VINÍCIUS MOLL

DETECÇÃO DE INTRUSÃO USANDO TÉCNICAS DE APRENDIZAGEM DE MÁQUINAS

FLORIANÓPOLIS

2010

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA DE AUTOMAÇÃO E SISTEMAS

DETECÇÃO DE INTRUSÃO USANDO TÉCNICAS DE APRENDIZAGEM DE MÁQUINAS

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia de Automação e Sistemas.

VINÍCIUS MOLL

Florianópolis, Abril de 2010

DETECÇÃO DE INTRUSÃO USANDO TÉCNICAS DE APRENDIZAGEM DE MÁQUINAS

Vinícius Moll

"Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia de Automação e Sistemas, Área de Concentração em Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia de Automação e Sistemas da Universidade Federal de Santa Catarina."

_____

Prof. Joni da Silva Fraga, Dr.
Orientador

_____

Prof. Eugênio de Bona Castelan Neto, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia de Automação e Sistemas

Banca Examinadora:

_____

Prof. Joni da Silva Fraga, Dr.
Presidente

_____

Prof. Carlos Alberto Maziero, Dr.

_____

Prof. Jomi Fred Hubner, Dr.

_____

Prof. Rafael Rodrigues Obelheiro, Dr.

Agradecimentos

Agradeço a Jeová Deus pelo dom imerecido da vida e pela força concedida a cada instante de mais esta conquista.

Agradeço ao meu orientador, Joni da Silva Fraga, que me ensinou muito ao longo destes dois anos que trabalhamos no desenvolvimento deste trabalho; aos demais professores do DAS (Eugênio Castelan, José Cury, Max Hering, Ricardo Rabelo, Guilherme Bittencourt, Eduardo Camponogara), que contribuíram para a minha formação.

Aos meus pais João Reinildes Moll e Iliane Gorete Moll, que sempre fizeram o seu melhor para me proporcionar a melhor educação possível. Agradeço também aos meus irmãos João Jr., Samir e Ernani pelo apoio e incentivo em seguir em frente nos meus estudos.

Um agradecimento especial à minha querida esposa Gabriela, pelo amor, carinho, paciência e por contribuir para o caminho ser mais fácil de ser percorrido.

Ao Paulo Mafra, que me ajudou neste trabalho e nas publicações. Agradeço aos colegas de laboratório, Marcos Camada, ao Carlos, ao Jim Lau, à Maiara, ao Rafael Deitos, ao Mathias e a todos que estou esquecendo de citar.

Quero agradecer também ao Alexandre Back e a Miriam Rolt, e aos meus colegas de trabalho, que direta ou indiretamente, contribuíram para a realização deste trabalho e não mediram esforços em me ajudar a alcançar mais esta conquista.

Não posso deixar de agradecer também à Maria Gorete, ao Rafael Callegaro e a sua esposa Vera, que me ajudaram muito durante a minha chegada em Florianópolis... (*The city of blinding ligths!*).

Obrigado aos amigos, Jorge Bidarra, Selmo Bonato, Márcio e Denise Veronez, Liege Ciupak e Neusa Carneiro que também me incetivaram a ir em busca da realização deste trabalho.

Enfim, a todas as pessoas que me ajudaram direta ou indiretamente para o meu crescimento, profissional e pessoal... Muito Obrigado.

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para obtenção do grau de Mestre em Engenharia de Automação e Sistemas.

DETECÇÃO DE INTRUSÃO USANDO TÉCNICAS DE APRENDIZAGEM DE MÁQUINAS

Vinícius Moll

Abril/2010

Orientador: Joni da Silva Fraga, Dr.

Área de Concentração: Controle, Automação e Sistemas.

Palavras-chave: detecção de intrusão, sistemas inteligentes, segurança computacional
Número de Páginas: 150

Sistemas de detecção de intrusão (IDSs) têm sido projetados para analisar dados coletados em redes ou em máquinas. Um IDS pode aplicar uma abordagem de análise baseada em assinaturas, similar a um *software* anti-vírus. Contudo, um IDS baseado em assinaturas não detecta ataques desconhecidos, porque o seu banco de assinaturas está desatualizado ou porque nenhuma assinatura está disponível ainda. Para superar essa limitação, os pesquisadores têm desenvolvido IDSs baseados em anomalias.

A abordagem baseada em anomalias é capaz de detectar ataques desconhecidos, modificações de ataques conhecidos, novos ataques e ataques direcionados. Este trabalho apresenta os principais esforços presentes na literatura sobre IDSs fazendo uma comparação entre os sistemas atualmente disponíveis. Depois apresenta uma nova arquitetura de IDS baseado em anomalias que utiliza técnicas de inteligência artificial para construir um classificador neural em uma primeira camada e uma segunda camada de verificação, que visa altas taxas de detecção e baixas taxas de falsos alertas gerados pelo sistema construído.

Por fim, apresenta um protótipo do sistema, o Polvo-IIDS e analisa os resultados de testes efetuados, com algumas conclusões e perspectivas futuras.

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements for the degree of Master in Automation and Systems Engineering.

INTRUSION DETECTION USING MACHINE LEARNING TECHNIQUES

Vinícius Moll

April/2010

Advisor: Joni da Silva Fraga, Dr. Ing.
Area of Concentration: Control, Automation and Systems.
Keywords: intrusion detection, intelligent systems, computer security
Page count: 150

In this master dissertation, we show that Intrusion Detection Systems (IDSs) have been designed to analyze data collected from networks or hosts. An IDS may apply an analysis approach based on signatures, in this case is very similar to a software anti-virus. However, a signature-based IDS cannot detect unknown attacks, because its bank signature is outdated or because there is no signature available yet. In order to overcome this limitation, researchers have developed anomaly-based IDSs.

The anomaly-based approach can detect unknown attacks, modifications of known attacks, attacks as soon as they occur and targeted attacks. This work presents the main efforts by Intrusion Detection Systems in the literature, doing a comparison between the systems currently available. Then we present a new anomaly-based IDS architecture that uses artificial intelligence techniques to build a first layer with a neural classifier and a second layer of verification, which aims high detection rates and low rates of false alert generated by the system built.

Finally, we present a prototype system Polvo-IIDS, the results of tests performed, some conclusions and future perspectives on the proposed model.

Conteúdo

Lista de Figuras

Lista de Tabelas

Notação

Símbolos

$\Re$	:	*conjunto dos números reais*
$\Re^n$	:	*conjunto dos vetores reais de dimensão n*
M^T	:	*transposta da matriz* M
M^{-1}	:	*inversa da matriz* M
Φ	:	*alguma função de Projeção* Φ

Acrônimos

RNAs	:	Redes Neurais Artificiais - *Artificial Neural Networks*.
SVMs	:	Máquinas de Vetores de Suporte - *Support Vector Machines*.
CERT	:	*Computer Emergency Response Team*.
CERT.br	:	*Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança*.
IDSs	:	*Intrusion Detection Systems*.
IIDSs	:	*Inteligent Intrusion Detection Systems*.
NIDS	:	IDSs baseados em Rede *Network Intrusion Detection System*.
HIDS	:	IDSs baseados em Host *Host Intrusion Detection System*.
DARPA	:	*Defense Advanced Research Projects Agency*.
IPDI	:	Instituro de Peritos em Tecnologias Digitais e Telecomunicações.
OCR	:	*Optical Character Recognition.*

1. Introdução

1.1: Motivação

Devido aos avanços tecnológicos das últimas décadas, hoje existem milhares de computadores conectados à Internet. Uma ampla variedade de serviços, recursos e informações estão disponíveis através da rede mundial, o que torna a segurança desta um elemento crítico.

