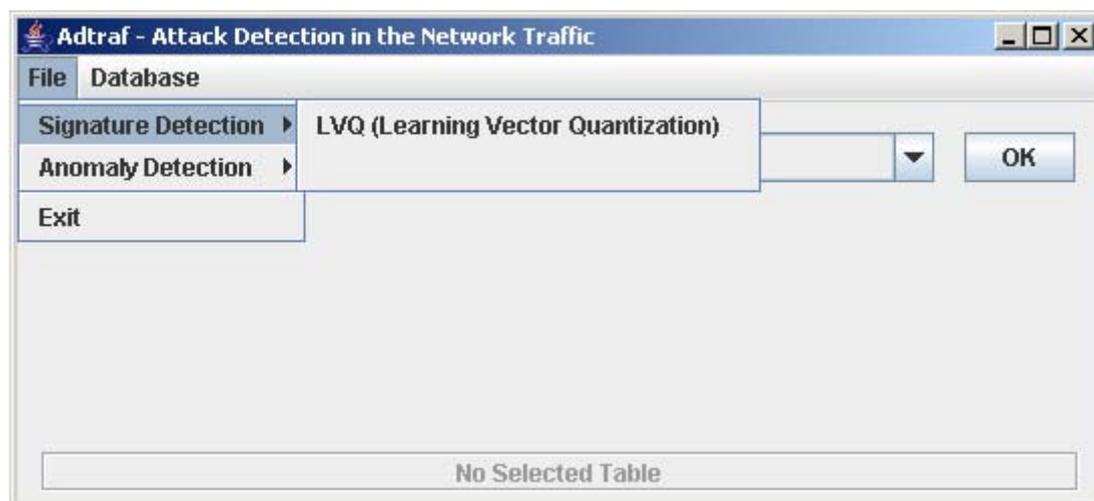


Figura 6.24 - Janela com as opções de classificação do tráfego

Ao selecionar a opção de menu "Learning Vector Quantization (LVQ)" do menu "File -> Signature Detection" é exibida a tela da Figura 6.25.

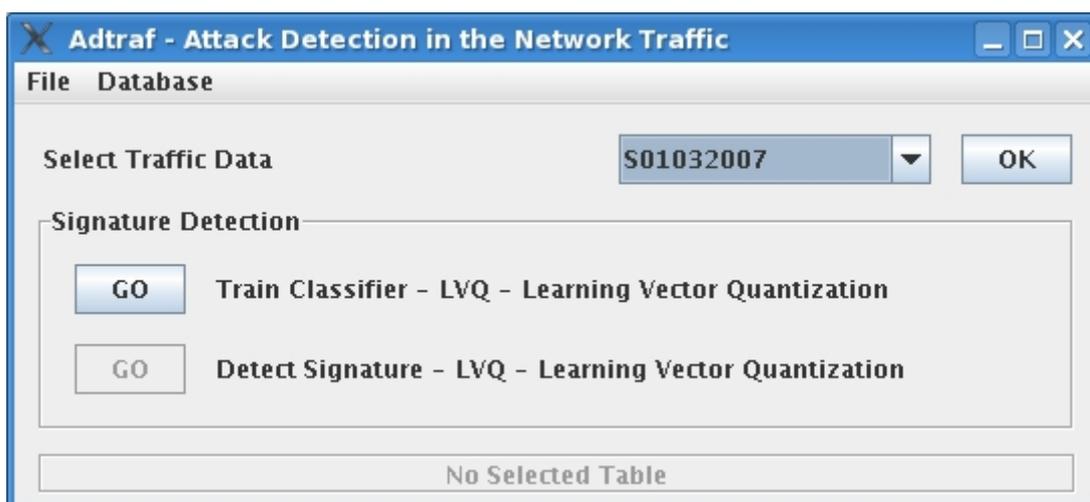


Figura 6.25 - Seleção de opções para classificação LVQ

Deve-se escolher a opção "Train Classifier LVQ" para introduzir na rede neural a base de assinaturas (treinamento supervisionado) e a opção "Detect Signature" para que o classificador realize a busca por padrões de ataques no tráfego.

O resultado da classificação do tráfego através da rede LVQ é apresentado pelo Módulo de Alerta que exibe na tela uma janela, como a da Figura 6.26, com informações sobre as sessões de ataque detectadas.

	Name	Source IP	Destination IP	Time
0	sugar	192.168.0.25	192.168.0.254	10:52
1	sugar	192.168.0.25	192.168.0.254	10:52
2	webi	192.168.0.25	192.168.0.254	15:05
3	sumo	192.168.0.25	192.168.0.254	15:46

Figura 6.26 - Janela de alerta de ataques

As informações apresentadas nesta janela, IP origem (atacante), IP destino (alvo), horário e tipo de ataque identificado, auxiliam na execução de ações preventivas ou na realização de futuras alterações na configuração da rede para impedir que os ataques previamente detectados voltem a ocorrer.

6.3.11.5 Detecção de Anomalias

As ferramentas para detecção de anomalias no tráfego são selecionadas a partir da tela apresentada na Figura 6.27.

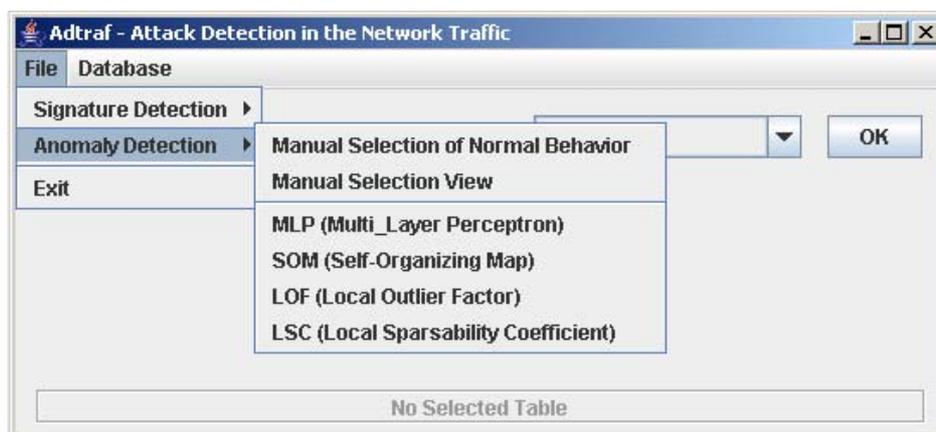


Figura 6.27 - Janela para a Seleção do Método de Detecção de Anomalias

Após seleção do tráfego desejado para análise, deve-se escolher o tipo de método de detecção de anomalias que será utilizado.

Ao selecionar a opção “Multi-Layer Perceptron” (MLP), a tela apresentada na Figura 6.28 é exibida e o analista pode escolher entre treinar o modelo, através

da opção “*MLP Supervised Training*” ou, utilizando os pesos de treinamento prévio, disparar a detecção de anomalias através da opção “*MLP Traffic Data Classification*”.

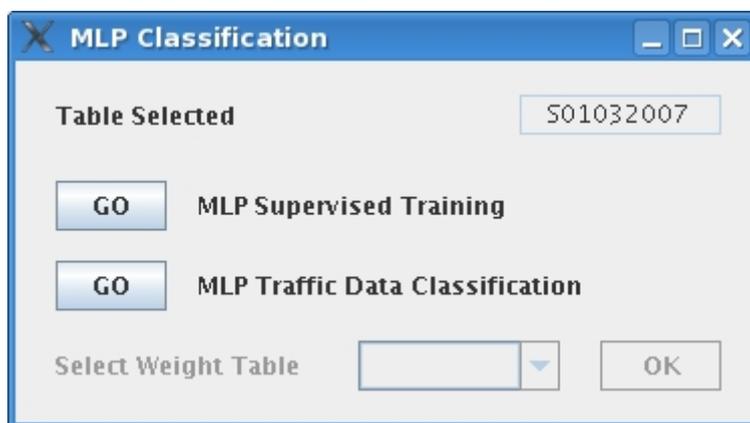


Figura 6.28 - Janela com Opções de Treinamento ou Classificação MLP

De modo recíproco, deve-se selecionar a opção para processamento do da rede neural SOM, utilizando a tabela apresentada na Figura 6.29.



Figura 6.29 - Janela com Opções de Treinamento ou *Clustering* SOM

6.3.11.6 Manipulação da Base de Dados

No sistema ADTRAF também são fornecidos recursos para facilitar a utilização das bases de dados do sistema. Dentre os recursos utilizados no ambiente de gerenciamento de base de dados MySQL, implementou-se um mecanismo para selecionar a localização da base de dados onde reside o tráfego a ser

analisado, mesmo que a base esteja em computador remoto. Através da janela mostrada na Figura 6.30 isto é possível.



Figura 6.30 - Janela para seleção da localização da base de dados

Outro recurso implementado permite excluir automaticamente tabelas do banco, utilizando-se o menu “Database”, opção “Drop Table”, como ilustrado na Figura 6.31.



Figura 6.31 - Janela para seleção de Tabelas a excluir do banco

Recursos adicionais para cálculo de valores médios e valores randômicos entre atributos de diferentes sessões encontram-se implementados no sistema, juntamente com o recurso de seleção de atributos, que cria uma tabela diferente a cada remoção de um atributo do conjunto. Estas ferramentas foram utilizadas em estudos de casos para verificar a contribuição dos atributos no processo de detecção e para testes de precisão do modelo.

7 ESTUDO DE CASOS

A fim de analisar a viabilidade de uso da metodologia desenvolvida para detecção de ataques à rede, alguns estudos de casos, envolvendo análise de tráfego HTTP, foram conduzidos.

As redes neurais Hamming Net e LVQ foram utilizadas nos estudos para a detecção de assinaturas no tráfego e as redes neurais SOM, LVQ e MLP foram empregadas para a detecção de anomalias no tráfego de rede.

Os atributos utilizados para análise das sessões de tráfego HTTP foram:

- TMPC - tamanho médio dos pacotes recebidos pelo cliente;
- TMPS - tamanho médio dos pacotes recebidos pelo servidor;
- NPC - número de pacotes recebidos pelo cliente;
- NPS - número de pacotes recebidos pelo servidor;
- DIR - direção do tráfego;
- PPP - porcentagem de pacotes pequenos;
- BRC - total de dados de *payload (bytes)* recebidos pelo cliente;
- BRS - total de dados de *payload (bytes)* recebidos pelo servidor;
- DUR - duração da sessão.

As fases, métodos, técnicas e ferramentas, que compõem a metodologia desenvolvida, foram utilizadas nesta tese, nos estudos de casos de detecção de ataques no tráfego HTTP e encontram-se resumidos na Tabela 7.1.

Tabela 7.1- Fases da Metodologia Proposta

Fases	Objetivo	Métodos e Técnicas	Ferramentas
Captura de tráfego de rede	Coletar dados brutos de tráfego cuja natureza e dinâmica será observada	<ul style="list-style-type: none"> Desenvolvimento de <i>script</i> para captura de pacotes de rede Gravação de dados do tráfego em janelas de 10 minutos 	Tcpdump
Seleção de tráfego Web	Extrair dados de tráfego HTTP para análise	<ul style="list-style-type: none"> Pré-filtragem de sessões relacionadas à porta 80 	Tcpdump
Armazenamento de Dados	Manter dados (assinaturas e tráfego histórico), em disco, para uso futuro nos sistemas e para análise dos resultados de classificação	<ul style="list-style-type: none"> Planejamento da estrutura da base de dados Gerenciamento da base de dados Desenvolvimento de recursos para facilitar a seleção de base de dados em estação remota e para remover Tabelas de dados de modo mais automatizado 	MySQL Adtraf
Tratamento e Modelagem de Dados	Preparar os dados para entrada nos algoritmos de detecção	<ul style="list-style-type: none"> Normalização de dados dividindo o valor de cada atributo pelo valor máximo dos atributos relacionados Uso de algoritmo de <i>hashing</i> para modelar entrada única para a detecção de assinaturas no conteúdo do pacote 	Adtraf Matlab Annida
Reconstrução de Sessões e seleção de atributos	Extrair atributos relevantes para análise de tráfego HTTP	<ul style="list-style-type: none"> Atualização de ferramenta de reconstrução de sessões para gravação de atributos em base de dados 	Recon
Detecção de Assinaturas	Identificar sessões de ataques conhecidos no tráfego analisado	<ul style="list-style-type: none"> Análise de conteúdo (<i>payload</i>) dos pacotes Análise de cabeçalho dos pacotes Uso de redes neurais de rápida classificação Hamming Net e LVQ 	Annida Adtraf

(continua)

Tabela 7.1 - Fases da Metodologia Proposta (conclusão)

Fases	Objetivo	Métodos e Técnicas	Ferramentas
Detecção de Anomalias	Identificar sessões anômalas no tráfego analisado	<ul style="list-style-type: none"> Análise de cabeçalho de pacotes Uso de redes neurais LVQ e MLP 	Adtraf Matlab
Análise de contribuição dos atributos	Verificar a influência dos atributos na identificação de sessões suspeitas	<ul style="list-style-type: none"> Análise visual do comportamento do tráfego Sugestão de remoção de um atributo por vez 	Adtraf Matlab
Geração de Tráfego Normal e Anômalo de Rede de Teste	Preparar conjuntos de dados simulados para treinamento supervisionado e não supervisionado e teste dos modelos de detecção	<ul style="list-style-type: none"> Lançamento de ataques em rede de teste, captura e reconstrução de sessão Acesso normal a serviços da rede de teste, captura e reconstrução de sessão Classificação manual de sessões anômalas (0) e normais (1) 	Adtraf
Geração de Tráfego Normal e Anômalo de Rede de Produção	Preparar conjuntos de dados reais para treinamento e teste dos modelos de detecção	<ul style="list-style-type: none"> Uso de mecanismo de detecção para identificar tráfego anômalo Remoção de tráfego anômalo do tráfego da rede produção Desenvolvimento de rotina para a remoção de tráfego alertado pelo Snort 	Snort Dragon <i>Script</i> remove_ataques
Observação de Tráfego e análise de respostas dos sistemas	Analisar o tráfego em busca de eventos de ataques ou desvios de comportamento	<ul style="list-style-type: none"> Observação de volumes diferentes de dados de tráfego Observação de tráfego em diferentes períodos Análise de resultados dos métodos de detecção, com auxílio de ferramenta gráfica e recursos de consulta a banco de dados 	Adtraf / Rgcom MySQL

Para os estudos realizados, conjuntos de sessões normais e anômalas de tráfego HTTP reconstruídas a partir de pacotes capturados do tráfego de redes internas do INPE foram utilizados. A descrição dos ataques selecionados encontra-se na seção 7.2.

Na Tabela 7.1 a coluna 'Ferramentas' relaciona as ferramentas utilizadas nos estudos de casos conduzidos nesta tese. Entretanto, a metodologia permite que ferramentas adequadas à realidade de cada usuário sejam utilizadas.

7.1 Geração de Tráfego Normal e Tráfego Anômalo

Com mencionado anteriormente, dois ambientes de rede internos do INPE foram utilizados para análise de tráfego: a RedeLab e a RedeBeta. A RedeLab é um ambiente de rede controlado do Laboratório de Pesquisa e Desenvolvimento em Redes da Divisão de Desenvolvimento de Sistemas de Segmento Solo. A RedeBeta é o ambiente de rede de produção do Prédio Beta.

Para a geração de tráfego normal da RedeLab foram instalados e configurados em um host nesta rede: serviço Web Apache, aplicação de Banco de Dados MySQL, software PHP e algumas páginas Web estáticas e dinâmicas desenvolvidas. Esta máquina também foi utilizada como *gateway* entre a RedeLab e a Internet, para a captura de dados de acesso normal a páginas Web externas através de solicitações provenientes da RedeLab. Em máquinas-cliente foram instalados os navegadores Microsoft Explorer e FireFox e ferramentas para compilação de *scripts* em linguagem C.

Através de máquinas-cliente na RedeLab foram realizados acessos normais ao serviço HTTP disponível na RedeLab e acesso normal a páginas Web da Internet. Os pacotes de rede do tráfego interno da RedeLab (comunicação entre máquinas clientes e servidora internas) e provenientes da Internet (comunicação entre máquinas clientes internas e servidoras externas) foram capturados através do software *tcpdump*, com o *sensor* IDS posicionado dentro

dos limites de proteção do *firewall* da RedeLab e antes do *gateway* da rede, conforme ilustrado na Figura 7.1. Isto foi possível pelo espelhamento do tráfego passante pela porta do *gateway* na porta do Sensor no *switch*. As sessões normais de rede foram reconstruídas e armazenadas no banco através do sistema Recon.

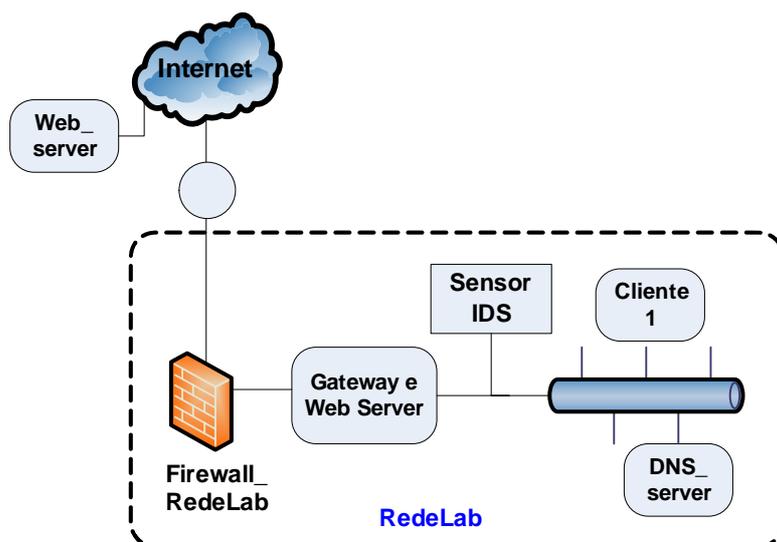


Figura 7.1 - Ambiente para Captura de Pacotes da RedeLab

Dados de tráfego anômalo da RedeLab foram capturados pelo sensor IDS. Através de uma estação cliente da rede interna foram lançados ataques contra o serviço Web disponibilizado nesta mesma rede. Toda a comunicação entre a estação cliente e a estação servidora foi registrada pelo sensor.

A principal dificuldade encontrada nesta etapa do trabalho foi a implantação do ambiente Web vulnerável e a busca de *exploits* (programas maliciosos de exploração de vulnerabilidades e falhas de serviços de rede e de aplicações em rede) adequados para exploração do serviço e aplicações Web configuradas no Laboratório.

Dados de tráfego legítimo da RedeBeta foram gerados da seguinte forma: os pacotes da RedeBeta foram capturados com o sensor IDS posicionado fora dos limites de proteção do *firewall* da RedeBeta conforme ilustrado na Figura 7.2. A porta do *firewall* da RedeBeta foi espelhada no *switch* da RedeBeta para a

porta do Sensor IDS, possibilitando a passagem de todo o tráfego de entrada e saída desta rede para o sensor IDS.

