UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM
CIÊNCIA DA COMPUTAÇÃO

Igor Vinícius Mussoi de Lima

UMA ABORDAGEM SIMPLIFICADA DE DETECÇÃO DE INTRUSÃO BASEADA EM REDES NEURAIS ARTIFICIAIS

Dissertação submetida à Universidade Federal de Santa Catarina como parte dos requisitos para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. João Bosco Mangueira Sobral

Florianópolis, fevereiro de 2005

UMA ABORDAGEM SIMPLIFICADA DE DETECÇÃO DE INTRUSÃO BASEADA EM REDES NEURAIS ARTIFICIAIS

Igor Vinícius Mussoi de Lima

Esta Dissertação foi julgada adequada para a obtenção do título de Mestre em Ciência da Computação Área de Concentração Sistemas de Computação e aprovada em sua forma final pelo Programa de Pós-Graduação em Ciência da Computação

Prof. Dr. Raul Sidnei Wazlawick
Coordenador do Curso de Pós-Graduação em Ciência da Computação

Prof. Dr. João Bosco Mangueira Sobral
(Orientador)

Prof. Dr. Jovelino Falqueto

Prof. Dr. Leandro Nunes de Castro Silva

Banca Examinadora

Prof. Dra. Mirela Sechi Moretti Annoni Notare

A vida só pode ser compreendida olhando-se para trás; mas só pode ser vivida olhando-se para a frente.

Soren Kierkegaard

Dedico este trabalho em memória de meu pai Edenilson Soares de Lima.

AGRADECIMENTOS

Agradeço ao professor Dr. João Bosco Mangueira Sobral por suas orientações, seu apoio e sua confiança durante todo o processo de desenvolvimento deste trabalho.

Ao grande amigo e colega de estudos Renato Bobsin Machado, que colaborou de forma decisiva para a realização deste trabalho, desde a definição do tema até sua conclusão.

Aos professores Arthur Ronald de Vallauris Buchsbaum e Rosvelter Coelho da Costa os quais tiveram papel fundamental sobre minha participação no programa Pós-Graduação da UFSC.

Ao professor Dr. Jovelino Falqueto pelos diversos esclarecimentos e revisões realizadas.

Aos amigos Alexandre dos Santos Pacheco e Eder Nicolau Cardoso pelo apoio incondicional em todos os momentos.

Aos professores Dr. João Alberto Fabro e Dr. Adriano Mauro Cansian pelos conhecimentos e experiências compartilhadas, bem como a inestimável colaboração relacionada ao principal componente deste trabalho.

Aos colegas Edney Matias da Silva e Wandrey Alves Rangão pela ajuda e prestatividade em vários momentos de dificuldade.

Agradeço a minha mãe Alba de Lourdes Mussoi de Lima, pelo grande apoio e incentivo que sempre me deu em tudo de produtivo que fiz na vida.

A Luciane Jussara Bezerra Kusbick pela importante contribuição na correção e adequação da dissertação as normas metodológicas.

A UFSC que me aceitou em seu programa de Pós-Graduação e possibilitou meu aperfeiçoamento técnico e pessoal.

De forma geral agradeço a todos aqueles que tiveram qualquer participação com este projeto, pois por menor que esta tenha sido, certamente teve sua parcela de importância.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

IA ..... Inteligência Artificial
AM ..... Aprendizado de Máquina
RNA ..... Rede Neural Artificial
MLP ..... *Multi Layer Perceptron*
IDS ..... *Intrusion Detection System*
CIDF ..... *Common Intrusion Detection Framework*
IDWG ..... *Intrusion Detection Work Group*
NIDS ..... *Network Intrusion Detection System*
I-IDS ..... *Inteligent Intrusion Detection System*
LBC ..... Lista da Base de Conhecimento
BPF ..... *BSD Packet Filtering*

RESUMO

Este trabalho apresenta uma abordagem simplificada de detecção de intrusão, aplicando técnicas de inteligência artificial para a classificação de eventos em redes de computadores. A base desta abordagem é o monitoramento do fluxo de dados da rede, o qual é considerado subsídio para análises posteriores, onde aplicam-se métodos especiais como análise semântica e redes neurais artificiais. O modelo proposto para esta abordagem possui arquitetura baseada em rede, método de detecção que incorpora funcionalidades de abuso e anomalia e geração de respostas passivas aos eventos detectados. Os resultados gerados pelo processo de análise são valores numéricos, os quais classificam os eventos analisados em padrões considerados intrusivos ou normais. A eficiência desta classificação é modelada durante a fase de aquisição de conhecimento, denominada treinamento. A forma de representação e aquisição do conhecimento, bem como as características adaptativas obtidas pelo processo de generalização do modelo, caracterizam contribuições efetivas da pesquisa realizada, cujas comprovações são demonstradas pelos experimentos realizados sobre o protótipo desenvolvido.

Palavras Chaves: Segurança de Redes, Redes Neurais Artificiais, Detecção de Intrusão.

ABSTRACT

This work presents an simplified approach to intrusion detection, applying artificial intelligence techniques for the classification of events in a computer network. This approach is based on the monitoring of the network traffic, which is considered as data for subsequent analysis, where methods involving artificial neural networks and semantic analysis are applied. The proposed model has a network-based architecture and its intrusion detection method incorporates the functions of misuse and anomaly as well the generation of passive responses to the detected events. The results generated by the analysis process are numeric values which classifies the analysed events either malicious or normal standard.The efficiency of this classification is shaped during the phase of acquisition of knowledge, called training. The form of representation and acquisition of knowledge, as well as the adaptive characteristics obtained through the model generalization process, characterize the effective contributions from this research, which are confirmed by the experiments carried out on the prototype developed.

Key Words: Network Security, Artificial Neural Networks, Intrusion Detection.

1 INTRODUÇÃO

Esta dissertação trata da classificação dos eventos ocorridos em redes de computadores, procurando utilizar uma abordagem simplificada de análise por meio de técnicas de inteligência artificial. Neste capítulo, o enfoque do trabalho será posicionado dentro do contexto científico em que se insere, serão apresentadas as motivações que levaram ao seu desenvolvimento, descritos os principais objetivos e, por último, apresentada a organização do texto.

1.1 Introdução

Devido aos grandes avanços tecnológicos e ao crescimento exponencial da Internet em todo o mundo, as soluções informatizadas abrangem todas as classes sociais e todos os ramos profissionais de atuação, proporcionando aumento de organização, produtividade e agilidade tanto em ambientes corporativos como para usuários domésticos.

Esse desenvolvimento acarretou quantidades cada vez maiores de atividades realizadas por intermédio de redes de computadores, tornando estes recursos soluções indispensáveis a todos os níveis da sociedade.

Em conjunto com essa popularidade e crescimento tecnológico surgiram dificuldades para manter a segurança da infra-estrutura envolvida, pois com o aumento na quantidade de serviços disponíveis, aumenta também a possibilidade de existirem vulnerabilidades.

Essa situação ocasionou o surgimento de diversas ações que comprometem a segurança desses ambientes, que foram impulsionadas pela popularização de ferramentas e técnicas cada vez mais sofisticadas de intrusão.

Os esforços realizados no sentido de proporcionar segurança em ambientes computacionais são motivados pelo fato de existirem riscos que podem comprometer o

objetivo principal dessa segurança. Esses riscos são avaliados de acordo com variáveis que determinam as chances do mesmo ocorrer e com os custos envolvidos para tratá-lo, e então podem ser prevenidos, corrigidos, transferidos ou ignorados (ABNT, 2002).

As técnicas de defesa vêm sendo aprimoradas, porém existem diversas limitações que as impedem de estarem efetivamente preparadas para o surgimento de formas inéditas de ataques , sendo necessárias soluções inovadoras para tratar os níveis de ameaças atuais e futuras.

Este cenário é a principal motivação deste trabalho que consiste em propor, modelar, implementar, realizar experimentos e mensurar resultados de uma solução para detecção de intrusão utilizando técnicas de inteligência artificial.

1.2 Proposta

Este trabalho propõe um modelo simplificado para classificação de eventos intrusivos em redes de computadores, o qual foi inspirado pela linha de pesquisa iniciada por meio do trabalho *Sistema Adaptativo de Detecção de Intrusos em Redes de Computadores* (CANSIAN, 1997).

De forma sucinta, o uso de técnicas de inteligência artificial, especificamente redes neurais artificiais, tem por objetivo utilizar suas características de reconhecimento de padrões e generalização para realizar a classificação dos eventos ocorridos em redes de computadores, classificando-os em normais ou intrusivos, permitindo a geração de respostas aos eventos considerados críticos.

Pretende-se analisar a viabilidade de aplicação desta abordagem, bem como detalhar suas vantagens ou desvantagens em relação a métodos convencionais de detecção de intrusão.

No decorrer deste trabalho são apresentados os conceitos e tecnologias relacionadas, as quais são necessárias para viabilizar a compreensão do modelo proposto.

1.3 Apresentação do Trabalho

Este trabalho é segmentado de forma a apresentar a conceituação necessária, seguida dos capítulos que discutem o modelo proposto e suas características.

O segundo capítulo apresenta o problema da segurança em redes de computa-

dores, descrevendo o cenário atual e detalhando os métodos de detecção de intrusão e as alternativas existentes.

No terceiro capítulo são apresentados os conceitos relativos à inteligência artificial, focando as características de aprendizado de máquina e redes neurais artificiais.

O quarto capítulo é destinado a detalhar o modelo proposto para a detecção neural de eventos intrusivos, bem como descrever os requisitos envolvidos na elaboração da proposta e as alternativas de implementação existentes.

O quinto capítulo detalha a implementação de um protótipo que segue as especificações do modelo proposto no capítulo anterior, incorporando opções de configuração e funcionalidades estatísticas as quais possibilitariam mensurar o desempenho e vantagens da solução proposta.

No sexto capítulo são apresentados os experimentos realizados sobre o protótipo do modelo, dispondo os resultados obtidos em situações adversas de configuração e treinamento.

No sétimo capítulo são tratadas as conclusões e discutidas as vantagens e dificuldades do modelo, descrevendo alternativas de trabalhos futuros os quais dariam continuidade à pesquisa e complementariam o modelo.

2 SEGURANÇA DE REDES E DETECÇÃO DE INTRUSÃO

Este capítulo aborda o tema de segurança aplicada a redes de computadores, procurando descrever a variedade e as características das ameaças existentes. Será apresentada a importância de mecanismos de defesa em ambientes computacionais, bem como descrita as funcionalidades, classificações e padronizações de sistemas de detecção de intrusão.

2.1 Introdução

A dependência da sociedade pela infra-estrutura computacional acarreta a preocupação com a segurança envolvida. Dessa forma, várias técnicas vêm sendo desenvolvidas e aprimoradas visando reduzir os riscos com vazamentos, erros, fraudes, sabotagens, uso indevido, roubo de informações e diversos outros problemas.

Apesar de vários esforços no sentido de prover a segurança em ambientes computacionais, o número, a variedade e complexidade dos incidentes relacionados à segurança tem crescido significativamente. Conforme a Figura 1, o número de incidentes reportados ao CERT (*Computer Emergency Response Team*) reafirma tal crescimento (CERT/CC, 2004).

Os principais objetivos da segurança computacional são manter a autenticidade, confidencialidade, integridade e disponibilidade das informações.

A *confidencialidade* é definida como sendo a garantia de que a informação esteja disponível somente para aqueles que tem autorização para obtê-la.

Integridade é a garantia de que a informação permanecerá inalterada mesmo sob situações críticas, como acidentes ou tentativas de manipulações hostis.

Disponibilidade consiste na proteção dos recursos e serviços prestados pelo sis-

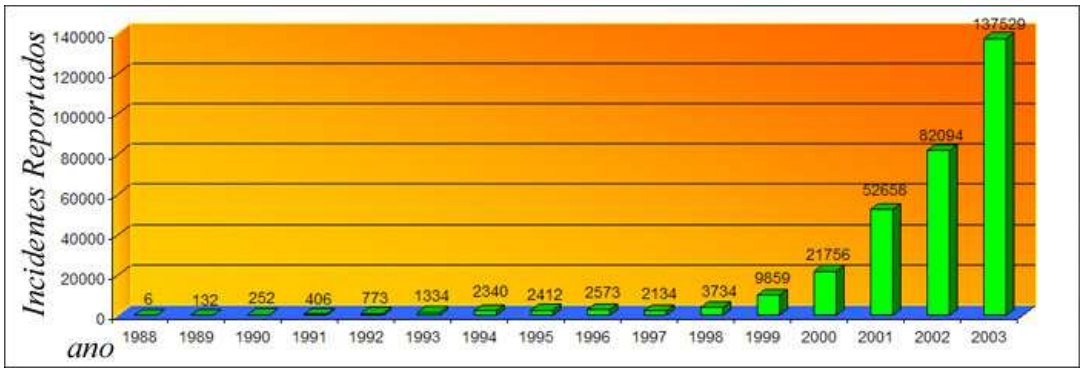


Figura 1: Estatística de incidentes reportados ao CERT. Fonte: (CERT/CC, 2004)

tema de forma que eles não sejam degradados ou se tornem indisponíveis, garantindo assim que a informação estará sempre acessível e pronta para o uso.

A *autenticidade* está associada à correta identificação de usuários ou computadores, visando proteger o sistema contra a personificação de intrusos e geralmente é garantida através de um mecanismo de senhas ou de assinatura digital.

Sendo assim, qualquer atividade que possa comprometer qualquer um desses objetivos é considerada uma violação às políticas de segurança.

2.2 Ameaças à Segurança

Os sistemas computacionais estão constantemente sujeitos a vários tipos de ameaças, sejam elas acidentais, maliciosas, internas ou externas, que podem desencadear intrusões explorando vulnerabilidades do sistema.

As explorações dessas vulnerabilidades são motivadas por objetivos específicos que podem variar desde simples atos de vandalismo até sofisticadas técnicas de espionagem industrial.

Em um relatório técnico gerado pelo *Sandia National Laboratories* (HOWARD; LONGSTAFF, 1998) é discutida uma taxonomia que se baseia em ações para classificar ameaças à segurança. Esta classificação trata sobre informações que estão em trânsito que podem ser visualizadas na Figura 2 e que possuem as seguintes características:

a) **Interrupção**: As informações em trânsito são interrompidas, impossibilitando que as mesmas cheguem até seu destino e prejudicando a questão da disponibilidade dos recursos.

b) **Interceptação**: As informações são interceptadas durante a transmissão, comprometendo a confidencialidade da mensagem.

c) **Modificação**: As informações são interceptadas e alteradas durante a transmissão, afetando não só a confidencialidade como a integridade da mensagem.

d) **Fabricação**: Trata-se da inserção de informações em determinada comunicação, comprometendo a autenticidade das informações recebidas.

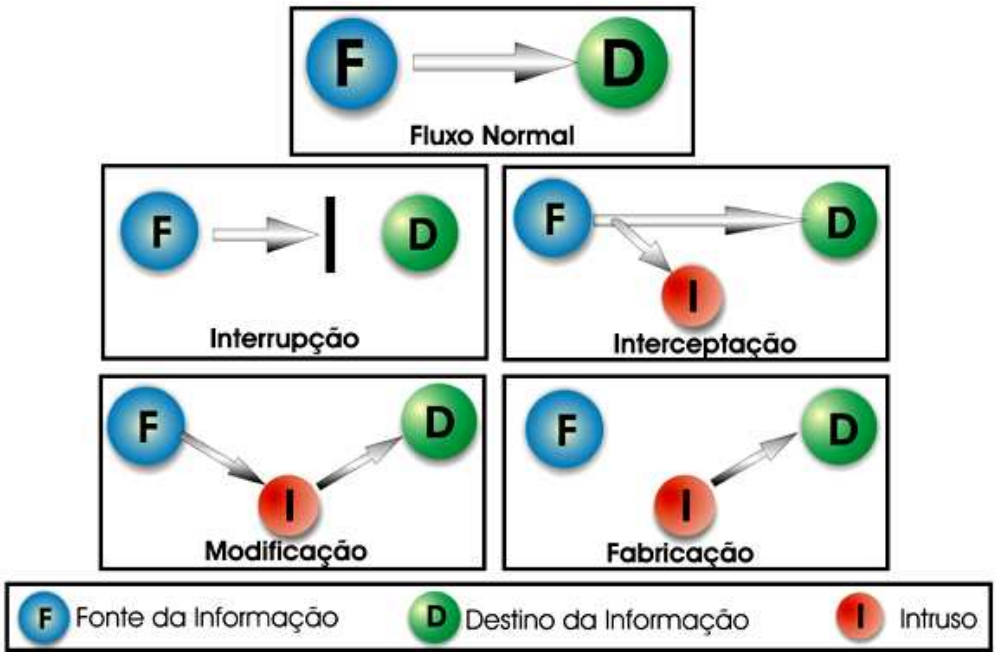


Figura 2: Taxonomia baseada em ações. Fonte: (STALLINGS, 2003)

O mesmo relatório descreve que toda atividade em um computador seja ela intrusiva ou não é realizada através de eventos, que são ações realizadas sobre determinados alvos com a finalidade de obter resultados esperados. Sugere ainda que em incidentes relacionados à segurança existem cinco fases inter-relacionadas, que são ilustradas na Figura 3, demonstrando que o atacante utiliza uma ferramenta para explorar uma vulnerabilidade e então realizar uma ação sobre determinado alvo com a finalidade de obter resultados não autorizados.

A preocupação com segurança em redes e sistemas de computadores ocasionou o surgimento de várias técnicas voltadas à prevenção de ataques, mas prevenir todas as possíveis quebras de segurança em um ambiente computacional é uma tarefa impossível de ser realizada, pois novas vulnerabilidades são descobertas constantemente. No entanto, é possível identificar tentativas ou violações ocorridas e então

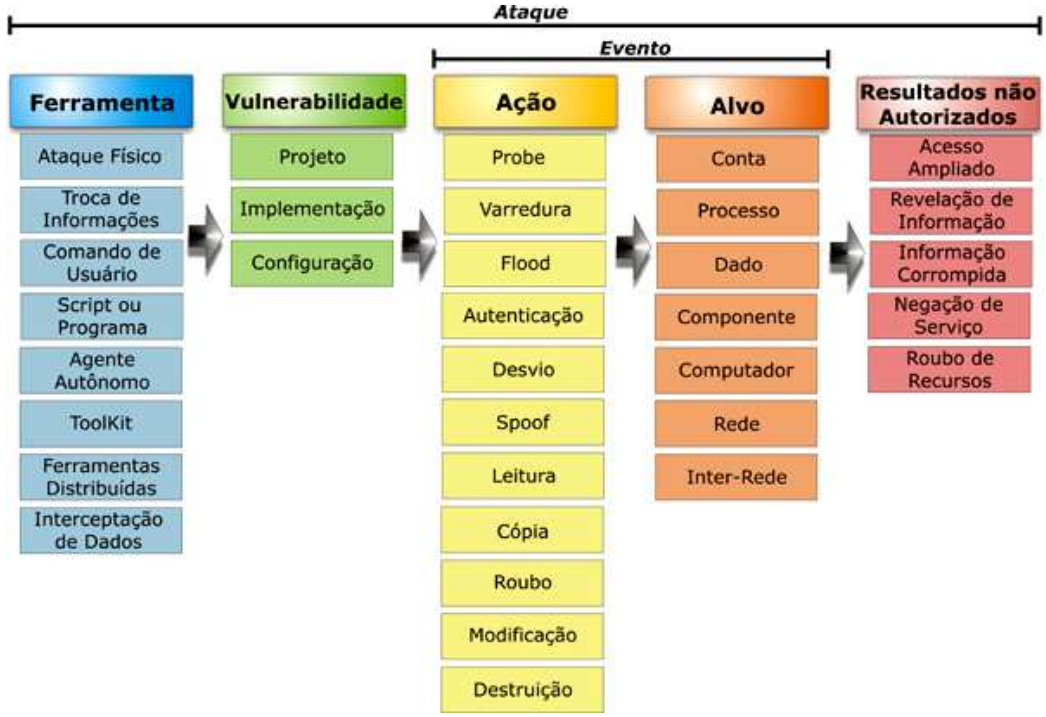


Figura 3: Evolução de um ataque. Fonte: (HOWARD; LONGSTAFF, 1998)

gerar respostas que possam minimizar os danos sofridos. Nesse cenário é que se destaca a detecção de intrusão, que é o processo de identificar e relatar e possivelmente reagir a qualquer atividade maliciosa agindo em computadores e recursos da rede.

Essa detecção é realizada através de conjuntos de software e hardware que cooperam de forma a executar análises sobre todas as atividades realizadas no sistema monitorado, identificando os eventos considerados como sendo ataques e posteriormente relatando que uma atividade maliciosa aconteceu, está acontecendo ou irá acontecer.

Na maioria das abordagens de detecção de intrusão utilizadas atualmente a identificação dos eventos maliciosos é realizada através da comparação das atividades correntes com ações esperadas de um intruso.

Os sistemas de detecção de intrusão possuem componentes que desempenham funções específicas como sensores, analisadores de eventos e unidades de respostas, que juntos provêem a capacidade de detectar, analisar e responder a cada evento de acordo com a gravidade do evento ocorrido.

2.3 Etapas de um Ataque

Ao observarmos em detalhes um sistema atacado com sucesso, notamos a presença de atividades sucessivas que caracterizam etapas do ataque.

Geralmente podemos constatar que o ataque foi precedido por investigações de reconhecimento, que na maioria dos casos são varreduras em busca de informações sobre recursos e vulnerabilidades existentes no host alvo e que examinam relações de confiança entre usuários, recursos e sistemas envolvidos (NORTHCUTT et al., 2001).

Essas investigações devem ser detectadas e encaradas com seriedade, conscientizando o sistema de defesa sobre os riscos envolvidos, pois sua identificação possibilita a execução de contra-medidas para evitar que tal exploração efetive uma intrusão.

Existem inúmeras ferramentas que aplicam vários tipos de técnicas de reconhecimento e varreduras, entre elas destacam-se SATAN (*Security Administrator Tool for Analyzing Networks*), SAINT (*Security Administrator's Integrated Network Tool*) e ISS (*Internet Security Scanner*).

Um exemplo de técnica de reconhecimento é o envio de pacotes de dados manipulados e mal formados (*crafted packets*) ao host alvo como estímulos de respostas, com isso é possível identificar o sistema operacional existente, analisando as respostas recebidas, visto que estas não são padronizadas entre as plataformas. Essa informação é importante para o atacante, pois a partir dela pode-se definir as vulnerabilidades a serem exploradas e suas respectivas ferramentas (MIIKA, 2000).

Após o reconhecimento, o intruso passa a comprometer a segurança do host alvo e a propagar o ataque explorando as possíveis falhas encontradas, através de técnicas e ferramentas específicas para atingir os objetivos do ataque. Podem também ser iniciados mecanismos de propagação de ataque que podem auto-iniciar novos ciclos de intrusões, como é o caso de códigos maliciosos como o *Code Red* e do *Ninda* que se propagaram a nível global em menos de 18 horas (STAFF, 2002).

Depois de ter realizado a exploração, o atacante passa a se preocupar em eliminar os indícios de sua presença ou atividade no sistema, bem como estabelecer formas de garantir acesso futuro ao host e coordenar a utilização das possíveis ferramentas intrusivas que foram implantadas no sistema. Para isso, manipula logs e registros de atividades, implanta *back-doors* ou ainda instala conjuntos de softwares denominados *rootkit's* que podem substituir aplicativos importantes do sistema operacional como também podem omitir processos, conexões, arquivos e logs manipulados pelo atacante.

2.4 Tipos de Ataques

Um ataque é uma ação maliciosa que viola as políticas de segurança e que compromete a integridade ou a disponibilidade dos recursos em um sistema, e tem por princípio básico a exploração das vulnerabilidades encontradas nas investigações realizadas, e que podem estar tanto em sistemas como nos protocolos existentes, podendo caracterizar efetivamente uma invasão ou acarretar indisponibilidade dos serviços providos pelo sistema alvo (BARBOSA; MORAES, 2000).

Segundo Stephen E. Smaha (SMAHA, 1989) os ataques podem ser classificados de acordo com as formas que agem sobre os sistemas, como:

a) **Negação de Serviço (*Denial of Service*)**: que é caracterizada pelo uso anormal do sistema de forma a exaurir os recursos disponíveis;

b) **Uso Malicioso (*Malicious Use*)**: que trata da execução de atividade maliciosa fazendo uso de privilégios concedidos e normalmente está relacionado a usuários legítimos do sistema;

c) **Tentativas de Invasão (*Attempted Breack-In*)**: que são executadas através de ações atípicas e que infringem as regras de segurança do sistema;

d) **Tentativas de Personificação (*Masquerade Attack*)**: sendo tentativas de invasão executadas de forma a se passarem por atividades de usuários válidos ou por máquinas confiáveis do domínio;

e) **Invasão no Controle de Segurança (*Penetration*)**: geralmente são detectados por monitoração de padrões específicos das atividades ocorridas e reportadas pelo sistema;

f) **Vazamento (*Leakage*)**: ataques que são detectados pelo uso anormal de recursos de entrada e saída do sistema e variações no padrão de utilização dos mesmos.