Entre os serviços amplamente utilizados na rede podemos destacar os serviços *e-mail*, *e-banking*, sites de compras e transações *on-line*. Estes e outros serviços precisam ser seguros no sentido de manter a confiança de seus usuários. Mas garantir a confiabilidade em transações financeiras e evitar fraudes ou roubos de informações sigilosas (por exemplo, senhas ou números de cartões de crédito) é sempre uma tarefa árdua se considerarmos ambientes abertos como a Internet.

É importante salientar que a maior disponibilidade de serviços e aplicações em geral, via Internet, certamente provocou a explosão de usuários que hoje presenciamos na rede mundial. Mas também é necessário reconhecer o aumento significativo de fraudes e de outras atividades maliciosas neste ambiente aberto. Diante destas atividades maliciosas, foram incrementadas as ações e controles preventivos nos serviços e aplicações, mas estes nem sempre se mostram efetivos. Métodos de detecção e de análises são, portanto complementos necessários para minimizar as perdas e para que se fortaleçam as defesas contra as atividades maliciosas nos sistemas. É neste último contexto que entram os sistemas de detecção de intrusão (*Intrusion Detection Systems* – IDSs). A partir da última década, o desenvolvimento e o uso de IDSs capazes tem sido intensificado no sentido de identificar e alertar violações à segurança de sistemas computacionais.

A proposta de sistemas de detecção de intrusão não é recente e sua origem remonta aos sistemas computacionais de auditoria concebidos na década de 1960. Naquela época, os sistemas realizavam auditorias baseadas exclusivamente na perícia de um operador humano [51]. Contudo, as tarefas de monitoramento e auditoria

em sistemas computacionais deixaram de ser triviais e exigem a geração automática de alertas junto com a detecção e classificação dos ataques lançados contra as aplicações e seus serviços.

1.2: Objetivos

O foco deste trabalho é abordar as questões relacionadas ao desenvolvimento de sistemas de detecção de intrusão flexíveis e adaptáveis. Para alcançar estes atributos, definimos como parte central deste trabalho o desenvolvimento de um classificador baseado em técnicas de Inteligência Artificial – IA.

Entre as muitas questões que direcionaram nossos objetivos neste trabalho estão:

- Como diminuir a taxa de alarmes falsos (falsos positivos) gerados por um sistema de detecção de intrusão?
- Quais técnicas um IDS deve utilizar na análise e classificação de ataques?
- Como avaliar a metodologia utilizada para automatizar o processo de classificação de ataques em um IDS?
- Como definir uma abordagem para controlar e ajustar o comportamento de um IDS, possibilitando sua adaptação?

Das questões citadas resultaram objetivos específicos neste trabalho de onde destacamos:

1. Fazer um estudo das principais técnicas de inteligência artificial, amplamente utilizadas ao longo dos últimos anos na construção e no aprimoramento dos sistemas de detecção de intrusão, destacando suas vantagens e limitações;
2. Apresentar uma nova abordagem de classificação e análise, através do uso de uma arquitetura em duas camadas, que usa uma primeira camada na classificação e a segunda camada na verificação dos dados classificados.

As pesquisas dos últimos anos, relacionadas ao desenvolvimento de IDSs mostram que os sistemas atuais têm feito uso de diferentes técnicas para definir o modelo de comportamento normal em um sistema ou rede. Neste trabalho, seguimos uma abordagem similar por apresentar uma arquitetura capaz de classificar ataques de modo automático, visando a redução da taxa de alarmes falsos gerados pelo sistema.

1.3: Organização do Trabalho

Esta dissertação está organizada em sete capítulos. Neste capítulo inicial, apresentamos o contexto geral do trabalho, sua motivação, objetivos pretendidos e suas contribuições. Os demais capítulos estão separados de acordo com a seguinte estrutura:

No capítulo 2 apresentamos os principais conceitos relacionados à segurança dos sistemas computacionais.

O capítulo 3 define os principais conceitos de sistemas de detecção de intrusão e apresenta exemplos de sistemas existentes.

O capítulo 4 aborda os principais conceitos relacionados às redes neurais artificiais – RNAs, segundo a sua utilização neste trabalho como classificadores de padrões.

No capítulo 5 tratamos dos principais conceitos envolvendo as máquinas de vetores suporte – *Support Vector Machines* – SVMs aplicadas à verificação e a detecção de novidades nos padrões classificados.

E o capítulo 6 apresenta o modelo de IDS Inteligente conforme a proposta inicial deste trabablho, analisando suas vantagens e desvantagens e os resultados obtidos.

Por fim, o capítulo 7 discute algumas das conclusões e perspectivas futuras referentes a este trabalho.

1.4: Contribuições

Como principais contribuições deste trabalho apontamos:

- Uma nova arquitetura de sistema de detecção de intrusão baseado em anoma-

lias, que emprega redes neurais de Kohonen (mapas auto-organizáveis) na classificação de ataques e verificadores (máquinas de vetores suporte) na camada subjacente.

- A verificação dos resultados obtidos com a aplicação das referidas técnicas de inteligência artificial e de aprendizado de máquina na construção de IDSs baseados em anomalias.

2. Segurança Computacional

Desde a década de 70 tem sido crescente a preocupação com a segurança dos sistemas computacionais. Com o passar dos anos, diversos mecanismos de segurança foram desenvolvidos tanto para a comunicação quanto para os sistemas operacionais e suas aplicações. Neste capítulo, apresentamos brevemente os conceitos básicos relacionados à segurança computacional.

2.1: Conceitos Básicos

O significado de segurança computacional tem sido modificado ao longo do tempo. Segundo o NIST[1][33], o termo segurança computacional refere-se a um conjunto de regras, técnicas e mecanismos que visam preservar a integridade, disponibilidade e confidencialidade do sistema e seus recursos de *hardware*, *software*, *firmware*, dados e telecomunicações.

A segurança em sistemas computacionais visa, portanto, a proteção da informação contra a destruição, vazamento ou modificação intencional ou acidental e a proteção contra o mau uso de recursos disponibilizados por um sistema.

Atualmente o conceito de segurança está padronizado pela norma ISO/IEC $17799:2005$ que regra o uso de recursos e informações dos sistemas com o objetivo de garantir as seguintes propriedades:

- **Confidencialidade**: garantia de proteção contra leitura ou cópia por usuário não autorizado.
- **Disponibilidade**: garantia de proteção de qualquer sistema ou serviço contra a degradação e/ou indisponibilidade sem autorização. Implica a informação ou serviço estará sempre disponível.
- **Integridade**: significa garantir a proteção de qualquer informação ou sistema contra modificações ou remoções sem a permissão explícita de seu proprietário ou do

[1] National Institute of Standards and Technology.

sistema que a disponibiliza.