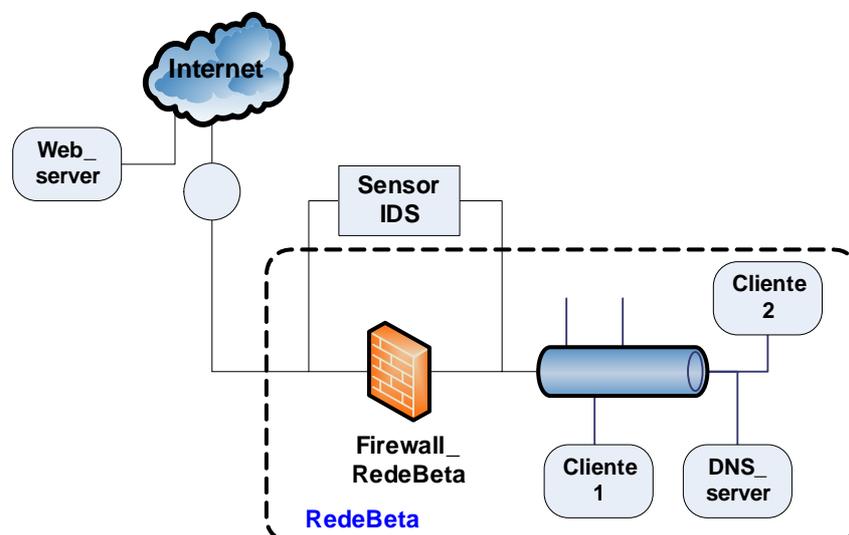


Figura 7.2 - Ambiente para Captura de Pacotes da RedeBeta

A principal dificuldade encontrada nesta etapa foi a configuração correta dos *scripts* para captura dos pacotes e *rotacionamento* dos logs. Uma questão enfrentada foi: ao matar o processo *tcpdump*, a cada janela de 10 minutos de gravação dos *dumps* em disco, utilizando o comando *kill -9*, a aplicação era interrompida de forma abrupta e o Recon não conseguia ler o arquivo de dump para reconstruir as sessões. Para resolver este problema foi utilizada a opção *kill -15*, para encerrar corretamente o processamento do *tcpdump*. De fato, conforme Ribeiro (2004), o comando *kill -9* termina o processo incondicionalmente de forma abrupta. Este comando pode deixar arquivos abertos e bases de dados corrompidas; por isto deve ser utilizado apenas quando o processo páre de responder ou em alguma situação de emergência. O comando *kill -15* termina o processo de forma prevista, possibilitando que o processo feche arquivos e realize suas rotinas de fim de execução.

Outro problema encontrado nesta etapa do trabalho refere-se à filtragem do tráfego HTTP da RedeBeta para análise. Ao espelhar a porta do *firewall* para a porta do sensor, o *switch* coloca um cabeçalho de VLAN nos pacotes que entram para a RedeBeta. Utilizou-se a opção (*tcp and port 80*) no comando

tcpdump para a gravação de dados do tráfego HTTP da RedeBeta, como havia sido utilizada para a RedeLab. Porém, as sessões RedeBeta foram reconstruídas de modo incorreto, considerando apenas os pacotes de saída da RedeBeta solicitando serviços Web. Este problema de uso de pacotes de uma única direção do tráfego foi corrigido com o uso das opções (*tcp and port 80 and (vlan <No VLAN RedeBeta> and port 80)* no comando *tcpdump*.

Em seguida, os *dumps* da RedeBeta corretamente capturados e armazenados foram passados pelo *IDS Snort* que gera um arquivo de alerta com os pacotes de tráfego anômalo identificados. Após a análise dos arquivos de alerta do *Snort* e do tráfego considerado normal pelo *Snort*, utilizou-se o *script "remove_ataques"*, que utilizando as quintuplas formadas por *IP_A*, *IP_B*, *port_A*, *port_B* e *timestamp* obtidas do arquivo de alerta do *Snort*, permitiu a separação dos dados de tráfego normal e tráfego anômalo desta rede.

Para os estudos envolvendo dados da RedeBeta também foram utilizados dados de tráfego normal destas redes concatenados com dados de tráfego anômalo sintético, produzido a partir do lançamento de ataques contra o ambiente da RedeLab, descritos na seção a seguir.

7.2 Descrição dos Ataques

Os ataques lançados contra o ambiente de rede controlado (RedeLab) para a geração de tráfego anômalo encontram-se especificados nas Tabelas 7.2 e 7.3 a seguir. Alguns destes ataques foram bem sucedidos, enquanto outros apenas representaram tentativas de ataques.

Tabela 7.2- Ataques do tipo R2L e U2R

NOME	TIPO
webi	Packet Injection
sumo	Nick Collider
m00-apache-w00t	Remote User Disclosure
gyan	Buffer Overflow
cam7k5nt	SQL injection
20050427.sql_expl	Stack Overflow
85mod_gzip	Buffer Overflow
apache	Buffer Overflow
xaran.pl	SQL Injection
sendfile.pl	CGI Remote Exploit
phpBB	Arbitrary Code Execution
PHPNuke	SQL Injection
Fuzzer - bed.pl	Buffer Overflow
dl-mancgi	Remote Command Execution
news_exp	Change Admin Password
phx	Buffer Overflow
phf	Buffer Overflow
ccbillx	CGI Remote Exploit
ishopcart-cgi-bof	Buffer Overflow
AWStats	Multiple Remote Exploit
aztek-spl0it	Database Dumper Exploit
sileFSBXspl	Remote Command Execution
sugar	Remote Code Execution

Tabela 7.3- Ataques do tipo DoS e Probe

NOME	TIPO
ttcp	Probe
syndrop	DoS
synflood	DoS
synful	DoS
probe	Probe
probe_tcp_ports	Probe
orgasm	Probe
napscan	Probe
killwin	DoS ou Probe
kkill	DoS ou Probe
more-sioux	DoS
httpscan	Probe
ipscan	Probe
portscan	Probe

Uma breve descrição de alguns ataques utilizados nesta tese é apresentada a seguir:

- **webi**: realiza injeção de pacotes em solicitação HTTP; foi escrito em C e é capaz de gerar cabeçalhos HTTP pequenos e utilizar todos os métodos apache;
- **m00-apache-w00t**: realiza varredura de hosts remotos com httpd (apache) nas versões Apache 1.3.* a 2.0.48 e descobre informação sobre contas de usuários existentes através da configuração default errada do módulo do apache denominado `mod_userdir`. Em seguida, tenta se logar no serviço ftp com os logins encontrados. Este ataque é antigo, mas é um problema ainda atual, pois 99% dos administradores de rede utilizam configuração default do servidor http apache;
- **gyan**: explora remotamente a vulnerabilidade "*memory_limit*" do PHP nas versões PHP 4 ($\leq 4.3.7$) e PHP 5 ($\leq 5.0.0RC3$). Este *exploit* constrói uma tabela *hash* maliciosa cujo espaço em memória a ser

alocado para esta conterá dados de solicitação prévia, incluindo um ponteiro destrutor que aponta para o *shellcode* do atacante;

- phf: realiza buffer overflow no Linux-x86;
- more-sioux: realiza, de modo simples, DoS em qualquer servidor Apache. Produz *memory leak* massiva pelo modo como manipula os cabeçalhos das solicitações que chegam no servidor. O consumo de memória do servidor aumenta de modo polinomial dependendo da quantidade de dados enviados;
- portscan: realiza a varredura de portas TCP de hosts apresentando todas as portas que aceitam conexões e, se conhecido, o nome do serviço. Este programa também pode ser facilmente alterado para varredura de portas UDP.

Os ataques do tipo R2L e U2R, descritos na Tabela 7.2 e utilizados nos estudos de casos desta tese, exploram as vulnerabilidades das aplicações Web classificadas como “b) Injeção de Falhas”, “c) Execução de Arquivo Malicioso” e “f) Manipulação Imprópria de Erros e Vazamento de Informação” na lista de “*Top Ten Vulnerabilities from Web Applications*” da OWASP, apresentada na seção 2.4 do segundo capítulo.

7.3 Conjuntos de Dados Utilizados

Para os estudos finais realizados, foram utilizados oito (08) conjuntos de dados da RedeLab e dez (10) conjuntos de dados da RedeBeta. Todos estes conjuntos foram gerados a partir de tráfego normal e anômalo conhecidos. A Tabela 7.4 contém a identificação dos conjuntos de dados utilizados e a quantidade de sessões normais e de ataques.

Tabela 7.4 - Conjuntos de dados utilizados nos estudos de casos

RedeLab	Normal	Ataques	Total	Descrição	
SETLAB_01	381	0	381	sem ataques	
SETLAB_02	381	4	385	R2L e U2R	
SETLAB_03	381	10	391	DoS e Probe	
SETLAB_04	381	70	451	R2L e U2R	
SETLAB_05	804	0	804	sem ataques	
SETLAB_06	804	4	808	R2L e U2R	
SETLAB_07	804	10	814	DoS e Probe	
SETLAB_08	804	70	874	R2L e U2R	
Filtro = (tcp and port 80)					
RedeBeta	Normal	Ataques	Total	Descrição	
SETBETA_01	3050	0	3050	10 minutos (15:00)	dia = 09082006
SETBETA_02	1989	0	1989	8 horas (00:00-07:59)	dia = 09082006
SETBETA_03	15304	0	15304	6 horas (18:00-23:59)	dia = 09082006
SETBETA_04	65602	0	65602	5 horas (08:00-12:59)	dia = 09082006
SETBETA_05	80938	0	80938	5 horas(13:00-17:59)	dia = 09082006
SETBETA_06	3050	3	3053	10 minutos (15:00)	dia = 09082006
SETBETA_07	1989	3	1992	8 horas (00:00-07:59)	dia = 09082006
SETBETA_08	15304	5	15309	6 horas (18:00-23:59)	dia = 09082006
SETBETA_09	65602	37	65639	5 horas (08:00-12:59)	dia = 09082006
SETBETA_10	80938	37	80975	5 horas(13:00-17:59)	dia = 09082006
Filtro = (tcp and port 80) or (vlan 16 and port 80)					

Aos dados normais da RedeLab, gerados com a captura de tráfego de acesso normal a páginas Web de serviço interno e da Internet, foram adicionados os dados de ataques simulados lançados contra a RedeLab.

Dados da RedeBeta do dia 09/08/2006 (quarta-feira) foram utilizados nos estudos de casos, envolvendo janelas de 10 minutos, 8 horas, 6 horas e 5 horas de tráfego, como apresentado na Tabela 7.4. O tráfego de ataques lançados contra a RedeLab foi utilizado para compor o tráfego anômalo na construção dos conjuntos de dados da RedeBeta.

7.4 Descrição dos Estudos de Casos

Os estudos de casos descritos neste capítulo são: análise visual de comportamento do tráfego; análise visual de contribuição dos atributos; detecção de assinaturas pelo conteúdo dos pacotes; detecção de assinaturas pelo cabeçalho dos pacotes; detecção de anomalias pelo cabeçalho dos pacotes.

7.4.1 Análise Visual do Comportamento do Tráfego

Uma sessão HTTP normal contém, em geral, algumas características próprias, incluindo: tempo de duração pequeno; pequena quantidade de dados enviados do cliente para o servidor; maior quantidade de dados enviados do servidor para o cliente. Portanto, atributos como tamanho médio dos pacotes, número de pacotes recebidos pelo cliente e servidor, e tempo de duração da sessão são dados importantes na análise deste tipo de tráfego.

Com base nestes atributos, uma análise visual do comportamento de três conjuntos de dados da RedeBeta, em uma janela de 10 minutos e em dois períodos diferentes do dia foi realizada. Tráfego de ataque conhecido foi inserido no tráfego analisado para visualização gráfica do perfil da rede. Sessões do tráfego normal são apresentadas na Figura 7.3 na cor verde e sessões de tráfego anômalo sintético, na cor azul. Os valores dos atributos são apresentados em coordenadas paralelas no intervalo entre mínimo e máximo de cada valor na base de dados.

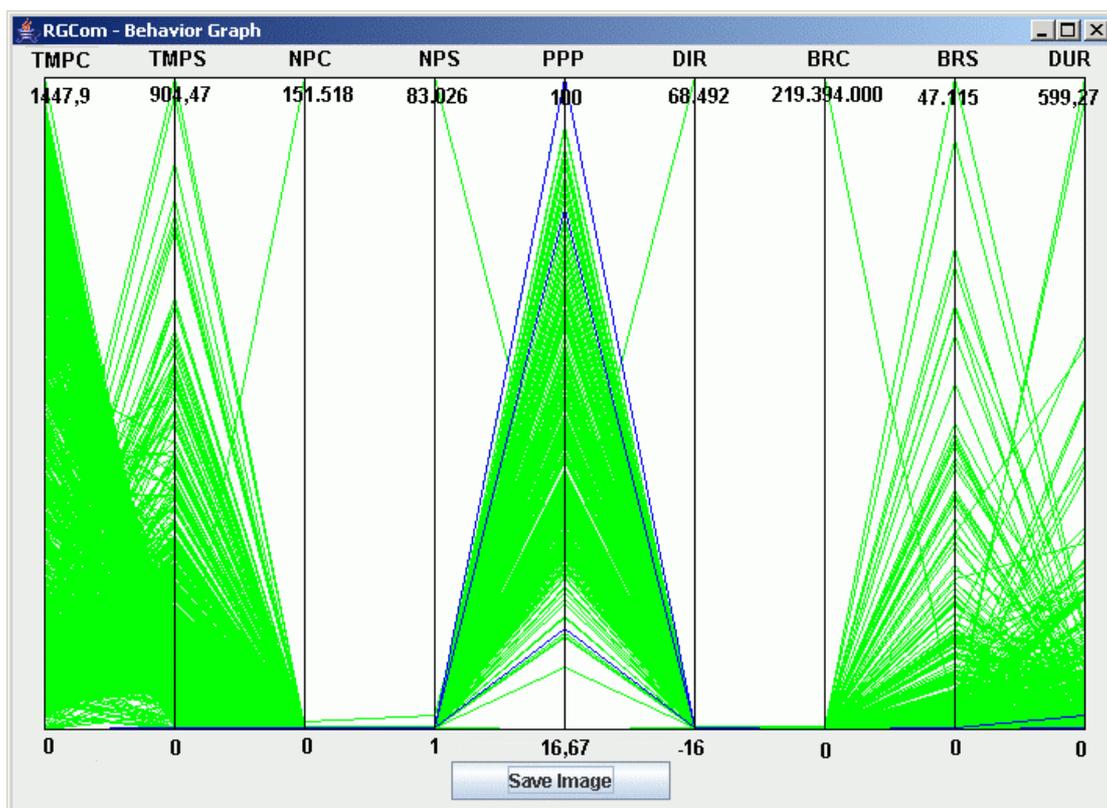


Figura 7.3 - Janela de 10 minutos de sessões do tráfego¹

O tráfego apresentado na Figura 7.3 corresponde à captura de pacotes em 10 minutos de tráfego do dia 09/08/2006 às 15:00h, com 3 registros (sessões) de ataques inseridos artificialmente (apenas para observação de comportamento da sessão anômala) no conjunto total de 3.053 sessões.

¹ TMPC - tamanho médio dos pacotes recebidos pelo cliente
 TMPS - tamanho médio dos pacotes recebidos pelo servidor
 NPC - número de pacotes recebidos pelo cliente
 NPS - número de pacotes recebidos pelo servidor
 DIR - direção do tráfego
 PPP - porcentagem de pacotes pequenos
 BRC - total de dados de *payload (bytes)* recebidos pelo cliente
 BRS - total de dados de *payload (bytes)* recebidos pelo servidor
 DUR - duração da sessão

Observando o gráfico da Figura 7.3 anteriormente mencionada, algumas considerações podem ser feitas:

- Pela feição triangular formada pelas linhas de sessão entre os eixos do primeiro (TMPC) e segundo (TMPS) atributos, e pela maior saturação de linhas em direção ao primeiro eixo, observa-se que neste tráfego, as estações cliente recebem mais pacotes que a(s) estação(ões) servidora(s);
- Algumas poucas linhas de sessão aparecem no segundo e terceiro eixos do gráfico (NPC e NPS), respectivamente, indicando que estas sessões são suspeitas por serem bem diferentes das demais, que nem aparecem no gráfico;
- Neste gráfico é ressaltado um valor grande de um ou mais atributos sobrepostos no terceiro e quarto eixos. Esta linha representa algo fora do padrão e pode indicar a ocorrência de uma sessão de *download* de arquivos de tamanho acima do usual ou acesso a rádio ou vídeo. Como a ferramenta apresenta valores máximos e mínimos dos atributos e os valores ressaltados nas colunas NPC e NPS são muito maiores que os demais (151.518 e 83.026 pacotes) os demais valores desaparecem no conjunto.
- O quinto atributo (PPP) refere-se à porcentagem de pacotes pequenos e varia de 0 a 100. Uma concentração de valores no meio do eixo indica que as sessões cujos pontos residem nas extremidades, muito próximos de 0 ou muito próximos de 100 deveriam ser investigados;
- Através do sexto atributo (DIR) é possível verificar o balanceamento das sessões (recebe valor +1 se cliente recebe pacote e -1 se servidor recebe pacote) e pode-se, por ele, reconhecer diferenças no tráfego. Neste caso, existe uma contagem muito grande para o cliente, valor muito acima dos demais, indicando que algum cliente recebeu muitos pacotes, provavelmente o mesmo sinalizado no eixo 3;

- Ocorre um grande número de bytes recebidos por cliente e menos pelo servidor, como mostra os atributos BRC (maior valor=219 Mbytes) e BRS (maior valor = 47 Kbytes) no gráfico. O ponto ressaltado no eixo BRC indica que o analista deveria investigar os endereços IP de origem e destino e tamanho dos pacotes para se certificar de que a operação em rede é devida;
- Outra observação é que o valor de duração da maioria das sessões, em geral, é pequeno, conforme apresentado no último eixo (DUR).

Através das observações descritas acima, pode-se concluir que existem sessões neste tráfego de 10 minutos que precisam ser investigadas. Mas, em resumo, o tráfego desta rede no intervalo observado, é normal.

O tráfego observado na Figura 7.4 a seguir refere-se a 5 horas de tráfego da RedeBeta (13:00-17:59h) com 37 registros de ataques inseridos artificialmente no conjunto contendo ao todo 80.975 sessões.

Este tráfego apresenta valores altos para número de pacotes recebidos pelo cliente e servidor (NPC e NPS), algumas sessões do tráfego desbalanceadas (visto em DIR), e alguns valores altos de bytes recebidos pelo cliente e servidor (BRC e BRS). A duração de algumas sessões também é alta. Porém, vale ressaltar que este é um conjunto de 5 horas de tráfego em horário de muito acesso à rede (15:00h). Para encontrar aberrações no tráfego através deste gráfico, os atributos NPC, NPS, PPP, DIR, BRC e BRS apresentam maior contribuição.