Estes comportamentos são realizados no intuito de explorar falhas de implementação nos protocolos e serviços, caracterizando ações que visam obter informações, paralisar serviços, subverter mecanismos, ou qualquer outro objetivo com características intrusivas.

2.5 Detecção de Intrusão

Existem vários mecanismos de segurança utilizados atualmente, que vão desde rotinas de backup, até complexos filtros, regras de controle de tráfego e detecção de intrusão, que são realizados por arquiteturas compostas por requisitos de hardware e software.

Os IDS (*Intrusion Detection System*) são ferramentas complementares no processo de gestão de segurança da informação, pois apesar dos esforços empregados para automatizar a tarefa de detecção e respectivas respostas, é imprescindível a interação humana na análise dos filtros, alertas e relatórios gerados, objetivando medidas apropriadas dadas as circunstâncias envolvidas.

2.5.1 Classificação dos Sistemas de Detecção de Intrusão

As técnicas utilizadas para procurar evidências de comportamento suspeito nos registros de utilização dos sistemas podem variar de acordo com a maneira que são implantadas, freqüência de operação, pelo tipo dos dados que analisam ou pelas análises que utilizam sobre esses dados. Estes métodos possuem determinadas aplicações, limitações e peculiaridades e podem ser classificadas de várias formas conforme ilustrado na Figura 4.

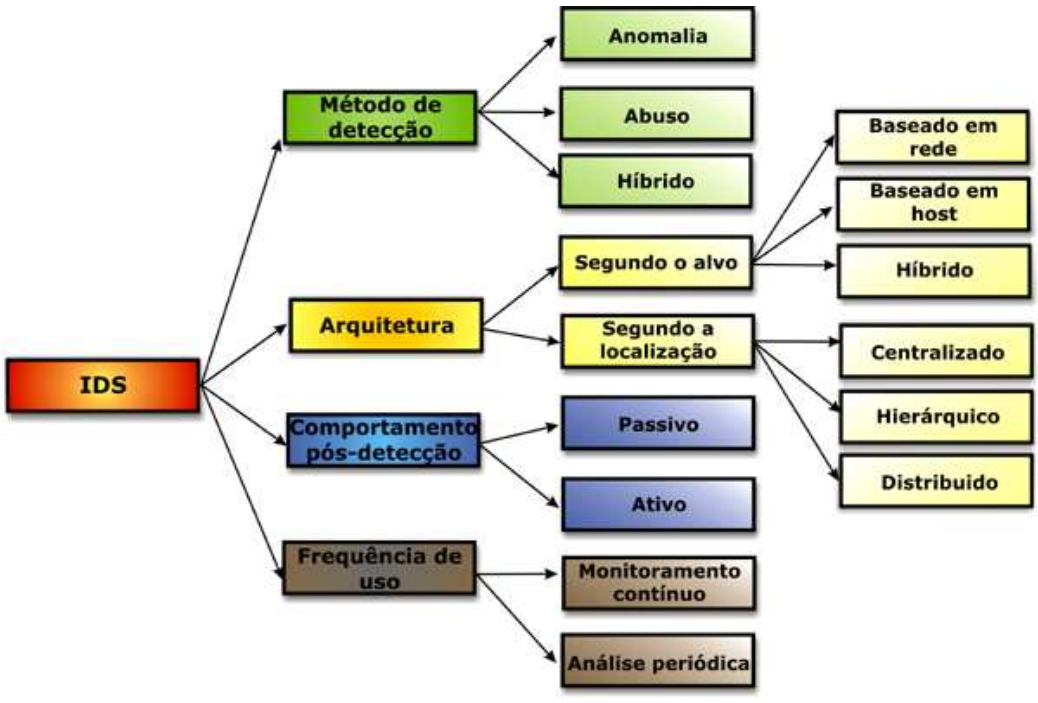


Figura 4: Classificação de IDS. Fonte: (CAMPELLO; WEBER, 2001)

Esta classificação permite identificar critérios ou características que são consideradas ao se projetar um sistema de detecção de intrusão. Em função disso diferentes ferramentas aplicam distintos mecanismos para obtenção, análise e tratamento dos dados, até mesmo utilizando arquiteturas híbridas para otimizar o modelo final.

2.5.2 Método de Detecção

O método de detecção empregado é um componente importante na construção de sistemas de detecção de intrusão, pois define a implementação do principal processo em soluções do gênero. Esses métodos definem formas de detecção por anomalia aplicando métodos estatísticos sobre a utilização do sistema, detecção por abuso desenvolvendo o monitoramento baseado em padrões de ataques conhecidos, e detecção híbrida aplicando ambas as abordagens.

2.5.2.1 Método de Detecção por Anomalia

Esta técnica considera que todo comportamento intrusivo é necessariamente anômalo, o sistema reage a todo comportamento que não se enquadre nos comportamento ditos como normais.

Em um tipo de análise denominada estatística estes comportamentos normais são previamente relacionados, baseado na observação do conjunto de atividades que caracterizam operações normais de um sistema, gerando estatísticas de uso dos recursos (CPU, unidades de armazenamento, memória, e outros periféricos) e das atividades dos usuários (tentativas de login, aplicativos utilizados, e outras ações).

Estas estatísticas podem ser constantemente atualizadas refletindo o estado atual do sistema (CROSBIE; SPAFFORD, 1995), o desvio de comportamento pode ser observado comparando o padrão de comportamento atual do sistema com as estatísticas geradas. Caso existam divergências abruptas nos parâmetros de comparação, é considerado que este é um comportamento anômalo e que, portanto pode vir a ser intrusivo.

Sendo assim, um atacante sabendo que está sob monitoramento de um sistema com estas características pode subverter o mecanismo de detecção alterando gradativamente seu comportamento de forma que o sistema os considere normais, e em dado momento efetivar o ataque sem que o sistema perceba um comportamento anômalo, ou ainda executando um conjunto de operações que individualmente não representam nenhuma ameaça, porém se caracterizando um ataque quando consideradas em

conjunto.

A detecção por anomalia leva em conta que qualquer comportamento anômalo é considerado como intrusivo, no entanto algumas atividades anômalas podem não ser intrusivas. Esta situação gera quatro estados de detecção (KUMAR, 1995):

a) **Intrusivo e anômalo**: a atividade é intrusiva e é apontada como tal por ser também anômala; são conhecidos como os verdadeiros positivos;

b) **Não intrusivo e não anômalo**: a atividade não é anômala e não é apontada como intrusiva; são denominados como verdadeiros negativos;

c) **Intrusivo mas não anômalo**: a atividade é intrusiva mas, como não é anômala não é reportada como tal, gerando uma falha em sua detecção; são consideradas como falso negativos;

d) **Não intrusivo mas anômalo**: atividade não é intrusiva, porém como é anômala, o sistema entende que se trata de uma atividade intrusiva, reportando de forma incorreta tal fato; estes são denominados falso positivos.

Altos índices de detecções sendo falso-positivos podem comprometer a eficiência do sistema de detecção de intrusão, pois uma grande quantidade de alertas pode ser gerada reportando como intrusivas as atividades normais dos usuários. Para contornar este problema é preciso redefinir os parâmetros que apontam tal comportamento como anômalo, porém ajustando-os de forma que não ocorram detecções falsas negativas.

Segundo (AXELSSON, 2000) os detectores por anomalia tendem a ser mais caros computacionalmente, pois muitos parâmetros e estatísticas precisam ser ajustados com freqüência, dependendo do tipo de atividade do sistema.

2.5.2.2 Método de Detecção por Abuso

A abordagem de detecção de intrusão por abuso baseia-se na observação de eventos que se assemelham a comportamentos intrusivos já conhecidos, comumente chamados de assinaturas de intrusão. Estas assinaturas são relacionadas e todos os eventos do sistema são comparados com as mesmas, a fim de identificar um padrão de comportamento que se encaixe nas especificações da assinatura.

Podem existir várias formas de se executar um mesmo ataque, sendo necessário que os aspectos, condições, posicionamento e inter-relações entre os eventos que

levam a uma intrusão sejam transcritos para sua assinatura de intrusão, a fim de detectar um mesmo ataque iniciado a partir de padrões diferentes.

Alguns ataques são desenvolvidos através de evoluções realizadas a partir de ataques já conhecidos. Sendo assim o uso de assinaturas na detecção por abuso colabora na localização de tentativas de quebra de segurança, de forma que a confirmação parcial da ocorrência de uma assinatura pode indicar uma tentativa de intrusão.

A principal limitação deste tipo de sistema é que a detecção dos eventos baseia-se em ataques já conhecidos, sendo, portanto ineficiente na detecção de comportamentos intrusivos que ainda não foram descobertos e publicamente divulgados. A eficiência deste método depende da freqüente atualização das assinaturas de intrusão do sistema.

2.5.2.3 Método de Detecção Híbrido

Abordagens de detecção por anomalia ou por abuso podem ser adequadas para casos distintos de formas de ataques, por isso algumas abordagens tratam sobre um método de detecção híbrido que incorpora os dois métodos, compondo uma solução de detecção mais eficiente, levando-se em conta a grande variedade de ataques existentes (BERNARDES, 1999).

2.5.3 Tipo de IDS

Várias abordagens e técnicas podem ser utilizadas e combinadas para prover a detecção dos eventos maliciosos ocorridos em determinado ambiente. A forma como estas técnicas são aplicadas na construção de sistemas de detecção de intrusão impacta diretamente sobre sua eficiência e desempenho.

As diferenças vão desde a lógica adotada para analisar os eventos até a abrangência do sistema dentro da infra-estrutura em que está inserido.

Podemos classificar os sistemas de detecção de intrusão em duas categorias principais, sendo IDS baseado em host e IDS baseado em rede. Freqüentemente os sistemas podem ser híbridos incorporando características das duas abordagens (CROSBIE; SPAFFORD, 1995).

2.5.3.1 IDS Baseado em Rede - NIDS

Esta categoria de IDS denominada NIDS - *Network Intrusion Detection System*, possui um mecanismo de detecção que analisa o tráfego da rede, e que implementa a detecção utilizando dois componentes principais, sendo eles os sensores e estação de gerenciamento.

Os sensores são componentes que são alocados estratégicamente em determinados segmentos de rede, passando a monitorar não somente a máquina onde estão instalados, mas todo o tráfego no seguimento em questão, pois a interface de rede é configurada para trabalhar em modo promíscuo capturando não apenas os pacotes de dados destinados à interface de rede da estação onde está instalado, mas toda e qualquer informação que trafegue pela rede.

Os dados capturados podem ser previamente selecionados para que posteriormente sejam analisados de acordo com o mecanismo de detecção adotado, que irá categorizar os eventos e determinar se o mesmo é ou não intrusivo, reportando os eventos de interesse para a estação de gerenciamento que gerencia os eventos e realiza o tratamento adequado para cada situação como, por exemplo, um alerta ou um e-mail para os responsáveis para que sejam tomadas as devidas providências. A Figura 5 ilustra a disposição dos componentes em uma estrutura de IDS baseado em rede.

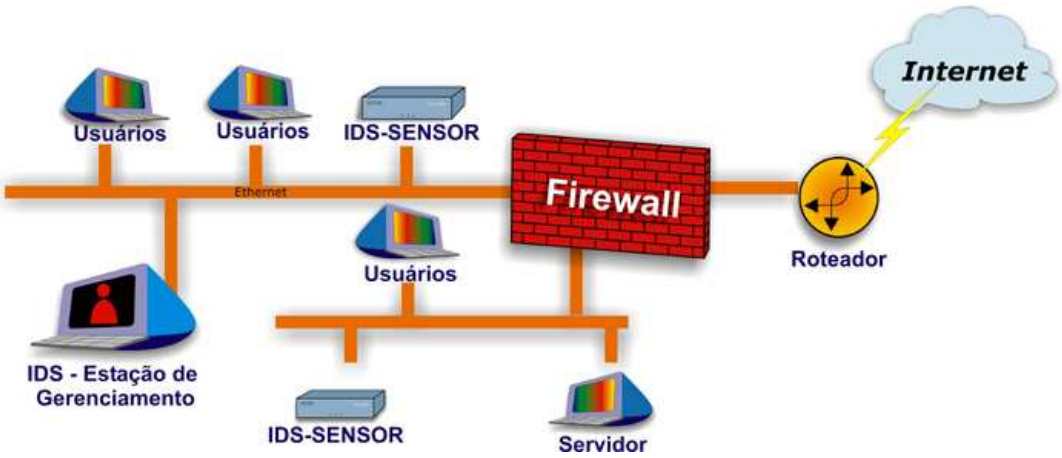


Figura 5: Estrutura de um IDS baseado em Rede. Fonte:(STANIFORD-CHEN, 1998)

O posicionamento dos sensores na rede pode impactar diretamente sobre a ocorrência dos eventos e os resultados esperados. Em uma situação onde os sensores estejam posicionados antes de um *firewall* estes estariam expostos a todas as atividades maliciosas da rede. Por outro lado, caso os sensores estejam posicionados na parte interna da rede o número e a severidade dos eventos a serem analisados

é consideravelmente menor, uma vez que o *firewall* realizou determinada filtragem bloqueando grande parte das tentativas de ataque (MIIKA, 2000).

Este tipo de implementação permite a descoberta de falhas na configuração dos mecanismos de segurança, e baseado em eventos reportados pelo sistema poderiam ser realizados ajustes nas regras de filtragem do *firewall*.

Existem algumas razões relacionadas ao ambiente e a infra-estrutura de rede que inviabilizam a utilização de um IDS baseado em rede. Em redes de alta velocidade o custo computacional exigido para realizar a captura e a análise do tráfego seria muito alto podendo trazer impactos sobre o tempo de resposta do sistema.

A utilização de *switches* como equipamentos de interconexão em redes de computadores limita a abrangência do sensor, devido à ausência de *broadcast*, pois o tráfego é comutado entre os elementos da rede. Portanto, mesmo em modo promíscuo o sensor não conseguiria capturar todos os pacotes de dados do segmento.

Existe também uma dificuldade relacionada à utilização de criptografia em aplicações de rede, pois em seções criptografadas o IDS estaria incapacitado de realizar análises, já que não seria possível reconhecer assinaturas ou padrões de ataque baseado em dados cifrados.

2.5.3.2 IDS Baseado em Host

Os sistemas de detecção de intrusão baseados em host possuem mecanismos e procedimentos de análise que verificam sinais de intrusão exclusivamente nas máquinas onde estão instalados, geralmente tomando como base para esta análise recursos locais, como registro de logs do sistema operacional e os registros de aplicações existentes, além de indícios de atividades não usuais, como tentativas de login sem sucesso, tentativas de acesso a arquivos, alterações de privilégios do sistema, entre outros.

Esta categoria de IDS possui algumas dificuldades relacionadas à captura e análise freqüente em um sistema específico, que pode acarretar problemas de degradação do desempenho do host, podendo ainda ser atacado e ter seu mecanismo de controle de logs comprometido, afetando a confiabilidade do mecanismo de detecção (MIIKA, 2000).

Além da análise dos registros de atividades do sistema, podem ser realizadas verificações de integridade de arquivos, que checam se determinados arquivos do sistema foram alterados desde a última verificação. Este processo é realizado gerando

um resumo de cada arquivo através de funções de *hash*. Estes resumos são armazenados e servirão como base em comparações futuras com novos resumos, determinando se houve ou não alteração nos arquivos em questão.

Independente do mecanismo utilizado, alguns tipos de ataques podem passar despercebidos pela maioria das ferramentas disponíveis, principalmente quando são utilizadas de maneira independente. Em função disso, é interessante possuir diversidade de mecanismos de detecção integrados no IDS, de forma a aproveitar as melhores características de cada técnica e tipo de análise.

2.5.4 Estrutura de IDS

Existe uma grande variedade de ferramentas de detecção de intrusão, que utilizam as mais variadas abordagens, formas de licenciamento e mecanismos utilizados para classificar os eventos.

Muitos esforços estão sendo realizados na busca pela interoperabilidade e padronização entre essas ferramentas, visando facilitar a integração de tecnologias complementares aumentando as chances de uma detecção bem sucedida. Vários padrões estão sendo propostos e aprimorados, entre eles destaca-se o CIDF (*Common Intrusion Detection Framework*) que define um conjunto de componentes funcionais que interagem entre si, formando um modelo para sistemas de detecção de intrusão utilizando a CISL (*Common Intrusion Specification Language*) como linguagem de especificação de eventos e comunicação entre os componentes.

Entre estes componentes estão geradores de eventos (*Caixas E*), mecanismos de análise (*Caixas A*), mecanismos de armazenamento (*Caixas D*), e contra medidas (*Caixas C*) (PTACEK; NEWSHAM, 1998), os quais podem ser observados na Figura 6.

As *caixas E* ou geradores de eventos são os sensores que monitoram o host ou a rede em busca de atividades anômalas gerando registros de ocorrências, disparando alertas e reportando relatórios relativos à atividade ocorrida no sistema.

Caixas A possuem mecanismos de análise que atuam sobre os registros de ocorrências reportados pelos sensores aplicando critérios de identificação e classificação dos eventos, podendo determinar uma linha de ação a ser executada mediante a análise realizada.

As *caixas D* definem procedimentos e formas de armazenamento seguro das informações relacionadas aos eventos reportados pelas *Caixas E* e *Caixas A*, bem

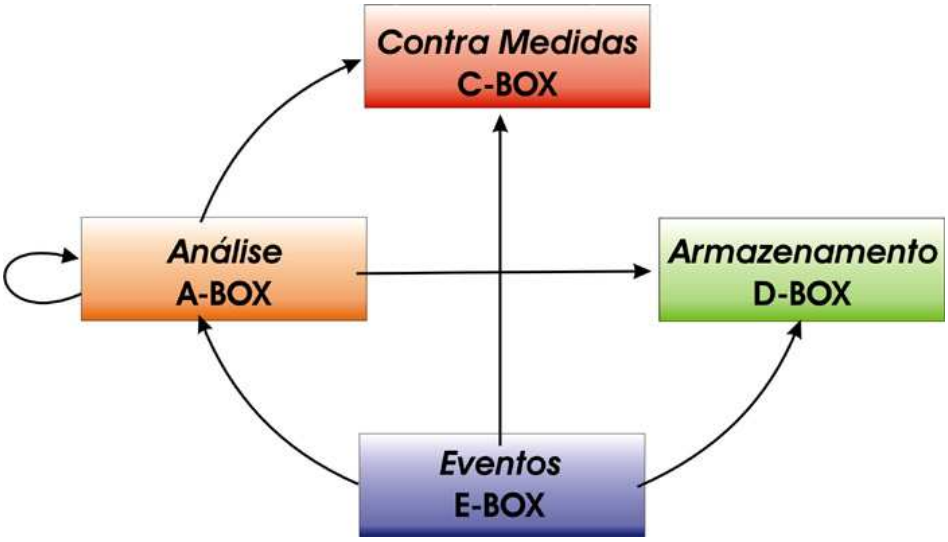


Figura 6: Relação de componentes no padrão CIDF. Fonte: (BARBOSA; MORAES, 2000)

como meios para torná-las disponíveis conforme a necessidade.

Depois de confirmado que determinado evento é considerado intrusivo, são tomadas ações para anular ou minimizar os seus efeitos. Estas ações são denominadas contra-medidas e são responsabilidade das *Caixas C* no modelo CIDF.

As respostas realizadas determinam o tipo de reatividade de cada sistema, podendo ser passivas quando providenciam emissão de alertas ou geração de relatórios informando sobre a ocorrência do evento, ou ativas quando implementam mecanismos de respostas aos ataques como recursos para desativar conexões TCP, modificar tabelas de roteamento ou regras de *firewall*, prevenindo dessa forma que o ataque efetivamente ocorra ou que este tome maiores proporções.

Entre os componentes do padrão CIDF destacam-se os geradores e analisadores de eventos, pois estes constam em todos as soluções de IDS tanto ativas quanto passivas. Segundo o grupo IDWG (*Intrusion Detection Working Group*) os principais componentes de um IDS são Sensor, Analisador e Gerente, onde o sensor provê subsídios para que o analisador possa reportar ao gerente o resultado de sua análise. Este gerente possui funções como configurar o sensor, analisar a configuração, avaliar e mensurar os eventos notificados (WOOD, 2003).

Estes componentes podem se relacionar de diversas formas na estrutura de um IDS, podendo ser combinados em um único módulo ou ainda segmentá-los. Uma estrutura pode possuir múltiplas instâncias de cada componente de acordo com a necessidade do ambiente. O diagrama da Figura 7 ilustra o relacionamento entre

estes componentes em determinado IDS.

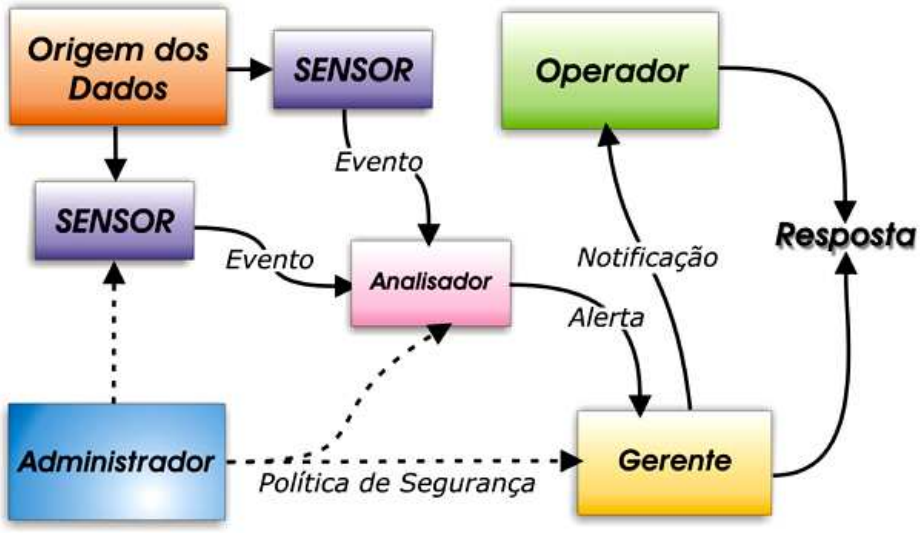


Figura 7: Relação de componentes no padrão IDWG. Fonte: (WOOD, 2003)

Existem outras propostas que tratam o tema com uma organização diferente, mas em todas as abordagens estão presentes os coletores de dados e algum mecanismo de análise.

3 REDES NEURAIS ARTIFICIAIS

Este capítulo apresenta uma conceituação básica sobre redes neurais artificiais, descrevendo algumas características estruturais relevantes e alternativas de aplicação, enfatizando técnicas de reconhecimento de padrões aplicadas a detecção de intrusão, as quais são o foco principal deste trabalho.

3.1 Introdução

O conceito de redes neurais artificiais (RNA's) envolve técnicas computacionais desenvolvidas através de modelos matemáticos baseados na constituição estrutural do cérebro. Esta inspiração originou-se pelo fato do cérebro possuir capacidades de processamento e organização poderosas e principalmente pelo fato de ser responsável pelo comportamento inteligente do indivíduo, com isso supõe-se que reproduzindo suas características, pode-se extrair resultados inteligentes.

O processamento de informações sensoriais e a capacidade de interação com ambientes pouco definidos, realizada pelos seres humanos e outros animais, motiva os pesquisadores a desenvolver sistemas artificiais capazes de desempenhar tarefas semelhantes, visando incorporar a sistemas computacionais capacidades de processamento de informações incompletas ou imprecisas e generalização (CASTRO, 1998).

As RNAs são aplicadas nas mais diversas áreas, pois suas vantagens constituem poderosas ferramentas para diversas aplicações. O poder computacional das redes neurais é extraído de sua estrutura paralelamente distribuída e de sua habilidade de aprender e generalizar.

3.2 Inspiração Biológica

O ser humano é dotado de complexos circuitos neurais cerebrais amplamente interconectados, estes circuitos são formados por células nervosas denominadas neurônios. Embora os neurônios possam apresentar diferentes formas e tamanhos, eles possuem quatro regiões especializadas servindo para funções específicas. São ilustrados na Figura 8 os dentritos, corpo celular, axônio e terminações do axônio (MACHADO, 2003).