Além destas propriedades, na literatura outros autores defendem que a segurança computacional deve garantir:

- **Autenticidade**: serve para atestar a validade das informações e sua origem, garantindo a identidade de entidades que interagem no sistema.
- **Não repúdio**: possibilidade de identificação de autoria de determinada ação sem equívocos, fornecendo prova irrefutável da realização de uma ação específica por parte de algum usuário.
- **Legalidade**: refere-se à aderência ou compatibilidade de um sistema ou informação à legislação ou cláusulas contratuais vigentes no país, localidade de disponibilização e uso da informação ou sistema.

2.1.1: Violações da Segurança

A violação da segurança é definida em [46] como sendo a atividade que determina a não verificação de uma das propriedades citadas anteriormente. Na literatura são identificadas basicamente três violações à segurança:

- Revelação não autorizada da informação (não verificação da propriedade de confidencialidade);
- Modificação não autorizada da informação (não verificação da propriedade de integridade);
- Negação de serviço (não verificação da propriedade de disponibilidade no sistema).

2.1.2: Vulnerabilidades

O termo vulnerabilidade normalmente é associado a recursos que por problemas de mau dimensionamento ou por falhas de projeto, tornam o sistema sensível a possíveis

invasões ou exposição acidental de informações e serviços. Ou seja, uma vulnerabilidade pode ser um ponto fraco ou um tipo de falha de projeto no sistema, em protocolos, serviços ou outros recursos que podem permitir acessos não autorizados no sistema computacional.

As vulnerabilidades existem em aplicações, sistemas operacionais, protocolos de rede e até mesmo no próprio *hardware*. Elas podem existir em partes (ou módulos) de um sistema de *software* que executa em um modo privilegiado, em uma senha fraca, numa regra de *firewall* mal configurada, ou ainda numa falha no projeto de *software*. Uma vulnerabilidade conhecida não representa perigo enquanto não existir a possibilidade de exploração da mesma.

2.1.3: Ameaças

Uma ameaça é qualquer circunstância ou evento com o potencial intencional ou acidental de explorar uma vulnerabilidade específica em qualquer sistema computacional, resultando na perda das propriedades de confidencialidade, integridade ou disponibilidade, ou seja, resultando em violações de segurança no sistema.

2.1.4: Ataques

Os ataques são ações que usam como base falhas de projeto e vulnerabilidades conhecidas do sistema, visando a violação da segurança. Qualquer sistema pode ser alvo de ataques, estando ou não conectado em rede (Internet/Intranet).

Em resumo, ataques são concretizados com qualquer ato intencional ou acidental que permita ataques em sistemas computacionais. Com estas invasões teremos a ocorrência das violações de segurança.

2.1.5: Riscos

Os riscos podem ser definidos com base nos impactos negativos resultantes da exploração de uma vulnerabilidade [73]. O risco é uma tentativa de quantificar as possi-

bilidades de violação e os prejuízos decorrentes de tais violações. Os riscos podem ser identificados e reduzidos, mas nunca totalmente eliminados.

2.1.6: Contra-medida

Uma contra-medida é uma ação preventiva ou resposta à ocorrência de um incidente que compromente a segurança. Antes de se adotar qualquer contra-medida, é necessário conhecer os principais tipos de ataques, as vulnerabilidades mais exploradas, os riscos envolvidos e os objetivos dos atacantes.

2.1.7: Ataques mais Freqüentes

Com relação aos tipos de ataques mais frequentes, em [46] citam-se os seguintes:

- Negação de serviço: tem por objetivo indisponibilizar um serviço por algum tempo. Por exemplo, nesse tipo de ataque o atacante inunda um servidor alvo com milhares de requisições até que o sistema não consiga mais atender a nenhuma requisição;
- Varredura (*scan*) de portas: visa identificar os serviços de rede disponíveis, as versões dos programas, sistema operacional, se existe um *firewall* no caminho, e outras informações;
- *Sites* falsos: sites que imitam os originais visando o roubo de dados bancários de usuários. O usuário recebe os *links* para estes *sites* por *e-mail* ou quando o servidor de nomes foi alterado.

Uma porcentagem grande do que foi relatado ao CERT.br[2] nos últimos anos envolveu vulnerabilidades de *buffer overflow*. Assim como muitos outros tipos de ataques, o que explora o estouro de buffer tem evoluído em complexidade e sofisticação. Os

[2]Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança no Brasil – é o Grupo de Resposta a Incidentes de Segurança para a Internet brasileira, mantido pelo Comitê Gestor da Internet no Brasil. É o grupo responsável por receber, analisar e responder a incidentes de segurança em computadores, envolvendo redes conectadas à Internet brasileira. `http://www.cert.br/`

primeiros ataques envolviam dados explicitamente lidos pelo programa alvo. Depois os ataques passaram a explorar registros usados para guardar valores de variáveis de ambiente sendo estas utilizadas como vetores de ataques. Recentemente tais ataques tem sido baseados no formato de strings.

O número de vulnerabilidades encontradas por ano, conforme a tabela 1, não tem variado muito, no entanto o número de incidentes conforme visto na tabela 2, tem sofrido um aumento considerável ao longo dos anos.

Ano	2000	2001	2002	2003	2004	2005
Vulnerabilidades	1.090	2.437	4.129	3.784	3.784	5.990

Table 1: Vulnerabilidades reportadas ao CERT.

Ano	2000	2001	2002	2003	2004	2005
Incidentes	5.997	12.301	25.092	54.607	75.722	68.000

Table 2: Incidentes reportados ao CERT.br.

Para ilustrar o aumento do número de incidentes relacionados a segurança, segundo o CERT.br, apenas no período de 1 de julho a 30 setembro de 2009, foram registrados mais de 20.000 incidentes, conforme é mostrado na **Figura 1** [13].

Sendo que os incidentes reportados na figura 1, referem-se desde ataques de negação de serviço, varredura de portas a fraudes e outros incidentes que não se enquadram nas categorias anteriores [13]

As estatísticas dos últimos dez anos revelam a necessidade de desenvolvimento e utilização de sistemas para auxiliar na detecção e no combate a fraudes e intrusões, o que pode ser comprovado na evolução do número de casos ao longo dos anos, conforme o gráfico da figura 2.