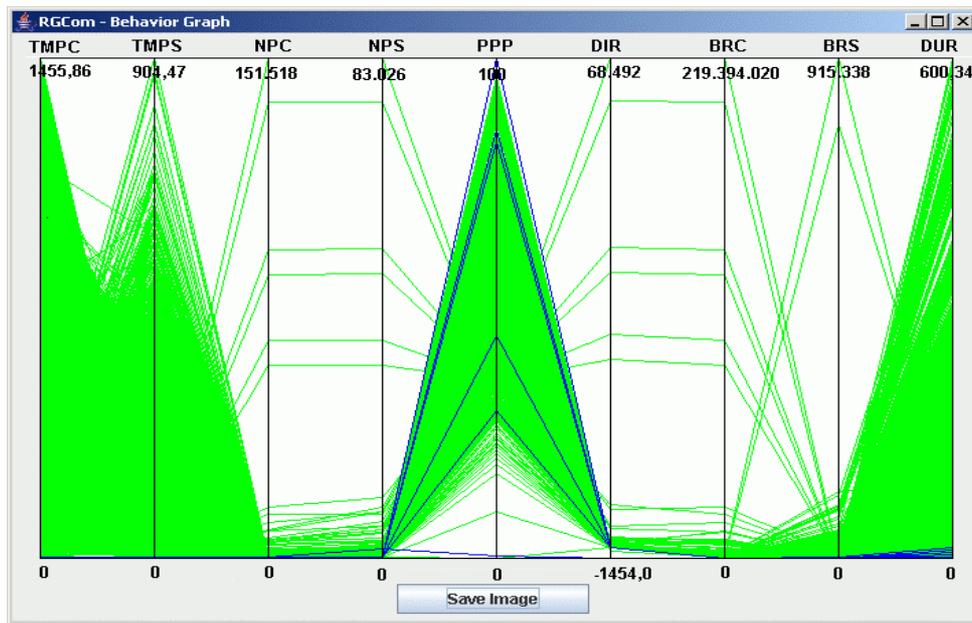


Figura 7.4 - Janela de 5 horas de sessões do tráfego em horário de pico

A representação visual de uma janela de 8 horas de tráfego da RedeBeta em horário de pouco uso da rede (00:00-07:59h), contendo 1992 sessões, dentre elas 3 sessões anômalas inseridas no conjunto, é ilustrada na Figura 7.5.

Observa-se nesta Figura que o valor Máximo do número de bytes recebidos pelo cliente e pelo servidor nesta janela de tempo é bem menor que o da Figura 7.4 anteriormente analisada, o que permite a observação de variação dos demais valores neste eixo.

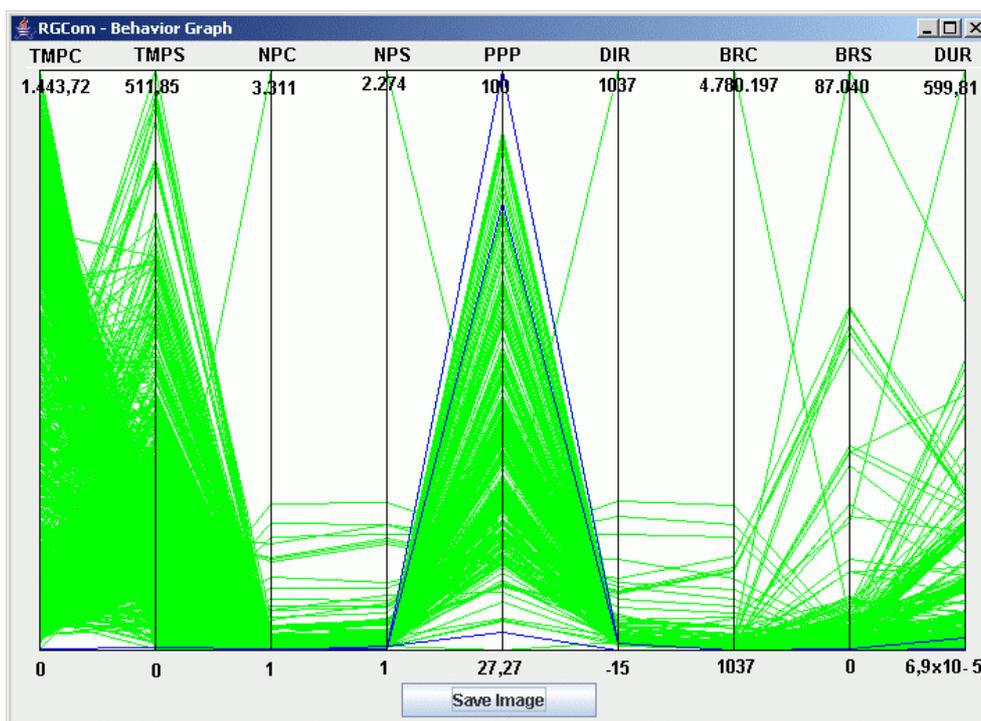


Figura 7.5 - Janela de 8 horas de sessões do tráfego de rede

A partir dos gráficos acima pode-se observar que a análise de janelas de tempo menores permite a extração de informação mais significativa do conjunto.

Pode-se usufruir o recurso implantado no sistema ADTRAF para seleção de região de normalidade no gráfico, fora da qual o sistema reconhece e sinaliza as sessões como anômalas e permite visualização destas de modo destacado, como mostra a Figura 7.6.

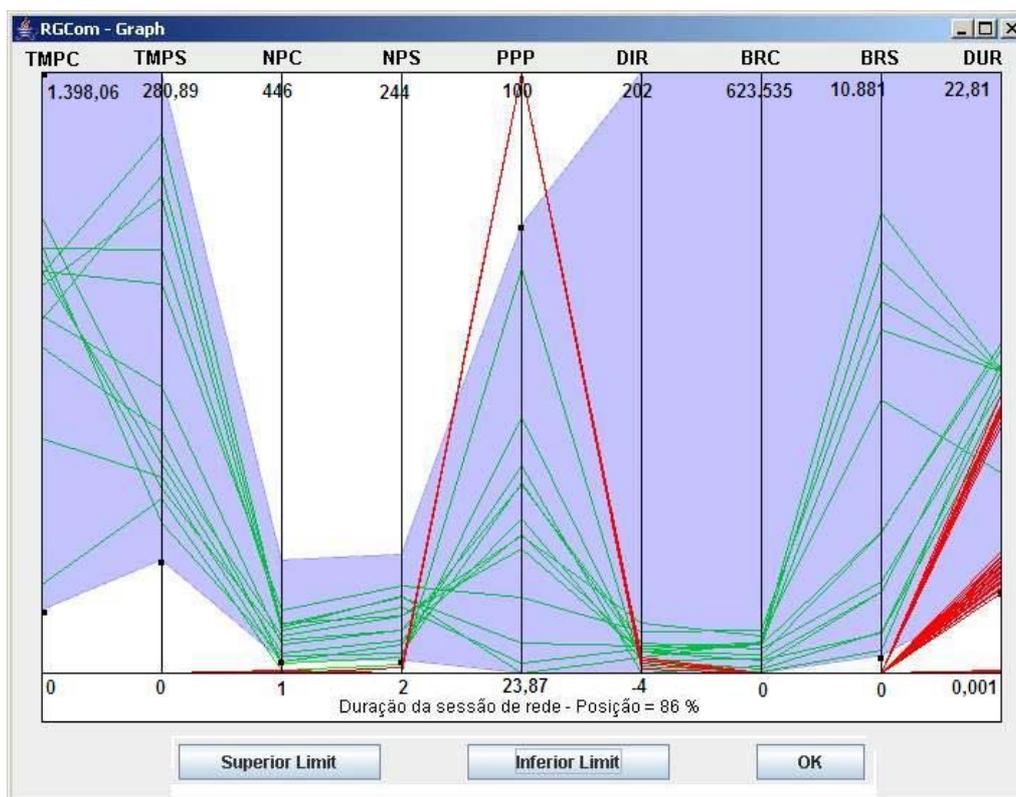


Figura 7.6 - Destaque de Sessões Anômalas fora da Região de Normalidade

Com os gráficos de sessões de ataques simulados gerados nesta tese observou-se que diferentes sessões do mesmo ataque tiveram os valores dos atributos PPP (porcentagem de pacotes pequenos) e DUR (duração do tráfego) alterados. Outro atributo observado cujo valor mudou para estas sessões foi o estado da sessão.

Entre ataques diferentes, observou-se variação maior dos seguintes atributos: número de pacotes recebidos pelo servidor (NPC), porcentagem de pacotes pequenos (PPP), direção do tráfego (DIR), duração (DUR) e estado da sessão. O estado da sessão é um atributo que não foi apresentado no gráfico de coordenadas paralelas do sistema, mas seus valores armazenados na base de dados foram observados.

A análise gráfica utilizando a ferramenta RGCOM é útil, pois permite o reconhecimento de desvios no tráfego, mas seu uso deve ser combinado com outras técnicas para uma detecção completa e mais precisa de ataques à rede.

7.4.2 Detecção de Assinaturas

Nos estudos de detecção de assinaturas no tráfego de rede foram utilizadas as ferramentas ANNIDA e ADTRAF.

A aplicação ANNIDA permite detectar assinaturas de ataques em *strings* de payload de pacotes de rede, através do uso das redes neurais Hamming Net e LVQ.

O sistema ADTRAF permite detectar assinaturas de ataques através de análise de atributos das sessões do tráfego de rede através da rede neural LVQ.

7.4.2.1 Análise de dados do Conteúdo dos Pacotes

Dois estudos de casos foram conduzidos, envolvendo o uso das redes neurais Hamming Net e LVQ na aplicação ANNIDA para análise de *payload* de pacotes de rede. A diferença entre estes estudos é que no primeiro, foi utilizada a aplicação na versão 2005 capaz de classificar dados simulados, ou seja, dados semelhantes ao conteúdo de *payload* de pacotes de rede, porém em ASCII, e no segundo estudo, foi utilizada a versão 2006 da aplicação para classificar dados reais de *payload* de pacotes de rede, em ASCII e em Hexadecimal.

No primeiro estudo, conforme apresentado na Tabela 7.5, foram utilizadas assinaturas Snort extraídas dos arquivos de regras *finger*, *icmp*, *ddos*, *dns*, *ftp*, *oracle* e *exploit*, contendo de uma a cinco *strings* de conteúdo malicioso, e armazenadas em arquivos do tipo texto. Neste estudo, os dados de entrada utilizados para classificação foram construídos artificialmente e correspondem a *strings* em formato ASCII.

Tabela 7.5 - Resultados de Classificação e Desempenho das Redes Neurais

Classe de Assinaturas	Tamanho da Entrada (char)	Total de Strings Maliciosas	Total de Exemplos	Classificação (HN e LVQ)	Tempo (seg) HN	Tempo (seg) LVQ
finger	99	1	NUMEXEMP= 15 MAXLEN= 12	100%	9,80	5,01
icmp	312	1	NUMEXEMP=11 MAXLEN=66	100%	8,45	4,08
		2	NUMEXEMP=11 MAXLEN=66	100%	9,20	5,32
ddos	78	1	NUMEXEMP=227 MAXLEN= 24	100%	50,8	39,7
dns	60	1	NUMEXEMP=19 MAXLEN=108	100%	8,35	3,66
		2			8,46	3,67
ftp	200	3	NUMEXEMP= 69 MAXLEN=24	100%	25,4	17,5
oracle	67	3	NUMEXEMP=200 MAXLEN=70	90%	45,7	26,9
exploit	89	4	NUMEXEMP=28 MAXLEN=24	100%	13,6	9,59
		5	NUMEXEMP=77 MAXLEN=12	90%	19,8	9,94

Para cada classe de assinaturas, os seguintes parâmetros foram utilizados: tamanho da *string* de entrada na rede neural para classificação (tamanho da entrada), total de *strings* de conteúdo malicioso apresentado por cada classe de assinatura, total de assinaturas (exemplares) existentes para cada classe (NUMEXEMP) e tamanho do maior exemplar em caracteres (MAXLEN).

De acordo com os resultados dos testes no primeiro estudo realizado, utilizando dados simulados, constatou-se que:

- O processamento da LVQ é mais rápido que o da Hamming Net para dados simulados;
- A mesma taxa de detecção foi obtida para ambas as redes. Um número pequeno de classificações ocorreu com precisão menor que 100%. Isto ocorreu porque não foi tratado o caso em que uma *string* de entrada pudesse estar inteiramente contida em uma string exemplar, antecedida e/ou precedida de outros caracteres;
- O número e o tamanho de *strings* de conteúdo malicioso por assinatura e o número de exemplares (assinaturas), influenciam no tempo de processamento das redes neurais. Portanto, utilizar classes de

assinaturas por tipo de serviço ou por tipo de aplicação pode acelerar o processo de detecção;

- Ocorreram colisões nos valores de entrada para as redes neurais, devido à chave de 4 dígitos gerada pelo algoritmo de *hashing fold-shift*.

O segundo estudo envolveu o uso da aplicação ANNIDA modificada para detecção de assinaturas em dados de carga útil de pacotes do tráfego real de rede.

Dois principais casos com dados reais do tráfego de rede foram analisados, sendo o primeiro caracterizado pela busca de assinaturas representadas por uma *string* única de conteúdo malicioso obtidas do *Snort* (testes 01 ao 11) e o segundo estudo refere-se à busca de várias assinaturas, obtidas através do lançamento de ataques ao serviço *Web* fornecido na RedeLab (testes 12 e 13). Para ambos os testes, o tráfego normal foi obtido através de acessos a páginas *Web* estáticas e dinâmicas fornecidas pelo servidor Apache.

Para o primeiro caso, testes 01 a 11, foram utilizadas assinaturas contidas no arquivo de regras *Snort* denominado “*web-attacks*” por explorarem o serviço *Web* utilizado nos testes: o Apache. Quarenta e seis assinaturas de ataques foram utilizadas nos testes. Um total de 14 máscaras de *payload* de tamanho variável entre 6 e 68 caracteres hexadecimal, sem repetição, foram utilizadas na varredura de *payload* dos pacotes de rede.

Para os testes de 01 a 11, como mostrado na Tabela 7.6, conjuntos de 208 pacotes de rede, em média, foram analisados, totalizando 135.157 bytes de *payload* examinados. As *strings* de ataques utilizadas como dados exemplares são exibidas na coluna 5 desta Tabela.

As *strings* de ataque utilizadas nos testes 08 a 11 não representam assinaturas de ataques para o serviço *web* utilizado nos testes, portanto, o sistema corretamente não as identificou como anômalas (Total *Strings* Detectadas = 0).

Neste estudo, ambas as redes apresentaram a mesma precisão (100% de acerto) na classificação de padrões de ataques conhecidos. Porém, a rede LVQ realizou classificação mais rapidamente que a Hamming Net.

Tabela 7.6 - Resultados dos testes de busca de *strings* de ataque

ID do Teste	Total Pacotes	Tamanho total de Payloads (bytes)	Assinatura ou String de Ataque	Tamanho Mascara Payload Casada	Total Assinaturas ou Strings Detectadas	Tempo Hamming Net	Tempo LVQ
01	208	135.143	/bin/ls	14	1	00:02:18	00:00:04
02	206	135.157	/bin/ls 7C	16	1	00:02:17	00:00:04
03	208	135.151	/bin/ps	14	1	00:02:17	00:00:05
04	208	135.182	/chmod	12	1	00:02:19	00:00:04
05	208	135.157	%20-display%20	40	1	00:02:09	00:00:04
06	207	135.144	wget%20	14	1	00:02:19	00:00:05
07	207	135.165	uname%20-a	20	1	00:02:08	00:00:04
08	209	135.158	ls%20-al	-	0	00:02:08	00:00:04
09	210	135.189	/usr/bin/id	-	0	00:02:08	00:00:05
10	206	135.190	cmd.exe	-	0	00:02:08	00:00:04
11	198	134.148	tráfego Normal	-	0	00:01:49	00:00:05

O sistema ANNIDA detectou corretamente todas as assinaturas existentes no conjunto de dados, como mostra o segundo campo da tabela (Total *Strings* Detectadas = 1) através do uso de máscaras de *payload* de diferentes tamanhos e levou cerca de dois minutos e oito segundos para detectar strings de conteúdo malicioso através do classificador Hamming Net e 4 segundos, em média, para detectar assinaturas através do classificador LVQ.

No segundo caso utilizando dados reais, foram analisados *payload* de pacotes em busca de assinaturas lançadas contra o servidor *Web* instalado. Os resultados destes testes são apresentados na Tabela 7.7.

Tabela 7.7 - Resultados dos testes de busca de múltiplas *strings* de ataque

ID Teste	Número de Assinaturas lançadas	Total de pacotes	Tamanho total de payloads (bytes)	Total de Assinaturas Detectadas	Tempo Hamming Net	Tempo LVQ
01	3	227	135.938	3	00:02:27	00:00:05
02	7	727	152.634	7	00:09:32	00:00:40

Para o teste 1, o tráfego de acesso a uma página Web estática foi registrado, bem como o tráfego de lançamento de três ataques contra este serviço, totalizando 227 pacotes capturados com tamanho total de *payloads* de 135.938

bytes. O teste 02 envolveu o registro de tráfego de acesso a uma página dinâmica, incluindo operações de consulta e registros, enquanto sete ataques foram simultaneamente lançados contra o serviço *Web* disponível, totalizando 727 pacotes capturados com tamanho total de *payloads* de 152.634 bytes.

A análise dos resultados obtidos no segundo estudo, em que a aplicação ANNIDA foi aperfeiçoada, revelou que:

- O processamento da LVQ manteve-se mais rápido que o da Hamming Net para análise de dados reais de *payload* de pacotes de rede. O tempo de processamento reduziu o tempo de minutos para segundos;
- A mesma taxa de detecção foi obtida para ambas as redes (100% de classificação correta);
- O número e o tamanho de *strings* de conteúdo malicioso por assinatura e o número de exemplares (assinaturas), influenciam no tempo de processamento das redes neurais. Portanto, utilizar classes de assinaturas por tipo de serviço ou por tipo de aplicação pode acelerar o processo de detecção;
- Não ocorreram colisões nos valores de entrada para as redes neurais, devido à chave de 128 dígitos gerada pelo algoritmo *n-hash* (NIPPON, 2006).

7.4.2.2 Análise de dados do Cabeçalho dos Pacotes

Através do sistema ADTRAF foi realizada a detecção de padrões de comportamento do tráfego a partir de dados do cabeçalho (atributos das sessões) dos pacotes de rede, utilizando a rede neural LVQ para classificação.

Para este estudo, foram utilizados os atributos das sessões de ataques do tipo R2L, U2R, DoS e Probe, descritos na seção 7.2, lançados contra os serviços instalados na RedeLab. Os pacotes capturados de cada tráfego de ataque lançado foram remontados, reconstruindo-se assim as sessões anômalas compostas por nove atributos. Cada conjunto de atributos (sessão) foi rotulado

com a identificação do respectivo ataque, descrevendo, portanto, uma assinatura de ataque específico. As múltiplas sessões de ataques foram todas rotuladas com a mesma identificação de nome do ataque associado.