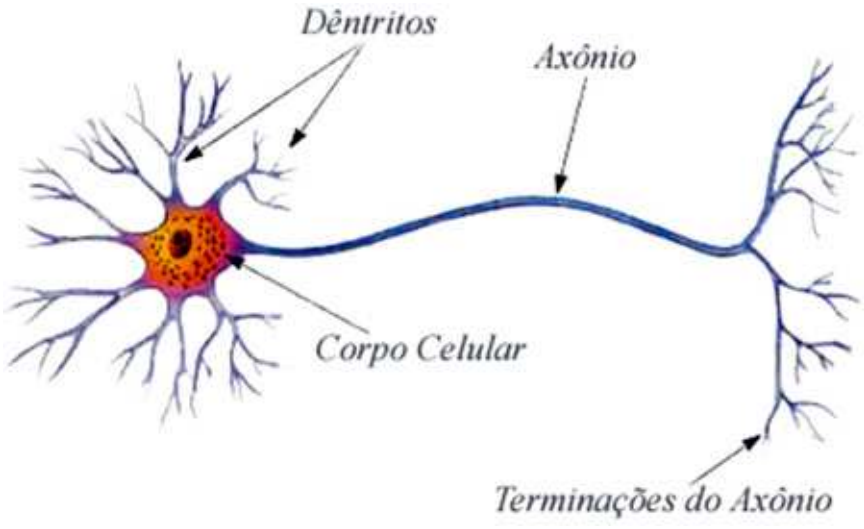


Figura 8: Neurônio Biológico. Fonte: (MACHADO, 2003)

Os dendritos formam vários conjuntos de fibras irregulares ao redor do corpo celular que funcionam como dispositivos de entrada recebendo sinais de outros neurônios. O corpo celular contém o núcleo e organelas críticas para manter a vitalidade do neurônio. O axônio corresponde a um filamento longo o qual conduz informação codificada em forma de potenciais de ação até pequenas ramificações denominadas terminações do axônio que funcionam como dispositivos de saída (MARTIN, 1998).

A transmissão de informação de um neurônio para outro acontece em locais de contato denominados sinapses, que geralmente estão em terminações do axônio de uma célula com os dendritos de outra. Os neurônios que enviam informações são denominados pré-sinápticos e aqueles que as recebem pós-sinápticos.

A transmissão dos sinais através das sinapses pode tanto excitar como inibir os neurônios pós-sinápticos, permitindo ou interrompendo respectivamente o fluxo desse sinal entre os neurônios envolvidos, assemelhando-se aos sinais elétricos digitais dos microcomputadores, representados pelo número 1 onde existe a presença de sinal e 0 significa sua ausência (LENT, 2001).

Cada neurônio do sistema nervoso é constantemente bombardeado por entradas sinápticas a partir de outros neurônios. Cada neurônio pode receber de 1.000 a 10.000 contatos sinápticos em seu corpo e dentritos podendo se projetar sobre milhares de neurônios alvos, criando assim uma enorme estrutura paralela de processamento que compõe o circuito neural (MACHADO, 2003) (KANDEL et al., 1995)

Os neurônios são de cinco a seis vezes mais lentos que as portas lógicas em silício, entretanto o cérebro compensa a taxa de operação relativamente lenta do neurônio, através do grande número de neurônios com conexões maciças entre si. (HAYKIN, 2001) *apud* (SHEPHERD; KOCH, 1990). As diversas vantagens do cérebro perante os computadores digitais motivaram o estudo de modelos artificiais inspirados no sistema neural biológico, os quais realizam comportamento coletivo em uma população muito grande de elementos computacionais simples (neurônios) que trocam sinais de cooperação ou competição, trabalhando em paralelo e de forma distribuída, sendo fortemente dependentes de suas conexões.

3.3 Fundamentos Teóricos

O processamento inteligente de uma rede neural é realizado através da interligação de células computacionais denominadas neurônios que são organizadas de forma complexa, não-linear e paralela (HAYKIN, 2001).

Vários esforços foram realizados no sentido de mimetizar o comportamento do cérebro humano e representar de forma artificial os neurônios biológicos, porém essas representações geralmente assumem diversas simplificações.

Um neurônio é a principal unidade de processamento de informações para a operação de uma rede neural artificial.

Pode-se modelar um neurônio artificial de acordo com a Figura 9, onde um neurônio k recebe sinais de entrada x_j (x_1,x_2, ..., x_m), que entram pela sinapse j onde são multiplicados pelos pesos w_{kj}. O resultado deste processo passa por um *somador* que soma os sinais de entrada ponderados pelos pesos das respectivas sinapses juntamente com um *bias* externo, representado por b_k. Este *bias* tem o efeito de aumentar ou diminuir o resultado antes de repassá-lo para uma função de ativação que irá transformar este resultado em um sinal de saída geralmente em um intervalo unitário fechado de [0,1] ou ainda [-1,1] o qual é transmitido para outros neurônios. Em termos matemáticos, um neurônio k pode ser descrito com o seguinte par de equações (FAUSETT, 1994):

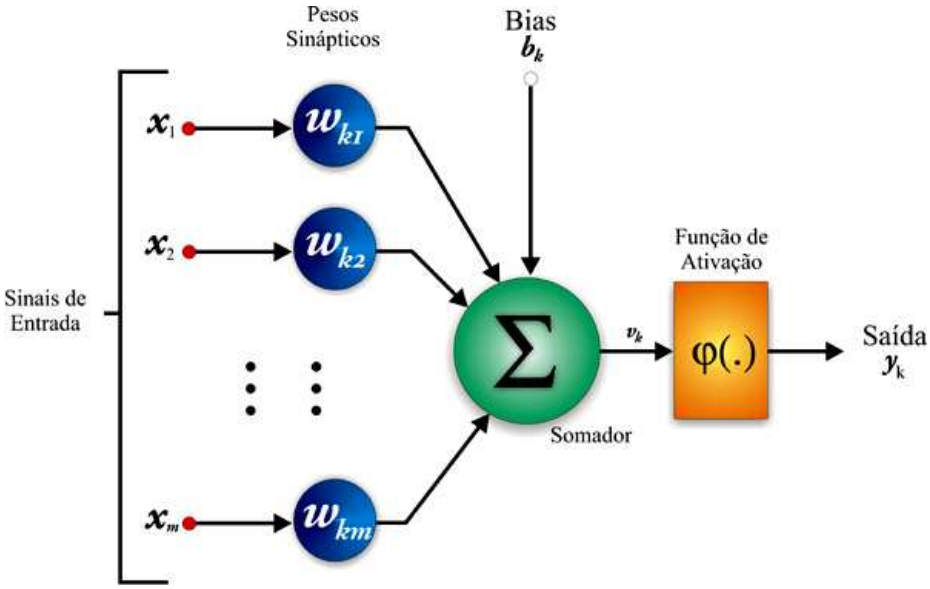


Figura 9: Modelo Simplificado de um Neurônio Artificial. Fonte: (HAYKIN, 2001)

$$v_k = \sum_{j=1}^{m} w_{kj} x_j$$

$$y_k = \varphi(v_k + b_k)$$

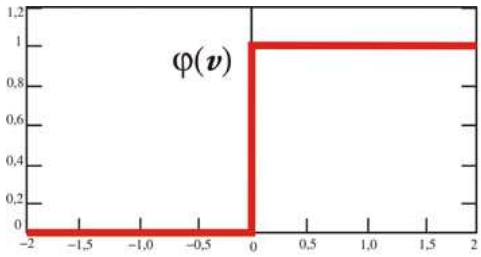
Os pesos sinápticos possuem importantes funções em um modelo neural. A influência desses pesos sobre a saída do neurônio pode ser de dois tipos onde pesos positivos tendem a incrementar o nível de ativação de um neurônio, e neste caso a conexão é chamada de excitatória. Ao contrário dessa situação um peso negativo tende a reduzir o sinal de saída, sendo chamadas de conexões inibitórias. O propósito dos pesos em uma rede neural é ponderar a influência dos sinais de entrada no neurônio pós-sináptico e armazenar o conhecimento (DALTON, 2000).

A função de ativação que está representada por $\varphi(.)$ na Figura 9, é muito importante no modelo neural, pois define qual será à saída de um neurônio de acordo com sua entrada(HAYKIN, 2001).

Vários tipos de função de ativação são propostas, entre elas:

a) **Função Limiar** que determina que a saída de um neurônio assume o valor 1 se o resultado do somador daquele neurônio dado por v for não negativo, e caso contrário o valor assumido é zero. O modelo de neurônio que utiliza essa função

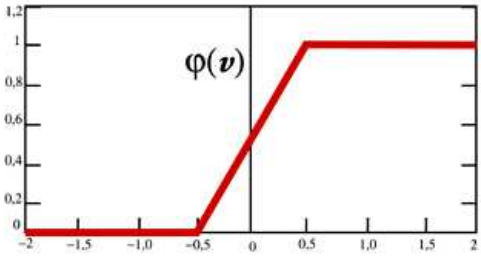
de ativação é conhecido como modelo McCulloch-Pitts, em reconhecimento ao trabalho pioneiro dos ciêntistas McCulloch e Pitts em 1943.



$$\varphi(v)=\begin{cases} 1 \text{ se } v \geq 0 \\ 0 \text{ se } v < 0 \end{cases}$$

Figura 10: Função de ativação Limiar. Fonte: (HAYKIN, 2001)

b) **Função Linear por Partes** possui duas situações particulares, sendo que seu resultado é uma função linear se a região de operação é mantida fora da região de saturação (ou seja se v estiver no intervalo entre -0,5 até 0,5 por exemplo). Caso a região de saturação seja atingida o comportamento se iguala a função limiar (com v estando no intervalo de -8 até -0,5 ou 0,5 até +8).



$$\varphi(v)=\begin{cases} 1 \text{ se } v \geq 0,5 \\ v \text{ se } 0,5 > v > -0,5 \\ 0 \text{ se } v \leq -0,5 \end{cases}$$

Figura 11: Função de ativação Linear. Fonte: (HAYKIN, 2001)

c) **Função Sigmóide** é uma função de ativação cujo gráfico tem forma de 'S' e que devido a sua adequada ponderação de resultados entre o comportamento linear e não-linear proporciona importantes implicações nos procedimentos de aprendizagem da rede, sendo a função de ativação mais comum na construção de redes neurais artificiais (DALTON, 2000). Um exemplo de função sigmóide é a *função logística* que pode ser descrita pela seguinte equação:

$$\varphi(v) = \frac{1}{1 + exp(-av)}$$

Onde a é o parâmetro de inclinação da função logística, quando este parâmetro tende a infinito esta aproxima-se da função limiar. Enquanto a função limiar assume valores zero ou um, a função logística assume valores contínuos que variam de zero a um, conforme ilustrado na Figura 12.

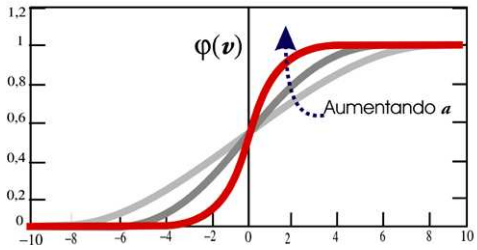


Figura 12: Função de ativação logística sigmóide. Fonte: (HAYKIN, 2001)

Um outro tipo de função sigmóide é a *função tangente hiperbólica* que possui características similares a função logística, porém seus valores contínuos variam de -1 a 1, sendo comumente aplicada na construção de redes neurais artificais onde o limiar de saída deve variar entre estes valores. Esta função pode ser definida por:

$$\varphi(v) = tanh(v)$$

O fato de se permitir que uma função de ativação do tipo sigmóide assuma valores negativos traz benefícios analíticos e vantagens durante a fase de aprendizado (HAYKIN, 2001). O modelo proposto neste trabalho faz uso desta função de ativação em sua implementação.

3.3.1 Processos de Aprendizagem

A principal característica das redes neurais artificiais herdadas de sua inspiração natural é a capacidade de adquirir e generalizar conhecimento a partir do ambiente em que está inserida. Esta característica é obtida por processos de aprendizagem que são realizados ajustando os pesos sinápticos de forma iterativa e direta através de técnicas denominadas treinamento.

Existem diversas alternativas para incorporar conhecimento à redes neurais artificiais. Entre elas destacam-se as técnicas de aprendizado por reforço, onde o treinamento consiste em reforçar os bons resultados gerados pela rede e penalizar os maus. Sendo assim se uma saída incorreta é gerada, os pesos das conexões envolvidas são ajustados de forma a evitar que isso ocorra, e em caso da saída estar correta, nenhuma providência é tomada.

Os ajustes dos pesos sinápticos durante o treinamento são realizados através de algoritmos de treinamento, proporcionando que o conhecimento seja adquirido a partir de exemplos. O aprendizado possibilita que a RNA incorpore a experiência

aprendida e passe a aplicá-la em suas próximas análises.

O processo de aprendizado é realizado através de regras de aprendizagem, que operam sobre uma rede neural de forma a lhe incorporar conhecimento.

Abaixo são apresentados de forma sucinta dois paradigmas de aprendizado comumente adotados:

a) **Aprendizagem supervisionada**: Neste caso a rede neural é treinada com o auxílio de um supervisor que possui conhecimento sobre o ambiente e o representa através de um conjunto de treinamento com entradas e as respectivas saídas desejadas. O supervisor repassa seu conhecimento e avalia o desempenho obtido pela rede de forma a corrigí-lo em caso de erro. Após o treinamento ser concluído o supervisor terá transferido seu conhecimento para a rede neural, podendo ser dispensado, deixando a rede lidar com as novas situações por si mesma.

b) **Aprendizagem não supervisionada**: Neste caso não existe o papel do supervisor durante o treinamento, que é realizado agrupando as entradas da rede se baseando em seus próprios critérios estatísticos. Este tipo de aprendizagem envolve processos de competição e colaboração entre os neurônios da rede.

Um algoritmo de aprendizado é constituído por um conjunto de regras para ajustar os pesos da rede neural. Basicamente, o modo como os ajustes dos pesos sinápticos são formulados é o que diferencia estes algoritmos.

Várias regras de aprendizagem são utilizadas atualmente e podem ser aplicadas no treinamento de redes neurais artificias, entre elas pode-se destacar:

O processo de *aprendizagem por correção de erro*, que é aplicável para treinamento em redes neurais com aprendizado supervisionado, é realizado determinando o sinal de erro entre a resposta gerada pela rede e a resposta desejada e então realizar ajustes nos pesos sinápticos de forma a minimizar este erro.

A Figura 13 ilustra o processo de aprendizagem por correção de erro. Considerando a saída y_k gerada por um neurônio k e a saída desejada d_k, é determinado um sinal de erro $e_k = d_k - y_k$. Este sinal será utilizado na determinação do novo valor dos pesos sinápticos deste neurônio, objetivando que o sinal de saída y_k se torne mais próximo de d_k. Pode-se para isso aplicar a regra delta definida como: $\Delta w_{kj} = \eta e_k x_j$, onde o η é um número real positivo que determina a taxa de aprendizado, e x_j compõe o vetor de estímulo. Sendo assim o valor do peso w_{kj} deverá

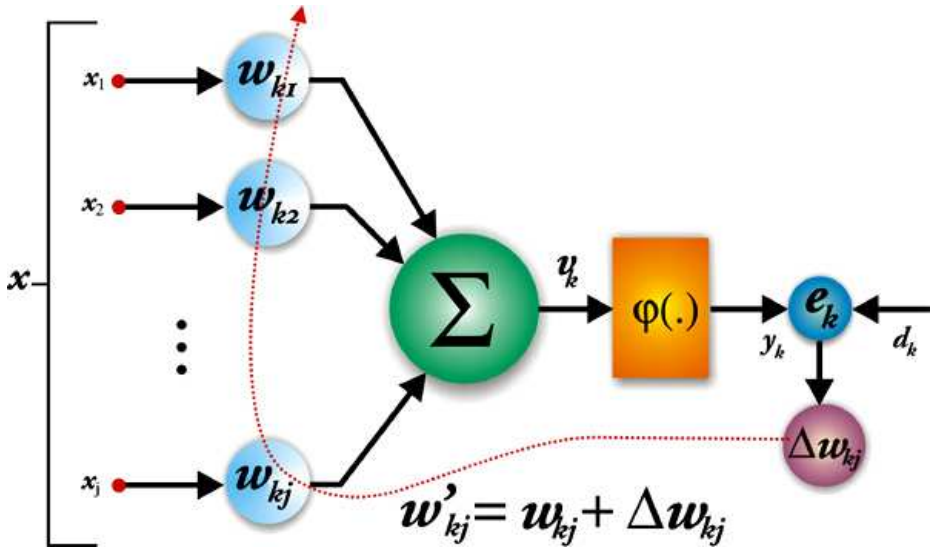


Figura 13: Processo de Aprendizagem por Correção de Erro.

ser ajustado produzindo w'_{kj}, recalculado através da correção do erro gerado pela função : $w'_{kj}=w_{kj}+\Delta w_{kj}$ (HAYKIN, 2001).

No *aprendizado baseado em memória* as situações apresentadas à rede são corretamente classificadas e armazenadas em uma memória que servirá como base nas próximas classificações, buscando-se pela similaridade dos itens armazenados em sua memória.

No *aprendizado competitivo* os neurônios de determinada rede neural competem entre si através de um processo de inibição mútua, de forma que esta competição determine apenas um neurônio ativo que será à saída da RNA. Este processo de aprendizado é não supervisionado e envolve três elementos básicos para seu funcionamento, sendo:

- Um conjunto de neurônios iguais entre si, porém diferentes quanto aos pesos de suas conexões sinápticas que são distribuídos randomicamente, ocasionando um comportamento diferente entre os neurônios diante de determinado conjunto de entradas.
- Um limite estipulado para a soma dos pesos das sinapses de determinado neurônio, de forma a regular a força de ativação.
- Um mecanismo que permita que os neurônios entrem em competição pelo direito de permanecerem excitados.

No aprendizado competitivo, entradas possuindo alguma semelhança, possuem

propensão a excitar o mesmo neurônio na saída (BARRETO, 1999).

O conhecimento da rede neural artificial se concentra nos pesos definidos para as conexões sinápticas da rede, formando uma representação compacta e distribuída desse conhecimento e proporcionando capacidades de generalização e adaptabilidade à rede neural. Porém, esta organização não baseada em regras impossibilita às redes neurais de explicar de forma abrangente o processo computacional pelo qual tomou a decisão exposta por suas saídas (HAYKIN, 2001).

3.3.2 Topologias de Redes Neurais

A arquitetura da rede neural está diretamente ligada ao algoritmo de aprendizado a ser aplicado. A seguir serão abordadas as principais arquiteturas destinadas ao projeto de redes neurais artificiais.

3.3.2.1 Redes Alimentadas Adiante

Nas redes neurais em camadas os neurônios da rede são organizados em camadas as quais são estruturadas de acordo com o propósito da rede.

As redes neurais alimentadas adiante também denominadas redes diretas ou *feedfoward* são aquelas que não possuem ciclos de realimentação entre os neurônios, o processo sináptico ocorre diretamente da camada de entrada em direção a camada de saída.

Em uma rede neural alimentada adiante com camada única ilustrada na Figura 14.a, existe uma camada de entrada de nós fontes que se projeta sobre uma camada de saída, e nunca o oposto. A nomenclatura camada única dada a este tipo de rede se refere à camada de saída da rede, pois a camada de entrada não é considerada por não realizar nenhum processamento.

Na Figura 14.b é ilustrada uma rede alimentada adiante com múltiplas camadas, estas camadas adicionais são denominadas camadas ocultas, e têm a função de intermediar de forma útil e não-linear o processamento entre a entrada e a saída da rede, proporcionando maior conectividade entre os neurônios e ampliando o poder de processamento da rede. Os sinais de saída da primeira camada servem como entrada para a segunda e assim por diante entre as camadas até a saída da rede(HAYKIN, 2001).

Uma rede neural é considerada totalmente conectada quando cada neurônio de

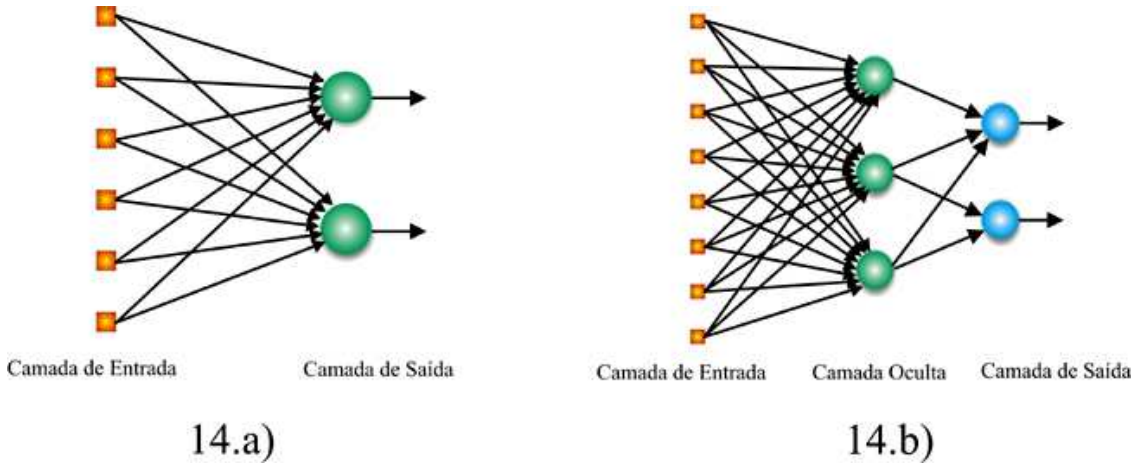


Figura 14: a)rede neural alimentada adiante com camada única; b) rede alimentada adiante com múltiplas camadas parcialmente conectada.

determinada camada está conectado diretamente a todos os outros nós da camada subseqüente. Caso esta conexão não seja completa entre os neurônios, como a Figura 14.b a rede é considerada parcialmente conectada.

3.3.2.2 Redes Neurais Recorrentes

Ao contrário de uma rede alimentada adiante, uma rede recorrente ou *feedback* como também é conhecida deve ter pelo menos um processo de realimentação entre seus neurônios.

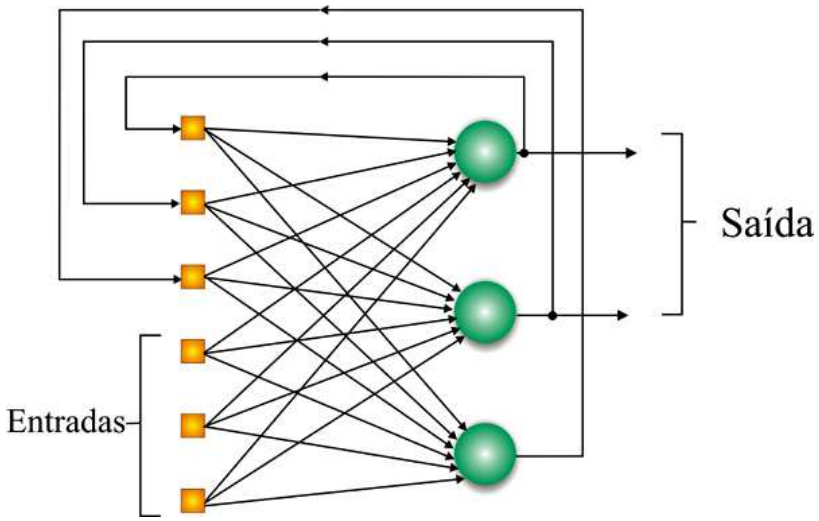


Figura 15: Rede Neural Recorrente. Fonte:(HAYKIN, 2001)

Essa realimentação pode ser realizada sobre neurônios de camadas anteriores ou da mesma camada, e até mesmo sua saída realimentar sua entrada novamente.

Neste último caso caracterizando uma situação de auto-realimentação.

A presença de laços de realimentação tem grande impacto tanto na capacidade de aprendizado da rede como também em seu desempenho (HAYKIN, 2001). Este tipo de rede pode exibir um comportamento dinâmico ao longo do tempo a partir de um único estímulo inicial (RAUBER, 1998).

3.3.3 Perceptron

Em 1958, Frank Rosenblatt descreveu em seu livro *"Principles of Neurodynamics"* o modelo dos perceptrons, onde neurônios eram organizados exclusivamente em camada de entrada e saída e os pesos das conexões eram adaptados a fim de se atingir melhor performance da rede.

Cada neurônio da camada de entrada é diretamente conectado a cada neurônio da camada de saída. Neste caso as entradas da rede são diretamente mapeadas em um conjunto de padrões de saída, não sendo possível a formação de uma representação interna (WAGNER, 1996). Sendo assim a codificação proveniente do mundo exterior deve ser suficiente para implementar esse mapeamento.

Esta restrição implica que padrões de entrada similares resultam em padrões de saída similares, o que leva o sistema à incapacidade de aprender importantes mapeamentos. Neste caso padrões de entrada com estruturas similares, mas que façam parte de classificações diferentes não são possíveis de serem identificados por redes sem camadas internas, como é o caso do perceptron simples.