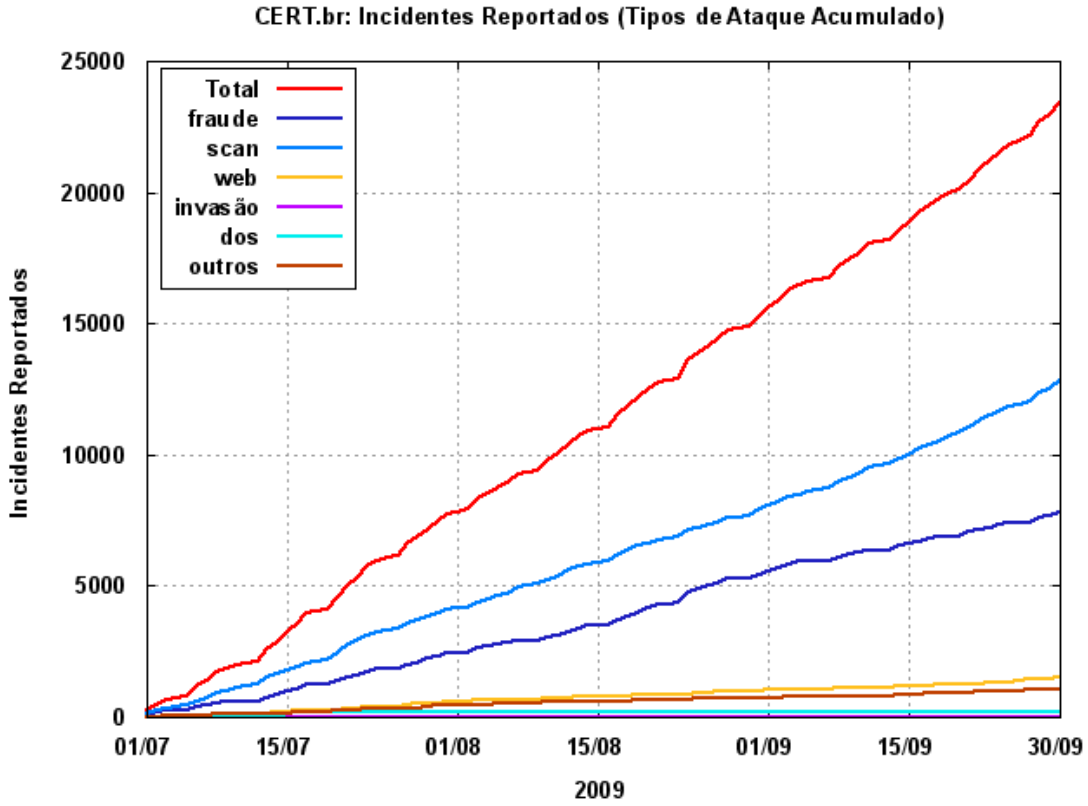


Figure 1: Número de incidentes relacionados a segurança na Internet no Brasil.

2.2: Políticas e Modelos de Segurança

2.2.1: Políticas

Uma política de segurança pode ser definida como um conjunto de regras pré-definidas no sentido de manter as propriedades de segurança no sistema [46]. Para [7], uma política de segurança forma a base para o estabelecimento do que é e o que não é permitido.

Formalmente, uma política de segurança divide os estados do sistema em um conjunto de estados autorizados, e em um conjunto de estados não autorizados. Dessa forma, um sistema seguro é um sistema que se inicia em um estado autorizado e evolui entre estados seguros.

As políticas de segurança podem seguir dois padrões de implementação distintos [32]:

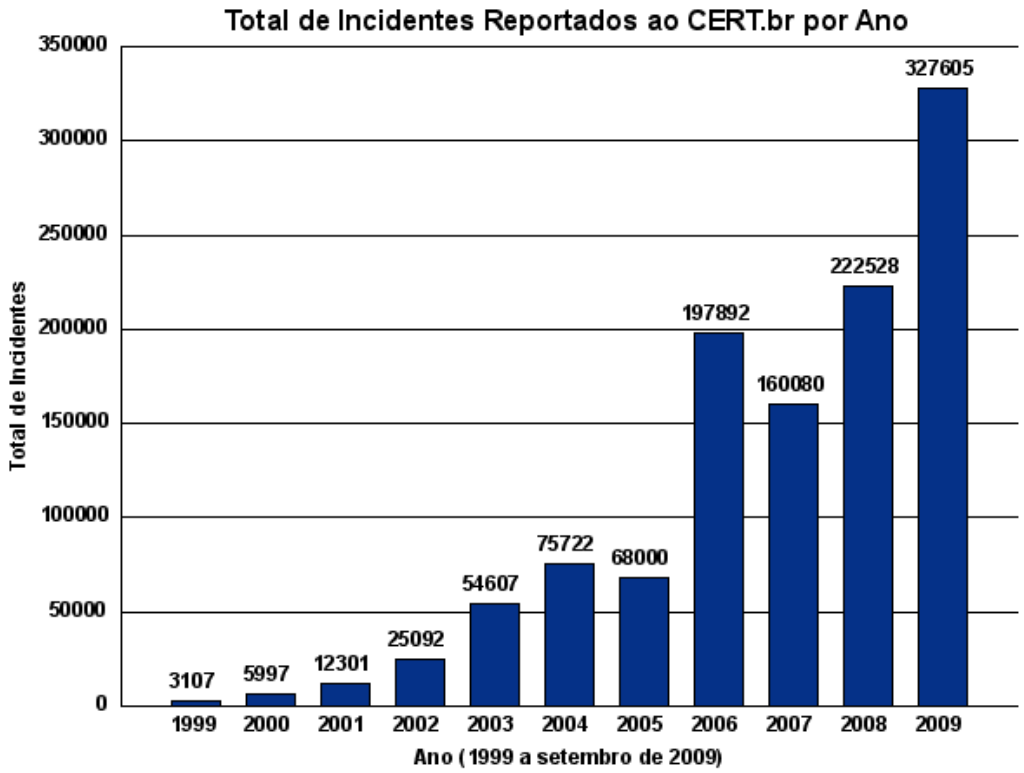


Figure 2: Evolução do número de incidentes reportados entre 1999 a 2009.

1. Padrão de negação: especificar apenas o que é permitido e negar o resto.

2. Padrão de permissão: especificar somente o que é proibido e liberar o resto.

O padrão de negação é o mais prudente. Mesmo havendo um equívoco na sua definição, nenhum serviço ficará exposto.

As políticas de segurança de sistemas definem três níveis de segurança: segurança física, segurança gerencial e segurança lógica.

A segurança física é voltada à proteção dos recursos físicos do sistema. Nesta política são definidas medidas contra desastres físicos e medidas para proteger o acesso físico ao sistema, através da proibição do acesso de pessoas não autorizadas.

A política de segurança gerencial preocupa-se com o ponto de vista da organização, sendo responsável pela definição dos processos para criação e manutenção das próprias políticas de segurança.

A política de segurança lógica define quais usuários terão direito de acesso ao

sistema e quais são estes direitos. Nesta política, são aplicados dois conceitos básicos: a autenticação, onde o usuário precisa se identificar ao sistema para obter acesso ao recurso; e a autorização, onde o usuário necessita provar que possui direitos sobre o recurso que deseja acessar.

Nesse sentido, a literatura apresenta os termos **principal** e **intruso**. O termo principal é utilizado para identificar usuários, processos ou máquinas atuando em nome dos usuários de um sistema, que são considerados aptos (pela política lógica) para executar determinada ação no sistema. O termo intruso, tem o sentido contrário e é usado para identificar usuários, processos e máquinas não autorizados pela política de segurança estabelecida.