A entrada da LVQ é um vetor de nove posições contendo valores de atributos de sessões do tráfego para análise. Para os estudos, tanto a parte normal, quanto a parte anômala do tráfego eram conhecidas para comparação dos resultados do classificador.

Neste estudo de caso, a rede neural LVQ recebe como valores de pesos os valores de atributos das N sessões anômalas (técnica de inserção de pesos). Os exemplares da rede recebem os valores de atributos das N sessões anômalas também e a rede LVQ possui N neurônios de saída, onde cada neurônio de saída está associado a um exemplar, na ordem em que foi inserido na rede.

Nesta rede, ocorre o casamento entre vetor entrada e o vetor exemplar, através de cálculo de distância Euclidiana entre entrada e exemplares (assinaturas de ataques), onde sempre o vetor exemplar mais semelhante (neurônio vencedor) ao vetor de entrada é identificado. No entanto, é utilizada nesta tese uma medida de similaridade que verifica a ocorrência de casamento perfeito (100% de similaridade) entre cada valor de atributo do vetor de entrada e do vetor exemplar vencedor. Ocorrendo um casamento perfeito, o algoritmo rotula a sessão de entrada como anômala e exibe um alerta com dados sobre esta sessão, como apresentado nas Figuras 7.7, 7.8, 7.9 e 7.10 a seguir. Este limiar pode ser ajustado para permitir casamentos não perfeitos entre entrada e exemplar, dando maior robustez ao classificador que poderá ser capaz de detectar variações de ataques de mesma categoria.

A Figura 7.7 ilustra a saída do classificador ao reconhecer sessões anômalas (de ataques do tipo R2L e U2R) existentes em um conjunto de 385 sessões da RedeLab apresentado à rede neural. São exibidas as informações: nome

identificador do ataque, os endereços IP participantes da comunicação e o *timestamp*, com horário de captura do primeiro pacote da sessão.

	Name	Lower IP	Higher IP	Time
372	Unknown Session	0.0.0.0	0.0.0.0	00:00
373	Unknown Session	0.0.0.0	0.0.0.0	00:00
374	Unknown Session	0.0.0.0	0.0.0.0	00:00
375	Unknown Session	0.0.0.0	0.0.0.0	00:00
376	Unknown Session	0.0.0.0	0.0.0.0	00:00
377	Unknown Session	0.0.0.0	0.0.0.0	00:00
378	Unknown Session	0.0.0.0	0.0.0.0	00:00
379	Unknown Session	0.0.0.0	0.0.0.0	00:00
380	Unknown Session	0.0.0.0	0.0.0.0	00:00
381	cam7knt	192.168.0.25	192.168.0.254	08:24
382	apache	192.168.0.25	192.168.0.254	08:30
383	ccbilx	192.168.0.25	192.168.0.254	09:29
384	webi	192.168.0.25	192.168.0.254	15:05

Figura 7.7 - Detecção de Sessões Isoladas de Ataque U2R e R2L

Outro exemplo, ilustrado na Figura 7.8, apresenta resultados do algoritmo de classificação de múltiplas sessões pertencentes a um mesmo ataque. Por exemplo, cinco sessões foram geradas para o ataque `m00-apache-w00t`, o qual explora, remotamente, contas de usuários em estações servidoras Apache, e seis sessões são provenientes do ataque `gyan` do tipo *buffer overflow*, que explora o limite de memória do PHP. Estas são algumas das 70 sessões de ataques do tipo U2R e R2L, registradas em um conjunto de 451 sessões da RedeLab.

	Name	Lower IP	Higher IP	Time
438	sugar	192.168.0.25	192.168.0.254	10:52
439	webi	192.168.0.25	192.168.0.254	15:05
440	m00-apache-w00t	192.168.0.25	192.168.0.254	16:09
441	m00-apache-w00t	192.168.0.25	192.168.0.254	16:09
442	m00-apache-w00t	192.168.0.25	192.168.0.254	16:09
443	m00-apache-w00t	192.168.0.25	192.168.0.254	16:09
444	m00-apache-w00t	192.168.0.25	192.168.0.254	16:09
445	gyan	192.168.0.25	192.168.0.254	16:12
446	gyan	192.168.0.25	192.168.0.254	16:12
447	gyan	192.168.0.25	192.168.0.254	16:12
448	gyan	192.168.0.25	192.168.0.254	16:12
449	gyan	192.168.0.25	192.168.0.254	16:12
450	gyan	192.168.0.25	192.168.0.254	16:12

Figura 7.8 - Detecção de Múltiplas Sessões de Ataque U2R e R2L

A Figura 7.9 ilustra a identificação de sessões de ataque DoS e Probe em conjunto de 391 sessões da RedeLab. Alguns traços de ataques são registrados em sessões isoladas, enquanto outros, em sessões múltiplas.

	Name	Lower IP	Higher IP	Time
379	Unknown Session	0.0.0.0	0.0.0.0	00:00
380	Unknown Session	0.0.0.0	0.0.0.0	00:00
381	ttcp	192.168.0.25	192.168.0.254	15:02
382	probe	192.168.0.25	192.168.0.254	15:54
383	probe_tcp_ports	192.168.0.25	192.168.0.254	15:55
384	orgasm	192.168.0.25	192.168.0.254	15:58
385	napsan	192.168.0.25	192.168.0.254	16:01
386	more-sioux	192.168.0.25	192.168.0.254	16:11
387	httpscan	192.168.0.25	192.168.0.254	16:13
388	ipscan	192.168.0.25	192.168.0.254	16:14
389	synful	1.46.251.194	192.168.0.254	15:12
390	synful	1.46.251.194	192.168.0.254	15:12

Figura 7.9 - Detecção de Sessões Mistas de Ataque DoS e Probe

Algumas sessões de ataque foram inseridas no conjunto de tráfego normal da RedeBeta formando um conjunto de 3053 sessões. Estas sessões também foram identificadas pela rede neural LVQ no sistema ADTRAF. A Figura 7.10 ilustra a saída do classificador.

	Name	Lower IP	Higher IP	Time
3041	Unknown Session	0.0.0.0	0.0.0.0	00:00
3042	Unknown Session	0.0.0.0	0.0.0.0	00:00
3043	Unknown Session	0.0.0.0	0.0.0.0	00:00
3044	Unknown Session	0.0.0.0	0.0.0.0	00:00
3045	Unknown Session	0.0.0.0	0.0.0.0	00:00
3046	Unknown Session	0.0.0.0	0.0.0.0	00:00
3047	Unknown Session	0.0.0.0	0.0.0.0	00:00
3048	Unknown Session	0.0.0.0	0.0.0.0	00:00
3049	Unknown Session	0.0.0.0	0.0.0.0	00:00
3050	20050427	192.168.0.25	192.168.0.254	08:27
3051	20050427	192.168.0.25	192.168.0.254	08:27
3052	phx	192.168.0.25	192.168.0.254	09:26

Figura 7.10 - Detecção de Sessões Mistas de Ataque U2R e R2L

Para todos os casos testados, a LVQ apresentou taxa de detecção de 100% de classificação de sessões de ataques, a partir de dados rotulados.

Outro estudo de caso envolveu a detecção de ataques através dos sistemas de detecção ADTRAF, *Dragon Network Sensor* (ENTERASYS, 2006) e Snort, instaladas na RedeLab. Na Tabela 7.8 constam os nomes e os tipos dos ataques reconhecidos por estes sistemas.

Tabela 7.8 – Reconhecimento de ataques com ADTRAF, Dragon e Snort

NOME	TIPO	DATA	ADTRAF	DRAGON	SNORT
ttcp*	Probe	Out/1991	SIM	NÃO*	NÃO
webi*	Packet Injection	24/jan/2002	SIM	NÃO*	NÃO
syndrop	DoS	03/nov/1997	SIM	SIM	NÃO
synflood	DoS	-	SIM	NÃO	NÃO
synful	DoS	-	SIM	NÃO	NÃO
sumo	Nick Collider	-	SIM	NÃO	SIM
probe	Probe	-	SIM	SIM	NÃO
probe_tcp_ports	Probe	27/jul/1994	SIM	SIM	NÃO
orgasm	Probe	-	SIM	NÃO	NÃO
napscan	Probe	-	SIM	NÃO	NÃO
killwin	DoS ou Probe	-	SIM	NÃO	NÃO
kkill	DoS ou Probe	-	SIM	NÃO	SIM
m00-apache-w00t	Remote User Disclosure	-	SIM	NÃO	NÃO
more-sioux*	DoS	11/ago/1998	SIM	NÃO*	SIM
gyan	Buffer Overflow	-	SIM	SIM	SIM
httpscan*	Probe	03/out/1999	SIM	NÃO*	NÃO
ipscan*	Probe	1997	SIM	NÃO*	NÃO
cam7k5nt	SQL injection	-	SIM	NÃO	NÃO
20050427.sql_expl	Stack Overflow	27/abr/2005	SIM	SIM	NÃO
85mod_gzip	Buffer Overflow	05/mai/2003	SIM	SIM	NÃO
apache	Buffer Overflow	-	SIM	NÃO	NÃO
xaran.pl	SQL Injection	-	SIM	SIM	NÃO
sendfile.pl	CGI Remote Exploit	-	SIM	NÃO	NÃO
phpBB	Arbitrary Code Execution	-	SIM	NÃO	NÃO
PHPNuke	SQL Injection	-	SIM	SIM	NÃO
Fuzzer - bed.pl	Buffer Overflow	-	SIM	NÃO	SIM
dl-mancgi	Remote Command Execution	-	SIM	NÃO	NÃO
news_exp*	Change Admin Password	21/out/2000	SIM	NÃO*	NÃO
phx	Buffer Overflow	2000	SIM	SIM	NÃO
ccbllx	CGI Remote Exploit	07/jul/2003	SIM	SIM	SIM
ishopcart-cgi-bof	Buffer Overflow	25/mai/2006	SIM	NÃO	SIM
AWStats	Multiple Remote Exploit	24/fev/2005	SIM	NÃO	NÃO
aztek-spoit	Database Dumper Exploit	05/mar/2005	SIM	NÃO	NÃO
sileFSBXpl	Remote Command Execution	10/mai/2005	SIM	NÃO	SIM
sugar	Remote Code Execution	-	SIM	NÃO	SIM

No campo 'Data' são apresentadas as datas de criação dos *scripts* de ataque utilizados nos testes. As datas sinalizadas com traço ('-') foram pesquisadas e não encontradas.

O valor 'SIM' nas colunas da Tabela 7.8 indica que o sistema detectou corretamente informações maliciosas nos pacotes analisados, ou seja, traços de ataques. O valor 'NÃO' significa que o sistema não detectou informações maliciosas nos pacotes.

O sistema ADTRAF, desenvolvido nesta tese, detectou todos os 35 ataques apresentados na Tabela 7.8 através da análise de dados do cabeçalho dos pacotes. Neste sistema, houve o reconhecimento dos padrões de entrada com base nos dados exemplares (valores dos atributos das sessões de ataques) inseridos previamente na rede neural LVQ.

O IDS comercial *Dragon Network Sensor* versão 6.2 da *Enterasys Network* também foi utilizado neste estudo, com uma configuração básica de parâmetros e uma base de assinaturas cuja última atualização foi realizada em 25 de março de 2004. O Dragon foi capaz de detectar 10 ataques (28%) de um total de 35 ataques lançados contra serviços de rede e aplicações Web da RedeLab. Seis ataques utilizados neste teste, apresentados na Tabela 7.8 com asterisco (*), possuem data de criação anterior a 2004 e, mesmo assim, o Dragon não os detectou.

Em seguida, os 35 arquivos *tcpdump* contendo traços de ataques foram analisados através do sistema Snort na versão 2.4.0. Foi utilizada a base de regras do Snort, com data de última atualização em 13 de maio de 2007, baixada da Internet (SOURCEFIRE, 2007). Antes de executar o Snort, foram configuradas algumas variáveis no arquivo *snort.conf*, como, por exemplo, o diretório onde as regras foram armazenadas e também foram configurados os pré-processadores que realizam a remontagem de pacotes, a decodificação de protocolos e a detecção baseada em anomalia de protocolo.

O Snort possui atualmente 13 pré-processadores, incluindo: frag3, stream4, flow, sfportscan, rpc_decode, performance monitor, http_inspect, smtp, fptelnet, ssh, dce_rpc, dns, asn.1. O formato da diretiva do pré-processador no arquivo de configuração do Snort é: `preprocessor <nome>: <opções>`

Dois experimentos foram conduzidos neste estudo: no primeiro, foram habilitados todos os pré-processadores do Snort em sua configuração *default* no arquivo `snort.conf` e, no segundo, foram habilitados cinco pré-processadores, frag3, stream4, flow, sfPortscan e HTTP_inspect, envolvidos na análise de pacotes do tráfego HTTP.

A configuração *default* dos cinco pré-processadores Snort utilizados nesta tese são apresentados a seguir:

```
preprocessor frag3_global
preprocessor frag3_engine
preprocessor stream4
preprocessor stream4_reassemble
preprocessor flow: stats_interval 0 hash 2
preprocessor sfportscan: proto { tcp } scan_type { all } sense_level {
high }
preprocessor http_inspect_server: server default profile all ports {
80 }
```

O Snort alertou corretamente sobre a ocorrência de 9 ataques (taxa de detecção de 26%), do total de 35 ataques do tipo DoS_Probe e R2L_U2R considerados. Exemplos de resultados de alerta apresentados pelo Snort incluem:

```
[**] [111:2:1] (spp_stream4) possible EVASIVE RST detection [**]
02/23-18:46:10.043722 192.168.0.25:32864 -> 192.168.0.254:80
TCP TTL:64 TOS:0x0 ID:4221 IpLen:20 DgmLen:52 DF
***A**R** Seq: 0x3AD784F2 Ack: 0xE2C18833 Win: 0x1920 TcpLen: 32
TCP Options (3) => NOP NOP TS: 2922002 96106739
```

```
[**] [111:24:1] (spp_stream4) possible EVASIVE FIN detection [**]
02/24-12:10:51.544945 192.168.0.254:80 -> 192.168.0.230:4987
TCP TTL:64 TOS:0x0 ID:38121 IpLen:20 DgmLen:52 DF
***A***F Seq: 0x2523DD95 Ack: 0xA3BA097D Win: 0x1920 TcpLen: 32
TCP Options (3) => NOP NOP TS: 346125 22332211
```

7.4.3 Detecção de Anomalias no Tráfego de Rede

Nesta seção são descritos os estudos de casos relacionados à detecção de anomalias no tráfego de rede, através do uso de atributos de sessões como dados de entrada para os métodos de detecção.

Os termos, taxa de detecção, taxa de erro, taxa de falsos positivos e taxa de falsos negativos são utilizados. Entende-se por taxa de detecção de anomalias a razão entre o número de anomalias corretamente detectadas e o número total de anomalias. A taxa de falsos positivos (ou alarmes falsos) é calculada pela razão entre o número de sessões normais incorretamente classificadas como anômalas e o número total de sessões normais. A taxa de falsos negativos corresponde à razão entre o número de sessões anômalas incorretamente classificadas como normais e o número total de sessões anômalas. A taxa de erro está relacionada à quantidade de anomalias que o detector não conseguiu identificar; este valor em percentual é calculado como cem menos a taxa de detecção.

A seguinte legenda foi considerada na construção das Tabelas desta seção:

- Tempo: tempo de resposta do detector;
- TD: taxa de detecção (NACD / NTA);
- TFP: taxa de falsos positivos (NSNIC / NTSN);
- TFN: taxa de falsos negativos (NSAIC/NTA).
- NTA: número total de sessões anômalas;
- NACD: número total de sessões anômalas corretamente detectadas;
- NTSN: número total de sessões normais;
- NSNIC: número total de sessões normais incorretamente classificadas como anômalas;
- NSAIC: número total de sessões anômalas incorretamente classificadas como normais.

O processo de validação dos resultados das ferramentas de detecção foi assistido pelos recursos de classificação manual e visualização do tráfego do

sistema ADTRAF e foram utilizados os valores armazenados no banco de dados para fins de comparação dos resultados.

Estudos preliminares de aplicação das redes neurais SOM e MLP foram conduzidos em busca de um método eficiente de detecção de anomalias no tráfego de rede.

Primeiramente, foram utilizadas redes SOM em cascata para *clustering* do tráfego de rede através de treinamento não supervisionado. Os mapas utilizados foram configurados com 9 neurônios de entrada e número fixo de *clusters* de saída. Quatro redes SOM com 50, 25, 10 e 2 *clusters*, respectivamente, foram empregadas. Os pesos de saída de uma rede constituíam a entrada para a próxima rede do conjunto. A tentativa era de reduzir gradativamente, através de agrupamento de entradas com as mesmas características pela rede neural, o grande volume de dados de entrada, organizando-o por etapas. Os *clusters* finais, da última rede em cascata, deveriam indicar duas classes para o tráfego global de entrada: tráfego normal e tráfego anômalo. Com esta técnica os resultados encontrados não foram satisfatórios.

Em seguida, uma rede SOM unidimensional com 9 neurônios de entrada e 2 neurônios (*clusters*) de saída foi utilizada e também não se obteve sucesso com os resultados desta rede para distinção entre tráfego normal e anômalo. Isto ocorreu porque a rede SOM possui a característica de não ser capaz de agrupar padrões que ocorrem esporadicamente em um conjunto, como é o caso dos padrões anômalos que ocorrem em quantidade muito menor que os padrões normais em qualquer tráfego de rede em correto funcionamento, mas dados freqüentes de natureza semelhante esta rede consegue agrupar bem.

Um terceiro passo de uso de rede neural para detecção de sessões de tráfego anômalas foi à aplicação da rede MLP, com treinamento supervisionado. Para isto, foram implementados 9 neurônios na camada de entrada (9 atributos de uma sessão de rede), 10 neurônios na camada oculta e 2 neurônios de saída

(valor próximo de 0 - normal ou próximo de 1 - anômalo). O processamento da rede foi realizado em no máximo 2000 épocas e foram utilizadas taxas de aprendizagem e momento variáveis. Porém os resultados de uso desta rede apenas também não foram satisfatórios.

7.4.3.1 Detecção de Anomalias utilizando as redes LVQ e MLP

Em estudo anterior, com o uso da rede MLP para classificar as sessões do tráfego como normal e anômalo, observou-se que esta rede polarizava na aprendizagem do tráfego normal, visto que a quantidade de sessões anômalas é muito menor do que a de sessões normais.

Melhores resultados foram obtidos com o uso combinado das abordagens LVQ e MLP, consecutivamente. Dado um conjunto de tráfego a ser analisado (tráfego total), este é separado em tráfego normal (T_n) e anômalo (T_a), como apresentado na Figura 7.11. Em seguida, o tráfego normal é apresentado à rede LVQ para *clustering*, gerando o tráfego normal clusterizado (T_{nc}). O tráfego normal clusterizado é concatenado ao tráfego anômalo e este conjunto é apresentado à rede MLP para treinamento da rede.