Para o perceptron funcionar adequadamente, os padrões a serem classificados devem ser linearmente separáveis como é o caso do conectivo E (AND) ou do OU (OR) ilustrados na Figura 16.a e 16.b. Um exemplo clássico de padrões não linearmente separáveis e que por isso estão além da capacidade do perceptron é o caso OU Exclusivo (XOR) onde, conforme a Figura 16.c, não é possível construir uma linha reta como fronteira de decisão entre as classes (HAYKIN, 2001).

A solução para problemas deste tipo seria a inclusão de uma camada a mais ao modelo perceptron, o que representaria uma generalização do perceptron de camada única.

Marvin Minsky e Seymour Papert analisaram matematicamente o Perceptron e demonstraram que redes de uma camada não são capazes de solucionar problemas que não sejam linearmente separáveis. Eles não acreditavam na possibilidade de se construir um método de treinamento para redes com mais de uma camada e con-

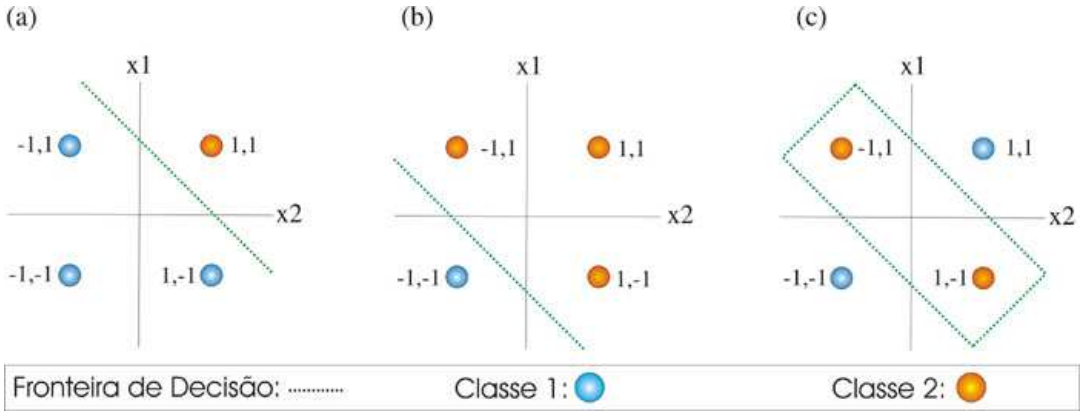


Figura 16: a) Representação gráfica de padrão linearmente separável (conectivo E); b) Representação gráfica do padrão linearmente separável (conectivo OU); Representação gráfica do padrão não linearmente separável (conectivo XOR)

cluíram em seu livro *"Perceptrons : An Introduction to Computacional Geometry"* que as redes neurais artificiais sempre seriam suscetíveis a esta limitação. A publicação deste livro causou desinteresse de alguns pesquisadores sobre o assunto, até que novas técnicas foram desenvolvidas para superar estas limitações.

3.3.3.1 Perceptrons de Múltiplas Camadas

Devido às limitações do perceptron simples surgiram as redes de múltiplas camadas alimentadas adiante conhecidas como *Multi Layer Perceptron* (MLP). Basicamente, este tipo de rede é composto por neurônios estruturados em uma camada de entrada, uma ou mais camadas ocultas e uma camada de saída, onde o sinal se propaga para frente entre as camadas, realizando funções específicas em cada passo.

A camada de saída recebe os estímulos das camadas ocultas e constrói o padrão que será a resposta da rede, enquanto as camadas ocultas recebem os estímulos da camada de entrada e operam como extratores de características, permitindo que a rede crie sua própria representação interna do problema.

A utilização de neurônios ocultos torna o processo de aprendizagem da rede neural mais complexo, pois é necessário ajustar o valor dos pesos das conexões que não são visíveis externamente, levando em conta características do padrão de entrada que devem ser representados nestes neurônios.

O treinamento é realizado de forma supervisionada baseado no aprendizado por correção de erro, através do algoritmo de retropropagação (*back-propagation*) que revolucionou as pesquisas sobre RNA por oferecer um método computacional eficiente para treinar perceptrons multicamadas.

O processo de aprendizagem por retropropagação é formado por dois passos principais, sendo que no primeiro um padrão apresentado à camada de entrada da rede propaga seu efeito através das camadas subseqüentes até que seja produzida uma resposta pela camada de saída.

No segundo passo esta resposta é comparada com uma resposta desejada produzindo um sinal de erro. Este sinal de erro é propagado para trás através da rede, fazendo com que os pesos sinápticos dos neurônios internos sejam ajustados de acordo com a regra delta conforme o erro é retropropagado, estimulando assim que as respostas geradas pela rede sejam estatisticamente mais próximas das respostas desejadas (HAYKIN, 2001).

O algoritmo de retroprapagação converge quando o sinal de erro estiver em um nível suficientemente pequeno, sendo este um dos critérios de parada, o qual define o fim do treinamento. Após este processo a rede é considerada como treinada e pronta para ser utilizada, passando a operar exclusivamente no modo progressivo (*feedfoward*).

O treinamento utilizando o algoritmo de *backpropagation* possui parâmetros como a taxa de aprendizado que deve ser ajustada de forma a otimizar o aprendizado da rede. Quanto menor a taxa de aprendizado definida, menor serão os ajustes dos pesos sinápticos da rede em cada ciclo de aprendizagem, o que proporcionará uma atualização gradativa dos pesos, porém em um tempo de treinamento consideravelmente longo.

Aumentando a taxa de aprendizagem o processo de treinamento é acelerado, provocando ajustes mais significativos a cada interação. No caso de serem definidas taxas muito altas podem ocorrer oscilações no treinamento comprometendo o processo de aprendizado. No entanto este problema pode ser contornado adicionando à regra delta um termo de momentum que tem por objetivo aumentar a velocidade de treinamento e reduzir os riscos de instabilidade (DANH et al., 1999).

O modo de treinamento é o que determina a forma de apresentação dos exemplos de treinamento para a rede neural, podendo ser classificado como modo seqüencial quando os pesos são atualizados após a apresentação de cada exemplo de treinamento, ou modo por lote quando os pesos são atualizados após todos os exemplos de treinamento terem sido apresentados.

3.4 Aplicações de Redes Neurais

As redes neurais artificiais podem ser aplicadas para a resolução de uma grande variedade de problemas. Devido a isso as pesquisas nesta área têm crescido consideravelmente. Suas características funcionais viabilizam sua utilização em casos onde há necessidade de manipulação de conhecimento impreciso ou ruidoso, além de possibilitar a construção de modelos a partir de exemplos.

Diversas aplicações envolvendo redes neurais têm sido desenvolvidas visando agregar vantagens a processos de tomada de decisão, aplicações médicas, mercado financeiro, indústria química entre outros. Em geral a principal aplicabilidade das redes neurais são problemas que necessitem tratar o reconhecimento e classificação de padrões e aproximação de funções.

3.4.1 Reconhecimento de Padrões

A área de pesquisa que aborda o reconhecimento de padrões tem por objetivo a classificação de padrões em determinadas categorias ou classes. Compreende a técnica pela qual um sistema computacional é instruído através de exemplos a identificar determinados padrões, sendo então capaz de generalizar seu conhecimento e reconhecê-los novamente, mesmo que o padrão observado não seja exatamente igual ao que lhe foi apresentado como exemplo.

A escrita manual é um exemplo clássico de reconhecimento de padrões, pois cada pessoa possui um estilo próprio de escrever sendo, portanto, impossível mapear todos os estilos de grafia. No entanto basta a apresentação de um pequeno conjunto de estilos de escrita para que através de comparações, seja possível identificar qualquer letra apresentada (CANSIAN, 1997).

O reconhecimento ou classificação de determinado padrão pode ser realizado de forma *supervisionada* onde o padrão de entrada é identificado como um membro de uma categoria definida pelos padrões de treinamento, ou *não supervisionada* onde o padrão é associado a uma categoria que é aprendida com base na similaridade entre os padrões apresentados durante o treinamento (RIPLEY, 1996).

3.4.2 Redes Neurais Aplicadas a Detecção de Intrusão

A aplicação de técnicas de inteligência artificial voltada à segurança computacional tem sido muito explorada nos últimos anos, objetivando otimizar os métodos

convencionais de detecção.

Como todo conhecimento das redes neurais está localizado nos pesos das conexões sinápticas não é necessário realizar comparações exaustivas em tempo de produção entre bases de assinaturas e padrões de comportamento, proporcionando otimização dos recursos computacionais no processo de detecção de intrusão.

A maioria dos sistemas de detecção de intrusão existentes incorporam sistemas especialistas, que codificam seu conhecimento através de um conjunto finito de regras, sendo necessárias atualizações constantes para detecção de novos ataques, mesmo que estes sejam apenas variações de ataques já conhecidos.

O poder de generalização das redes neurais habilita a rede a reconhecer variações de um mesmo tipo de ataque e classificá-los de forma correta. Porém, com o surgimento de técnicas de intrusão completamente diferentes faz-se necessário atualizar a base de treino e realizar o re-treinamento da rede neural, de forma a conferir o poder de adaptabilidade à rede. Algumas abordagens mais recentes visam otimizar esta situação, conforme detalhado em (MARKOU; SINGH, 2003).

Em (GEORGIA, 2000) são descritos processos de aprendizado que visam otimizar a adaptabilidade de sistemas de detecção de intrusão, possibilitando a detecção de novos ataques de forma autonôma. A incorporação de conhecimento é realizada por reforço contínuo, e não baseada na inserção de novas regras, como ocorre com os sistemas especialistas, nem no completo re-treinamento da rede como é o caso das abordagens tradicionais que utilizam redes neurais artificiais.

Diversas pesquisas têm sido realizadas de forma a explorar as características das redes neurais para a detecção de intrusão, como por exemplo:

- Em (RYAN et al., 1998) é apresentado um sistema de detecção de intrusão denominado NNID (*Neural Network Intrusion Detection*), o qual utiliza uma rede neural para analisar a variedade de comandos executados pelos usuários, e identificar alterações nos padrões de uso do sistema. Dependendo das características dessas variações, a utilização é considerada intrusiva.

- Em (JOO et al., 2003) é discutida a performance de sistemas de detecção de intrusão, sendo mensurada a partir de seus índices de falsos positivos e falsos negativos. Propõe ainda um modelo que utiliza redes neurais artificiais para aumentar a precisão da detecção realizada pelo sistema.

- A pesquisa apresentada em (DASGUPTA; BRIAN, 2001) descreve um modelo distribuído de detecção de intrusos, o qual aplica diversas alternativas compu-

tacionais em sua estrutura, como imunologia computacional, agentes móveis e redes neurais artificiais. O propósito dessa junção de tecnologias é criar um sistema robusto, eficaz e tolerante a falhas.

- A abordagem discutida em (CANNADY, 1998), leva em conta a construção de uma solução híbrida, que utiliza recursos de sistemas especialistas aliados as capacidades das redes neurais artificiais para otimizar o processo de detecção.

Este trabalho apresenta um modelo simplificado de detecção de intrusão, o qual aplica redes neurais artificiais ao processo de análise de eventos. Detalhes sobre este modelo serão discutidos nos próximos capítulos.

4 DETECÇÃO NEURAL DE EVENTOS INTRUSIVOS

Em (CANSIAN, 1997) foi proposto um modelo adaptativo para detecção de intrusos, o qual incorporava características adaptativas através do uso de redes neurais artificiais, prevendo a realização de uma série de tratamentos anteriores a esta análise.

Seguindo esta mesma linha, a proposta deste trabalho é apresentar um modelo funcional e simplificado de detecção de intrusos, que utiliza a propriedade de reconhecimento de padrões das redes neurais de forma a classificar a atividade da rede, identificando padrões intrusivos ou indicativos de ataque, através do conhecimento obtido durante a fase de treinamento.

Neste capítulo é apresentado um modelo que aplica os conceitos e vantagens das redes neurais em uma estrutura para análise e detecção de eventos em redes de computadores.

O modelo possui alguns módulos, onde são aplicadas técnicas de captura de pacotes, análise léxica, análise semântica e redes neurais de forma a obter, classificar e analisar o grau de severidade de cada seção de comunicação ocorrida na rede de computadores, auxiliando no processo decisório dos gerentes sobre a segurança da rede.

4.1 Modelo Computacional

O modelo proposto trata de um mecanismo de segurança capaz de detectar o comportamento intrusivo em determinadas seções de rede.

Esta detecção é realizada através de um módulo que possui características desejáveis em sistemas de detecção de intrusão, como a utilização de redes neurais para o reconhecimento dos padrões de ataque de forma a incorporar à estrutura habilidades

como adaptabilidade e generalização.

Estas capacidades possibilitariam a correta classificação de padrões semelhantes, mesmo que alguns destes padrões não façam parte do conjunto de treinamento. Esta características colaboraria na composição de uma solução robusta e adaptável a mudanças no ambiente onde está inserida.

4.2 Propriedades do Modelo

Conforme discutido no *Capítulo 2* existem diversos aspectos a serem considerados em relação a um sistema de detecção de intrusão. Dentro deste contexto foram avaliadas diferentes abordagens de registro e classificação de eventos em redes de computadores, as quais subsidiaram a presente proposta. O resultado da pesquisa é um sistema que captura e analisa o tráfego gerado na rede, classificando-o entre atividade normal e intrusiva.

Durante a etapa de captura as seções são monitoradas e registradas, de forma a compor uma representação da atividade da rede, obedecendo aos filtros estabelecidos pelo administrador no módulo de captura de pacotes. As seções capturadas são analisadas utilizando comparações em uma base de conhecimento, a qual é composta por *strings* de caracteres relevantes em perfis de ataques, auxiliando na identificação de atividades suspeitas que serão devidamente formatadas e irão compor o vetor de estímulo para a análise da rede neural.

A rede neural é o componente que efetivamente realiza a classificação dos eventos considerados suspeitos, tomando como base o conhecimento obtido durante a fase de treinamento, onde a partir de diversos padrões de comportamento e suas respectivas classificações é possível ajustar os pesos sinápticos da rede neural de forma a otimizar a detecção dos eventos intrusivos.

De acordo com a classificação de sistemas detectores de intrusos este modelo compõe uma arquitetura baseada em rede que incorpora métodos de detecção por abuso e anomalia, freqüência de uso periódico e provendo reações passivas aos eventos detectados.

4.2.1 Método de Detecção Híbrido

A saída gerada pelo modelo é obtida pelo resultado da análise neural dos vetores de estímulos, que são as entradas da rede neural, subsidiadas pelos módulos

inferiores, onde é realizada a captura, pré-seleção e a formatação do fluxo de dados da rede.

Esta pré-seleção é fundamentada pelo resultado gerado pelo analisador semântico que localiza possíveis assinaturas de intrusão nas seções capturadas o que caracteriza um método de detecção por abuso. Porém, a análise final é realizada por um componente de inteligência artificial que não se baseia em comparações diretas de assinaturas de intrusão e que tem a capacidade de tratar casos não previstos através da generalização, o que caracteriza os métodos de detecção por anomalia.

Desta forma a classificação de cada evento depende tanto da observação dos eventos que se assemelham a comportamentos intrusivos quanto da generalização de situações desconhecidas. Essas premissas permitem definir que o modelo aplica os dois métodos de detecção, constituindo uma solução híbrida.

4.2.2 Arquitetura Baseada em Rede

As seções analisadas pelo modelo são obtidas através de um módulo de captura de pacotes que atua sobre a rede de computadores, filtrando e tratando o fluxo de dados de forma a compor o conjunto de informações que serão analisadas pelos módulos superiores.

Pode-se posicionar estrategicamente o módulo de captura para obter pacotes referentes a determinado conjunto de computadores ou redes, ou ainda refinar os filtros de captura focando determinados serviços ou protocolos existentes na rede monitorada.

Devido a essas características o modelo proposto é classificado como sendo baseado em rede, o que permite maior flexibilidade quanto à obtenção de recursos a serem analisados.

4.2.3 Reação Passiva

O objetivo principal da solução proposta é detectar eventos considerados intrusivos em redes de computadores e a partir deste reconhecimento apresentar os resultados probabilísticos aos administradores, que efetivamente poderão tomar ações reativas de acordo com a gravidade de cada caso específico, o que caracteriza um comportamento passivo por parte da solução.

4.2.4 Freqüência Periódica de Uso

No modelo proposto a captura dos dados é subsídio para os tratamentos realizados. Portanto, se faz necessário efetuar a captura do fluxo de dados da rede para posteriormente analisá-los. Sendo assim, classifica-se o modelo como tendo uma freqüência periódica de uso, onde um período é definido como o intervalo de tempo entre a captura e sua respectiva análise.

Esse intervalo deve ser definido pelo administrador de forma a buscar melhores índices de respostas por captura, otimizando a utilização da solução proposta.

4.3 Arquitetura do Modelo

A arquitetura do modelo em questão procurou seguir as recomendações propostas pelo modelo CIDF e IDWG, descritas no *Capítulo 1*, onde existem módulos bem definidos para cada função do sistema de detecção de intrusão.

O foco principal deste trabalho é o reconhecimento dos eventos realizados na rede monitorada. Portanto, a arquitetura foi modelada principalmente sobre os componentes E-BOX e as A-BOX do modelo CIDF, equivalentes aos sensores e analisadores do modelo IDWG, que são, respectivamente, responsáveis pela captura e análise dos eventos da rede.

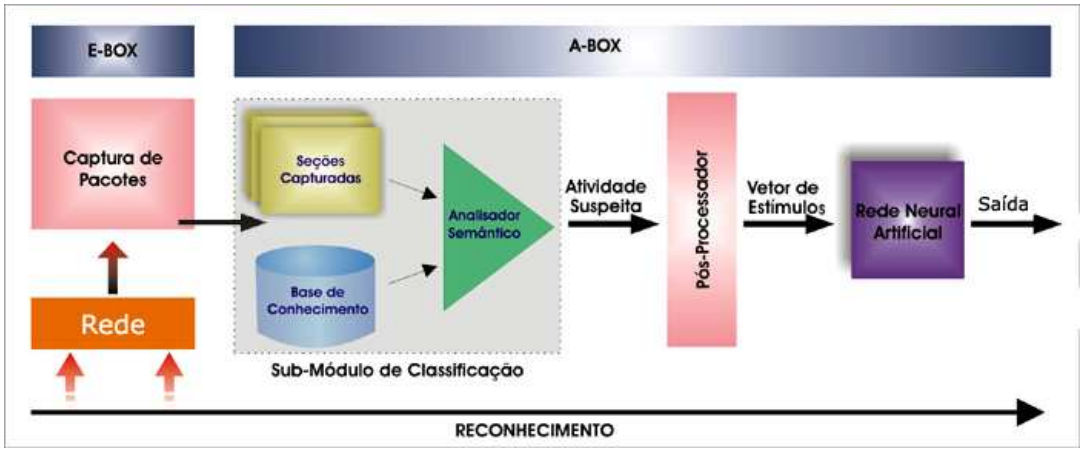


Figura 17: Arquitetura do Modelo Neural de Detecção de Intrusos.

A arquitetura modular da solução proposta é detalhada na Figura 17 onde a E-BOX é o nível mais baixo da estrutura e realiza apenas a captura do fluxo de dados da rede de acordo com os filtros estabelecidos, e posteriormente os repassa devidamente ordenados para o módulo superior denominado A-BOX.

Este módulo por sua vez contém um sub-módulo de classificação, um pós-processador e uma rede neural que juntos realizam as tarefas de classificação e análise dos eventos reportados pela camada inferior.

O componente principal de análise deste modelo é uma rede neural, cujo conhecimento é representado pelos pesos de suas conexões sinápticas, e que são ajustados através de treinamento. Portanto, pode-se criar uma analogia entre o treinamento e os respectivos padrões de treino com os componentes D-BOX do modelo CIDF, de forma que em caso de surgimento de novos padrões de ataque o ideal seria re-treinar a rede, de forma a aumentar o índice de acerto da rede.

A resposta passiva aos eventos detectados é uma das características previstas nas especificações do modelo IDWG (WOOD, 2003), sendo que qualquer reação aos eventos seria responsabilidade do administrador da estação de gerenciamento.

4.3.1 Módulo de Captura

Neste módulo são realizados além do processo de captura do fluxo de dados da rede uma pré-filtragem dos eventos de interesse e a reconstrução ordenada das seções capturadas.

Este processo é realizado através da obtenção e tratamento de informações detalhadas sobre a origem, destino e os serviços envolvidos nas conexões estabelecidas. Ao final do processo de captura é gerado um resumo ordenado seqüencialmente por evento, caracterizando um fluxo de dados bidirecional onde irão constar todos os parâmetros negociados entre as seções capturadas.

Figura 18: Captura de seções em uma rede de computadores.

Este resumo é a saída do módulo de captura que será tratado pelas camadas superiores do modelo.

4.3.2 Módulo de Análise

O módulo de análise é o responsável pela classificação de cada seção obtida pelo módulo de captura. Isto é feito através de três sub-módulos descritos a seguir, que interagem de forma a classificar, formatar e analisar os dados indicando como saída a probabilidade de cada seção representar risco à segurança.

4.3.2.1 Sub-Módulo de Análise Semântica

A principal função do analisador semântico é classificar os dados capturados de forma a identificar seções suspeitas, formando uma representação intermediária contendo tanto as informações de estabelecimento da conexão, como as *strings* que caracterizam atividade suspeita e os respectivos códigos binários para cada classe de *string* localizada.

Este módulo utiliza como parâmetro de comparação uma base de conhecimentos composta por um código binário para cada categoria de *strings* consideradas suspeitas, bem como os equivalentes para as portas de conexão e os protocolos envolvidos.

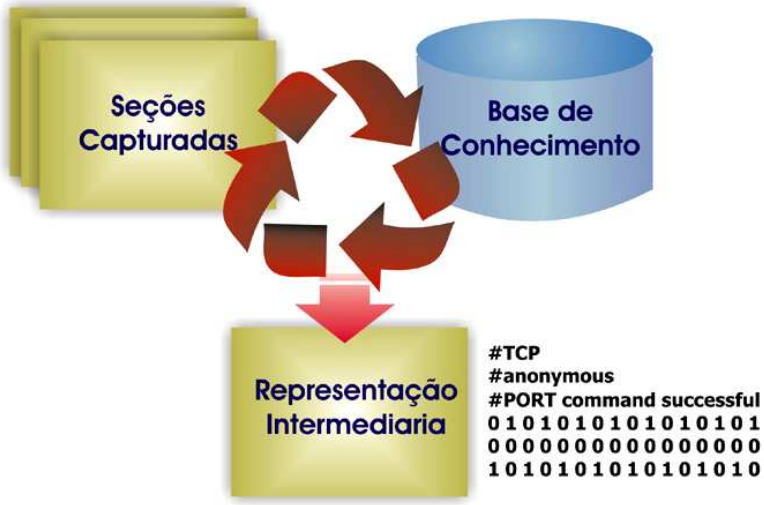


Figura 19: Representação da análise realizada pelo analisador semântico.

O objetivo é converter cadeias de *strings* que podem formar assinaturas de ataque em valores binários, pois esta é a forma de representação que deve ser apresentada à rede neural definida na arquitetura do modelo.

Dessa forma, este tratamento é realizado com todos os dados contidos nas seções capturadas, reportando quantos e quais deles combinam com supostos perfis de ataques para o sub-módulo de pós-processamento.

4.3.2.2 Sub-Módulo de Pós-Processamento

Este módulo interage com a rede neural tratando a representação intermediária gerada pelo analisador semântico, de forma a compor um vetor de estímulos para a rede neural.

Se a intenção é treinar a rede neural com as seções capturadas e previamente tratadas, o pós-processador irá compor uma representação contendo todos os vetores de estímulos e as respectivas respostas do supervisor do treinamento para cada caso.

Caso o objetivo seja analisar as seções capturadas com uma rede neural previamente treinada, é realizado um processo semelhante de forma a compor um vetor de estímulos contendo apenas as representações binárias das assinaturas de intrusão sem as respectivas classificações.

4.3.2.3 Sub-Módulo de Rede Neural Artificial

O componente que incorpora a rede neural é responsável por analisar os vetores de estímulos reportados pelas camadas inferiores procurando inferir sobre as entradas e informar o grau de suspeita de cada seção analisada. A partir de um padrão de aceitação definido pelo administrador pode-se classificar o tráfego entre normal e intrusivo.

Como é sabido, grande parte dos ataques possuem similaridades quanto às técnicas aplicadas, sendo necessário um conjunto reduzido de padrões durante o treino para que a rede neural seja treinada e possa identificar corretamente novos ataques que sigam a mesma estratégia.