2.2.2: Modelos de Segurança

Os modelos de segurança são diferenciados de mecanismos de segurança. Modelos de segurança determinam como os acessos são controlados e mecanismos de segurança são as funções de *software* e *hardware* que são configuradas a partir de um modelo de segurança para implementar uma política lógica (explicitada pelo modelo). Um modelo de segurança é uma expressão muitas vezes formal de uma classe de políticas, abstraindo detalhes e concentrando-se em um conjunto de comportamentos. Este modelo é utilizado como base para a definição de políticas de segurança [44]. Na literatura, os modelos de segurança são apresentados e divididos em três tipos básicos: discricionários (*discretionary*), obrigatórios (*mandatory*) e os baseados em papéis (*role-based*).

2.2.3: Modelos Discricionários

Nos modelos discricionários (Discretionary Access Control – DAC) os controles de acesso são definidos segundo políticas construídas pelos proprietários dos objetos. Cada requisição de um usuário para acessar um objeto requer a verificação das autorizações previamente especificadas. Caso exista uma autorização permitindo que o usuário possa acessar o objeto, no modo especificado (leitura, escrita, etc), o acesso é

permitido ou em caso contrário é negado [65].

Em [43] foi introduzido o modelo de matriz de acesso, um modelo conceitual discricionário que especifica os direitos que cada usuário possui sobre cada objeto do sistema. Os sujeitos (usuários, processos ou máquinas) representam o usuário (**principal**). Fazendo uma analogia com o controle de acesso de um sistema operacional, cada linha da matriz representa um sujeito do sistema, cada coluna da matriz representa um objeto deste sistema (arquivo, processo, etc) e as células indicadas pelas linhas e pelas colunas especificam os direitos de autorização do sujeito (linha) sobre o objeto (coluna).

Os modelos discricionários não impõem qualquer restrição na forma de uso da informação por um usuário, permitindo inclusive que um principal propague a informação acessada sem levar em conta suas autorizações associadas.

2.2.4: Modelos Obrigatórios

Os Modelos obrigatórios (Mandatory Access Control – MAC) caracterizam políticas globais incontornáveis em um sistema. Estes modelos determinam controles que definem os fluxos permitidos de informação no sistema. Estes modelos são baseados na classificação de sujeitos e objetos no sistema. Cada sujeito e cada objeto no sistema é associado a um nível de segurança.

O nível de segurança associado a um objeto reflete a sensibilidade da informação contida neste objeto, ou seja, quanto mais sensível for a informação, maior será o nível de segurança deste objeto. O nível de segurança associado a um sujeito (*clearance*) reflete a confiança no sujeito em não revelar informações sensíveis para outros sujeitos [65].

Em um exemplo simples, o nível de segurança é um elemento de um conjunto (ultra-secreto (US), secreto (S), confidencial(C) e não classificado (NC)) ordenado hierarquicamente, onde US $>$ S $>$ C $>$ NC. Cada sujeto de um nível de segurança possui acesso a objetos de seu próprio nível e a todos os outros em níveis mais baixos do seu na hierarquia.

Além dos níveis de segurança hierárquicos, categorias também podem ser asso-

ciadas com objetos e sujeitos. Neste caso, os rótulos de classificação de segurança são associados a sujeitos e objetos, formando um par composto por um nível de segurança e um conjunto de categorias. O conjunto de categorias associado com um usuário reflete as áreas específicas às quais o usuário pertence. O conjunto de categorias associadas a um objeto reflete a área na qual as informações contidas neste objeto são referenciadas. O uso de categorias provê um refinamento na classificação de segurança.

Na literatura existem diversas descrições de modelos obrigatórios, entre os quais podemos citar o modelo Bell-LaPadula [6], baseado nos procedimentos usuais de manipulação de informação em áreas ligadas à segurança nacional americana. Neste modelo, o sistema é descrito através de uma máquina de estados finitos onde as transições de estado obedecem a determinadas regras. Bell e Lapadula demonstraram por indução que a segurança do sistema é mantida partindo de um estado seguro e as únicas transições de estado permitidas são as que conduzem o modelo a um outro estado seguro.

2.2.5: Modelos Baseados em Papéis

Os modelos discricionários e obrigatórios não preenchem todos os requisitos necessários para a maioria das organizações. Desta forma, surgiram algumas alternativas para estes modelos, dentre os quais estão os modelos baseados em papéis (Role-Based Access Control – RBAC) [28, 65].

Um papel pode ser definido como um conjunto de ações e responsabilidades associadas com uma atividade do sistema [65]. Ao invés de especificar todos os acessos que cada principal pode executar, as autorizações de acesso a objetos são especificadas através de papéis. Um principal pode executar todos os acessos concedidos àquele papel. Papéis e privilégios são atribuídos e gerenciados por administradores. Em geral, um principal pode fazer uso de diferentes papéis em diferentes ocasiões e o mesmo papel pode ser atribuído a vários principais e ser utilizado simultaneamente por estes. Pode-se atribuir uma permissão a um ou mais papéis, sendo que a um papel podem estar associadas uma ou mais permissões. Um estudo [29] afirma que o uso de modelos baseados em papéis em organizações é uma abordagem que possui muitas vantagens. Algumas delas são discutidas a seguir:

- Gerência de autorização: a tarefa de especificar autorizações é dividida em duas partes, a associação de principais aos papéis e a associação de direitos aos papéis, simplificando o gerenciamento de segurança.

- Hierarquia de papéis: em muitas aplicações existe uma hierarquia natural de papéis, permitindo que as permissões sejam herdadas ou compartilhadas, simplificando a gerência de autorizações.

- Privilégio mínimo: papéis permitem que um principal trabalhe com o privilégio mínimo necessário para uma determinada tarefa, minimizando o risco de danos ao sistema na ocorrência de falhas ou intrusões.

- Separação de papéis: a separação de papéis é fundamental quando existem conflitos de interesses. Em uma sessão um usuário com vários papéis não pode atuar simultaneamente em papéis conflitantes. Por exemplo, um usuário não pode assumir papéis como "gerente de banco" e "cliente" do mesmo banco em uma mesma sessão de computação. Nesta situação, deve ser levado em conta que nenhum principal pode obter privilégios que permitam ao mesmo o uso de forma maliciosa, em benefício próprio.

2.3: Princípios de Segurança

Em geral, o esforço para se manter um sistema seguro é proporcional ao seu tamanho e complexidade. Minimizar a variedade, espaço e complexidade dos componentes confiáveis no modelo do sistema pode reduzir custos e riscos. Com isso é mais fácil assegurar que tais sistemas sejam corretamente especificados e implementados. Se um modelo centraliza toda a verificação de segurança em um componente que executa em um domínio separado e protegido, ele será freqüentemente invocado e isso pode exigir um grande número de operações. Os *kernels* centralizados sofrem desse problema [45].

2.4: Mecanismos de Segurança e Autenticação

Uma vez apresentadas as políticas e modelos de segurança, abordaremos alguns mecanismos utilizados para evitar ou prevenir violações de segurança. Um mecanismo de segurança pode ser um método, uma ferramenta ou um procedimento para viabilizar e implementar uma política de segurança [7].

Em [46] são definidos mecanismos de segurança como sendo aqueles que são desenvolvidos para detectar, prevenir ou recuperar o sistema de um ataque a sua segurança. Algumas ferramentas e mecanismos utilizados na prevenção e detecção de ataques são apresentados a seguir.