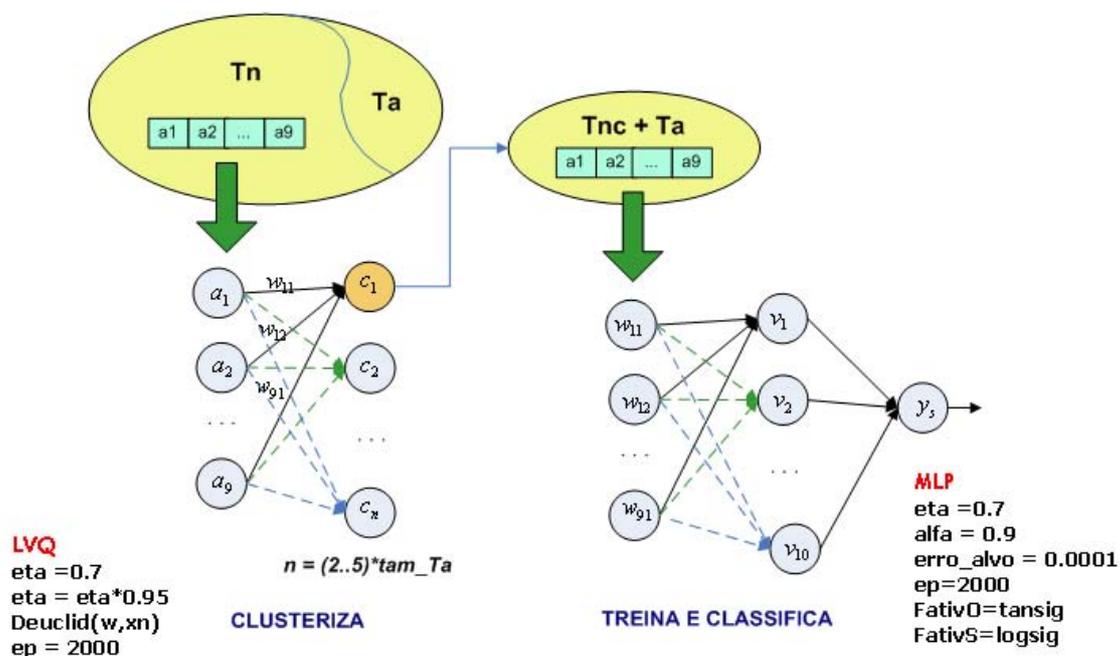


Figura 7.11 - Arquitetura da LVQ e MLP para detecção de anomalias

Em busca de boa generalização da rede, múltiplos conjuntos de dados de tráfego são processados cada um da seguinte maneira: agrupa-se a primeira metade do grupo de sessões normais do conjunto de dados de tráfego a ser analisado com a primeira metade do grupo de sessões anômalas deste conjunto, formando um conjunto de treinamento. O restante do conjunto de dados, segunda metade de sessões normais e segunda metade de sessões anômalas é agrupado e forma o conjunto de testes. Para a composição do conjunto de entrada para *clustering*, que posteriormente participará do conjunto de treinamento para a rede MLP, os conjuntos de treinamento resultantes são concatenados e formam o tráfego total a ser analisado, o qual é separado em tráfego normal (T_n) e anômalo (T_a) e introduzido na LVQ para *clustering*.

A rede LVQ é implementada em sua forma mais simples: nove neurônios de entrada e N neurônios ou clusters de saída, onde N é calculado como X (duas a cinco) vezes a quantidade de sessões de tráfego anômalo inseridas na rede. O número X (duas a cinco vezes) de clusters foi definido empiricamente nesta tese, mas estudos devem ser realizados em busca de dedução destes valores.

Após a separação do tráfego total em tráfego anômalo e normal, o tráfego normal é clusterizado através da LVQ utilizando o cálculo da distância Euclidiana entre os pesos da rede e os vetores de entrada, processando nove atributos de sessão por vez. Deste modo, os vetores característicos representativos do tráfego normal são gerados a partir da clusterização do conjunto de entrada de tráfego normal e compõem um conjunto reduzido de sessões normais. Este conjunto resultante é então agrupado com o conjunto original de tráfego anômalo e obtém-se um vetor de treinamento para a rede neural MLP.

A rede neural MLP realiza a classificação do tráfego após o modelo ser treinado com as sessões normais do tráfego reduzido e as sessões anômalas. O tipo de sessão (normal ou anômala) é passado à rede através do campo 'decisão', além dos valores dos nove atributos.

A arquitetura da rede neural MLP utilizada nesta etapa é composta por nove neurônios na camada de entrada, dez neurônios na camada oculta e um neurônio na camada de saída. A saída da rede é comparada com um limiar pré-definido para classificação da sessão como normal ou anômala.

Os valores do vetor de entrada para treinamento e teste da rede MLP são normalizados pelo valor máximo do atributo correspondente no vetor de treinamento. O critério de parada do treinamento desta rede é o número máximo de 2000 épocas ou quando o erro na saída do treinamento da rede para uma determinada época for menor que o erro desejado.

Os dados utilizados para os estudos destas abordagens são apresentados na Tabela 7.9.

Tabela 7.9 - Conjuntos de dados utilizados nos testes LVQ e MLP

RedeBeta	Normal	Ataques	Total
SETBETA_06	3050	3	3053
SETBETA_07	1989	3	1992
SETBETA_08	15304	5	15309
SETBETA_09	65602	37	65639
SETBETA_10	80938	37	80975
RedeLab	Normal	Ataques	Total
SETLAB_01	381	0	381
SETLAB_02	381	4	385
SETLAB_03	381	10	391
SETLAB_04	381	70	451
SETLAB_05	804	0	804
SETLAB_06	804	4	808
SETLAB_07	804	10	814
SETLAB_08	804	70	874

Tabela 7.10 - Resultados de Detecção de Anomalias utilizando LVQ e MLP

RedeBeta	Normal	Ataques	Total
SETBETA_06	3050	3	3053
SETBETA_07	1989	3	1992
SETBETA_08	15304	5	15309
SETBETA_09	65602	37	65639
SETBETA_10	80938	37	80975

Teste	Conjunto Utilizado
1	SETBETA_06treina e SETBETA_07treina
2	SETBETA_06treina e SETBETA_07treina e SETBETA_08treina
3	SETBETA_06treina e SETBETA_07treina e SETBETA_08treina
4	SETBETA_06treina e SETBETA_08treina
5	SETBETA_06treina e SETBETA_08treina
6	SETBETA_06treina e SETBETA_08treina

Teste 1		treino: 5,68 s - epocas=2.000		clusters=2*tam_ta	
Dados Teste	Tempo Teste	TFP (%)	TFN (%)	TD (%)	TE (%)
SETBETA_06	0.01	2.36	0	99.86	0.13
SETBETA_07	0.01	6.13	0	99.79	0.2
SETBETA_08	0.18	18.63	0	99.93	0.06
SETBETA_09	0.21	3.71	0	99.88	0.11
SETBETA_10	0.29	3.68	0	99.92	0.09
Média= 6,9%					
Teste 2		treino: 16,01 s - epocas=2.000		clusters=2*tam_ta	
Dados Teste	Tempo Teste	TFP (%)	TFN (%)	TD (%)	TE (%)
SETBETA_06	0.03	89.5	0	99.86	0.13
SETBETA_07	0.01	46.83	0	99.79	0.2
SETBETA_08	0.06	29.96	0	99.93	0.06
SETBETA_09	0.15	68.08	0	99.88	0.11
SETBETA_10	0.23	74.69	0	99.91	0.09
Média= 61,81%					
Teste 3		treino: 33,26 s - epocas=2.000		clusters=3*tam_ta	
Dados Teste	Tempo Teste	TFP (%)	TFN (%)	TD (%)	TE (%)
SETBETA_06	0.03	18.36	0	99.86	0.13
SETBETA_07	0.12	7.03	0	99.79	0.21
SETBETA_08	0.09	17.68	0	99.93	0.06
SETBETA_09	0.34	9.2	0	99.88	0.11
SETBETA_10	0.83	9.03	0	99.92	0.08

Média=12,26%

Teste 4		treino: 12,35 s - epocas=2.000		clusters=2*tam_ta	
Dados Teste	tempo Test	TFP (%)	TFN (%)	TD (%)	TE (%)
SETBETA_06	0.02	83.14	0	99.86	0.13
SETBETA_07	0.01	31.99	0	99.79	0.21
SETBETA_08	0.11	28.61	0	99.93	0.06
SETBETA_09	0.29	58.18	0	99.88	0.11
SETBETA_10	0.45	67.52	0	99.92	0.08
Média= 53,88%					
Teste 5		treino: 56,37 s - epocas=2.000		clusters=3*tam_ta	
Dados Teste	tempo Test	TFP (%)	TFN (%)	TD (%)	TE (%)
SETBETA_06	0.03	17.51	0	99.86	0.14
SETBETA_07	0.06	6.83	0	99.79	0.21
SETBETA_08	0.1	17.64	0	99.93	0.07
SETBETA_09	0.33	8.54	0	99.88	0.12
SETBETA_10	0.35	8.91	0	99.9	0.09
Média= 11,86%					
Teste 6		treino: 68,46 s - epocas=2.000		clusters=5*tam_ta	
Dados Teste	tempo Test	TFP (%)	TFN (%)	TD (%)	TE (%)
SETBETA_06	0.03	12.91	0	99.86	0.14
SETBETA_07	0.04	6.53	0	99.79	0.2
SETBETA_08	0.11	17.55	0	99.93	0.06
SETBETA_09	0.28	7.05	0	99.88	0.12
SETBETA_10	0.33	7.81	0	99.9	0.09

Média=10,37%

TFP = no. SN incorretamente classificadas / total SN

TFN = no. SA incorretamente classificadas / total A

TD = no. SA corretamente detectadas / total A

Na tabela 7.10 são apresentados os resultados de detecção de ataques em conjuntos de dados do tráfego HTTP da RedeBeta, utilizando a combinação de técnicas LVQ e MLP.

Para o primeiro caso estudado, referente ao teste 1, foram concatenados os conjuntos SETBETA_06treina e SETBETA_07treina, o tráfego normal do conjunto concatenado foi clusterizado pela LVQ gerando o conjunto de vetores característicos do tráfego normal que foi concatenado ao conjunto de vetores do tráfego anômalo original para a composição do conjunto de treinamento para a MLP. Este mesmo procedimento foi efetuado para os demais casos. Os conjuntos de dados da Tabela 7.11 foram utilizados nos testes.

Tabela 7.11 - Descrição dos conjuntos de teste

Teste	Conjunto Utilizado
1	SETBETA_06treina e SETBETA_07treina
2	SETBETA_06treina e SETBETA_07treina e SETBETA_08treina
3	SETBETA_06treina e SETBETA_07treina e SETBETA_08treina
4	SETBETA_06treina e SETBETA_08treina
5	SETBETA_06treina e SETBETA_08treina
6	SETBETA_06treina e SETBETA_08treina

Em cada teste, o método de detecção apresentou taxa de detecção acima de 99,79 % e 0% de falsos negativos.

A média de valores para a taxa de falsos positivos em cada teste foi, respectivamente, 6,9%, 61,81%, 12,26%, 53,88%, 11,86% e 10.37%. Os piores resultados de taxa de falsos positivos foram encontrados para os testes 2 e 4 . No teste 2 foi incluído um maior volume de dados (SETBETA_08treina), contendo acima de 7.500 registros ao conjunto concatenado SETBETA_06treina e SETBETA_07treina. A tarefa de clusterização continuou gerando vetores característicos num total de o dobro do valor do número de vetores anômalos. Ao definir um maior número de *clusters* (teste 3) o classificador gerou taxa menor de falsos positivos.

O mesmo ocorreu no teste 4, e foi corrigido através do teste 5, com o aumento do número de clusters. Observou-se, portanto, que a definição do número de *clusters* para agrupamento dos vetores do tráfego normal influencia nos resultados do classificador, conforme visto também no teste 6. O número de *clusters* para agrupamento foi definido empiricamente nesta tese como duas a cinco vezes a quantidade de sessões de tráfego anômalo. Portanto, para maiores conjuntos de dados de tráfego, o uso de maior número de *clusters* na rede LVQ melhora o resultado de classificação.

A partir dos resultados obtidos neste estudo verifica-se a capacidade de processamento rápido das redes neurais para as tarefas de classificação do tráfego estudado, apresentando taxas satisfatórias de detecção e de erro.

A combinação das técnicas LVQ e MLP gerou melhor resultado de detecção de tráfego HTTP anômalo com os conjuntos de dados utilizados. Porém, é uma técnica que deve ser mais explorada, considerando um escopo maior de casos de testes e observando se os resultados se manterão satisfatórios.

Para aplicação adequada desta técnica em ambiente de produção, deve-se considerar a diversidade de aplicações e serviços fornecidos na rede, a começar pela verificação se os atributos utilizados neste trabalho se aplicam aos demais tipos de tráfego de rede.

7.4.3.2 Detecção de Anomalias utilizando as Técnicas LOF e LSC

Um dos primeiros estudos realizados envolveu o uso de técnicas estatísticas LOF e LSC para detecção de tráfego anômalo em conjuntos de dados da RedeLab, contendo sessões normais e anômalas. Alguns dos conjuntos testados através destes métodos estão descritos na Tabela 7.12.

Tabela 7.12 - Exemplos de Conjuntos de Dados Analisados

Tráfego	Normal	Ataque	Total de Sessões
SET_01	1685	253	1938
SET_02	1585	17	1602
SET_03	206	4	210
SET_04	206	253	459
SET_05	804	253	1057
SET_06	804	0	804
SET_07	1.010	0	1.010

As técnicas LOF e LSC utilizam o parâmetro *MinPts*, que corresponde ao número máximo de pontos vizinhos de um ponto P (no caso, um registro de sessão de rede, com nove atributos) a ser analisado no processo de detecção de ataques. Foram avaliados os resultados de classificação destas técnicas, utilizando diferentes valores para *MinPts* e os resultados são apresentados a seguir.

As taxas de detecção de sessões anômalas, bem como a taxa de falsos positivos, em que sessões normais são incorretamente classificadas como anômalas, podem ser vistos nas Tabelas 7.13 e 7.14. O tempo de processamento do algoritmo é dado em segundos e as taxas, em percentagem.

Tabela 7.13 - *Clustering* do tráfego com LOF e variação de Minpts

Tráfego	MINPOINTS = 2			MINPOINTS = 5			MINPOINTS = 10		
	Tempo	TD (%)	TFP(%)	Tempo	TD (%)	TFP(%)	Tempo	TD (%)	TFP(%)
SET_01	7,5790	97,23	44,69	7,4840	97,23	44,69	7,8130	97,23	44,69
SET_02	4,8590	100,00	61,96	4,8290	100,00	59,31	5,0470	100,00	61,96
SET_03	0,0780	100,00	0,97	0,0780	100,00	0,99	0,0780	100,00	98,06
SET_04	0,3750	97,23	27,67	0,3750	97,23	27,67	0,4220	97,23	27,67
SET_05	2,1410	96,84	63,06	2,1880	96,84	54,60	2,2500	97,23	62,94
SET_06	1,1090	-	18,53	1,1250	-	20,52	1,1720	-	22,01
SET_07	1,7820	-	14,85	1,7810	-	16,34	1,8900	-	17,52
Média	2,5604	98,26	33,10	2,5514	98,26	32,01	2,6674	98,33	47,83

Analisando os resultados apresentados na Tabela 7.13, verificou-se que a técnica LOF apresentou taxa média de detecção de 98,2% e taxa média de falsos positivos de 37,64%. Para maior valor de *MinPts* (*MinPts*=10) o algoritmo gerou maior taxa de falsos positivos e para *MinPts*=2 e *MinPts*=5 foi encontrada taxa média aproximada de 32,5% de falsos positivos. Tempo médio de processamento de 2,59 segundos foi gasto por este método para processamento dos conjuntos de dados.

Tabela 7.14 - *Clustering* do tráfego com LSC e variação de Minpts

Tráfego	MINPOINTS = 2			MINPOINTS = 5			MINPOINTS = 10		
	Tempo	TD (%)	TFP(%)	Tempo	TD (%)	TFP(%)	Tempo	TD (%)	TFP(%)
SET_01	7,1090	95,65	4,33	7,2190	95,65	4,33	7,1250	95,65	4,33
SET_02	4,6400	94,12	17,67	4,6100	97,51	17,67	4,7040	94,12	17,79
SET_03	0,062	43,06	0,00	0,0790	100,00	0,00	0,0780	0,00	99,03
SET_04	0,3600	61,66	27,64	0,3590	61,66	0,00	2,0310	61,66	23,79
SET_05	2,0310	95,65	17,29	2,0630	95,65	17,65	2,0310	95,65	25,87
SET_06	1,1250	-	1,24	1,1250	-	6,72	1,1250	-	3,61
SET_07	1,7650	-	0,99	1,7970	-	5,35	1,1750	-	2,87
Média	2,4417	78,02	9,88	2,4645	90,04	7,38	2,6098	69,41	25,32

Através da técnica LSC, conforme apresentado na Tabela 7.14, foram obtidas taxa média de detecção de ataques de 79,15% e taxa média de falsos positivos de 14,20%. Para valor de *MinPts*=5, o algoritmo gerou menor taxa média de falsos positivos. O tempo médio de processamento do método LSC para diferentes valores de *MinPts* apresentou pequena variação. Tempo de processamento de 2,50 segundos, em média, foi utilizado por este método para processamento de todos os conjuntos de dados.

Comparando os resultados de ambas as técnicas de detecção de *outliers*, cujos valores médios são apresentados na Tabela 7.15, constatou-se que a técnica de LSC apresentou tempo médio de processamento muito próximo ao gasto pela técnica LOF para análise dos mesmos conjuntos de dados. Maior taxa de detecção, em média, foi alcançada pela técnica LOF, mas menor taxa de falsos positivos foi obtida através da técnica LSC.

Tabela 7.15 - Valores médios de Tempo de Processamento e Taxas

	MINPOINTS = 2			MINPOINTS = 5			MINPOINTS = 10		
	Tempo	TD (%)	TFP(%)	Tempo	TD (%)	TFP(%)	Tempo	TD (%)	TFP(%)
LOF	2,5604	98,26	33,10	2,5514	98,26	32,01	2,6674	98,338	47,83
LSC	2,4417	78,028	9,88	2,4645	90,04	7,3885	2,6098	69,41	25,32

8 CONCLUSÕES

Diversas técnicas para reconhecimento de eventos de intrusão têm sido implementadas, entretanto, observa-se a necessidade de uma metodologia de fácil aplicação para auxílio aos analistas na detecção de ataques à rede.