Uma característica interessante desta abordagem é que o espaço necessário para armazenar o conhecimento é mínimo e invariável, pois não está localizado em regras ou assinaturas de intrusão, mas sim nos pesos das conexões sinápticas da rede os quais foram ajustados durante o treinamento. Esta característica também acarreta o fato das redes neurais serem consideradas como *caixas-pretas* por não apresentarem subsídios óbvios para explicitar o tipo de inferência que lhe levou a obter determinada resposta.

De acordo com o representado na Figura 20 a resposta gerada por um sistema com estas características é um índice em uma escala entre -1 a 1, o qual determinada se a atividade pode ser considerada intrusiva ou normal. Este resultado é diferenciado, com relação a maioria das ferramentas de detecção de intrusão, onde as respostas são apenas uma classificação booleana de cada atividade.

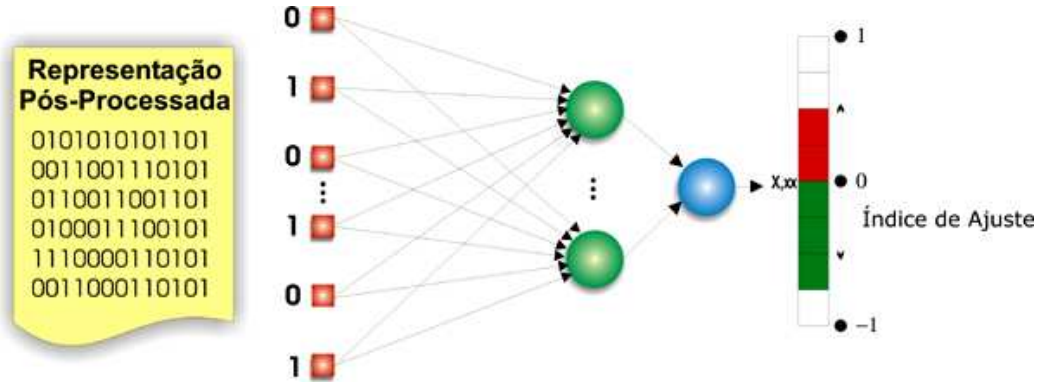


Figura 20: Análise neural da representação pós-processada da captura.

4.4 Treinamento

Para que seja possível identificar corretamente as entradas que podem compor perfis de ataque se faz necessário que a rede neural tenha sido devidamente treinada com um conjunto suficiente de padrões que representem tanto comportamento normal como intrusivo.

O treinamento utilizado neste modelo é classificado como sendo supervisionado, pois se faz necessário o auxílio de um supervisor que efetivamente possui o conhecimento e o representa através de respostas corretas para cada padrão de treino, possibilitando que a estrutura adquira o conhecimento necessário para classificações posteriores na fase de execução.

Podem existir situações onde a rede neural esteja reportando níveis de suspeitas inferiores para padrões que sejam potencialmente intrusivos. Neste tipo de caso estes padrões devem ser incorporados aos dados de treinamento e a rede neural deve ser re-treinada, o que acarretaria mudanças no comportamento da rede neural possibilitando a correta identificação do padrão de ataques.

Após a rede neural ter sido treinada, ela está apta a realizar quantas classificações forem necessárias, utilizando o conhecimento adquirido para inferir sobre situações reais de análise.

Através de técnicas de persistência pode-se armazenar o conhecimento obtido durante o treinamento em um arquivo de pesos, de forma a utilizar o mesmo conhecimento futuramente sem que seja necessário realizar novamente o treinamento.

5 IMPLEMENTAÇÃO DE UM PROTÓTIPO DO MODELO

Com o propósito de realizar testes e mensurar resultados sobre o modelo proposto no *Capítulo 4*, foi implementado um protótipo denominado I-IDS, o qual segue todas as especificações e incorpora todos os módulos definidos pela arquitetura proposta. Neste capítulo serão apresentados detalhes da implementação deste protótipo.

5.1 Introdução

O protótipo desenvolvido, resumidamente inicia a captura das seções de rede em um servidor com sistema operacional *Unix Like*, as quais são exportadas para uma estação de gerenciamento com sistema operacional *Microsoft Windows*, onde são realizados os tratamentos e análises destas seções, reportando cada nível de suspeita ao administrador que opera a estação de gerenciamento através de uma interface gráfica.

Os módulos foram desenvolvidos de forma independente, possibilitando que os sensores que são compostos pelo módulo de captura possam residir em máquinas separadas do sistema de gerenciamento, composto pelo analisador semântico e a rede neural. Esta estrutura possibilita monitorar diversas redes, cada qual com um sistema de captura de pacotes, e centralizando as informações de alerta em um agente central de segurança.

Conforme discutido no capítulo anterior, os módulos envolvidos foram implementados procurando reproduzir os componentes de padronizações como os modelos CIDF e IDWG.

O protótipo foi desenvolvido em linguagem *C/C++*, adequando o ambiente de programação de acordo com a necessidade da implementação. Para o módulo de captura utilizou-se apenas o compilador *gcc* e as respectivas bibliotecas, e para o

módulo de análise o ambiente *Borland C++ Builder*. O processo aplicado para o desenvolvimento bem como as características de implementação são descritas nas seções subseqüentes.

5.2 Captura de Pacotes

Este módulo do protótipo realiza a captura dos pacotes da rede, aplicando determinados filtros, técnicas de pré-seleção e análise inicial. Essas funcionalidades preparam os dados para análises posteriores.

A implementação deste módulo foi realizada a partir do projeto *tcpflow* (ELSON, 2001), que utiliza como base a biblioteca de captura de pacotes denominada *libpcap* (MCCANNE et al., 2002), que dispõe de diversos recursos para sistemas de captura de pacotes, se adequando, portanto, ao domínio do problema.

Esses recursos permitem ao módulo ter acesso aos dados que trafegam na rede, os quais são devidamente mapeados, filtrados e formatados gerando um arquivo que representa a atividade bidirecional da rede de acordo com os filtros estabelecidos.

Estes filtros determinam o resultado da captura. Caso não tenha sido definido nenhum tipo de filtro, todo o tráfego IP da rede é capturado, descartando todo o restante.

Por utilizarem a mesma biblioteca este módulo possui a mesma estrutura de filtragem utilizada pelo *tcpdump*, sendo possível definir vários tipos de filtros de acordo com a necessidade do utilizador, podendo ainda importar arquivos capturados pelo próprio tcpdump e dar o tratamento equivalente.

A captura das seções pode ser iniciada e finalizada remotamente por meio da interface gráfica da estação de gerenciamento, a qual importa o resultado fornecido pelo módulo de captura e disponibiliza os demais recursos previstos para este protótipo.

5.2.1 A Biblioteca libpcap

Essa biblioteca desenvolvida em C permite a captura de vários tipos de pacotes como Ethernet, FDDI, IP, ARP, RARP, DECNET, LAT, SCA, MOPRC, MOPDL, TCP e UDP (ELSON, 2001), bastando ajustar configurações de filtros a serem aplicados sobre suas funções de captura.

Estes filtros determinam o tráfego que será capturado ou descartado pelo módulo. O filtro de pacotes aplicado nesta biblioteca se baseia na arquitetura BPF - *BSD Packet Filtering* que aplica diversos componentes, conceitos e técnicas visando otimizar o desempenho da filtragem, conforme detalhado em (MCCANNE; JACOBSON, 1993).

Por meio de funções como *pcap_ lookupdev()*, *pcap_ lookupnet()* e *pcap_ datalink()* é possível determinar características da interface, da rede e dos pacotes a serem monitorados, possibilitando que seja dado o tratamento adequado a cada pacote independente do protocolo utilizado (MCCANNE et al., 2002).

Através do modo de operação denominado promíscuo é possível ter acesso a todos os pacotes distribuídos na rede em forma de broadcast, possibilitando que se monitore atividades que ocorrem entre conexões que não envolvem diretamente a estação de captura. A função *pcap_ openlive()* além de iniciar o tratamento da captura permite definir se o modo promíscuo deve ou não ser ativado.

A coleta e processamento dos pacotes é realizada por funções como *pcap_ loop()* ou *pcap_ dispatch()*, que provêm recursos para que os pacotes sejam mapeados em estruturas na memória e devidamente tratados.

Muitas outras funções são disponibilizadas por esta biblioteca e podem ser encontradas detalhadamente em (MCCANNE et al., 2002).

5.2.2 Técnicas de Captura

Para tratar os pacotes capturados são levados em consideração os parâmetros contidos em áreas do datagrama como cabeçalho IP, TCP e *payload*, pois estas informações são subsídios para tratamentos posteriores que irão compor a representação do fluxo da rede.

Para montar esta representação é realizado um mapeamento dos dados relevantes em estruturas de dados na memória, visando construir um arquivo para cada seção e para cada sentido do fluxo de dados capturados. Estes arquivos seriam identificados da seguinte forma:

```
192.168.101.102.02345-010.001.001.013.00023_TCP
```

Esta identificação representa um fluxo TCP unidirecional originado pelo host 192.168.101.102 na porta 2345 para o host 10.1.1.13 na porta 23, geralmente associada ao serviço de telnet. Cada linha deste arquivo contém um identificador que

representa a ordem de ocorrência de cada evento nesta seção, conforme exemplificado na Tabela 1.

Fluxo TCP unidirecional
00000000001 Welcome to Debian/Linux.
00000000002 Linux login:
00000000004 Password:
00000000006 Login Incorrect
00000000007 Linux login:

Tabela 1: Fluxo de informação TCP unidirecional.

À medida que as seções são desenvolvidas os respectivos arquivos identificadores são atualizados. Ao final do processo de captura são processados todos os arquivos de fluxo gerados, procurando montar uma representação única e bidirecional de todo o tráfego.

Esta representação única é obtida através da validação, mesclagem e ordenação de todos os arquivos unidirecionais obtidos, contendo portando todas as informações sobre a captura realizada.

Cada registro de seção dentro deste arquivo é composta por diversos caracteres de formatação e informações de fluxo, bem como marcadores que identificam o sentido da comunicação, conforme representado na Tabela 2.

Identificadores de Início de seção
#####_I-IDS_#####
TCP 192.168.101.102:2345 -> 10.1.1.13:23
####_I-IDS_#####:1:

Tabela 2: Caracteres marcadores de início de seção.

Além destes marcadores que identificam cada seção capturada, são detalhados logo em seguida no arquivo os respectivos eventos gerados, todos devidamente marcados por identificadores que representam o sentido deste fluxo.

Os eventos antecedidos pelo marcador `I-IDS->` representam o fluxo de informação proveniente do endereço que originou a seção para o endereço destino, ao passo que `I-IDS<-` representa o fluxo de dados gerados a partir do host destino em direção à origem.

Uma seção de FTP não intrusiva originada a partir do host 10.1.1.4 para o host 200.195.169.61 seria representada conforme a Tabela 3, possibilitando mapear em detalhes o fluxo de dados gerado durante o período de captura.

Seção FTP não intrusiva.

```
#####_I-IDS_#####
TCP 10.1.1.4:32771 -> 200.195.169.61:21
####_I-IDS_#####:5:
I-IDS<-220 ProFTPD 1.2.0pre10 Linux Server
I-IDS->USER igor
I-IDS<-331 Password required for igor...
I-IDS->PASS teste123
I-IDS<-230 User igor logged in...
I-IDS->SYST
I-IDS<-215 UNIX Type:  L8..
I-IDS->PWD
I-IDS<-257 "/home/igor"is current directory...
I-IDS->QUIT..
I-IDS<-221 Goodbye...
I-IDS:END
```

Tabela 3: Exemplo de seção FTP não intrusiva.

Ao final de cada seção registrada é inserida uma última linha contendo o marcador `I-IDS:END` caracterizando que a seção foi finalizada.

A possibilidade de filtrar o conteúdo a ser capturado, proporciona flexibilidade ao módulo de captura, pois é possível estabelecer determinados alvos antes de realizar qualquer processamento posterior. Dessa forma é possível iniciar uma captura baseada em fatores críticos da rede monitorada, como determinados hosts, redes, portas, protocolos, ou mesmo combinações entre esses parâmetros.

5.3 Análise das Seções Capturadas

Ao final do processo de captura e geração da respectiva representação, as informações obtidas serão devidamente restauradas e tratadas pelos módulos superiores, de forma a classificar as seções potencialmente intrusivas e avaliar os respectivos níveis de suspeita, através das propriedades de reconhecimento de padrões da rede neural artificial.

O módulo que compõe a implementação da A-BOX do modelo é constituído por sub-módulos, onde cada um deles oferece subsídios ao sub-módulo imediatamente

superior. Basicamente processos como busca de assinaturas suspeitas, conversão das *strings* para binário, preparação do vetor de estímulos e análise neural são realizados neste módulo, compondo fases obrigatórias e bem definidas para o método de análise.

5.3.1 Análise Semântica

O analisador semântico é o módulo que irá processar a representação gerada pelo módulo de captura em busca de *strings* que podem representar uma atividade suspeita. Essas *strings* avaliadas podem ser a definição de determinados protocolos ou portas envolvidas na conexão, mas tratam-se principalmente das mensagens trocadas entre os hosts durante o desenvolvimento da seção analisada.

Esta análise é apoiada por determinados arquivos que compõem a base de conhecimentos do modelo. Estes arquivos contêm as *strings* que representam atividade suspeita e seus respectivos identificadores de categoria, os quais são associados a um código binário que será utilizado para compor a representação intermediária da seção.

A base de conhecimentos contém determinadas palavras que sozinhas não representam necessariamente alguma atividade suspeita, porém combinadas a outros parâmetros podem representar uma atividade potencialmente intrusiva.

O principal arquivo que compõe a base de conhecimentos é composto por várias linhas onde cada uma delas possui um código decimal que identifica sua categoria, bem como a respectiva *string* que servirá como parâmetro de busca nos arquivos capturados pelo módulo inferior. Na Tabela 4 pode-se visualizar um pequeno exemplo da estrutura de um arquivo do gênero. A lista completa dos arquivos que compõem a base de conhecimentos pode ser encontrada no *Apêndice A*.

Com a finalidade de otimizar a busca por assinaturas de intrusão no tráfego da rede, além de palavras que representam atividade suspeita foram incorporadas à base de conhecimentos algumas *strings* encontradas na base de regras do *Snort*, que é um dos sistemas de detecção de intrusão por assinatura largamente utilizados. Estas strings foram extraídas do campo *content* das regras mais significativas deste IDS.

A seguir um exemplo de uma regra do *Snort* a qual possibilita a identificação de um ataque web, que procura manipular permissões de arquivos. A *string* "*/bin/chmod*"da seção *content* é uma das *strings* extraídas que irão compor a base de conhecimentos do protótipo.

Lista da Base de Conhecimento
1 login:administrator
1 login:acess
2 permission denied
3 Login incorrect
4 /etc/inetd.conf
4 /etc/passwd
4 /etc/services
4 /etc/shadow
5 vi
5 less
6 logged in
7 USER (none)
8 anonymous

Tabela 4: Exemplo parcial do principal arquivo que compõe a base de conhecimento.

```
alert tcp $EXTERNAL_NET any->$HTTP_SERVERS $HTTP_PORTS(msg:"WEB-ATTACKS
chmod command attempt";flow:to_server,established;content:"/bin/chmod";
nocase; sid:1336; classtype:web-application-attack; rev:4;)
```

Apesar das regras do *Snort* serem compostas por outros parâmetros além da *string content*, leva-se em consideração que a ocorrência destas *strings* aliadas a outras atividades registradas potencialmente caracteriza uma atividade suspeita.

A base de conhecimentos foi enriquecida ainda com diversos parâmetros que são geralmente utilizados por ferramentas de varredura conferindo ao protótipo a capacidade de identificar seções de reconhecimento. Além disso várias *strings* comuns em seções suspeitas foram adicionadas ao conjunto, proporcionando maior abrangência ao levantamento inicial de seções suspeitas.

As formas de realizar buscas por estas assinaturas nos arquivos possuem algumas peculiaridades que devem ser levadas em consideração, e que serão discutidas mais adiante neste mesmo capítulo.

5.3.1.1 Representação Binária das Assinaturas

A base de conhecimentos é composta por outros arquivos que auxiliam o processo de representação binária da atividade suspeita, mapeando cada categoria definida no arquivo apresentado anteriormente em seu respectivo código binário, conforme a

Tabela 5.

Apoio à Conversão Binária
1 1000000000000001
2 1000000000000010
3 1011000000000001
4 0011000000000000
5 0101000000000000
6 1000000110000001
7 1000001111000000
8 0011010000100000

Tabela 5: Exemplo parcial do arquivo de apoio à conversão binária das categorias da LBC.

Este mapeamento binário de cada classe deve ser realizado seguindo alguns princípios, pois a forma de representação de cada categoria impacta diretamente sobre o tempo de treinamento e os resultados das análises geradas pela rede neural.

Caso a codificação binária das categorias não seja corretamente definida poderá levar a rede a confundir padrões que deveriam ser considerados diferentes, sendo assim a rede levaria mais tempo de treino para poder criar uma representação correta de cada padrão e depois de treinada sua taxa de acerto poderia ser comprometida ao tentar generalizar novos padrões.

Os códigos binários atribuídos às categorias devem possuir determinada *distância de hamming* entre si, ou seja, devem ser diferentes dos demais em um número mínimo de bits para garantir que todas as representações sejam suficientemente diferentes, sendo possível minimizar as dificuldades ao representar os padrões e obter ganhos significativos de rendimento, tanto no treinamento quanto em execução. Conforme discutido em (CANSIAN, 1997) esta é uma abordagem adequada para tratar este tipo de problema.

No modelo implementado cada categoria definida é associada a um código binário de 16 bits, onde é definido o valor da *distância de hamming* de acordo com o número de combinações necessárias para representar as *strings*.

Como a base de conhecimentos do protótipo é formada por aproximadamente 100 categorias, podemos compor representações binárias diferentes no mínimo em 6 bits uns dos outros, em 129 categorias possíveis. Caso seja necessário representar um número maior de categorias, basta utilizar o gerenciador desenvolvido especificamente para determinar os possíveis códigos binários baseado no valor estipulado

para o número de bits e a respectiva *distância de hamming*.

Através desse gerenciador é possível adicionar ou remover registros de categorias ou assinaturas, e até mesmo otimizar a diferenciação da codificação binária existente. Como os arquivos de codificação binária são separados do arquivo de *strings*, não se faz necessário manipular o arquivo de *strings* para redefinir o grau de diferença binária entre as categorias cadastradas. Sendo assim o uso deste gerenciador simplifica e flexibiliza o processo de manutenção na base de conhecimentos.

Da mesma forma que o mapeamento binário realizado para as *strings*, as portas e protocolos envolvidos na comunicação também são mapeados para seus correspondentes códigos binários, que, por serem encontrados em menores quantidades, podem ser representados com uma diferença de bits maior entre eles, conforme representado na Tabela 6.

Portas
21 0000000011111110
22 0000111100001110
23 0000111111110001
25 0011001100110010
53 0011001111001101
79 0011110000111101
80 0011110011000010
109 0101010101010100
110 0101010110101011
111 0101101001011011
113 0101101010100100
Protocolos
TCP 0101010101010101
UDP 1010101010101010

Tabela 6: Demais arquivos de apoio à conversão binária.

Estes arquivos formam o componente denominado base de conhecimento deste sub-módulo, que será utilizada como parâmetro de comparação e representação para formar a saída gerada.

5.3.1.2 Reconhecimento de Atividade Suspeita

A análise inicial do cabeçalho de determinada seção que compõe o arquivo capturado é realizada catalogando os protocolos e portas envolvidos na conexão. Logo após são analisadas as ações ocorridas procurando localizar possíveis assinaturas de intrusão.

A busca pelas ocorrências de *strings* nas seções que constem na base de conhecimento é um processo que requer técnicas mais funcionais do que uma simples busca exaustiva entre os arquivos.

Ao se procurar por *login* podem existir *login administrator* ou *login admin*, ou simplesmente *login*, o que exemplifica que ao se executar determinada análise podem existir ocorrências de *strings* que não caracterizam uma assinatura, mas que pode vir a fazer parte de uma assinatura completa.

Neste modelo foram desenvolvidos métodos que otimizam a busca de assinaturas nos arquivos capturados, procurando contornar dificuldades decorrentes deste tipo de processo. Para isso foi construído um autômato finito simples baseado na descrição proposta em (CANSIAN, 1997) quando abordava o mesmo problema, o qual é descrito no *Apêndice C*.

Nesta implementação cada linha do arquivo capturado é analisada em busca de palavras. Como geralmente os caracteres que separam palavras não são letras nem dígitos, foi definido que cada ocorrência destes caracteres identifica o término de uma palavra.

A palavra localizada é utilizada como parâmetro de busca na base de conhecimentos, onde será verificado se ela por si só constitui uma assinatura de intrusão ou se ela combina com o início de uma assinatura específica. Neste último caso se faz necessário armazenar cada parte da palavra para que seja possível identificar a assinatura completa.

Supondo que a palavra procurada seja *unknown user name* e na base de conhecimento conste as palavras *user name* e *unknown user loged*, o autômato buscaria na base de conhecimento cada palavra, começando por *unknown* que seria localizada parcialmente, então se marcaria este ponto e continuaria a busca por *user* que também constitui parte de ambas as assinaturas. Novamente os pontos de coincidência são marcados e é dada seqüência ao processo de busca pelo componente *name* que não coincide com o termo *loged* da segunda assinatura, portanto descartando esta hipótese. Porém o termo *name* que aliada a *user* já identificada na varredura ante-

rior compõe uma assinatura completa, identificando a *string user name* como uma *string* suspeita.

5.3.1.3 Representação da Atividade Suspeita

O processo de reconhecimento e mapeamento binário é realizado com todo o arquivo capturado, identificando todas as ocorrências de *strings* suspeitas de cada seção que irão compor um arquivo juntamente com os respectivos códigos binários de suas categorias. Este arquivo é uma representação intermediária entre a captura e a entrada da rede neural.

A Tabela 7 apresenta um exemplo desta representação intermediária obtido em determinada seção capturada.

Representação Intermediária
#8:3
#TCP
#21
#USER
#login:administrator
#mail
#PASS
#anonymous
#mail
#user name
#PORT command successful
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0
1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Tabela 7: Representação intermediária de uma seção de FTP.

Nas seções iniciadas pelo marcador # constam informações referentes à seção tratada e o significado de cada código binário que compõe sua representação. Neste exemplo a primeira linha #8:3 representa que foram localizadas oito *strings* suspeitas na terceira seção analisada. A segunda e terceira linha indicam respectivamente o protocolo e a porta utilizada durante a conexão. As demais linhas formam a relação de *strings* localizadas e ao final da relação o respectivo código binário contendo todas as informações da seção. Nesta implementação uma seção deve conter 16 representações binárias de 16 bits cada, onde duas delas são destinadas ao protocolo e porta utilizada, e os demais para eventuais ocorrências de *strings* suspeitas. Caso não existam 14 *strings* suspeitas em uma seção, a representação do restante é preenchido com 0000000000000000. Este processo se faz necessário, pois com isso será gerado o vetor de estímulos da rede neural que deve possuir número fixo de entradas.

5.3.2 Pós-processamento

Este sub-módulo é responsável por compor o vetor de estímulos que será usado como entrada para a rede neural, e que será composto a partir da representação intermediária gerada pelo sub-módulo anterior.

Este processo é utilizado tanto para gerar entradas que serão analisadas, quanto para gerar entradas que serão utilizadas durante o treinamento supervisionado da rede neural.

Ao processar uma entrada para análise, cada seção registrada no arquivo intermediário é processada individualmente de forma a gerar outro arquivo que na primeira linha informará o número de padrões que constam no arquivo e os códigos binários de cada seção dispostos linha por linha. Como cada seção é composta por 16 padrões de 16 bits, o vetor de estímulos possuirá 256 entradas por seção, que serão analisadas individualmente pela rede neural, a qual irá utilizar o conhecimento obtido durante o treinamento para definir o grau de suspeita de cada padrão informado pelo vetor de estímulos.

Caso a ação solicitada seja usar as seções capturadas para treinar a rede neural o processo é similar, o único diferencial é que a cada seção analisada os padrões são exibidos ao supervisor de treino para que ele possa classificar o padrão entre intrusivo ou não. A resposta do supervisor será armazenada ao final da seqüência de bits de cada seção, sendo 1 caso a seção caracterize uma ação intrusiva e -1 caso seja uma atividade normal.

Outra opção do protótipo permite exportar o arquivo de treino para o formato

utilizado pelo simulador *JavaNNS* que será discutido mais adiante neste capítulo. Esta função possibilitaria utilizar os mesmos padrões de treino diretamente no protótipo ou no simulador neural. Mais detalhes sobre a formatação deste arquivo estão disponíveis em (FISCHER et al., 2001).

O sub-módulo de rede neural ao treinar a rede irá utilizar este arquivo de treino para ajustar seus pesos e adquirir conhecimento de acordo com as informações passados pelo instrutor.