2.4.1: Domínios de proteção

Um domínio de proteção é definido como sendo um conjunto de permissões necessário para a manipulação de objetos deste domínio [43]. Um principal autorizado a entrar neste domínio terá então todos os direitos necessários para a sua computação. Esta definição é decorrente do princípio de privilégio mínimo. Além dos mecanismos para criação de domínios e de controle de entrada em domínios, são necessários mecanismos para controlar a comunicação entre domínios. Domínios podem ser construídos usando características de *hardware*, usando somente *software* ou, como é o caso usual, alguma combinação de *hardware* e *software*.

2.4.2: Identificação e Autenticação

A identificação é o processo em que o usuário de um sistema informa quem ele é, geralmente, ao informar seu nome de usuário ou *login*.

A autenticação determina se uma identificação fornecida (*login*) por um usuário é legítima.

O processo de autenticação em sistemas computacionais consiste tanto da autenticação de usuários como a autenticação mútua necessária na comunicação entre pares comunicantes em um sistema distribuído. Na autenticação de usuários, um con-

junto de procedimentos e mecanismos são usados na identificação de agentes externos (usuários, dispositivos, etc) como principais autorizados segundo as políticas de segurança adotadas no sistema.

O par nome de usuário e senha continua sendo a forma mais comum de identificação e autenticação de usuários [46]. Entretanto, o uso de cartões (tipo *smart cards*) e biometria está crescendo. Estas duas formas de autenticação dependem de fatores como: algo que o usuário conhece (senha), algo que o usuário tem (uma chave) ou de alguma característica física ou de comportamento do próprio usuário (impressão digital, íris).

Na autenticação entre partes comunicantes de um sistema distribuído, como elemento chave no processo de autenticação, é necessária a existência de um caminho de confiança entre a fonte de informação e o receptor da mesma. Este caminho de confiança é resolvido com a introdução de uma Terceira Parte Confiável (TPC) que será mediador entre os pares comunicantes no sentido de garantir a autenticidade dos mesmos e das informações trocadas.

Esta terceira parte confiável pode ser assumida por "serviços de autenticação" que em sistemas distribuídos concentram funções de autenticação de usuários e na construção de canais de comunicação confiáveis e seguros.

2.4.3: Controle de Acesso

O controle de acesso ou autorização, define quais direitos e permissões um usuário tem no sistema. Após a identificação e autenticação, a autorização determina o que um usuário pode fazer em um sistema. O controle de acesso faz a mediação de requisições entre usuários e objetos em um sistema. Existe uma entidade chamada de monitor de referência que recebe as requisições de acesso dos sujeitos e autoriza ou nega o acesso de acordo com a política de segurança utilizada.

Mecanismos de controle de acesso ou autorização existem em uma grande variedade em sistemas de computação. Normalmente, estes são expressão do modelo discricionário "Matriz de Acesso" e são representados basicamente em duas formas

clássicas: "listas de acesso" e "lista de competências". Em listas de acesso, as permissões são armazenadas junto aos objetos. Os usuários se apresentam com seus identificadores e realizam verificações nas listas para determinar se os mesmos possuem permissões para os acessos desejados nos objetos considerados. As listas de competências ("*capabilities*") definem o posicionamento dos direitos junto aos sujeitos. Quando deseja acessar um objeto, um sujeito deve apresentar a permissão necessária ao controlador do objeto. Este último verifica somente a autenticidade da permissão apresentada.

O sistema pode decidir se um sujeito está autorizado a acessar alguns arquivos em particular e uma aplicação como um sistema de gerência de banco de dados pode decidir quais os campos o sujeito pode acessar.

2.5: Detecção de Intrusão e Tolerância a Intrusão

Os sistemas de detecção de intrusão monitoram redes ou máquinas, com o objetivo de identificar ataques e possíveis violações de segurança. Tais sistemas podem ser centralizados, híbridos ou distribuídos.

Prevenir intrusões diante da complexidade dos sistemas atuais é uma tarefa árdua e muito difícil de garantir a eficácia do objetivo. Atualmente, grande parte das pesquisas estão tomando outros rumos. A idéia básica destas novas abordagens é conviver com intrusões mas tentando minimizar os danos das mesmas. Estes sistemas de tolerância a intrusão tentam confinar a ação dos atacantes ou, a partir da detecção de ações intrusivas (uso de IDSs), montar estratégias que permitam ao sistema manter o seu serviço correto mesmo diante destas ações [31].

2.5.1: Auditoria

Auditoria é a análise de registros que visa apresentar informações sobre o sistema de forma clara e compreensível [7]. Fazer auditoria em registros é extremamente útil para administradores de sistemas, pois através da auditoria é possível identificar violações do sistema ou falhas em programas. Estes registros são usados para detectar invasões e

para reconstruir as atividades no sistema de possíveis intrusos.

Atualmente, a auditoria de segurança baseada em registros é provavelmente a maneira mais comum de verificar se ocorreram violações, além de ser uma fonte de dados para os sistemas de detecção de intrusão que atuam *on-line* no sistema [46].

2.5.2: Controles Criptográficos

Controles criptográficos são mecanismos que fazem uso de técnicas criptográficas para transformar um texto inteligível numa forma ilegível e vice-versa [16].

Estes controles são usados para proteger documentos de entidades não autorizadas mantendo a confiabilidade dos mesmos e também como base para técnicas de autenticação (assinaturas digitais, MACs, etc)

Os criptosistemas usados nestes controles podem ser tanto simétricos como assimétricos. Normalmente, os assimétricos são usados na autenticação e na distribuição de chaves simétricas. A criptografia simétrica é limitada normalmente nestes controles ao papel de chaves de sessão (nos canais de comunicação).

2.6: Conclusões do Capítulo

Neste capítulo abordamos conceitos básicos relacionados a segurança dos sistemas computacionais. Apresentamos as principais propriedades que um sistema precisa garantir para ser considerado seguro. Apresentamos os conceitos de violação de segurança, ameaças, vulnerabilidades, ataques e riscos.

Analisamos ainda a definição de políticas e modelos de segurança, mecanismos de autenticação e controle de acesso. No próximo capítulo vamos abordar aspectos relacionados aos sistemas de detecção de intrusão.

3. Sistemas de Detecção de Intrusão

"That general is skillful in attack whose opponent does not know what to defend; and he is skillful in defense whose opponent does not know what to attack."

Sun Tzu, "The Art of War."

Nos últimos anos, observou-se um aumento do número de vulnerabilidades descobertas em sistemas computacionais e conseqüentemente um aumento do número de intrusões. Diversos sistemas de detecção de intrusão (*Intrusion Detection Systems* – IDS) têm sido desenvolvidos para ajudar administradores de redes a detectar violações das políticas de segurança adotadas pelos sistemas computacionais das organizações.

Este capítulo aborda essencialmente os sistemas de detecção de intrusão. Inicialmente, são apresentadas algumas definições de sistemas de detecção de intrusão presentes na literatura e sua evolução ao longo dos últimos anos. A parte inicial do capítulo descreve os elementos básicos que formam um IDS. Na seqüência, são apresentadas algumas abordagens utilizadas por estes sistemas para a coleta de dados. Em seguida, alguns sistemas de detecção de intrusão distribuídos são brevemente abordados, destacando-se algumas tendências. Por fim, são apresentados alguns exemplos de IDSs existentes com foco em suas características e técnicas de detecção.