Nesta tese é proposta uma metodologia de apoio à detecção de ataques no tráfego de redes, englobando um conjunto de fases, métodos, técnicas e ferramentas.

As principais contribuições desta tese são os métodos desenvolvidos de detecção de assinaturas e de anomalias no tráfego de redes de computadores, baseados em redes neurais.

Três principais aplicações de segurança foram implementadas, incluindo: a aplicação ANNIDA, para detecção de assinaturas no conteúdo de pacotes de rede; a ferramenta RGCOM, para auxílio na análise visual do comportamento do tráfego de rede; e o sistema ADTRAF, para detecção de anomalias no tráfego. Através destas ferramentas, tornou-se possível a detecção de ataques no tráfego HTTP das redes monitoradas.

A aplicação das redes neurais de rápida classificação 'Hamming Net e LVQ' para detecção de assinaturas no *payload* dos pacotes de rede foi bem sucedida, e constatou-se que a rede LVQ é mais rápida que a Hamming Net na detecção de assinaturas, reduzindo o tempo de detecção de assinaturas da ordem de minutos para segundos. Embora ambas as redes tenham apresentado elevada taxa de detecção, como a rede LVQ apresentou melhor tempo de processamento, esta foi readaptada para a detecção de assinaturas a partir de dados de cabeçalho dos pacotes de rede, produzindo resultados muito satisfatórios. Constatou-se, portanto, neste trabalho, que a rede neural LVQ é uma técnica supervisionada viável para a detecção de padrões de ataques em dados de carga útil e dados de cabeçalho dos pacotes de rede e

um método promissor a ser comparado, em termos de desempenho, com outros métodos existentes de busca de padrões.

Uma limitação da aplicação ANNIDA é a incapacidade de detectar novos padrões de ataques, diferentes daqueles previamente cadastrados na base de exemplares. Como os vetores exemplares e de entrada da aplicação são *strings* contidas no *payload* dos pacotes, pré-processadas e representadas em formato hexadecimal criptografado e único, esta aplicação não permite atualmente a detecção de variações de *strings* maliciosas de ataque, que seria possível se comparados os caracteres das *strings* de entrada e exemplares.

A rede neural LVQ utilizada na ANNIDA também foi aplicada com êxito para detecção de padrões de comportamento do tráfego a partir de dados do cabeçalho (atributos de sessões) dos pacotes de rede no sistema ADTRAF. Embora uma medida de 100% de similaridade tenha sido realizada para casamento entre entrada e exemplar, o uso de diferentes medidas de similaridade no sistema ADTRAF pode permitir a detecção de ataques que pertençam à mesma categoria DoS, Probe, U2R ou R2L ou variações de ataques.

Para detecção de tráfego anômalo em conjuntos de dados, foram aplicadas as redes neurais SOM, LVQ e MLP. O uso da rede SOM unidimensional para *clustering* de todo o tráfego sob análise e o uso da rede neural MLP para classificação do tráfego, separando-o em tráfego normal e anômalo, não foram técnicas bem sucedidas. Resultados positivos, com taxa de falsos negativos e taxa de falsos positivos aceitáveis, foram obtidos com a combinação das técnicas LVQ e MLP para detecção de tráfego anômalo nos conjuntos de dados utilizados.

Técnicas estatísticas de detecção de *outliers* também foram utilizadas nos estudos preliminares para identificação de tráfego anômalo. A técnica LSC apresentou melhor taxa de falsos positivos quando comparada a LOF,

enquanto esta última apresentou melhor taxa de detecção. Porém, ambas as técnicas requerem testes com maiores conjuntos de dados.

Recentemente, têm-se ampliado as pesquisas sobre diferentes formas de representação gráfica do tráfego de rede, visto que uma imagem vale mais que mil dados para rápida interpretação. De fato, técnicas de visualização do comportamento do tráfego de rede podem auxiliar os analistas na identificação de evidências de possíveis anomalias para análise posterior mais criteriosa.

Um primeiro esforço nesta direção foi a implementação do software RGCOM com o uso de software livre Java e apresentação de atributos do cabeçalho dos pacotes. Através desta ferramenta, em estágio básico de desenvolvimento, é possível identificar evidências de possíveis desvios no tráfego, mas seu uso deve ser combinado com outras técnicas para uma análise mais criteriosa, completa e precisa do tráfego de rede.

A metodologia proposta auxilia os analistas na implantação de camada adicional de segurança à rede; é de fácil aplicação; incentiva o uso de soluções de domínio público; proporciona rapidez de processamento com aplicação de redes neurais para detectar assinaturas e anomalias; provê recursos gráficos para facilitar análise do tráfego; e permite a adaptação de técnicas para análise de tráfego de rede diversificado.

Dentre as limitações existentes no protótipo desenvolvido, incluem-se: análise restrita a dados do tráfego de rede, testes de detecção de ataques no tráfego HTTP apenas; detecção de ataques em modo *off-line*; dados criptografados não são tratados e as técnicas de detecção adotadas não são reativas.

Porém, a metodologia proposta é escalável e pode ser expandida para melhorar o processo de detecção de ataques, através de utilização de dados de hosts e de sistemas para análise, além de dados do tráfego de rede; inclusão ou remoção de atributos para análise de dados de tráfego diferente de HTTP; aperfeiçoamento dos métodos para detecção de ataques em tempo real; desenvolvimento de solução para tratamento de dados criptografados; e inclusão de técnicas para tornar os métodos de detecção reativos.

Utilizou-se o tráfego do serviço de rede HTTP para aplicar a metodologia proposta porque este serviço é atualmente o mais acessado pelos usuários, produzindo, assim, maior quantidade de tráfego a analisar, e por ser o serviço de rede mais explorado pelos atacantes, devido a sua disponibilidade na *World Wide Web* e as suas vulnerabilidades.

Com relação à análise de comportamento do tráfego, ou seja, detecção por anomalia ou desvios de comportamento do tráfego, é possível aplicar a metodologia para análise de tráfego DNS, por exemplo, ou para análise de qualquer outro tipo de tráfego, diferente de HTTP. Porém, isto requer uma investigação dos atributos de sessões que caracterizam cada tipo de tráfego e inclusão ou remoção de atributos relevantes ao protótipo desenvolvido.

Nesta tese foram enfrentados vários desafios, desde a preparação do ambiente de captura de pacotes de rede, ataques ao ambiente, ao desenvolvimento das ferramentas de detecção. Alcançar menor taxa de falsos positivos para os métodos de detecção de anomalias foi a tarefa mais complexa e requer continuação. Modelar as *strings* para entrada nas redes neurais dos modelos detectores de assinaturas foi um problema que demandou tempo razoável de pesquisa, bem como a criação de uma estratégia para a varredura de conjuntos de dados em busca de conteúdo malicioso, entre outros.

A partir desta tese, vários caminhos poderão ser trilhados, englobando os seguintes trabalhos futuros:

- Testes e ajustes das técnicas aplicadas para detecção de ataques em um domínio mais amplo de atuação, envolvendo estudos com conjuntos de dados maiores e mais diversificados, para observar o comportamento dos métodos em termos de precisão de resultados;
- Busca de técnicas alternativas para detecção de anomalias que produzam altas taxas de detecção e baixas taxas de alarmes falsos;

- Adaptação dos métodos de detecção para análise de diferentes tipos de tráfego de rede;
- Adoção de estratégias para acelerar o processamento dos dados, visando análises de tráfego em tempo real;
- Estudo criterioso de contribuição de atributos para a escolha de dados mais significativos para a representação de diferentes tipos de tráfego de rede. O uso de critérios de categorização de atributos baseados em desempenho e tempo de processamento dos métodos de detecção é uma alternativa para análise de contribuição de atributos;
- Ampliação da capacidade de detecção de assinaturas da aplicação ANNIDA e acoplamento desta ao sistema ADTRAF;
- Melhoria da ferramenta RGCOM para permitir ao usuário obter mais informações sobre as sessões visualmente identificadas como evidências de possíveis anomalias.

REFERÊNCIAS BIBLIOGRÁFICAS

ABDI, M.; SZU, H. Independent component analysis (ICA) and self-organizing map (SOM) approach to multidetection system for network intruders. In: SPIE INTERNATIONAL SOCIETY FOR OPTICAL ENGINEERING, 2003, [S.l.]. **Proceedings...** [S.l.]: SPIE Digital Library, 2003, v. 5102, p. 348-353.

AGRAWAL, R.; SRIKANT, R. Fast algorithms for mining association rules. In: INTERNATIONAL CONFERENCE ON VERY LARGE DATA BASES, 20., 1994, Santiago, Chile. **Proceedings...** Santiago: VLDB, 1994, p. 487-499. ISBN 1558601538.

AGYEMANG, M.; EZEIFE, C. I. LSC-Mine: algorithm for mining local outliers. In: INFORMATION RESOURCE MANAGEMENT ASSOCIATION (IRMA) INTERNATIONAL CONFERENCE, 15., 2004, New Orleans. **Proceedings...** Hershey, PA: IRM press, 2004, v. 1, p. 5-8.

AMBWANI, T. Multiclass support vector machine implementation to intrusion detection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2003), 20., 2003, Portland, Oregon, USA. **Proceedings...** Piscataway, New Jersey, USA: IEEE, 2003. v. 3, p. 2300-2305. ISBN 0780378989.

BARFORD, P.; PLONKA, D. Characteristics of network traffic flow anomalies. In: ACM SIGCOMM INTERNET MEASUREMENT WORKSHOP, 1., 2001, San Francisco, CA. **Proceedings...** New York, NY: ACM Press, 2001. p. 69-73. ISBN 1581134355.

BARFORD, P.; KLINE, J.; PLONKA, D.; RON, A. A signal analysis of network traffic anomalies. In: ACM SIGCOMM INTERNET MEASUREMENT WORKSHOP, 2., 2002, Marseille, France. **Proceedings...** New York, NY: ACM Press, 2002. p. 71-82. ISBN 158113603X.

BEARAVOLU, R.; LAKKARAJU, K.; YURCIK, W.; RAJE, H. A visualization tool for situational awareness of tactical and strategic security events on large and complex computer networks. In: IEEE MILITARY COMMUNICATIONS CONFERENCE (MILCOM), 2003, Urbana, IL, USA. **Proceedings...** IEEE Xplore, 2003. p. 850-855. Digital Object Identifier 10.1109/MILCOM.2003.1290234.

BLOEDORN, E.; CHRISTIANSEN, A. D.C.; HILL, W. ; SKORUPKA, C.; TALBOT, L.M.; TIVEL, J. **Data mining for network intrusion detection: how to get started.** Disponível em: <www.mitre.org/work/tech_papers/tech_papers_01/bloedorn_datamining/bloedorn_datamining.pdf>. Acesso em: 12 ago. 2003.

BONIFACIO, J. M.; CANSIAN, A. M.; CARVALHO, A. C. P. L. F.; MOREIRA, E. Neural networks applied in intrusion detection system. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'1998), 1998, Anchorage, Alaska, USA. **Proceedings...**, Los Alamitos, CA: IEEE Computer Society Press, 1998. p. 205-210.

BREUNIG, M. M.; KRIEGEL, H. P.; NG, R. T.; SANDER, J. LOF: Identifying density-based local outliers. **ACM SIGMOD Record**, v. 29, n. 2, p. 93-104, 2000. ISSN:0163-5808.

BRIDGES, S. M.; HOSSAIN, M.; VAUGHN, R. B. Adaptive intrusion detection with data mining. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2003, Washington, DC, USA. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2003. v. 4, p. 3097-3103. ISSN: 1072922X.

CABRERA, J. B. D.; RAVICHANDRAN, B.; MEHRA, R. K. Statistical traffic modeling for network intrusion detection. In: IEEE INTERNATIONAL SYMPOSIUM ON MODELING, ANALYSIS AND SIMULATION OF COMPUTER AND TELECOMMUNICATION SYSTEMS, 8., 2000, San

Francisco, CA, USA. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2000. p. 466.

CANSIAN, A M. **Desenvolvimento de um sistema adaptativo de detecção de intrusos em redes de computadores.** 1997. 154 p. Tese (Doutorado em Física) -Universidade de São Paulo, São Carlos, 1997.

CASWELL, B.; BEALE, J.; FOSTER, J. C.; POSLUNS, J. **Snort 2 - sistema de detecção de intruso open source,** Rio de Janeiro: Alta Books, 2003. 384 p. ISBN 8576080125.

CERT.br - Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil, **Estatísticas sobre notificações de incidentes.** Disponível em: <<http://www.cert.br>>. Acesso em: 23 jul. 2006.

CHAVES, C.H.P.; MONTES, A. Detecção de backdoors e canais dissimulados. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA, 5., 2005, INPE, São José dos Campos, SP. **Anais...** São José dos Campos: INPE, 2005a. Disponível em : <http://hermes2.dpi.inpe.br:1905/col/dpi.inpe.br/hermes2@1905/2005/10.03.12.10/doc/dbcc_worcap_2005.pdf>. Acesso em: 3 jul. 2006.

CHAVES, C.H.P.; **Detecção de backdoors e canais dissimulados.** 2005. 145 p. Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, SP, 2005b.

CHAVES, M.H.P.C. **Análise de estado de tráfego de redes TCP/IP para aplicação em detecção de intrusão.** 2002. 172 p. (INPE-9625-TDI/845). Dissertação (Mestrado em Computação Aplicada) - Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2002.

DENNING, D. E. An intrusion-detection model. **IEEE Transactions on Software Engineering**, v.13, n. 2, p. 222-232, February 1987.

DICKERSON, J.E.; DICKERSON, J.A. Fuzzy network profiling for intrusion detection. In: INTERNATIONAL CONFERENCE OF THE NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY, 19., 2000, Atlanta, GA, USA. **Proceedings...** local: editora, 2000. p. 301-306. ISBN 0780362748. Disponível em: <<http://home.eng.iastate.edu/~julied/publications/NAFIPSpaper2000.pdf>>. Acesso em 21 ago 2005.

DICKERSON, J.E. ; JUSLIN*, J. ; KOUKOUSOULA*, O. ; DICKERSON, J.A. Fuzzy intrusion detection. In: NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY INTERNATIONAL CONFERENCE, 20., 2001, Vancouver, British Columbia. **Proceedings...** Piscataway, NJ: IEEE, 2001. v.3, p. 1506-1510.

DOKAS, P.; ERTOZ, L.; KUMAR, V.; LAZAREVIC,A.; SRIVASTAVA, J.; TAN, P. **Data mining for network intrusion detection**. Disponível em: <http://www.cs.umn.edu/research/minds/papers/nsf_ngdm_2002.pdf>. Acesso em: 12 out. 2003.

ENTERASYS networks, Inc. **DRAGON IDS/IPS**. Disponível em: <<http://www.enterasys.com/products/advanced-security-apps/dragon-intrusion-detection-protection.aspx>> . Acesso em: 25 set. 2006.

ERTOZ, L.; EILERTSON, E.; LAZAREVIC, A.; TAN, P.; SRIVASTAVA, J.; KUMAR, V.; DOKAS, P. **The MINDS - Minnesota Intrusion Detection System**. Disponível em: <<http://www.cs.umn.edu/research/minds/>>. Acesso em: 12 nov. 2003a.

ERTOZ, L., LAZAREVIC, A., EILERTSON, E., TAN, P., DOKAS, P., KUMAR, V., SRIVASTAVA, J. Protecting against cyber threats in networked information systems. In: SPIE ANNUAL SYMPOSIUM ON AEROSENSE, BATTLESPACE DIGITALIZATION AND NETWORK CENTRIC SYSTEMS, 3., 2003, Orlando, FL. **Proceedings...** Bellingham, WA: International Society for Optical Engineering, 2003b. n. 5101, p. 51-56. ISSN: 0277-786X.

ERTOZ, L.; EILERTSON, E.; LAZAREVIC, A.; TAN, P.; DOKAS, P.; SRIVASTAVA, J.; KUMAR, V. **Detection and summarization of novel network attacks using data mining**. Technical Report. Minneapolis, USA: University of Minnesota, 2003c. 20 p.

FAGUNDES, L. L. **Metodologia para avaliação de sistemas de detecção de intrusão**. 2002. 68 p. Trabalho de Conclusão de Curso (Graduação em Informática) - Universidade do Vale do Rio dos Sinos, São Leopoldo, RS, 2002.

FAUSETT, L. **Fundamentals of neural networks: architectures, algorithms, and applications**. 1. ed. New Jersey: Prentice-Hall, 1994. 461 p. ISBN 0133341860.

FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery: an overview. In: FAYYAD, U. M.; PIATETSKY-SHAPIRO, G.; SMYTH, P.; UTHURUSAMY, R. (Ed.) **Advances in Knowledge Discovery and Data Mining**. Menlo Park, CA: American Association for Artificial Intelligence, 1996a, p. 1-34.

GONÇALVES, E. C.; PLASTINO, A. Mineração de regras de associação híbridas. In: ENCONTRO NACIONAL DE INTELIGÊNCIA ARTIFICIAL, CONGRESSO DA SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC-ENIA), 5., 2005, São Leopoldo, RS. **Anais...** [S.l.]: [s.n.], 2005. p. 811-820.

GOLDSCHMIDT, R.; PASSOS, E. **Data Mining: um guia prático**. 1. ed. Rio de Janeiro: Elsevier Editora, 2005. 255 p. ISBN 8535218777.

GOTO, K.; KEENI, K. On classification of alarms from network intrusion detection system using multi-layer feed-forward neural networks. In: IASTED INTERNATIONAL CONFERENCE ON NEURAL NETWORKS AND COMPUTATIONAL INTELLIGENCE (NCI'2003), 2003, Cancun, Mexico. **Proceedings...** Anaheim, CA: IASTED/ACTA Press, 2003. p. 163-168.

HAYKIN, S., **Redes neurais princípios e práticas**, 2. ed. Porto Alegre: Bookman, 2001. 900 p. ISBN 8573077182.

HEBERLEIN, L.T.; DIAS, G. V. ; LEVITT, K. N.; MUKHERJEE, B.; WOOD, J.; WOLBER, D. A Network Security Monitor. In: IEEE SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY, 1990, Oakland, CA.

Proceedings... New Jersey, USA: IEEE Press, 1990. p. 296–304.

HOFMANN, A.; SCHMITZ, C.; SICK, B. Intrusion Detection in Computer Networks with Neural and Fuzzy Classifiers. In: JOINT INTERNATIONAL CONFERENCE ON ARTIFICIAL NEURAL NETWORKS AND NEURAL INFORMATION PROCESSING (ICANN/ICONIP'2003), 2003, New York, Istanbul. **Proceedings...** Berlin: Springer Verlag, 2003. p. 316 - 324.

JACOBSEN, V.; LERES, C.; MCCANNE, S. **tcpdump**. LBNL, University of California, June 1997. Disponível em: <ftp://ftp.ee.lbl.gov/tcpdump.tar.Z>. Acesso em: 12 jan. 2005.

JIANG, J.; ZHANG, C.; KAMEL, M. RBF-based real-time hierarchical intrusion detection systems. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2003), 2003, Portland, Oregon, USA. **Proceedings of...** Piscataway, NJ, USA: IEEE, 2003. v. 2 , p. 1512-1516.