5.3.3 Análise Neural

Este sub-módulo é o principal componente do modelo, pois todos os sub-módulos inferiores preparam os dados para que efetivamente sejam analisados pela rede neural que informará como saída o nível de suspeita de cada seção analisada.

O principal propósito para se aplicar redes neurais no processo de detecção de intrusão é a possibilidade de se definir um pequeno conjunto de padrões para serem usados como arquivo de treino representando assinaturas conhecidas, e que seriam suficientes para viabilizar a detecção de novos ataques a partir da similaridade entre ataques conhecidos.

5.3.3.1 Rede Neural

A seleção de uma rede neural artificial para ser aplicada ao domínio do problema deve ter por subsidio a avaliação de importantes critérios, tais como topologia, algoritmo de treinamento, taxa de aprendizado entre outros. Esses parâmetros devem ser ajustados de forma a melhor atender as necessidades do modelo proposto.

A fim de determinar qual seria a melhor estrutura de rede neural a ser aplicada, foram avaliadas diversas alternativas através de um ambiente de simulação denominado *JavaNNS* (*Java Neural Network Simulator*)(FISCHER et al., 2001) desenvolvido pelo *Wilhelm-Schickard-Institute for Computer Science* (WSI) em Tubingen na Alemanha. Este simulador, desenvolvido em Java, oferece diversas opções para se criar redes neurais artificiais de vários tipos, bem como treinar e mensurar seu desempenho.

De acordo com a estrutura das entradas compostas pelo vetor de estímulos gerado pelo sub-módulo de pós-processamento, que possui 16 padrões de 16 bits, e da necessidade de classificação em apenas 2 categorias (intrusivo e não intrusivo), fixou-se a entrada da rede neural em 256 neurônios e a saída em 1 neurônio, que

informa determinados índices que classificam o padrão de entrada.

De forma a evitar a ocorrência de falsos positivos ou falsos negativos a margem de saída estabelecida para caracterizar uma atividade normal ou intrusiva pode ser configurada de acordo com o cenário. Como padrão, o sistema trata que o valor de saída estando no intervalo -1 e 0 seria uma atividade normal e valores no intervalo 0 e 1 caracterizariam atividades intrusivas.

Devido a capacidade de resolver problemas não linearmente separáveis das redes neurais *feedfoward* do tipo *multilayer perceptrons*, elas foram escolhidas para compor este componente do protótipo. Para o treinamento foi escolhido o algoritmo de retro-propagação de erro (*backpropagation*), que apresentou resultados satisfatórios diante de simulações realizadas com outros algoritmos disponíveis no simulador. A função tangente hiperbólica foi utilizada como função de ativação da rede, pois determina resultados dentro do limiar de -1 a 1, que foi definido para a saída da rede neural.

Através de simulações foram testadas diversas configurações deste tipo de rede e optou-se por determinar uma camada oculta composta por 21 neurônios, os quais atuam como extrator de características e se conectam ao neurônio de saída, que irá efetivamente informar o grau de suspeita de cada vetor de estímulo analisado.

Após determinar as características da rede neural, optou-se por implementá-la diretamente em um protótipo, o qual incorpora todos módulos do modelo. Essa implementação gerou independência de ferramentas externas, como é o caso do simulador *JavaNNS*, e facilidades de utilização a partir da estação de gerenciamento do administrador.

A implementação desse componente foi realizada através de uma biblioteca em C, denominada *neuro.h* (FABRO, 2003) a qual contempla em sua estrutura todas as funcionalidades necessárias para criação, treinamento e utilização da rede neural artificial com as características estipuladas.

Por meio de uma função implementada no protótipo é possível reconfigurar parâmetros como taxa de aprendizado, erro máximo e momentun de acordo com a necessidade do treinamento, bem como reestruturar o número de neurônios de cada camada da rede caso seja necessário.

A inicialização da rede neural é realizada através da função que gera pesos randômicos que calcula determinados valores dentro do intervalo estabelecido para as conexões sinápticas, sendo necessário, portanto treiná-la após sua criação de forma a ajustar todos os pesos para a solução do problema.

A rede neural irá usar o conhecimento adquirido durante o treinamento para classificar todos os outros padrões reais. A principal pretensão é que ela passe a reconhecer com certo grau de certeza, todo tipo de ataque que possua alguma similaridade com os apresentados à rede na fase de treino. Esta propriedade conferiria ao modelo vantagens sobre as análises convencionais de eventos em sistemas detectores de intrusão.

5.3.3.2 Treinamento

O treinamento tem como função agregar conhecimento à rede neural, atualizando os pesos das conexões sinápticas de acordo com as informações contidas nos dados de treinamento.

Uma das opções do protótipo permite realizar o treinamento da rede neural a partir do arquivo de treino gerado pelo pós-processador. As opções implementadas permitem realizar o treinamento da rede neural diretamente no protótipo ou no simulador neural *JavaNNS* que possui uma grande diversidade de algoritmos para treinamento.

O algoritmo de treinamento utilizado no protótipo em questão é o *backpropagation*, onde cada época ou ciclo de aprendizagem representa uma apresentação completa de todo o conjunto de padrões. Após a apresentação de cada padrão é calculado o valor do erro da rede, que é retro-propagado atualizando os pesos sinápticos.

O protótipo implementado pode realizar o número necessário de ciclos até se atingir o nível de erro definido nas configurações do treino. Porém, é possível configurá-lo para que este passe a treinar a rede baseado em determinado número de ciclos, possibilitando um controle mais amplo sobre o número de épocas de treinamento.

O treinamento pode ser acompanhado por um gráfico, onde se pode analisar a curva de aprendizado gerada pela variação do erro à medida que os ciclos ocorrem, bem como observar o ponto de convergência do processo de treino. Ao se alcançar este ponto, dentro dos níveis aceitáveis de erro, a rede neural é considerada suficientemente treinada.

O treinamento se comporta de acordo com os parâmetros estabelecidos para taxa de aprendizado, momentum ou erro máximo. Para se definir os parâmetros ideais foram realizadas simulações com conjuntos de padrões durante a fase de testes.

Após o treinamento ser concluído é gerado um arquivo de pesos que representa todo o conhecimento adquirido. Estes pesos são atribuídos à rede neural tornando-a apta a realizar análises.

A geração deste arquivo de pesos representa uma forma de persistência do conhecimento adquirido, pois é possível a aquisição de conhecimento carregando os pesos deste arquivo para a rede neural, sem a necessidade de realizar novamente o mesmo treinamento. Desta forma é possível transferir todo conhecimento adquirido para outras redes com a mesma estrutura, bastando atualizar seus pesos de acordo com o arquivo estabelecido.

Este arquivo possui um tamanho fixo de aproximadamente 50 KB, independente do treinamento realizado, sendo esta uma das vantagens da utilização desta abordagem. Um breve exemplo deste arquivo de persistência do conhecimento pode ser observado no *Apêndice B*.

5.4 Monitoração de Processos

Uma importante função do protótipo desenvolvido é a monitoração do processo de análise, realizado por meio da interface gráfica disponível na estação de gerenciamento, a qual possibilita o acompanhamento intuitivo do processo de detecção.

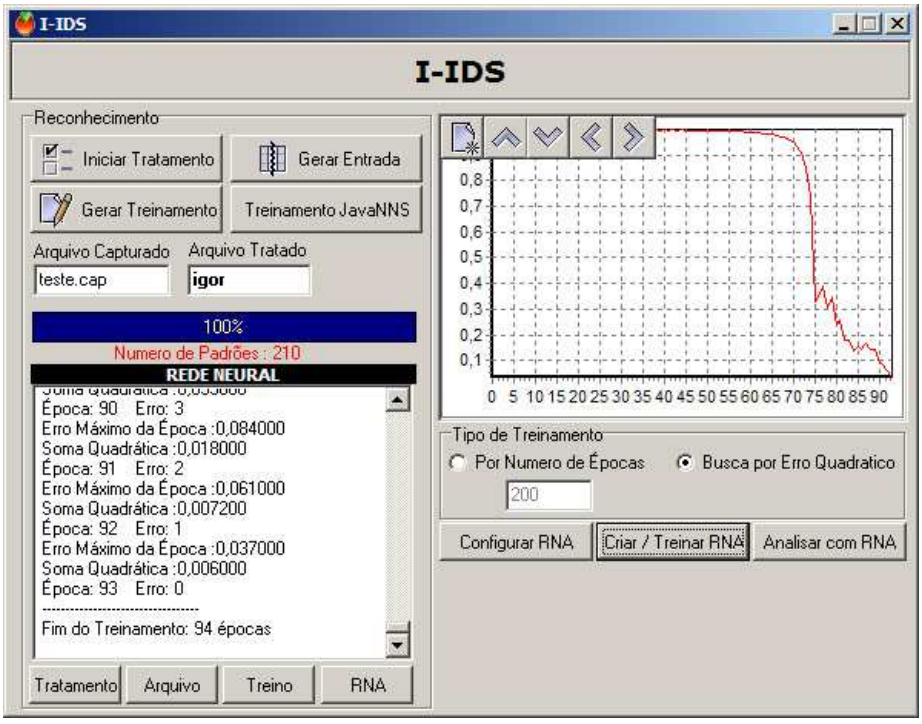


Figura 21: Interface gráfica do protótipo durante a fase de treino.

A interface disponibiliza várias ações que podem ser executadas pelo administrador sobre o conjunto de dados tratados. A evolução de cada etapa é apresentada em um painel informativo, que permite acompanhar os resultados obtidos por todas as fases.

Nas Figuras 21 e 22, pode-se observar todas as opções disponibilizadas pela interface gráfica.

As etapas da análise neural também podem ser acompanhadas através do respectivo painel, onde são informados detalhes sobre a configuração, treinamento e performance da rede neural. Todo o processo de aprendizado pode ser visualizado através de gráficos que representam a evolução do processo de treinamento.

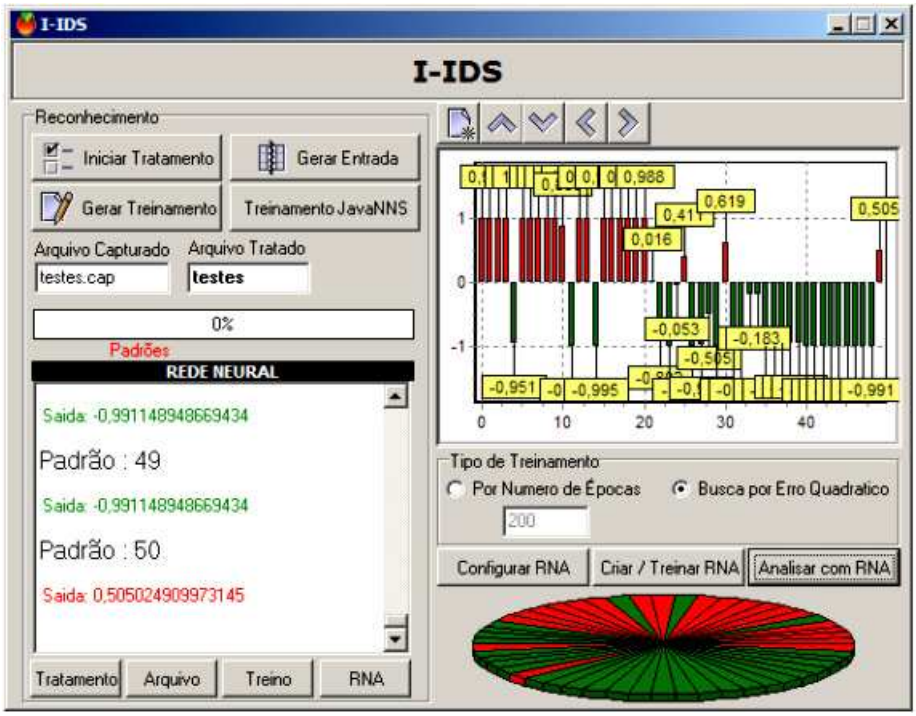


Figura 22: Interface gráfica do protótipo durante a fase de análise.

Os resultados obtidos pela análise durante a tarefa de detecção podem ser acompanhados através dos mesmos recursos. No painel são exibidos os índices das saídas para cada padrão de entrada e nos gráficos são representados suas variações em números e cores, proporcionando ao gerente maior controle sobre o resultado da análise e uma forma gráfica e intuitiva de avaliar a gravidade destes eventos.

6 EXPERIMENTOS E RESULTADOS OBTIDOS

Este capítulo detalha os experimentos realizados e seus respectivos resultados, os quais possibilitam visualizar os diferenciais desta abordagem, em relação a outros métodos de análise.

6.1 Introdução

Com o objetivo de avaliar o modelo proposto, vários experimentos foram efetuados com o protótipo implementado, os quais apresentaram resultados que demonstram as vantagens da abordagem proposta neste trabalho.

Para a elaboração dos testes foram necessárias simulações de comportamentos intrusivos e não intrusivos, todos devidamente capturados pelos sensores que monitoravam o ambiente de testes.

A partir da captura destes comportamentos foram definidas assinaturas de ataques e não ataques. Após a devida conversão binária essas assinaturas foram utilizadas como dados de treinamento e avaliação para a rede neural.

O protótipo foi submetido a várias situações de treinamento e avaliação, visando obter resultados comparativos da estrutura em diferentes cenários de treino e execução.

A seguir são detalhadas as fases dos experimentos realizados bem como os resultados obtidos.

6.2 Ambiente de Testes

De modo a comprovar a eficiência do protótipo foram simulados experimentos utilizando uma estrutura comumente encontrada em cenários reais. Foram realizadas varreduras e explorações originadas de estações em uma rede doméstica em direção a um servidor corporativo o qual disponibiliza uma série de serviços com e sem vulnerabilidades conhecidas e que possuía sensores estrategicamente posicionados e configurados para capturar todo o tráfego gerado pela atividade intrusiva e atividade normal daquele período.

Durante os experimentos realizados foram efetuados diversos ensaios para se determinar quais serviços teriam maior relevância e qual seria o conjunto ideal de padrões para os testes. Observou-se que vários ataques gerados possuíam comportamentos muito parecidos o que resultaria em um único padrão utilizável para vários casos.

Os experimentos foram realizados utilizando-se portas referentes aos serviços comumente utilizados nos mais diversos ambientes computacionais.

Depois de capturado o tráfego de interesse, é gerado um arquivo que é encaminhado para uma estação de gerenciamento dentro da rede, onde são realizadas as análises que determinaram o nível de suspeita de cada seção.

O servidor corporativo onde foi realizada a captura possui as seguintes características:

- Processador *INTEL Pentium IV* 2.0 *GHz*, 256 MB de memória DDR.
- Sistema operacional *Linux*, *kernel 2.2.19*, distribuição *Debian Potato*.
- Principais serviços oferecidos monitorados:

  DNS (*Dynamic Name Service*) - BIND

  FTP (*File Transfer Protocol*) - Pro-FTPD 1.2.Opre10

  Proxy - Squid 2.2.STABLE5

  SMTP (*Simple Mail Transfer Protocol*) - EXIM

  TELNET - TELNETD

  WebServer - APACHE 1.3.9

A estação de gerenciamento utilizada possui as seguintes características:

- Processador *INTEL Pentium IV* 2.4 *GHz*, 512 MB de memória DDR, placa de rede 10/100 Mbps.
- Sistema operacional *Microsoft Windows XP Professional*.

Os experimentos foram realizados procurando analisar o desempenho individual de cada módulo do sistema, a fim de se ter noção mais específica do impacto de cada módulo sob o desempenho da solução como um todo.

6.3 Performance de Captura

A captura do tráfego que seria tratado foi efetuada estabelecendo-se filtros os quais limitaram a captura somente às portas envolvidas nos testes, desconsiderando todos os pacotes que não atendem à regra estipulada.

As portas definidas para os experimentos e os respectivos serviços que geralmente atendem nessas portas são representadas na Tabela 8.

Porta	**Serviço**
21	FTP
23	TELNET
25	SMTO
53	DNS
80	HTTP
8080	HTTP (alternativo)

Tabela 8: Definições de portas monitoradas durante os experimentos.

Com o objetivo de mensurar o impacto da captura de tráfego sobre a performance do servidor, foram acompanhadas através do software de monitoração *top* (BINNS; GEYER, 2004), as variações do consumo de memória e CPU em dois cenários distintos. No primeiro cenário foram submetidos 20 MB de tráfego ao servidor sem o agente de captura estar ativo, e em um segundo instante o mesmo tráfego foi gerado, porém com o sistema de captura devidamente iniciado. A comparação gráfica pode ser observada na figura 23.

Deve-se levar em conta nesta análise que além do tráfego gerado, todos os serviços mencionados anteriormente estavam devidamente iniciados e atendendo re-

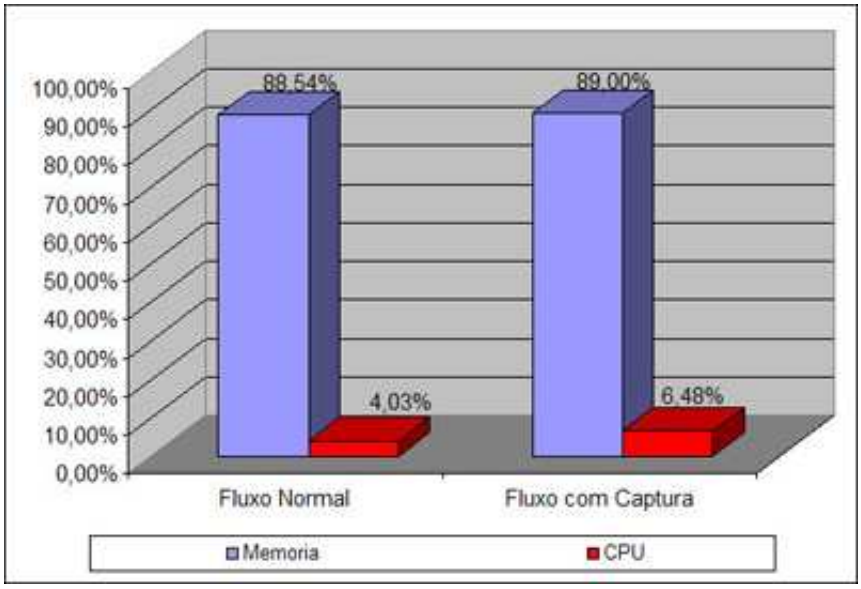


Figura 23: Gráfico do impacto de performance durante o processo de captura.

quisições, bem como o próprio sistema operacional e as respectivas ferramentas de monitoria.

Como o agente de captura foi estruturado com o uso da biblioteca *libpcap* que usa a arquitetura otimizada de filtragem BPF (MCCANNE; JACOBSON, 1993), o impacto relacionado à performance durante o monitoramento e captura das conexões não foi considerado significativo, não prejudicando portando as demais atividades do servidor.

6.4 Performance da Análise Semântica

Grande parte do esforço computacional aplicado pela estação de gerenciamento é responsabilidade do módulo de análise semântica, pois este módulo identifica todos os padrões suspeitos existentes nas seções capturadas. Além disso realiza o mapeamento dos padrões identificados em seus respectivos códigos binários, formando uma representação intermediária da análise que será subsídio para a formação do vetor de estímulos.

A performance da análise semântica está diretamente ligada ao volume de dados a serem tratados e ao conjunto de *strings* que compõe a base de conhecimentos, pois todo o tráfego capturado é comparado com cada entrada da base de conhecimentos, através do autômato finito desenvolvido com esta finalidade.

Os testes realizados com o analisador semântico indicam que deve existir uma

coerência entre o volume de dados capturados em relação ao tempo máximo de análise esperado, pois estas variáveis devem ser ajustadas para que não ocorra o monopólio dos recursos da estação em determinada análise.

6.5 Composição das Assinaturas

Para o aprendizado da rede neural é necessário um conjunto de padrões que representem ataques, bem como um conjunto de padrões correspondentes a comportamentos normais, pois estes serviriam de exemplos durante o treinamento supervisionado aplicado neste modelo.

Para compor estes padrões foram capturados separadamente dados de cada uma destas categorias. A captura do tráfego não intrusivo foi obtida através da execução e monitoria de atividades normais da grande maioria de usuários, como envio de e-mails, navegação em páginas web, conexões FTP e TELNET. Em todos os casos foi representado um comportamento normal e sem risco algum para a segurança da estrutura existente.

Os procedimentos intrusivos capturados foram desenvolvidos através de ferramentas específicas para a descoberta e exploração de vulnerabilidades, como SAINT (SAINT, 2004), SARA (CORPORATION, 2004) e NESSUS (DERAISON, 2004). Além dessas ferramentas foram utilizadas diversas técnicas reais de intrusão documentadas em relatórios e publicações especializadas, como relatórios do CERT e demais órgãos do gênero. Em complemento a estes padrões de testes aplicou-se também o conjunto de padrões cedidos pelo laboratório ACME/UNESP (CANSIAN, 2004), o qual realiza pesquisas nesta área.

O número de seções capturadas em cada porta também foi limitado de acordo com o conjunto de padrões de ataque obtidos pelas simulações realizadas. De forma a não prejudicar o treinamento (favorecendo determinado tipo de comportamento) foi fixada a mesma quantidade de padrões em cada porta monitorada para ambos os tipos de atividade.

A Tabela 9 exibe a quantidade de seções utilizadas em cada porta monitorada e sua respectiva classificação.

O conjunto de seções intrusivas compõe 105 assinaturas de intrusão, sendo esta a mesma quantidade para as seções ditas normais, portanto totalizando 210 padrões considerados válidos para o treinamento e experimentos.

Porta	Seções Intrusivas	Seções Normais	Total
21	30	30	60
23	20	20	40
25	12	12	24
53	07	07	14
80	30	30	60
8080	06	06	12
Total	105	105	210

Tabela 9: Volume de seções consideradas nos experimentos.

Estes conjuntos de padrões foram divididos em dois sub-conjuntos:

a) Treinamento: conjunto utilizado durante o treinamento supervisionado da rede neural, objetivando agregar conhecimento ao principal módulo de análise.

b) Testes: definido com a finalidade de testar o conhecimento adquirido na fase de treino. Apresentando a rede neural padrões diferentes dos exibidos na fase de treino, pode-se estimar sua capacidade de generalização.

Na tabela 10 pode-se visualizar a distribuição dos padrões entre as respectivas subdivisões, levando-se em conta que os padrões não se repetem dentro desta estrutura.

	Treinamento			***Testes***			
Porta	**Ataque**	**Normal**	**Total**	**Ataque**	**Normal**	**Total**	**Total**
21	23	23	46	07	07	14	60
23	15	15	30	05	05	10	40
25	09	09	18	03	03	06	24
53	05	05	10	02	02	04	14
80	23	23	46	07	07	14	60
8080	05	05	10	01	01	02	12
Total Geral	80	80	160	25	25	50	210

Tabela 10: Divisão do conjunto de padrões em treinamento e testes.

Levando-se em conta esta divisão foram criados dois conjuntos de dados, os quais possuem uma organização diferente entre os 210 padrões considerados nos experimentos. Estes dois conjuntos foram compostos de forma aleatória e submetidos aos mesmos experimentos, no intuito de visualizar o comportamento e desempenho da rede sobre diferentes conjuntos e situações de treinamento.

6.6 Treinamento

O treinamento da rede neural ocorre em ciclos ou épocas onde em cada época todos os 160 padrões de treino são apresentados à rede neural e a cada apresentação é calculado o erro quadratico médio, através da diferença entre o resultado apresentado pela rede neural e o valor esperado para determinado padrão.

Os pesos das conexões sinápticas foram inicializados com valores aleatórios entre -1 e 1, os quais seriam ajustados no decorrer do treinamento.

No intuito de definir a melhor configuração para a taxa de aprendizado foram realizados experimentos variando seu valor, procurando observar o efeito desta variação sobre a convergência da rede. Os dois conjuntos de treinamento formados pelos 160 padrões devidamente categorizados foram submetidos às mesmas variações de taxa de aprendizado e à mesma quantidade de 350 épocas, possibilitando uma comparação entre os conjuntos e uma amostra do efeito da variação desta taxa sobre o treinamento.

As taxas foram fixadas em 0.05, 0.08 e 0.1, sendo que a configuração de 0.05 foi a que apresentou melhores resultados ao se levar em conta os dois conjuntos. Portanto, esta taxa de aprendizado foi fixada em todos os demais experimentos realizados.