3.1: Conceitos de IDSs

Um sistema de detecção (IDS) monitora sistemas locais (máquinas ou sistemas operacionais) ou suportes de comunicação (protocolos da rede de comunicação) para determinar a ocorrência de atividades maliciosas (intrusões). Cada vez que uma atividade maliciosa é detectada, o IDS gera um alerta que cumpre a função de notificar aos administradores sobre possíveis intrusões nos sistemas computacionais. Segundo [51], a detecção de intrusão pode ser definida como a tentativa do sistema de identificar violações à segurança e seus danos em sistemas computacionais. De acordo com Sandhu

[65] a detecção de intrusão corresponde a uma auditoria de sistema sendo executada *on-line*.

Os IDSs são programas de computador elaborados para detectar possíveis intrusões, comparando o comportamento observado nas atividades computacionais com padrões normais de comportamento, preferencialmente em tempo de execução [37]. Após detectar uma possível violação à segurança um IDS deve ser capaz de gerar alertas para notificar os administradores dos sistemas computacionais. As violações envolvem desde uso não autorizado até abusos do sistema por usuários legítimos ou intrusos.

Entre as noções comuns aos sistemas de detecção de intrusão, temos o que é chamado de **verdadeiro positivo**, que nada mais é do que um alerta real gerado em resposta a uma intrusão (ou tentativa) que tenha ocorrido no sistema. Já um **falso positivo** é um falso alerta, gerado em resposta a um comportamento não malicioso. Por outro lado, um **verdadeiro negativo** ocorre quando nenhum alerta (ou detecção) é gerado quando no sistema não existem tentativas de ataque ou intrusões. E, por fim, um **falso negativo** ocorre quando nenhum alerta é gerado diante da ocorrência de tentativas ou intrusões no sistema.

Os falsos positivos influenciam diretamente o uso de um sistema. Um sistema que dispara falsos alertas em boa parte do tempo acaba tendo seus alertas ignorados.

3.2: Evolução dos Sistemas de Detecção de Intrusão

As primeiras auditorias em sistemas computacionais foram realizadas por volta da década de 60. Essas auditorias eram baseadas exclusivamente na perícia de um operador humano [51]. Em 1972, James Anderson [2] publicou o *Computer Security Technology Planning* onde levantou questões sobre o que deve ser detectado, como realizar a análise e como proteger o sistema e seus dados contra ataques e violações.

3.2.1: Primeiros Sistemas de Auditoria

Em 1980, Anderson publicou um relatório [3] para um cliente que processava grande quantidade de dados usando *mainframes*. A equipe de segurança fez um manual detalhado de dados para serem auditados, procurando informações que pudessem indicar violações de segurança. Contudo, devido ao grande volume de dados processados, tal tarefa consumia bastante tempo, além de não ser trivial, uma vez que algumas informações necessárias não eram capturadas e outras eram capturadas repetidas vezes. Com isso, Anderson sugeriu formas alternativas para reduzir ou restringir a quantidade de dados a ser analisada, estabelecendo que dados estatísticos do comportamento de usuários e grupos fossem comparados com dados levantados do processamento, para viabilizar a detecção de comportamentos anormais.

Em 1984, Dorothy Denning [24] e Peter Neumann iniciaram o projeto IDES (*Intrusion Detection Expert System*), que foi um dos mais significativos no início das pesquisas sobre IDSs.

O modelo do IDES é baseado na suposição de que é possível estabelecer perfis para caracterizar a interação normal de usuários com objetos (arquivos, programas ou dispositivos). Os perfis são modelos estatísticos do comportamento de usuários e o sistema tenta detectar comportamentos anormais. O perfis são complementados por um sistema que usa uma base de regras para descrever atividades que representam violações de segurança conhecidas. Isso evita que um usuário possa gradualmente treinar o sistema para aceitar comportamentos ilegais.

Por volta do início da década de 1990, um grande número de IDSs havia sido desenvolvido, em sua grande maioria como uma combinação de abordagens estatísticas e sistemas especialistas. Um exemplo desses sistemas foi o de NADIR [35], onde os motores de análise incorporaram sistemas de gerenciamento de bases de dados comerciais tais como o Oracle e o Sybase, aproveitando a vantagem da habilidade que estes SGBDs apresentam na organização dos dados a serem auditados. O NADIR ainda é utilizado e está em atividade de refinamento para acomodar novas ameaças e para adaptar-se aos novos sistemas alvos.

Outro sistema deste tipo é o MIDAS, que foi desenvolvido pelo *National Computer Security Center* para monitorar seus sistemas Multics e Dockmaster [68]. O MIDAS era um sistema híbrido que usava a abordagem de análise estatística para examinar registros de dados do *Multics* que controlava o *login* de usuários. O MIDAS foi um dos primeiros sistemas que trabalhou conectado à Internet e foi utilizado no período de 1989 até metade da década de 1990.

Desde seu surgimento até os tempos atuais, ainda persistem alguns problemas e questões relacionadas aos IDSs. Entre os problemas podemos citar a dificuldade de obtenção de dados livres de intrusões para realizar o treinamento e o alto índice de falsos positivos [38]. Além disso, a técnica que melhor se adequa para a análise dos dados obtidos em diferentes fontes é uma das questões que devem ser consideradas durante o projeto e o desenvolvimento de um IDS.

3.3: Modelo de Sistemas de Detecção de Intrusão

Geralmente o projeto e desenvolvimento de IDSs segue um modelo padrão. A diferença entre os IDSs existentes reside nos módulos que os compõem, mas alguns elementos básicos são os mesmos independente da arquitetura ou do método de detecção empregado.

Nos últimos anos, um grande esforço vem sendo feito para padronizar as funções dos elementos de um IDS visando facilitar a integração entre diferentes IDSs. O CIDF[3] [70] definiu um modelo genérico e propos a padronização (ainda em *draft*) de elementos de um IDS, como contendo os seguintes componentes:

- E-Box – Gerador de Eventos: componente responsável pela geração e pela padronização do formato de dados dos eventos;
- A-Box – Analisador de Eventos: componente responsável pela detecção de intrusos. este componente recebe os dados do gerador de eventos e faz uma análise na procura de padrões que caracerizam um possível ataque;
- D-Box – Base de dados de Eventos: componente de armazenamento dos eventos;

[3] http://www.isi.edu/gost/cidf/

- C-Box – Contramedidas: componente encarregado pelas ações a serem tomadas como respostas a incidentes. Pode ser um bloqueio de conexão, um sinal para encerrar algum processo, entre outras ações que caracterizem uma resposta do sistema.

O CIDF foi uma tentativa de padronização dos sistemas de detecção de intrusão que tem por objetivo definir a estrutura de um IDS. Juntamente com o CIDF, foi desenvolvida uma linguagem para especificação de eventos denominada CISL[4], que padroniza a troca de informações entre os componentes sobre os eventos ocorridos no ambiente.