KAYACIK, H.G.; ZINCIR-HEYWOOD, A.N.; HEYWOOD, M.I. On the capability of an SOM based intrusion detection system. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS, (IJCNN'2003), 2003, Portland, Oregon, USA. **Proceedings of...** Piscataway, NJ, USA: IEEE, 2003. v. 3, p. 1808-1813.

KIM, S.S.; REDDY A. L. N.; VANNUCCI M. Detecting traffic anomalies through aggregate analysis of packet header data. **Lecture notes in Computer Science**, v. 3042, p. 1047–1108, 2004.

KIM, S.S.; REDDY, A. L. N. A study of analyzing network traffic as images in real-time. In: ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER

AND COMMUNICATIONS SOCIETIES (IEEE INFOCOM'2005), 24., 2005, Miami, Florida, USA. **Proceedings...** Piscataway, NJ, USA: IEEE, 2005a. v. 3, p. 2056-2067.

KIM, S.S.; REDDY, A. L. N. Modeling network traffic as images. In: IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS (ICC'2005), 41., 2005, Seoul, Korea. **Proceedings...** [S.I.]: IEEE Explore, 2005b. V. 1, p. 168-172. ISSN : 0536-1486. ISBN: 0-7803-8938-7.

KIM, S.S.; REDDY, A. L. N. Netviewer: A network traffic visualization and analysis tool. In: USENIX SYSTEM ADMINISTRATION CONFERENCE (LISA'2005), 19., 2005, San Diego, CA. **Proceedings...** Berkeley, CA: USENIX Association, 2005c. p. 185-196.

KNORR, E.; NG, R. Algorithms for mining distance-based outliers in large data sets. In: VERY LARGE DATABASES (VLDB) CONFERENCE, 24., 1998, local. **Proceedings...** New York: Morgan Kaufmann Publishers, 1998. p. 392-403.

KOHONEN, T. **Self-organizing and associative memory.** 3. ed. New York: Springer-Verlag, 1989.

LABIB, K.; VEMURI, R. **NSOM: A real-time network-based intrusion detection system using self-organizing maps.** Technical report, Davis, CA: Dept. of Applied Science, University of California, 2002. 6 p.

LAKHINA, A.; CROVELLA M.; CHRISTOPHE, D. Diagnosing network-wide traffic anomalies. In: ACM SIGCOMM 2004 CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION (ACM'2004), 2004, Portland, Oregon, USA. **Proceedings...** New York, NY: ACM Press, 2004. v. 34, n. 4, p. 219-230.

LAZAREVIC, A.; ERTOZ, L.; OZGUR, A; SRIVASTAVA, J.; KUMAR, V. A comparative study of anomaly detection schemes in network intrusion

detection. In: SIAM INTERNATIONAL CONFERENCE ON DATA MINING (SDM'2003), 3., 2003, San Francisco, CA. **Proceedings...** [S.l.]: [s.n.], 2003.

LEE, S.C.; HEINBUCH, D.V. Training a neural-network based intrusion detector to recognize novel attacks. **IEEE transactions on systems, man, and cybernetics, part A (systems and humans)**, v. 31, n. 4, p. 294-299, July 2001.

LEE, W.; STOLFO, S.J. Data mining approaches for intrusion detection. In: USENIX SECURITY SYMPOSIUM, 7., 1998, San Antonio, TX, USA. **Proceedings...** [S.l.]: [s.n.], 1998. p. 79-94.

LEE, W. **A data mining framework for constructing features and model for intrusion detection systems**. 1999, PhD Thesis - Columbia University, New York, 1999.

LI, Z. J.; WU, Y.; WANG, G. Y.; HAI, Y. J.; HE, Y. P. A new framework for intrusion detection based on rough set theory. In: DATA MINING AND KNOWLEDGE DISCOVERY: THEORY, TOOLS, AND TECHNOLOGY, 6., 2004, Orlando, Florida. **Proceedings...** [S.l.]: SPIE Digital Library, 2004. v. 5433, p. 122-130.

LINGXUAN, H.; ZHIJUN, H. Neural network-based intrusion detection systems. In: INTERNATIONAL CONFERENCE FOR YOU COMPUTER SCIENTIST: IN COMPUTER SCIENCE AND TECHNOLOGY IN NEW CENTURY, 6., 2001. **Proceedings...** [S.l.]: [s.n.], 2001. p. 296-298.

LUO, J.; BRIDGES, S. Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. **International Journal of Intelligent Systems**, v. 15, n. 1, p. 687-703, August 2000.

MAHONEY, M.; CHAN, P. Learning nonstationary models of normal network traffic for detecting novel attacks. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING

(KDD'2002), 8. 2002, Edmonton, CA. **Proceedings...** New York, NY: ACM Press, 2002. p. 376-385.

MANCILHA, T. D.; SILVA, L. S.; SALGADO, A. E. M; MONTES, A.; PAULA, A. R. Desenvolvimento em java de uma ferramenta de visualização gráfica do tráfego de rede, **Revista Univap**, v. 13, n. 24, out. 2006. ISSN 1517-3275.

MARTINEZ, J. P. Proteção para ativos valiosos. **Tecnologia da Informação, Software e Serviços**, v. 1, n. 6, p. 52, set. 2006.

MARTINS, R. X. **Redes neurais artificiais, o aprendizado reproduzido em máquinas**. Disponível em: <<http://www.univir-mg.br/engconh/rna.pdf>>. Acesso em: 24 nov. 2003.

MOORE, S. J. **The boyer-moore fast string searching algorithm**. Disponível em: <<http://www.cs.utexas.edu/users/moore/best-ideas/string-searching/index.html>>. Acesso em: 18 fev. 2002.

MORADI, M.; ZULKERNINE, M. A neural network based system for intrusion detection and classification of attacks. In: IEEE INTERNATIONAL CONFERENCE ON ADVANCES IN INTELLIGENT SYSTEMS - THEORY AND APPLICATIONS (AISTA'2004), 2004, Luxembourg-kirchberg, Luxembourg. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2004. p. 148:1-6.

MORAES, C.R., **Estruturas de dados e algoritmos: uma abordagem didática**. 1. ed. São Paulo: Berkeley, 2001. 362 p.

MUKKAMALA, S.; JANOWSKI, G.; SUNG, A. H. Intrusion detection using support vector machines. In: HIGH PERFORMANCE COMPUTING SYMPOSIUM (HPC'2002), 10., 2002, San Diego, CA. **Proceedings...** [S.l.]: [s.n.], 2002a. p.178-183.

MUKKAMALA, S.; SUNG, A.H; JANOSKI, G. Intrusion detection using neural networks and support vector machines. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2002), 2002, Honolulu, HI. **Proceedings...** [S.l.]: [s.n.], 2002b. v. 2, p. 1702-1707.

MUKKAMALA, S.; SUNG, A. H. Detecting denial of service attacks using support vector machines. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS, 12., 2003, St. Louis, Missouri, USA. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2003a. v. 2, p. 1231-1236.

MUKKAMALA, S.; SUNG, A. H. A comparative study of techniques for intrusion detection. In: INTERNATIONAL CONFERENCE ON TOOLS WITH ARTIFICIAL INTELLIGENCE (ICTAI'2003), 15., 2003, Sacramento, California, USA. **Proceedings...** DBLP: IEEE Computer Society, 2003b. p. 570-577. ISBN: 0-7695-2038-3.

MUKKAMALA, S.; SUNG, A.H Artificial intelligent techniques for intrusion detection. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, no., 2003, Washington, DC, USA. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2003c. v. 2, p. 1231-1236.

MUKKAMALA S.; SUNG A. H. Feature selection for intrusion detection using neural networks and support vector machines. **Journal of the Transportation Research Board of the National Academics**, Transportation Research Record n. 1822, p. 33-39, 2003d.

MUKKAMALA S.; SUNG A. H. Identifying significant features for network forensic analysis using artificial intelligence techniques. **International Journal on Digital Evidence**, v. 1, n. 4, 2003e.

NEUMANN, P. G.; PORRAS, P. A. Experiences with Emerald to date. In: USENIX WORKSHOP ON INTRUSION DETECTION AND NETWORK MONITORING, 1., 1999, Santa Clara, CA. **Proceedings...** Menlo Park, CA: SRI International, 1999. p. 73-80.

NIPPON telephone and telegraph. **N-HASH Implementation**, Disponível em: <<http://www.ussrback.com/crypto/source/hash/nhash.c>>. Acesso em: 12 mar. 2006.

NORTHCUTT, S.; NOVAK, J. **Network intrusion detection**. 3. ed. New York: New Riders Publishing, 2002. 460 p. ISBN: 0735712654.

ONUT, L-V.; ZHU; B.; GHORBANI, A. A. A novel visualization technique for network anomaly detection. In: ANNUAL CONFERENCE ON PRIVACY SECURITY AND TRUST (PST'2004), 2., 2004, Fredericton, CA. **Proceedings...** [S.l.]: [s.n.], 2004. p. 167-174.

ONUT, L-V.; ZHU; B.; GHORBANI, A. A. SVision: a novel visual network-anomaly identification technique. **Computer & Security**, v. 26, n. 3, May 2007 Disponível em: < <http://dx.doi.org/10.1016/j.cose.2006.10.001> >

ORFILA, A.; CARBO, J.; RIBAGORDA, A. Fuzzy logic on decision model for IDS. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE'2003), 12., 2003, St. Louis, MO, USA. **Proceedings...** New Jersey, USA: IEEE Press, 2003. v. 2, p. 1237-1242.

OUYANG, M-G.; WANG, W-N.; ZHANG, Y-T. A fuzzy comprehensive evaluation based distributed intrusion detection. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS, 1., 2002, Beijing, China. **Proceedings...** Alamos, CA: IEEE Computer Society Press, 2002. v. 1, p 281-284.

OWASP: the open web application security project. **The ten most critical web application security vulnerabilities 2004**. Disponível em: <<http://www.owasp.org/index.php>>. Acesso em: 12 dez. 2006.

PAXSON, V. Experiences learned from Bro, **The Usenix Association Magazine**, p. 21-22, September 1999a.

PAXSON, V. Bro: A system for detecting network intruders in real-time. **Computer Networks**, v. 31, n. 23-24, p. 2435-2463, December 1999b.

PAXSON, V.; SOMMER, R. Enhancing byte-level network intrusion detection signatures with context. In: ACM Conference on Computer and Communications Security (ACM CCS'2003), Washington, DC, .USA. **Proceedings...** New York, NY: ACM Press, 2003. p. 262-271.

PETROVSKIY, M. A fuzzy kernel-based method for real-time network intrusion detection. **Lecture Notes In Computer Science**, v. 2877, n. 8, p.189-200, 2003.

PLONKA, D. FlowScan: A network traffic flow reporting and visualization tool. In: USENIX SYSTEM ADMINISTRATION CONFERENCE, 14., 2000, New Orleans, LA. **Proceedings...** Berkeley, CA: USENIX Association, 2000. p. 305-318.

POLITI, J. **Implementação de uma metodologia para mineração de dados aplicada ao estudo de núcleos convectivos**. 2005. 149 p. (INPE-14165-TDI/1082) Dissertação (Mestrado em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2005.

PORRAS, P. A.; NEUMANN, P. G. EMERALD: Event monitoring enabling responses to anomalous live disturbances. In: NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE (NISSC'1997), 20., 1997, Baltimore, Maryland. **Proceedings...** [S.l.]: [s.n.], 1997. p. 353-365.

QUANMIN, W.; WEIMIN, L. A model for intrusion detection based on fuzzy match and neural network. In: INTERNATIONAL SYMPOSIUM ON TEST AND MEASUREMENT (ISTM'2001), 4., 2001, Shanghai , China. **Proceedings...** [S.l.]: [s.n.], 2001. v. 1, p. 411-414.

RAHMAN, Z.; RAHMAN, A. S. S.; KHAN, L. **Survey reports on four selected research papers on data mining based intrusion detection**

system. Disponível em:

<<http://web2.uwindsor.ca/courses/cs/aggarwal/cs60564/surveys/ZillurRahmanKhan.doc>>. Acesso em: 12 jan. 2006.

RAPAKA, A.; NOVOKHODKO, A.; WUNSCH, D. Intrusion detection using radial basis function network on sequences of system calls. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2003), 2003, Portland, Oregon, USA. **Proceedings...** Piscataway, New Jersey, USA: IEEE, 2003. v. 3, p. 1820-1825.

REN; P.; GAO, Y.; LI, Z.; CHEN, Y.; WATSON, B. IDGraphs: intrusion detection and analysis using stream compositing. **Computer Graphics and Applications**, v. 26, n. 2, p. 28- 39, April 2006. ISSN: 0272-1716.

RIBEIRO, U. **Certificação linux**. 1. ed. Rio de Janeiro: Axcel Books do Brasil, 2004. 472 p. ISBN: 8573232323.

ROSSI, G.B.; TAVARES, D.M.; CASTEJON, E. F. **Acme! (Advanced Counter Measures Environment). Um mecanismo de captura e análise de pacotes para aplicação em detecção de assinaturas de ataque.**

Disponível em: <<http://www.cerias.purdue.edu/coast/intrusion-detection/ids.html>>. Acesso em: 18 jul. 2006.

SANDRI, S.; CORREA, C., Lógica nebulosa. In: ESCOLA DE REDES NEURAI, CONSELHO NACIONAL DE REDES NEURAI, 5., 1999, ITA, São José dos Campos , São Paulo. **Anais...** São José dos Campos: ITA, 1999. p 73-90.

SHAH, H.; UNDERCOFFER, J.; JOSHI, A. Fuzzy clustering for intrusion detection. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS (FUZZ-IEEE'2003), 12., 2003, St. Louis, MO, USA. **Proceedings...** New Jersey, USA: IEEE Press, 2003. v. 2, pp. 1274-1278.

SHEMA, M.; TRADUCAO DE VIEIRA, M. **Hack notes: segurança na web: referência rápida**. 1. ed. Rio de Janeiro: Elsevier, 2003. 232 p.

SILVA, L. S.; SANTOS, A. C. F.; SILVA, J. D. S.; MONTES, A. Neural network application for attack detection in computer networks. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2004), 2004, Budapeste, Hungria. **Proceedings...** Piscataway, NJ, USA: IEEE, 2004. (INPE-11626-PRE/7007).

SILVA, L. S.; SANTOS, A. C. F.; SILVA, J. D. S.; MONTES, A. ANNIDA: Artificial Neural Network for Intrusion Detection Application - Aplicação da hamming net para detecção por assinatura. In: CONGRESSO BRASILEIRO DE REDES NEURAIAS (CBRN'2005), 7., 2005, Natal, RN, Brasil. **Anais...** [S.l.]: [s.n.], 2005a.

SILVA, L. S.; SANTOS, A. C. F.; SILVA, J. D. S.; MONTES, A. Estudo do uso da hamming net para detecção de intrusão. In: SIMPÓSIO DE SEGURANÇA EM INFORMÁTICA, 7., ITA, São José dos Campos, São Paulo, 2005. **Anais...** São José dos Campos: ITA, 2005b. Disponível em: <www.linorg.cirp.usp.br/SSI/SSI2005/Short/14241.pdf>. Acesso em: 4 jan 2006. (INPE-14109-PRE/9256).

SILVA, L.S.; MONTES, A.; SILVA, J.D. S Evolução dos trabalhos em detecção de anomalias na rede. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA, 5., 2005, INPE, São José dos Campos, SP. **Anais...** São José dos Campos: INPE, 2005c. Disponível em: <hermes2.dpi.inpe.br:1905/rep-/dpi.inpe.br/hermes2@1905/2005/10.04.01.42>.

SILVA, L. S.; SANTOS, A. C. F.; SILVA, J. D. S.; MONTES, A. Hamming net and LVQ neural networks for classification of computer network attacks: a comparative analysis. In: BRAZILIAN NEURAL NETWORKS SYMPOSIUM, 9., 2006, Ribeirão Preto, São Paulo. **Anais...** [S.l.]: IEEE Explore Digital Library, 2006a. p. 13. ISBN 0769526802
<http://doi.eeeecomputersociety.org/10.1109/SBRN.2006.21>

SILVA, L.S.; MONTES, A., SILVA, J.D. S; MANCILHA, T. D.; SANTOS, A. C. F. A framework for analysis of anomalies in the network traffic. In: WORKSHOP DOS CURSOS DE COMPUTAÇÃO APLICADA, 6., 2006, INPE, São José dos Campos, SP. **Anais...** São José dos Campos: INPE, 2006b. Disponível em: <eprint.sid.inpe.br/rep-sid.inpe.br/ePrint@80/2006/12.20.23.21> Acesso em: 13 dez 2006.

SILVA, L.S.; SANTOS, A. C. F.; MANCILHA, D. T.; SILVA, J.D.S.; MONTES, A. **Detecting attack signatures in the real network traffic with Annida**. Aceito pela revista Expert Systems with Application. Acesso em: 29 mar. 2007.

SILVA, L. S. Detecção de anomalias em redes utilizando diferentes atributos do tráfego de rede e análise gráfica, In: SIMPÓSIO SEGURANÇA EM INFORMÁTICA (SSI), 8., 2006, São José dos Campos, São Paulo. **Anais...** São José dos Campos: CTA/ITA, 2003.

SILVA, R. M.; MAIA, M. A. G. M. Redes neurais artificiais aplicadas à detecção de intrusos em redes TCP/IP. In: SIMPÓSIO SEGURANÇA EM INFORMÁTICA, 6., 2004, ITA, São José dos Campos, São Paulo. **Anais...** São Paulo: USP, 2004. Disponível em: <www.linorg.cirp.usp.br/SSI/SSI2004/Artigos/Art019_ssi04.pdf>. Acesso em: 17 dez 2004.

SIRAJ, A.; BRIDGES, S.; VAUGHN, M.; RAYFORD, B. Fuzzy cognitive maps for decision support in an intelligent intrusion detection system. In: ANNUAL CONFERENCE OF THE NORTH AMERICAN FUZZY INFORMATION PROCESSING SOCIETY (NAFIPS'2001), 20., 2001, Vancouver, Canada. **Proceedings...** [S.l.]: IEEE Xplore, 2001. v. 4, p. 2165-2170.

SOURCEFIRE vulnerability research team. **Sourcefire VRT Certified Rules**, Disponível em: <<http://www.snort.org/rules>>. Acesso em: 09 mar. 2007.

TAPIADOR, J.M. E.; TEODORO, P. G.; VERDEJO, J.E. D. Measuring normality in http traffic for anomaly-based intrusion detection. **Computer Networks**, v. 45, n. 2, p. 175-193, June 2004a.

TAPIADOR, J.M. E.; TEODORO, P. G.; VERDEJO, J.E. D. Anomaly detection methods in wired networks: a survey and taxonomy. **Computer Communications**, v. 27, n. 16, p. 1569-1584, October 2004b.