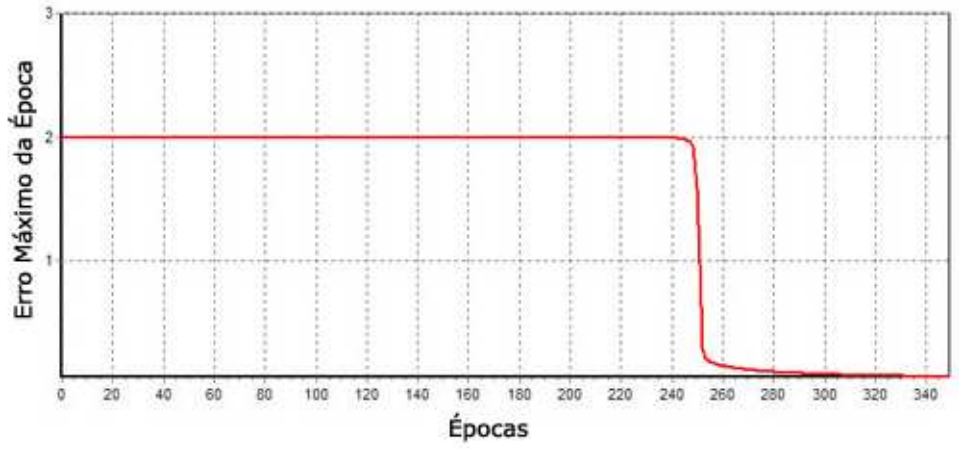


Figura 24: Variação do erro máximo por época durante o treinamento do conjunto 1 com taxa de aprendizado de 0.05.

À medida que o treinamento se desenvolve o erro máximo apresentado em cada época tende a diminuir estabilizando em um ponto de convergência específico. A variação deste erro à medida que as épocas foram desenvolvidas deu origem aos gráficos ilustrados nas Figuras 24 e 25 que representam o treinamento do conjunto

1 e do conjunto 2 respectivamente.

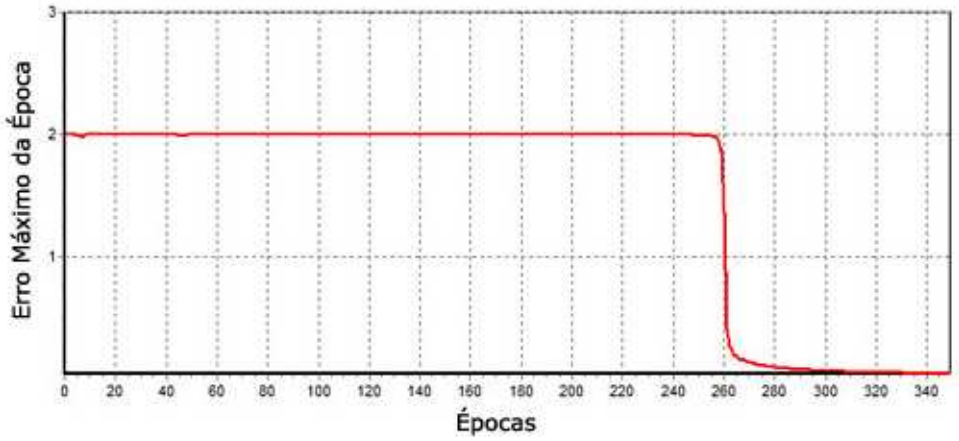


Figura 25: Variação do erro máximo por época durante o treinamento do conjunto 2 com taxa de aprendizado de 0.05.

Pode-se observar nos gráficos que o comportamento de ambos os conjuntos durante o treinamento foi bastante similar. Esta situação ocorreu, pois os 160 padrões que compõem cada conjunto foram escolhidos aleatoriamente a partir do mesmo conjunto total de padrões iniciais. Nota-se ainda que durante os experimentos, o conjunto 1 iniciou o processo de convergência em um número menor de épocas em relação ao conjunto 2.

Como a função de ativação tangente hiperbólica utilizada no protótipo opera em um limiar de -1 à 1, o erro máximo de cada época que a saída poderia apresentar é 2, sendo que este erro tende a diminuir com o desenvolvimento do treinamento, conforme pode ser observado nas Figuras 24 e 25.

O erro máximo obtido durante o treinamento ao final das 350 épocas com relação aos experimentos do conjunto 1 foi de 0,0652, sendo que para o conjunto 2 este valor foi de 0,0511. Estes valores de erro são efetivamente baixos, pois representam validações realizadas sobre o próprio conjunto de treino.

A resposta produzida pela rede neural para avaliação dos padrões de testes possui maior relevância ao mensurar o desempenho da rede, do que a apresentação dos próprios padrões utilizados no treinamento. Dessa forma é possível avaliar a capacidade de generalização e adaptação do modelo baseado nas respostas geradas, obtendo-se resultados que retratam o comportamento da rede ao tratar casos não vistos durante fase de aprendizado.

6.7 Processo de Validação

O processo de validação da rede neural é utilizado para acompanhar o aprendizado da rede à medida que o treinamento é realizado.

Durante a validação os padrões de testes são submetidos periodicamente para avaliação neural e a cada apresentação o erro quadrático médio é calculado, proporcionando uma estimativa do desempenho da rede treinada.

Para que o processo de aprendizado evolua, é necessário que ambos os erros de treinamento e validação decaiam com o passar das épocas.

Nos experimentos realizados neste trabalho o processo de validação não contribui efetivamente com o treinamento, pois nenhuma alteração nos pesos da rede é realizada baseado no erro de validação. Portanto, trata-se apenas de um índice que permite estimar o desempenho da rede treinada quando a novos padrões são apresentados.

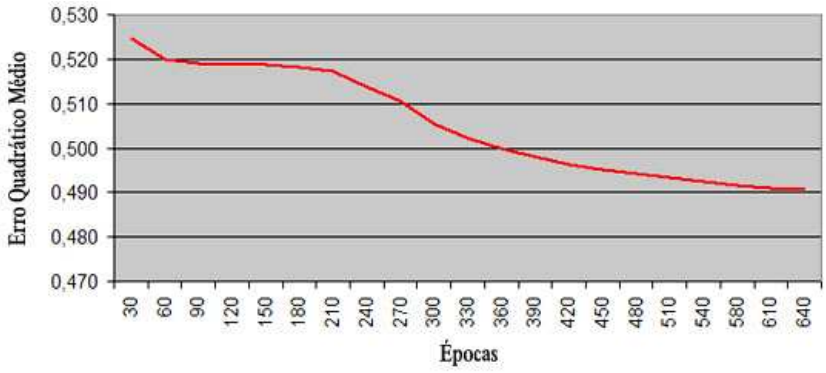


Figura 26: Gráfico de variação do erro quadrático médio por época durante a validação do conjunto 1.

A validação foi realizada nos dois conjuntos a cada 30 épocas durante um período de 640 épocas para que fosse possível acompanhar a variação do erro quadrático médio após a convergência da rede.

No conjunto 1 durante a validação observa-se que o erro quadrático médio tende a cair durante o ponto de convergência do treinamento, estabilizando e não sofrendo reduções consideráveis a partir deste ponto, conforme representado no gráfico da Figura 26 que ilustra os dados disponíveis na Tabela 11.

No conjunto 2 observa-se que apesar do treinamento apresentar um ponto de convergência a um erro mínimo, isto não ocorre com o erro de validação que cresce com o decorrer das épocas e caindo levemente após o período de convergência, con-

Conjunto 1		
Épocas	**Erro Quadrático Médio**	**Erro de Classificação**
30	0,524612016824972	26,23 %
60	0,520032056320175	26,00 %
90	0,519180553607711	25,96 %
120	0,518905378876376	25,95 %
150	0,518656819710304	25,93 %
180	0,518215482134226	25,91 %
210	0,517251937435471	25,86 %
240	0,513648796453054	25,68 %
270	0,510556685067159	25,53 %
300	0,505565085642267	25,28 %
330	0,502232385100059	24,99 %
360	0,499809920817359	24,90 %
390	0,49794091016853	24,82 %
420	0,496447492512702	24,76 %
450	0,495215808583849	24,71 %
480	0,494179142597000	24,66 %
510	0,493299741847191	24,63 %
540	0,492534621100885	24,58 %
570	0,491665214770662	24,55 %
600	0,491093786276127	24,54 %
630	0,490587114520757	24,52 %
Desvio Padrão	**0,0119620641474335**	

Tabela 11: Variação do erro quadrático médio por época durante a validação do conjunto 1.

forme ilustrado na Figura 27 que representa de forma gráfica os dados dispostos na Tabela 12. Esta situação indica que a capacidade de generalização da rede neural está sendo degradada e é geralmente chamada de *over-fitting* ou *over-trainning*.

Análises realizadas sobre o conjunto 2 de padrões, indicam que esta variação indesejável do erro sobre o conjunto de validação foi ocasionada devido a maior ocorrência de determinado tipo de padrão e suas variantes no conjunto de treino, o que levou a rede a obter resultados tendenciosos. Este fato foi ocasionado devido à característica aleatória da formação dos conjuntos, não sendo manipulados os padrões de treinamento ou testes.

Este tipo de formação aleatória dos conjuntos utilizados nos experimentos, foi assim definida para não interferir de forma proposital no processo de generalização, o que poderia levar a resultados distorcidos ou manipulados sobre a estrutura em questão.

Esta situação comprova a importância da correta formação dos conjuntos de padrões destinados ao treinamento da rede neural, pois o processo de generalização depende principalmente do treinamento adequado.

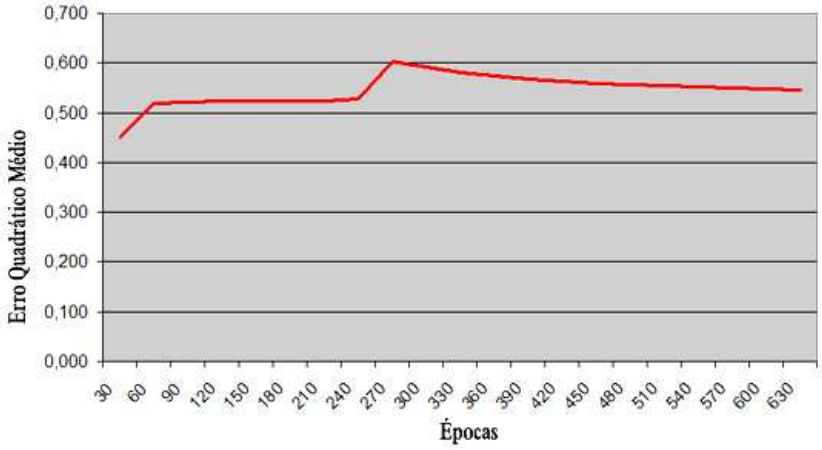


Figura 27: Variação do erro quadrático médio por época durante a validação do conjunto 2.

Conjunto 2		
Épocas	**Erro Quadrático Médio**	**Erro de Classificação**
30	0,450621587585267	22,53 %
60	0,518283530203774	25,91 %
90	0,522159370351231	26,11 %
120	0,522296552544275	26,11 %
150	0,522248177327997	26,11 %
180	0,522523265616999	26,13 %
210	0,523570194750481	26,18 %
240	0,527410789635102	26,37 %
270	0,603334200426312	30,17 %
300	0,591293912379487	29,56 %
330	0,581274913310486	29,06 %
360	0,573882182221454	28,69 %
390	0,568146148448215	28,41 %
420	0,563546743702423	28,18 %
450	0,559745829860524	27,99 %
480	0,556556040082241	27,83 %
510	0,553828490611322	27,69 %
540	0,551492896048151	27,57 %
570	0,549428979831774	27,47 %
600	0,547614854797835	27,38 %
630	0,54599591392121	27,30 %
Desvio Padrão	**0,0328016474054302**	

Tabela 12: Variação do erro quadrático médio por época durante a validação do conjunto 2.

6.8 Resultados dos Experimentos

Após o devido treinamento a rede neural é considerada apta a reconhecer e classificar corretamente qualquer padrão similar aos contidos na base de treino.

Ao submetermos para análise o mesmo conjunto de padrões usados durante o treinamento a rede neural atingiu 100% de acertos e um erro quadrático médio insignificante, isso ocorreu porque ela foi condicionada a reconhecer estes padrões.

Para efetivamente testar a eficiência e a capacidade de generalização da estrutura, a rede neural foi devidamente treinada com o respectivo conjunto de treino, e os resultados da análise neural foram mensurados a partir do conjunto de testes, que foi formado por padrões não apresentados à rede neural durante o treinamento conforme disposto na Seção 6.3. Dessa forma os valores determinados pela rede neural podem ser considerados inferências realizadas a partir do processo de aprendizagem.

Na Tabela 13 está representado um dos padrões de ataque analisados, o qual não estava presente durante o treinamento e que, no entanto, foi classificado corretamente pela rede neural, pois o índice informado em sua saída estava mais próximo de 1 que de -1.

Padrão de intrusão:
#TCP
#25
#EXPN
#decode
#EXPN
#uudecode
#tail
#cat > /tmp
#bad
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 1 1 0 0 0 0 0 1 1 0 1 1 0 1 0
0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 1
0 1 1 0 0 0 0 0 1 1 0 1 1 0 1 0
0 0 0 0 1 0 1 1 0 1 0 1 0 0 1 1
0 0 0 0 1 1 0 0 1 1 0 1 1 1 1 1
0 1 1 0 0 0 0 0 1 1 1 0 0 1 0 1
0 1 1 0 0 1 1 0 1 0 1 0 1 1 1 0
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Saída Desejada: 1
Saída Informada: 0,887257993221283

Tabela 13: Padrão de intrusão analisado pela RNA.

No caso específico deste padrão o erro cometido pela rede neural seria de 0,11274 que foi obtido subtraindo a saída desejada pela saída da rede, uma classificação realizada com 94,36% de certeza que se tratava de uma intrusão.

Ao apresentarmos para a rede devidamente treinada um padrão de não ataque que não constava no conjunto de treino, observa-se um bom índice de classificação, conforme representado na Tabela 14.

Padrão de não intrusão:
#TCP
#21
#PASS
#logged in
#PWD
#command successful
#LIST
#command successful
#RETR
#QUIT
0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 0
0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0
0 1 0 1 0 1 1 0 1 1 1 0 0 0 1 1
0 1 1 0 0 1 1 1 0 1 1 0 1 0 0 1
0 1 1 0 0 1 1 0 1 0 0 1 0 0 0 1
0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0
0 1 1 0 0 1 1 0 1 0 0 1 0 0 0 1
0 1 1 0 1 0 1 1 1 0 0 0 1 0 0 0
0 1 1 0 0 0 0 1 0 0 0 1 1 1 0 1
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Saída Desejada: -1
Saída Informada: -0,592655003070831

Tabela 14: Padrão de não intrusão analisado pela RNA.

O erro cometido para este padrão é de -0,40734 uma classificação realizada com 79,63% de certeza que o padrão não se tratava de uma intrusão.

Levando-se em conta todos os padrões dos conjuntos de testes, nota-se que o erro quadrático médio obtido por meio dos experimentos possui uma escala de valores que variam de 0 a 2 para cada padrão de treinamento, pois os valores de saída da rede neural implementada podem variar de -1 a 1.

Porém em aplicações de redes neurais que utilizam a função de ativação sigmóide do tipo logística, este limiar varia apenas de 0 a 1, o que apresentaria menores índices de erro quadrático médio para os mesmos experimentos.

No intuito de melhorar a visão sobre o desempenho da rede em comparação com outras implementações de redes neurais, os erros quadráticos médios obtidos para ambos os conjuntos de padrões estão sendo apresentados na tabela 15 em uma escala percentual, onde pode-se observar a diferença entre os conjuntos devido as características individuais de composição dos padrões em cada caso.

Conjunto 1	**Conjunto 2**	**Média**
24,52 %	27,29 %	25,91 %

Tabela 15: Erros de classificação em escala percentual após o treinamento da rede.

Em função da formação dos padrões o conjunto 1 teve resultado mais satisfatório em relação ao outro conjunto, demonstrando que a composição do padrão de treino tem papel fundamental sobre o desempenho da rede neural. O gráfico representado na Figura 28 ilustra a diferença entre os erros de classificação entre os dois conjuntos analisados.

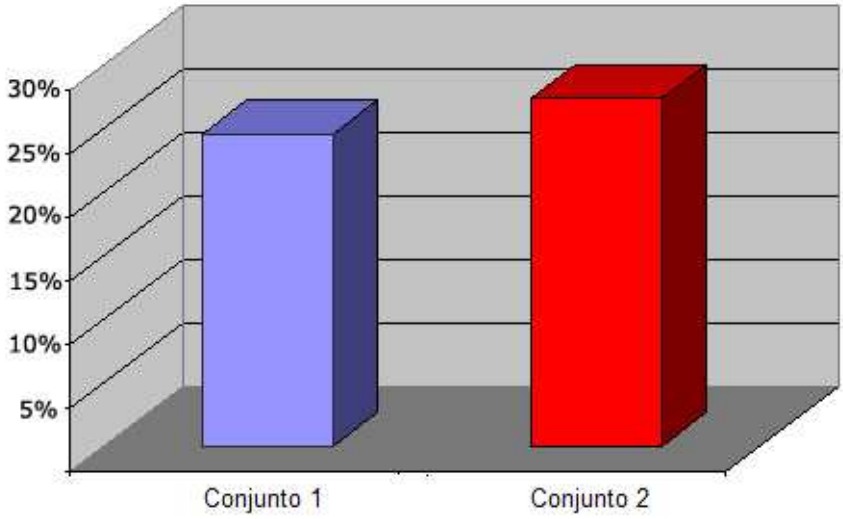


Figura 28: Gráfico de erro quadrático médio da rede em escala de 0 a 1 para cada conjunto de testes.

Conforme a média de erro de aproximadamente 26% disposto na Tabela 15, nota-se que a rede neural classificou corretamente cerca de 74% dos padrões apresentados durante a fase de execução. Este índice de resultado é satisfatório, pois reflete o conhecimento obtido através de um treinamento onde não houve nenhum tipo de critério ao pré-selecionar os conjuntos de treino em relação ao conjunto de testes.

Dessa forma a rede neural é capaz de reconhecer novamente qualquer padrão já visto durante o treinamento, e ainda classificar de forma correta qualquer tipo de variação destes padrões.

Os erros quadráticos médios informados levam em consideração o valor desejado em relação ao valor informado pela rede neural, porém os erros ou acertos reais cometidos pela rede neural podem variar, dependendo do valor fixado para o que seria ou não um comportamento intrusivo, pois em todas as situações de análise, sejam elas acertos ou erros, existem taxas divergentes entre o padrão informado e o padrão esperado.

Conforme discutido no *Capítulo 4*, ao considerarmos este índice em 0, todos valores de saída dentro da faixa de 0 a 1 seriam considerados intrusivos e de -1 a 0 comportamentos normais. Dessa forma determinado padrão pode apresentar um erro de 0,9 e ainda assim ser considerado um acerto.

Portanto, pode-se considerar este índice como uma variável de ajuste para falsos positivos e falsos negativos, onde quanto maior for este valor menos ocorrências de comportamentos intrusivos o sistema reportaria.

7 CONCLUSÃO E TRABALHOS FUTUROS

As funcionalidades de reconhecimento de padrões proporcionadas pelo uso de redes neurais artificiais permitiram a definição de um modelo simplificado de detecção de intrusão em redes de computadores, o qual atende a algumas carências de outras abordagens utilizadas.

A definição deste modelo e a respectiva implementação de um protótipo que atenda suas especificações foram contempladas neste trabalho. Esta pesquisa possibilitou mensurar resultados positivos, os quais contribuem para o desenvolvimento de técnicas de segurança.

7.1 Conclusão Geral

A capacidade de generalização e adaptação obtida pela rede neural incorporada ao módulo de análise proporcionou vantagens significativas como a possibilidade de detectar com índices satisfatórios ações não previstas durante a fase de treinamento, conforme apresentado pelos experimentos realizados. Este recurso agrega sensibilidade ao processo de detecção, pois atacantes que tentem usar variações de ataques conhecidos seriam detectados.

A aquisição de conhecimento sobre novos comportamentos intrusivos ou normais pode ser facilmente obtida através da simulação e captura desse comportamento, incorporando-o ao conjunto de treino antes de realizar o re-treinamento da rede neural.

A manutenção da base de conhecimentos que é apoiada pelo gerenciador binário de assinaturas trata-se de processo simples e que colabora com as características adaptativas e com a incorporação de novas capacidades ao sistema.

Devido ao fato do conhecimento ser adquirido durante a fase de aprendizado e

todas as análises passarem a ser realizadas a partir deste conhecimento adquirido, não se faz necessário dispensar recursos computacionais com armazenamento, tratamento e padronizações de bases de regras, como é o caso de sistemas de detecção de intrusão convencionais. Sendo assim, após a obtenção de uma rede neural treinada adequadamente, seu conhecimento pode ser replicado de forma simples para outras estruturas do mesmo tipo.

Como a definição do modelo foi realizada em módulos independentes que podem residir em diferentes ambientes computacionais, é possível que o administrador posicione os sensores em locais estratégicos de acordo com a necessidade, e ainda centralizar todos os eventos reportados por estes em uma estação de gerenciamento, a qual seria responsável pelo processo de análise e resposta passiva, portanto não interferindo na performance da rede ou estações monitoradas.

O ajuste de falsos positivos e falsos negativos proporcionado pela variação do índice de aceitação na saída da rede neural é uma importante característica do modelo proposto, pois flexibiliza ao administrador configurar o sistema para atender as necessidades do ambiente onde está sendo aplicado.

7.2 Trabalhos Futuros

A abordagem utilizada agrega diversas vantagens ao modelo, porém o protótipo implementado apresenta algumas dificuldades não tratadas neste trabalho e que são sugestões de trabalhos futuros, em pesquisas que venham a otimizar a solução. Entre elas destacam-se:

- Inviabilidade de captura de dados em ambientes criptografados, pois a biblioteca e funcionalidades implementadas não permitem decifrar o conteúdo deste tipo de seção.
- O modo promíscuo da interface de rede que é ativado pelo módulo de captura necessita que o segmento monitorado opere com broadcast. Caso contrário, existe certa limitação na abrangência da monitoração e os sensores devem ser estrategicamente posicionados para minimizar estes impactos.
- A comunicação entre o módulo de captura e a estação de gerenciamento necessita ser otimizada, de forma a agregar funcionalidades gerenciais ao protótipo.
- Todas as seções capturadas são repassadas para análise da rede neural sem existir qualquer pré-seleção entre os eventos. Adicionar métricas que controlem

o nível de suspeita de cada seção diminuiria o esforço e o tempo de análise.

- A reatividade do sistema é composta apenas por respostas passivas, sendo que em situações críticas haveria a necessidade que o sistema respondesse de forma ativa a eventos intrusivos.

- A arquitetura baseada exclusivamente em eventos da rede impossibilita a detecção de intrusões que não utilizem este recurso, como é o caso de eventos locais realizados por usuários válidos ou atacantes com acesso físico ao sistema.

7.3 Conclusão Final

Por meio deste trabalho pôde-se comprovar que técnicas de inteligência artificial podem contribuir efetivamente para otimização de mecanismos aplicados à segurança como os sistemas de detecção de intrusão.

Os experimentos realizados comprovam a eficácia desta abordagem, pois demonstram que o processo de generalização contribui significativamente para a adaptação do modelo a variações nos padrões de ataque. Além disso a flexibilidade de configuração e as características de aquisição de conhecimento bem como as formas de representá-lo contribuem positivamente com o desenvolvimento dessa abordagem.

Acredita-se que estas funcionalidades aliadas a outras técnicas existentes possam compor soluções mais eficientes e robustas no tratamento de violações à segurança de computadores.

REFERÊNCIAS

ABNT. **Tecnologia da Informação - Código de prática para a gestão de seguranças da informação**. NBR ISO/IEC 17799, 2002. ISBN 85-07-00214-5. Disponível em: <Associação Brasileira de Normas Técnicas>.

AXELSSON, S. **Intrusion Detection Systems: A survey and Taxonomy**. [S.l.]: New Riders, 2000.

BARBOSA, A. S.; MORAES, L. F. **Sistema de Detecção de Intrusão**. Rio de Janeiro: [s.n.], 2000. Seminários Ravel - CPS760: Laboratório de Redes de Alta Velocidade, UFRJ.

BARRETO, J. M. **Inteligência Artificial no Limiar do Século XXI**. [S.l.]: Duplic Edições, 1999.

BERNARDES, M. C. **Avaliação do Uso de Agentes Móveis em Segurança Computacional**. Dissertação (Mestrado) — Instituto de Ciências Matemáticas e de Computação (ICMC - USP), São Carlos, SP, 1999.

BINNS, R.; GEYER, H. **TOP monitor Software**. Linux Man page (man top), Outubro 2004. Acessado em 08/09/2004. Disponível em: <http://www.hmug.org/man/1/top.html>.

CAMPELLO, R. S.; WEBER, R. F. Sistemas de detecção de intrusão. In: **Anais do Simpósio Brasileiro de Redes de Computadores**. [S.l.: s.n.], 2001. Instituto de Informática UFRGS.

CANNADY, J. **Artificial Neural Networks to Misuse Detection**. 1998. Disponível em: <http://csrc.nist.gov/nissc/1998/proceedings/paperF13.pdf>.

CANSIAN, A. M. **Desenvolvimento de um Sistema Adaptativo de Detecção de Intrusos em Redes de Computadores**. Tese (Tese de Doutorado) — Instituto de Física de São Carlos - Universidade de São Paulo, Novembro 1997.

CANSIAN, A. M. **Padrões Intrusivos e Não Intrusivos**. Laboratório ACME de Pesquisa em Segurança de Computadores e Redes, Outubro 2004. UNESP - Universidade Estadual Paulista - Campus de São José do Rio Preto. Disponível em: <http://www.unesp.br/>.

CASTRO, L. N. de. **Análise e Síntese de Estratégias de Aprendizado para Redes Neurais Artificiais**. Dissertação (Mestrado) — Universidade Estadual de Campinas - UNICAMP, Campinas, SP, Setembro 1998.

CERT/CC. **CERT/CC Statistics 1988-2004**. Computer Emergency Response Team (Coordenation Center), Outubro 2004. Acessado em 08/02/2004. Disponível em: <http://www.cert.org/stats/cert_stats.html/>.

CORPORATION, A. R. **SARA - Security Auditor's Research Assistant**. Advanced Research Corporation, January 2004. Acessado em 08/08/2004. Disponível em: <http://www-arc.com/sara/>.

CROSBIE, M.; SPAFFORD, G. Defending a computer system using autonomous agents. **West Lafayetee: Dept. of Computer Sciences**, West Lafayetee, Purdue University, p. 12–16, 1995.

DALTON, A. D. J. Artificial neural networks - an approach to increasing machine intelligence. **IEEE POTENTIALS**, April 1991, v. 01, p. 33–36, 2000.

DANH, N. T.; PHIEN, H. N.; GUPTA, A. D. Neural network models for river flow forecasting. **Asian institute of Techonology**, Thailand, Pahtumthani- Thailand, 1999.

DASGUPTA, D.; BRIAN, H. Tmobile security agents for network traffic analysis. **DARPA Information Survivability Conference and Exposition**, June 2001, v. 01, p. 1332, 2001.

DERAISON, R. **NESSUS Project**. Project founder and current leader, January 2004. Acessado em 08/07/2004. Disponível em: <http://www.nessus.org>.

ELSON, J. **tcpflow - TCP Flow recorder**. TCP DUMP Research Team, January 2001. Acessado em 08/05/2004. Disponível em: <http://www.circlemud.org/ jelson/software/tcpflow/>.

FABRO, J. A. **Uma abordagem neuro-nebulosa para controle preditivo de processos nebulos multi-estágios**. Tese (Tese de Doutorado) — CEFET-PR CPGEI, Fevereiro 2003.

FAUSETT, L. **Fundamentals of Neural Networks: architectures, algorithms and applications**. [S.l.]: Prentice-Hall, 1994.

FISCHER, I. et al. **Java Neural Network Simulator User Manual, Version 1.1**. [S.l.]: Wilhelm-Schickard-Institute for Computer Science, Outubro 2001. University Of Tubingen.

GEORGIA, J. C. **Next Generation Intrusion Detection: Autonomous Reinforcement Learning of Network Attacks**. 2000. Disponível em: <citeseer.ist.psu.edu/649594.html>.

HAYKIN, S. **Redes Neurais - Princípios e Práticas**. [S.l.]: Bookman, 2001. Segunda Edição.

HOWARD, J. D.; LONGSTAFF, T. **A Common Language for Computer Security Incidents**. Livermore, CA: Sandia Nacional Laboratories, 1998. Acessado em 17/11/2004.

JOO, D.; HONG, T.; HAN, I. The neural network model for ids based on the asyimmetric costs of false negatives erros and false positivies erros. **Expert System with applications**, April 1991, v. 01, p. 69–75, 2003.

KANDEL, E. R.; SCHWARTZ, J. H.; JESSEL, T. M. **Essentials of Neural Science and Behaviour**. [S.l.]: Appleton Lange, 1995.

KUMAR, S. **Classification and Detection of Computer Intrusions**. Tese (Doutorado) — Purdue University, W, Purdue, IN, 1995. Acessado em 20/11/2004. Disponível em: <citeseer.ist.psu.edu/kumar95classification.html>.

LENT, R. **Cem bilhões de neurônios: Conceitos fundamentais de neurociência**. [S.l.]: Editora Atheneu, 2001.

MACHADO, A. B. M. **Neuroanatomia Funcional**. [S.l.]: Atheneu, 2003. Segunda edição - São Paulo-SP.

MARKOU, M.; SINGH, S. **Novelty Detection: A Review**. 2003. Disponível em: <citeseer.ist.psu.edu/668059.html>.

MARTIN, J. H. **Neuroanatomia: Texto e Atlas**. [S.l.]: Editora Artes Médicas Sul Ltda, 1998.

MCCANNE, S.; CRAIG, V.; JACOBSON. **PCAP**. Lawrence Berkeley National Laboratory, November 2002. University of California. Disponível em: <http://www.tcpdump.org>.

MCCANNE, S.; JACOBSON, V. The bsd packet filter: A new architecture for user-level packet capture. **USENIX Technical Conference**, USENIX Technical Conference, San Diego. CA. USA, January, v. 01, 1993.

MIIKA, T. Intrusion detection systems. **Seminar on Distributed Security**, Department of Computer Science : University of Helsinki, v. 01, n. 6, 2000.

NORTHCUTT, S. et al. **Intrusion Signatures and Analysis**. New Jersey: New Riders, 2001.

PTACEK, T. H.; NEWSHAM, T. N. Insertion, evasion, and denial of service: Eluding network intrusion detection. In: **Magazine**. [S.l.: s.n.], 1998. Secure Networks Inc.

RAUBER, T. W. Redes neurais artificiais. **Encontro Regional de Informática**, Sociedade Brasileira de Computação, Nova Friburgo-RJ e Vitória-ES, v. 01, p. 201–228, 1998.

RIPLEY, B. D. **Pattern Recognition and Neural Networks**. [S.l.]: Cambridge University Press, 1996.

RYAN, J.; LIN, M.-J.; MIIKKULAINEN, R. Intrusion detection with neural networks. In: JORDAN, M. I.; KEARNS, M. J.; SOLLA, S. A. (Ed.). **Advances in Neural Information Processing Systems**. The MIT Press, 1998. v. 10. Disponível em: <citeseer.ist.psu.edu/ryan98intrusion.html>.

SAINT. **SAINT - Security Administrator's Integrated Network Tool**. SAINT, January 2004. Acessado em 12/08/2004. Disponível em: <http://www.saintcorporation.com>.

SHEPHERD, G.; KOCH, C. Introduction to synaptic circuits. **The Synaptic Organization of the Brain**, New Yourk Oxfor University Press, p. 03–31, 1990.

SMAHA, S. E. Haystack: An intrusion detection system. **roceedings of the Fourth Aerospace Computer Security Applications Conference**, Washington, DC: IEEE Computer Society Press, Orlando, Florida, v. 01, n. 6, p. 12–16, 1989.

STAFF, C. **Overview of Attack Trends**. Computer Emergency Response Team (Coordenation Center), November 2002. Carnegie Mellon University. Disponível em: <http://www.cert.org/>.

STALLINGS, W. **Cryptography and Network Security: Principles and Practice**. 3th editon. ed. New Jersey: Prentice Hall, 2003.

STANIFORD-CHEN, S. **Common Intrusion Detection Framework(CIDF)**. Computer Emergency Response Team (Coordenation Center), Outubro 1998. Acessado em 08/02/2004. Disponível em: <http://seclab.cs.ucdavis.edu/cidf/>.

WAGNER, M. M. B. Understanding neural networks as statisticals tools. In: **Department of Preventive Medicine and Biometrics**. [S.l.: s.n.], 1996. Department of Preventive Medicine and Biometrics, Depatment of Mathematical and Computer Sciences - University of Colorado.

WOOD, M. Intrusion detection message exchange requirements. **Intrusion Detection Working Group**, Harvey Mudd College, http://www.ietf.org/html.charters/idwg-charter.html, p. 01–18, 2003.

APÊNDICE A – PALAVRAS CHAVES DA LISTA DA BASE DE CONHECIMENTO

A.1 Arquivo LBC.LBC

1 USER admin
1 USER apply
1 USER auditor
1 USER adm
1 USER access
1 USER account
1 USER info
1 USER lib
1 USER local
1 USER sys
1 USER tests
1 USER server
1 USER supervisor
1 USER school
1 USER slip
1 USER sync
1 USER project
1 USER remote
1 USER www-data
1 USER manager
1 USER newuser
1 USER telnet
1 USER toor
1 USER tty
1 USER visitor
1 USER guest
1 USER help
1 USER daemon
1 USER ftp

1 USER bin
1 USER demos
1 USER backup
1 USER application
1 USER irc
1 USER 4DGifts
1 USER +
2 Croot
3 sleep 2 ; echo quit
3 sleep 2 ;echo quit
4 su
5 command unrecognized
5 Command unrecognized
5 Invalid command
6 Permission denied
6 permission denied
7 uudecode
7 UUDECODE
7 decode
7 DECODE
8 User Unknown
8 user unknown
9 tail
9 vi
9 less
9 vim
9 emacs
9 joe
10 Invalid sender address
11 Need MAIL before RCPT
12 Need MAIL command
13 syntax error
13 Syntax Error
14 Malformed from address
15 chsh
16 access denied
16 Login incorrect
16 command not understood
17 debug
17 Debug
17 DEBUG
18 Login with USER first
19 /bin/mail
20 Content-Disposition
20 .pif
20 .shs
20 .exe

20 .vbs
20 .bat
20 .scr
20 .hsq
21 Invalid request
21 Invalid Request
22 /etc/passwd
22 /etc/shadow
22 /etc/service
22 /etc/inet.d
23 /etc/aliases
23 /etc/inetd.conf
23 /.rhosts
23 /etc/network/interfaces
23 /etc/fstab
23 /etc/services
23 /dev/null
23 /etc/hosts
23 /etc/group
23 /winnt/win.ini
23 /users.pwd
24 GET /n0nexi5tent$_{f}i1e.html$
24 GET /$_{v}ti_{p}vt/administrators.pwd$
24 GET /exec/show/config/cr
25 /%2E%2E/%E%2E/%2E%2E/%2E%2E/%2E%2E/
25 /scripts/..
26 gcc
26 g++
26 make
27 GET /scripts/cart32.exe/cart32clientlist
28 cd /tmp
29 mkdir
29 rmdir
29 wget
30 Entering Passive Mode
31 scanning "postmaster"
32 lpr
32 lpd
33 chattr
33 fuser
34 logged in
35 password file
35 PASSWORD FILE
35 ALIASES
35 INETD.CONF
35 .RHOSTS
35 SKEY

35 HOSTS.EQUIV
35 SERVICES
35 HOSTS
35 UUCP
35 SHADOW
35 SYSTEM
35 INTERFACE
35 ROUTE
36 administrador
36 administrator
36 ADMIN
36 ADMINISTRATOR
36 root
36 ROOT
37 Sorry
37 sorry
38 chmod 777
38 chmod 666
38 chmod a+rwx
38 chmod ugo+rwx
39 rewt
39 wank
40 chown
41 Not owner
42 can't change
42 can not change
43 /bin/sh
44 |mail
44 | mail
44 |/bin/mail
44 |/bin/echo
45 VRFY
45 vrfy
45 EXPN
46 cat > /tmp
47 QUIT
48 nobody
49 command successful
49 Command successful
50 bad
50 Bad
51 ls -la
51 cp
51 mv
51 rm
52 NetBus
52 GetInfo|0d|

52 PWD
52 Ahhhh My Mouth Is Open
52 Wtzup Use
52 Connected.
52 Remote|3A| You are connected to me.
52 WHATISIT
52 NetSphere
52 GateCrasher
52 c|3A|
52 pINg
52 r00t
52 backdoor
52 wh00t!
52 StoogR
52 lrkr0x
52 connected. time/date
53 Content-Type
53 CAL
53 USR
53 NICK
53 PRIVMSG
53 JOIN
54 AAAAAAAAAA
54 *HELLO*
54 betaalmostdone
54 killme
54 l44adsl
54 alive tijgu
54 sicken
54 ficken
54 spoofworks
54 gesundheit!
54 skillz
54 niggahbitch
54 skillz
55 CWD
55 SITE
55 PASS
55 RMDIR
55 REST
55 DELETE
55 MODE
55 STAT
55 SAINT
55 root
55 LIST
55 RETR

56 AUTHENTICATE
56 AUTH
56 RENAME
56 FIND
56 PARTIAL
57 |show databases
58 |ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff ff|
58 |90 90 90 90 90 90 90 90 90|
58 |d840 cd80 e8d9 ffff ff|/bin/sh
58 |23|list
58 aim $AddExternalApp$?
58 aim $AddGame$?
58 NOTIFY *
58 6ISS ECNRA Built-In
59 User-Agent $Quicktime$
59 Content-type $audio/x-ms-wma$
59 Content-type $video/x-ms-asf$
59 Content-type $audio/x-scpls$
59 Content-type $audio/x-mpegurl$
60 DELE
60 UIDL
60 AUTH
60 XTDN
60 APOP
60 XTND
60 RSET
60 FOLD
61 charset = |22 22|
61 expn decode
61 expn root
61 expn *@
61 mail from|
61 ETRN
61 ehlo cybercop
61 Content-Transfer-Encoding
62 Content-Disposition
62 phpbb$_{r}oot_{p}ath$ =
62 file=http //
62 ForumLang=../
62 PHP$_{A}UTH_{U}SER = boogieman$
62 password=12345
62 password=admin
62 Server[path]=http
62 |ba49feffff f7d2 b9bfffffff f7d1|
62 new XMLHttpRequest
63 GET /cgi-bin/webgais
63 GET /cgi-bin/csh

63 GET /cgi-bin/ksh
63 GET /cgi-bin/wwwboard
63 GET /cgi-bin/test-cgi
63 GET /cgi-bin/tcsh
63 GET /cgi-bin/guestbook.pl
63 GET /cgi-bin/info2www
63 GET /cgi-bin/wais.pl
63 GET /cgi-bin/db2www.exe
63 GET /cgi-bin/webplus.exe
63 GET /cgi-bin/finger.cgi
63 GET /cgi-bin/statsconfig.pl
63 GET /cgi-shl/win-c-sample.exe
63 GET /cgi-bin/guestbook.cgi
63 GET /cgi-bin/console.exe
63 GET /cgi-bin/query?mss=../config
63 GET /cgi-dos/args.cmd
63 GET /cgi-bin/mmstdod.cgi?ALTERNATE$_{T}EMPLATES$ =
63 GET /cgi-bin/saint$_{t}est_{c}gi$
63 GET /cgi-bin/infosrch.cgi?cmd=getdocdb=manfname=
63 GET /cgi-bin/webplus.cgi
63 GET /cgi-bin/ncommerce3/ExecMacro/orderdspc.d2w/report
63 GET /cgi-bin/w3-msql/index.html
63 GET /cgi-bin/wwwboard.pl
63 GET /cgi-bin/db2www
63 GET /cgi-bin/perl.exe
63 GET /cgi-bin/ash
63 GET /cgi-bin/jj
63 GET /cgi-bin/post-query
63 GET /cgi-bin/aglimpse
63 GET /cgi-bin/handler
63 GET /cgi-bin/webplus
63 GET /cgi-bin/finger.pl
63 GET /cgi-bin/htsearch?-c/n0nexi5tent$_{f}1le$
63 GET /cgi-bin/nph-test-cgi
63 GET /cgi-bin/wrap
63 GET /cgi-dos/args.bat
63 GET /cgi-bin/perl
63 GET /cgi-bin/zsh
63 GET /cgi-bin/websendmail
63 GET /cgi-bin/pi
63 GET /cgi-bin/glimpse
63 GET /cgi-bin/phf
63 GET /cgi-bin/finger
63 GET /cgi-bin/wais/wais.pl
63 GET /cgi-bin/excite
63 GET /cgi-bin/dumpenv.pl
63 GET /cgi-win/uploader.exe

63 GET /cgi-bin/sh
63 GET /cgi-bin/bash
63 GET /cgi-bin/imagemap.exe
63 GET /cgi-bin/websync.exe
63 GET /cgi-bin/www-sql
63 GET /cgi-bin/wwwwais
63 GET /cgi-bin/textcounter.pl
64 PUT /cgi-bin/saint.txt HTTP/1.0.Content-Length:
65 GET /scripts/cart32.exe/cart32clientlist
65 GET /scripts/..
65 GET /scripts/emurl/RECMAN.dll
65 GET /scripts/..
65 GET /scripts/..
65 GET /scripts/root.exe?/c+dir+
65 GET /scripts/Admin.dll
65 GET /scripts/..
65 GET /scripts/c32web.exe/ChangeAdminPassword
66 GET /msadc/root.exe?/c+dir
66 GET /msadc/Admin.dll
66 GET /msadc/samples/selector/showcode
66 GET /msadc/msadcs.dll/ActiveDataFactory.Query
67 GET /c/winnt/system32/cmd.exe?/c+dir+
67 GET /d/winnt/system32/cmd.exe?/c+dir+
68 GET /cfdocs/expeval/openfile.cfm
68 GET /cfdocs/exampleapps/publish/welcome.cfm
68 GET /cfdocs/exampleapp/docs/sourcewindow.cfm
68 GET /cfdocs/exampleapps/email/login.cfm
68 GET /cfdocs/snippets/viewexample.cfm
68 GET /cfdocs/expeval/exprcalc.cfm
68 GET /cfdocs/cfmlsyntaxcheck.cfm
69 GET /phpMyAdmin/
69 GET /admin.php
70 GET /iissamples/exair/howitworks/code.asp
70 GET /$_{v}ti_{p}vt$
70 GET /$_{p}rivate/shopping_{c}art.mdb$
70 GET /$_{n}cl_{s}ubjects.shtml$
70 GET /site/eg/source.asp
70 GET /pbserver/pbserver.dll
70 GET /process$_{b}ug.cgi$
70 GET /search/query.idq
70 GET /home.nsf/<img
70 GET /$_{v}ti_{p}vt/authors.pwd$
70 GET /level/16/exec/show/config/cr
70 GET /ncl$_{s}ubjects.html$
70 GET /iissamples/sdk/asp/docs/codebrws.asp
70 GET /pls
70 GET /lcgi/ndsobj.nlm

70 GET /pccsmysqladm/incs/dbconnect.inc
70 GET /$_vti_bin/_vti_aut/$
70 GET /n0nexi5tent$_fi1e.html$
70 GET /search97cgi/vtopic
70 GET /iissamples/exair/howitworks
70 GET /eManager/cgi-bin/register.dll
70 GET /dsgw/bin/search
70 GET /servlet/sunexamples.BBoardServlet
70 GET /interscan/cgi-bin/FtpSaveCSP.dll
70 GET /query.idq
70 GET /CFIDE/Administrator/startstop.html
70 GET /$_vti_pvt/service.pwd$
70 GET /shop/product.asp
70 GET /pls/admin$_/gateway.htm$
70 GET //WEB-INF/
70 GET /SWEditServlet?station$_path$
70 GET /ddrint/bin/ddicgi.exe
70 GET /CFIDE/Administrator/docs/releasenotes.htm
70 GET /interscan/cgi-bin/FtpSaveCVP.dll
70 GET /bb$_smilies.php?user$ =
70 GET /globals.pl
70 GET /search/query.asp
70 GET /catinfo
71 worm
71 sasser
71 exploit
71 msds

A.2 Arquivo PORT.POR

21 0000000011111110
22 0000111100001110
23 0000111111110001
25 0011001100110010
53 0011001111001101
79 0011110000111101
80 0011110011000010
109 0101010101010100
110 0101010110101011
111 0101101001011011
113 0101101010100100
119 0110011001100111
139 0110011010011000
143 0110100101101000
445 0110100110010111
513 1001011001101000

514 1001011010010111
515 1001100101100111
1433 1001100110011000
3128 1010010101011011
3389 1010010110100100
5632 1010101001010100
5555 1010101010101011
6000 1100001100111101
6666 1100001111000010
6699 1100110000110010
7777 1100110011001101
8875 1111000000001110
8080 1111000011110001
8888 1111111100000001

A.3 Arquivo LAPRNA.BIN

1 0000011001001010
2 0000011001110101
3 0000011110001101
4 0000011110110010
5 0000101010010100
6 0000101010101011
7 0000101101010011
8 0000101101101100
9 0000110011011111
10 0000110011100000
11 0000110100011000
12 0000110100100111
13 0011000001001100
14 0011000001110011
15 0011000110001011
16 0011000110110100
17 0011011000000111
18 0011011000111000
19 0011011111000000
20 0011011111111111
21 0011101011011001
22 0011101011100110
23 0011101100011110
24 0011101100100001
25 0011110010010010
26 0011110010101101
27 0011110101010101
28 0011110101101010
29 0101000010010111

30 0101000010101000
31 0101000101010000
32 0101000101101111
33 0101011011011100
34 0101011011100011
35 0101011100011011
36 0101011100100100
37 0101101000000010
38 0101101000111101
39 0101101111000101
40 0101101111111010
41 0101110001001001
42 0101110001110110
43 0101110110001110
44 0101110110110001
45 0110000011011010
46 0110000011100101
47 0110000100011101
48 0110000100100010
49 0110011010010001
50 0110011010101110
51 0110011101010110
52 0110011101101001
53 0110101001001111
54 0110101001110000
55 0110101110001000
56 0110101110110111
57 0110110000000100
58 0110110000111011
59 0110110111000011
60 0110110111111100
61 1001001001010110
62 1001001001101001
63 1001001110010001
64 1001001110101110
65 1001010000011101
66 1001010000100010
67 1001010111011010
68 1001010111100101
69 1001100011000011
70 1001100011111100
71 1111100000100111

APÊNDICE B – ARQUIVO DE PERSISTÊNCIA DA RNA

B.1 Arquivo neuralnet.padrao

A transferência de conhecimento na estrutura proposta, é realizada através do carregamento do arquivo de persistência, o qual é composto por 5422 valores, que representam individualmente os valores que devem ser atribuidos a cada conexão sináptica da rede neural artificial, bem como alguns parametros de configuração.

A seguir são apresentados os primeiros 20 valores deste conjunto, para simplificar o entendimento dessa estrutura.

-0.480000
0.956128
-0.640000
-0.347866
-0.532000
-0.591872
0.660000
0.820128
-0.496000
-0.955871
-0.768000
-0.687872
0.516000
0.996128
0.440000
0.676128
-0.208977
0.759023
-0.835974
-0.547975

APÊNDICE C – AUTÔMATO FINITO

O autômato implementado é composto de estados (S) e transições (T), de um estado inicial (S0) e de estados finais (S6 e S7), representados por um círculo duplo. A figura a seguir descreve seu esquema:

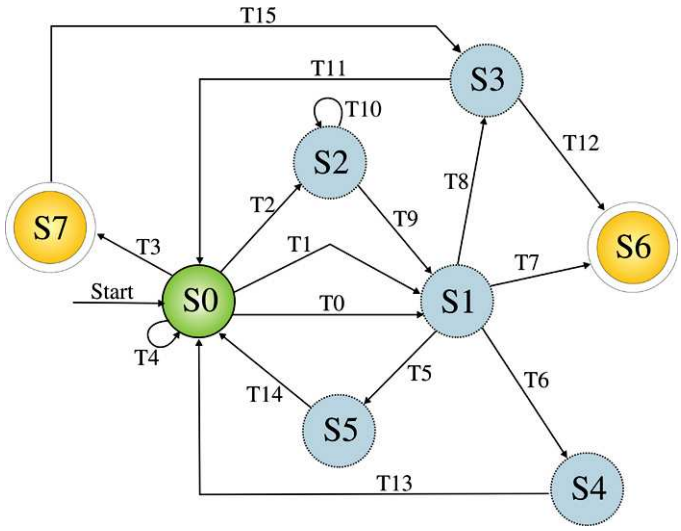


Figura 29: Esquema do autômato finito reconhecedor de assinaturas

Este autômato é composto por 7 estados e 14 transições, sendo controlado por algumas variáveis globais a todos os estados. São elas:

- ***ini*** **e** ***fim***: ponteiros para o início e fim das palavras a serem localizadas.
- ***indexAnt***: se o que estiver sendo procurado é identificado como sendo componente de assinatura, e também é por si só uma assinatura, é este ponteiro que guarda o fim desta primeira assinatura encontrada. Se nenhuma assinatura for encontrada, está é retornada e a análise léxica continua a partir deste ponto.
- ***indexProx***: ponteiro para a segunda palavra que estiver sendo procurada.
- ***lexema***: Um flag indicando se encontrou um caractere especial, que deva ser tratado de forma diferente das letras e dígitos.