TAROUCO, L.; BERTHOLDO, L.M.; ANDREOLI, A. V. Compreendendo ataques denial of services. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES, 1., 2003, Porto Alegre. **Anais...** Porto Alegre, [s.n.], 2003, v.1, p.71-76.

TILLAPART, P.; THUMTHAWATWORN, T.; SANTIPRABHOB, P. Fuzzy intrusion detection system. In: WORLD MULTICONFERENCE ON SYSTEMICS, CYBERNETICS AND INFORMATICS (SCI 2002), 6., 2002, Orlando, Florida, USA. **Proceedings...** [S.l.]: [s.n.], 2002. p. 272-276. ISBN: 980-07-8150-1.

VIGNA, G.; KEMMERER, R.A. NetSTAT: a network-based intrusion detection system. **Journal of Computer Security**, v. 7, n. 1, p. 37-71, 1999.

XIN*, J. Q.; DICKERSON, J. E.; DICKERSON, J. A. Fuzzy feature extraction and visualization for intrusion detection. In: IEEE INTERNATIONAL CONFERENCE ON FUZZY SYSTEMS, 12., 2003, St. Louis, MO. **Proceedings...** [S.l.]: IEEE Xplore, 2003. v. 2, p. 1249-1254. Disponível em: <ieeexplore.ieee.org/iel5/8573/27148/01206610.pdf>

WITTEN, I. **Data mining: practical machine, learning tools and techniques with Java implementations**. San Diego, USA: Academic Press, 2000. 369 p. ISBN 1558605525.

WIKIPEDIA. **Teorema de Tales**, Disponível em: <http://pt.wikipedia.org/wiki/Teorema_de_Tales>. Acesso em: 12 jun. 2007.

YAMANISHI, K.; TAKEUCHI, J. ; WILLIAMS, G.; MILNE, P. On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms. In: ACM SIGKDD INTERNATIONAL CONFERENCE ON KNOWLEDGE DISCOVERY AND DATA MINING (SIGKDD'2000), 6., Boston, MA, 2000. **Proceedings...** New York, USA: ACM (Association for Computing Machinery), 2000. p. 320- 324.

ZANERO, S. Analyzing TCP traffic patterns using self organizing maps. In: SPECIAL SESSION ON PATTERN IN COMPUTER SECURITY, 2005, Cagliari, Italy. **Proceedings...** Heidelberg: Springer Berlin, 2005a. v. 3617, p. 83-90. ISBN: 978-3-540-28869-5.

ZANERO, S. Improving self organizing map performance for network intrusion detection. In: SIAM CONFERENCE ON DATA MINING (SDM'2005), 5., 2005, Newport Beach, CA.. **Proceedings...** [S.l.]: [s.n.], 2005b. Disponível em: <<http://citeseer.ist.psu.edu/zanero04improving.html>>. Acesso em: 24 abr 2006.

ZHANG, Z.; LI, J. HIDE: a hierarchical network intrusion detection system using statistical preprocessing and neural network classification. In: IEEE WORKSHOP ON INFORMATION ASSURANCE AND SECURITY, 6., 2001, United States Military Academy, West Point, New York. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2001. p. 175-178.

ZHANG, C. L.; JIANG, J.; KAMEL, M. Comparison of BPL and RBF network in intrusion detection system. **Lecture Notes In Computer Science**, v.2639, p. 466-470, 2004. ISSN 03029743.

ZACHARY, J.M.; MCEACHEN, J.C. Real-time representation of network traffic behavior for enhanced security. In: INTERNATIONAL CONFERENCE ON INFORMATION TECHNOLOGY AND APPLICATIONS (ICITA'2005), 3., 2005, Sydney, Australia. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2005. v.2, p. 214- 219. ISBN: 0-7695-2316-1.

BIBLIOGRAFIA COMPLEMENTAR

- AMINI, M.; RASOOL, J. Network-based intrusion detection using unsupervised adaptive resonance theory (ART). In: INTERNATIONAL CONFERENCE ON ENGINEERING OF INTELLIGENT SYSTEMS, 4., 2004, Madeira, Portugal. **Proceedings...** [S.l.]: [s.n.], 2004.
- ARI, I.; HONG, B.; MILLER, E. L.; BRANDT S. A.; LONG D. E. **Modeling, analysis and simulation of flash crowds on the internet**, Technical Report UCSC-CRL-03-15, Santa Cruz, CA: University of California, 2004.
- BARBARA, D.; WU, N.; JAJODIA, S. Detecting novel network intrusions using bayes estimators. In: SIAM CONFERENCE ON DATA MINING, 1., Chicago, IL, USA, 2001. **Proceedings...** [S.l.]: [s.n.], 2001.
- BASHAH, N.; SHANMUGAM, I. B.; AHMED, A.M. Hybrid intelligent intrusion detection system. **Transactions On Engineering Computing and Technology**, v. 6, pp. 291-294, June 2005.
- CAMPELLO, R. S; WEBER, R. F. **Sistemas de Detecção de Intrusão**, Instituto de Informática, UFRGS. Disponível em: <www.inf.ufrgs.br/~gseg/producao/minicurso-ids-sbrc-2001.pdf>. Acesso em: 10 ago. 2005.
- CHO, S-B. Incorporating soft computing techniques into a probabilistic intrusion detection system. **IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews**, v. 32, n. 2, p. 154-160, May 2002.
- DASGUPTA, D.; BRIAN, H. Mobile security agents for network traffic analysis. In: DARPA INFORMATION SURVIVABILITY CONFERENCE AND EXPOSITION (DISCEX II), 2., Anaheim, CA, USA, 2001. **Proceedings...** New Jersey, USA: IEEE Press, 2001. V. 2, p. 331-340.

- DASGUPTA, D.; GONZALEZ, F. An immunity-based technique to characterize intrusions in computer networks. **IEEE Transactions on Evolutionary Computation**, v. 6, n. 3, p 281-291, June 2002.
- DEBAR, H.; BECKER, M.; SIBONI, D. A neural network component for an intrusion detection system. In: SYMPOSIUM ON SECURITY AND PRIVACY, 1992, Oakland, CA, USA. **Proceedings...** New Jersey, USA : IEEE Press, 1992. p. 240-250.
- DHANJANI, N. **Hack Notes segurança no Linux e Unix referência rápida**. 1. ed. Rio de Janeiro: Elsevier, 2003. 264 p. ISBN 8535213414.
- DOUMAS, A.; MAVROUDAKIS, K.; GRITZALIS, D.; KATSIKAS, S. Design of a neural network for recognition and classification of computer viruses. **Computers & Security**, v. 14, n. 5, p. 435-448, 1995.
- DRAELOS, T.; DUGGAN, D.; COLLINS, M.; WUNSCH, D. Adaptive critic designs for host-based intrusion detection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2002), 2002, Honolulu, Hawaii, USA. **Proceedings...** New Jersey, USA: IEEE Press, 2002. V. 2, p. 1720-1725.
- ESKIN, E.; ARNOLD, A.; PRERAU, M. ; PORTNOY, L. ; STOLFO, S. A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. **Applications of Data Mining in Computer Security**, v. 6, p. 272, 2002. ISBN 9781402070549.
- GHOSH, A.; SCHWARTZBARD, A. A study in using neural networks for anomaly and misuse detection. In: USENIX SECURITY SYMPOSIUM, 8., 1999, Washington, DC, USA. **Proceedings...** [S.I.]: [s.n.], 1999.
- HAN, S-J.; CHO, S-B. Detecting intrusion with rule-based integration of multiple models. **Computers & Security**, v. 22, n. 7, p. 613-623, 2003.

HENDERSON, S.J AND BRODLIE, K. **VisEd Visualization Education**

Tool. Disponível em:

<<http://www.siggraph.org/education/materials/HyperVis/vised/VisTech/vtmain.html>>. Acesso em: 12 out. 2006.

HOFMANN, A.; SCHMITZ, C.; SICK, B. Rule extraction from neural networks for intrusion detection in computer networks. In: IEEE INTERNATIONAL CONFERENCE ON SYSTEMS, MAN AND CYBERNETICS, 2003, Washington, DC. **Proceedings...** Piscataway, New Jersey, USA : IEEE Press, 2003. V. 2, p 1259-1265.

HOFMANN, A.; SICK, B. Evolutionary optimization of radial basis function networks for intrusion detection. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2003), 2003, Portland, Oregon, USA. **Proceedings...** Piscataway, New Jersey, USA: IEEE, 2003. v. 1, p. 415-420.

HOFMEYR, S. A. ; FORREST, S.; SOMAYAJI, A. Intrusion detection using sequences of system calls. **Journal of Computer Security**, v. 6, p.151-180, 1998.

JAVITZ, H.S.; VALDES, A. **The NIDES statistical component: description and justification**, Technical Report A010, Menlo Park, California: SRI International, March 1994.

JUNG, J.; KRISHNAMURTHY, B.; RABINOVICH, M. Flash crowds and denial of service attacks: characterization and implications for CDNs and web sites. In: WORLD WIDE WEB CONFERENCE, 11., 2002, Honolulu, Hawaii, USA. **Proceedings...** New York, NY: ACM Press, 2002. p. 252-262.

KRUEGEL, C.;TOTH, T. Using decision trees to improve signature-based intrusion detection. **Lecture Notes In Computer Science**, v. 2820, p. 173-191, 2003.

KUN, Z.; MAN-WU, X., HONG, Z.; FENG-YU, L. Intrusion detection method (RHDID) based on relative hamming distance. **Chinese Journal of Computers**, v. 26, p. 65-70, 2003.

LAKHINA, A., CROVELLA M., CHRISTOPHE, D. Mining anomalies using traffic feature distributions. In: ACM SIGCOMM CONFERENCE ON APPLICATIONS, TECHNOLOGIES, ARCHITECTURES, AND PROTOCOLS FOR COMPUTER COMMUNICATION, Philadelphia, Pennsylvania, USA, 2005. Proceedings... New York, NY: ACM Press, 2005. p. 217-228.

LEE, S.C.; HEINBUCH, D.V. Building a true anomaly detector for intrusion detection. In: MILITARY COMMUNICATIONS CONFERENCE (MILCOM), 2000, Los Angeles, CA. **Proceedings...** New York: Institute of Electrical and Electronics Engineers, 2000. v. 2, p. 1171-1175.

LEE, W.; STOLFO, S.; CHAN, P. K.; ESKIN, E.; FAN, W.; MILLER, M.; SHLOMO HESHKOP, S.; ZHANG, J. Real-time data mining-based intrusion detection. In: DARPA INFORMATION SURVIVABILITY CONFERENCE AND EXPOSITION (DISCEX'01), 2., 2001, Anaheim, CA. **Proceedings...** New Jersey, USA: IEEE Press, 2001. p. 89-100.

LEE, W.; XIANG, D. Information-theoretic measures for anomaly detection. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 2001, Oakland, CA. **Proceedings...** CA: IEEE Computer Society Press, 2001. p. 130-143.

LEE, W.; ESKIN, E.; STOLFO, S. J. Modeling system calls for intrusion detection with dynamic window sizes. In: DARPA INFORMATION SURVIVABILITY CONFERENCE AND EXPOSITION (DISCEX II), 2., 2001, Anaheim, CA. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, v. 1, 2001. p. 165.

LEE, W.; STOLFO, S.; MOK, K. W. A data mining framework for adaptive intrusion detection. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1999, San Antonio, TX. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 1999.

LICHODZIJEWski, P.; ZINCIR-HEYWOOD, A.; HEYWOOD, M. I. Host-based intrusion detection using self-organizing maps. In: INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN'2002), 2002, Honolulu, Hawaii, USA. **Proceedings...** New Jersey, USA: IEEE Press, V. 2, 2002. p. 1714-1719.

MARIN, J.; RAGSDALE, D.; SURDU, J. A hybrid approach to the profile creation and intrusion detection. In: DARPA INFORMATION SURVIVABILITY CONFERENCE AND EXPOSITION (DISCEX'2001), 2001, Anaheim, California. **Proceedings...** New Jersey, USA: IEEE Press 2001. p. 69-73.

MCHUGH, J. 1998 Lincoln Lab intrusion detection evaluation (a critique). In: INTERNATIONAL WORKSHOP ON RECENT ADVANCES IN INTRUSION DETECTION, 3., 2000, Toulouse, France. **Proceedings...** [S.l.]: [s.n.], 2000.

MELO, S. **Exploração de vulnerabilidades em redes TCP/IP**. 1. ed. Rio de Janeiro: Alta Books, 2004. 236 p. ISBN 8576080559.

MIT Lincoln Laboratory. **DARPA Intrusion Detection Evaluation**. Disponível em: <<http://www.ll.mit.edu/IST/ideval>>. Acesso em: 15 jan. 2007.

NASCIMENTO, H. A. D.; FERREIRA, C. B. R. **Visualização de Informações - Uma Abordagem Prática**. Disponível em: <<http://www.inf.ufg.br/funcomp/infovis/topico7.html>>. Acesso em: 16 out. 2006.

RAMASWAMY, S.; RASTOGI, R. ; SHIM, K. Efficient algorithms for mining outliers from large data sets. **ACM SIGMOD Record**, v. 29, n. 2, p. 427-438, 2000. ISSN:0163-5808.

RANUM, M. J.; LANDFIELD, K. ; STOLARCHUK, M. ; SIENKIEWICZ, M.; LAMBETH, A.; WALL, E. NFR, Implementing a generalized tool for network monitoring. In: SYSTEMS ADMINISTRATION CONFERENCE, 11., 1997, San Diego, CA. **Proceedings...** San Diego: California,1997. p. 1-19.

RAO, X.; DONG, C.; YANG, S. Statistic learning and intrusion detection. **Lecture Notes in Computer Science**, v. 2639, p. 652-659, May 2003. ISBN 3540140409.

SANTOS, A C. F, **Estado da arte no uso de tecnologias de inteligência artificial na detecção de padrões de ataques em redes de computadores**. 2006. 46 p. Dissertação (Qualificação em Computação Aplicada) – Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2006.

SILVA, A R. A. **Um modelo representativo de assinaturas de ataque para sistemas detectores de intrusão**. UNESP, São José do Rio Preto, SP, Brasil, 2002, 53 p. Disponível em: <www.acmesecurity.org/hp_ng/files/testes_monografias/acme-monografia-AISF-2002-artur.pdf>. Acesso em: 23 maio 2005.

SMAHA, S. E.; GRANCE, T. ; TEAL, D. M. DIDS - motivation, architecture, and an early prototype. In: NATIONAL COMPUTER SECURITY CONFERENCE, 14., 1991, Washington, DC. **Proceedings...** Davis: Dept. of Computer Science, University of California, 1991, p. 167-176.

STANIFORD, S.; HOAGLAND, J.; MCALERNEY, J. Practical automated detection of stealthy portscans. **Journal of Computer Security**, v. 10, n. 1-2, p. 105-136, 2002.

STANIFORD-CHEN, S. ; CHEUNG, S. ; CRAWFORD, R. ; DILGER, M.; FRANK, J.; HOAGLAND,J.; LEVITT, K. ; WEE, C. ; YIP, R.; ZERKLE, D. GRIDS - a graph-based intrusion detection system for large networks. In: NATIONAL INFORMATION SYSTEMS SECURITY CONFERENCE, 19.,

1996, Baltimore, MD. **Proceedings...** New York: National Institute of Standards and Technology, v. 1, p. 361-370, 1996.

STEVENS, W.R **TCP/IP Illustrated (vol. 1):** the protocols, Indianapolis: Addison-Wesley Longman Publishing, 1993. 576 p. ISBN 0201633469.

TAVARES, D.M. **Avaliação de técnicas de captura para sistemas detectores de intrusão.** 2002. 98 p. Dissertação (Mestrado em Ciências da Computação e Matemática Computacional) – Universidade de São Paulo (ICMC/USP), São Carlos, 2002.

THANGAVEL, K.; SHEN, Q.; PETHALAKSHMI, A. Application of clustering for feature selection based on rough set theory approach, **AIML Journal**, v. 6, n. 1, January 2006.

TSAI, D. R.; TAI, W. P.; CHANG, C. F. A hybrid intelligent intrusion detection system to recognize novel attacks. In: ANNUAL INTERNATIONAL CARNAHAN CONFERENCE ON SECURITY TECHNOLOGY, 37., 2003, Taipei, Taiwan. **Proceedings...** IEEE Xplore: 2003. p.428- 434. ISBN: 0780378822.

WANG, X. T. The IDS model of intelligent design system **Computers & Structures**, v. 61, n. 3, p. 579-586, November 1996.

YE, N. A markov chain model of temporal behavior for anomaly detection. In: 2000 IEEE SYSTEMS, MAN, AND CYBERNETICS INFORMATION ASSURANCE AND SECURITY WORKSHOP, 2000. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 2000. p. 171-174.

YE, N.; CHEN, Q. An anomaly detection technique based on a chi-square statistic for detecting intrusions into information systems. **Quality and Reliability Engineering International**, v. 17, n. 2, p. 105 – 112, March 2001.

ZHAO, X. L.; SUN, J. Z. A parallel scheme for IDS. In: INTERNATIONAL CONFERENCE ON MACHINE LEARNING AND CYBERNETICS, 2., 2003, Xi'an, China. **Proceedings...** Piscataway: Institute of Electrical and Electronics Engineers Inc., 2003. p. 2379- 2383. ISBN 0780378652.

PUBLICAÇÕES TÉCNICO-CIENTÍFICAS EDITADAS PELO INPE

Teses e Dissertações (TDI)

Teses e Dissertações apresentadas nos Cursos de Pós-Graduação do INPE.

Manuais Técnicos (MAN)

São publicações de caráter técnico que incluem normas, procedimentos, instruções e orientações.

Notas Técnico-Científicas (NTC)

Incluem resultados preliminares de pesquisa, descrição de equipamentos, descrição e ou documentação de programa de computador, descrição de sistemas e experimentos, apresentação de testes, dados, atlas, e documentação de projetos de engenharia.

Relatórios de Pesquisa (RPQ)

Reportam resultados ou progressos de pesquisas tanto de natureza técnica quanto científica, cujo nível seja compatível com o de uma publicação em periódico nacional ou internacional.

Propostas e Relatórios de Projetos (PRP)

São propostas de projetos técnico-científicos e relatórios de acompanhamento de projetos, atividades e convênios.

Publicações Didáticas (PUD)

Incluem apostilas, notas de aula e manuais didáticos.

Publicações Seriadas

São os seriados técnico-científicos: boletins, periódicos, anuários e anais de eventos (simpósios e congressos). Constam destas publicações o Internacional Standard Serial Number (ISSN), que é um código único e definitivo para identificação de títulos de seriados.

Programas de Computador (PDC)

São a seqüência de instruções ou códigos, expressos em uma linguagem de programação compilada ou interpretada, a ser executada por um computador para alcançar um determinado objetivo. São aceitos tanto programas fonte quanto executáveis.

Pré-publicações (PRE)

Todos os artigos publicados em periódicos, anais e como capítulos de livros.