Os esforços empregados no desenvolvimento do CIDF e da linguagem CISL tinham por objetivo abranger toda a estrutura de um IDS, dividindo-o em módulos e padronizando tanto a implementação destes módulos quanto a forma de interação entre IDSs.

Em 1999 o projeto CIDF foi descontinuado e a maioria dos esforços têm sido concentrados em outro importante trabalho de padronização dos IDSs que vem sendo desenvolvido pelo grupo IDWG, criado em 1998. O objetivo do grupo IDWG[5] [87], pertencente ao IETF[6], é definir formatos de dados e procedimentos para o compartilhamento de informações em IDSs.

O IDWG desenvolveu uma proposta de padronização do formato de dados e protocolos de comunicação para viabilizar a troca de informações entre IDSs, resultando na especificação de um formato padrão para troca de mensagens: o IDMEF[7] e a especificação do protocolo IDXP[8] um protocolo de comunicação para o transporte das mensagens IDMEF.

O IDXP é o protocolo que proverá um canal de comunicação para as trocas de mensagens IDMEF entre os componentes do IDS. Este protocolo é implementado como um novo perfil do protocolo BEEP[9]. O grupo IDWG também propôs um modelo básico de elementos de um IDS conforme pode ser visualizado na figura 3 a seguir:

[4] Common Intrusion Specification Language
[5] Intrusion Detection Exchange Format Working Group
[6] Internet Engineering Task Force – `http://www.ietf.org`
[7] Intrusion Detection Message Exchange Protocol)
[8] Intrusion Detection Exchange Protocol
[9] Block Extensible Exchange Protocol

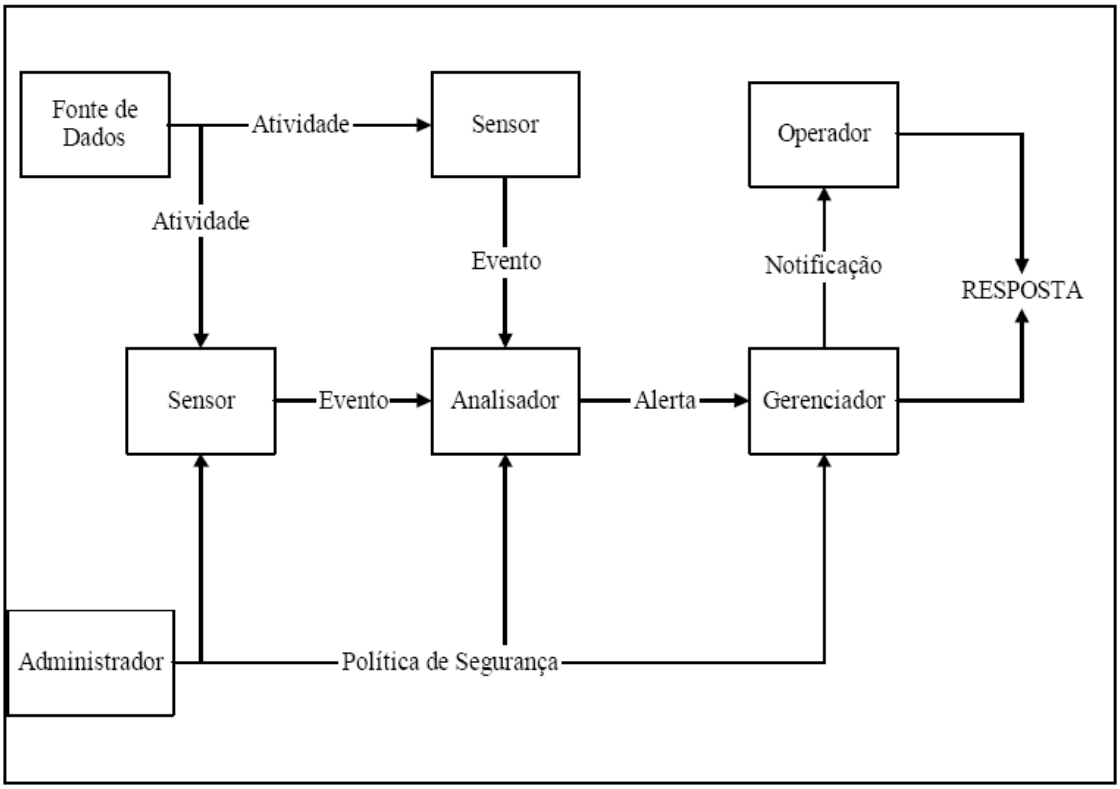


Figure 3: Componentes de um IDS segundo o IDWG.

Neste modelo, tem-se os seguintes elementos:

- Fonte de dados: elemento onde ocorrem atividades relacionadas com o processamento no sistema;
- Sensor: elemento responsável pela geração de eventos a partir das atividades na fonte de dados e que deverão ser utilizados para detecção de intrusões e auditoria;
- Analisador de Eventos: elemento responsável pela detecção de intrusão. Este elemento recebe os dados dos sensores e faz uma análise, buscando anomalias ou padrões que caracterizam um possível ataque;
- Gerente: elemento encarregado pela configuração do sistema e pela notificação dos eventos ocorridos ao operador;
- Operador: usuário responsável pela análise dos alertas gerados e encarregada da resposta ao ataque;

- Administrador: usuário responsável pela definição e manutenção das políticas de segurança.

No modelo de IDS atualmente proposto pelo IDWG, apenas o formato de eventos (alertas) está especificado. Caso o operador queira tomar ações contra um evento, terá que fazê-las separadamente em cada IDS de seu ambiente. Isto dificulta o gerenciamento dos eventos e causa um atraso na execução das contra-medidas, exigindo que o operador tenha conhecimentos específicos sobre cada um dos IDSs instalados em seu ambiente.

No modelo apresentado na figura 3 observamos que o sensor, o analisador e o gerente devem estar presentes em todos os sistemas IDSs pois correspondem à coleta, análise dos dados e tomada de decisão, sendo estas funcionalidades básicas de qualquer IDS. Em alguns IDSs, os elementos do modelo IDWG são decompostos em vários sub-elementos, porém as funções básicas são sempre mantidas. Alguns modelos definem ainda interfaces, geradores de alarmes e outros elementos.

As funções de respostas de um IDS podem ser divididas em:

- Respostas ativas - que são ações automáticas tomadas quando certos tipos de intrusões são detectadas.
- Respostas passivas - fornecem respostas aos usuários do sistema dependendo de intervenção humana para a execução de ações subseqüentes. Existem diversos IDSs comerciais baseados unicamente em respostas passivas.

O uso de respostas ativas do sistema ao ataque em tempo real, reconfiguração de *firewall* ou o encerramento de um processo pode ser perigoso. Se o sistema sempre reage automaticamente recusando uma conexão de rede ou de uma máquina que ele suspeita ser responsável pelo ataque, então existe o risco de um atacante criar ataques de negação de serviços, fazendo um *spoofing*[10] de uma fonte crítica.

Muitos IDSs optam pelo uso de uma abordagem de resposta passiva ao ataque, onde é feita a coleta de informações e identificação do sujeito, sem modificação (em

[10] Ocorre quando um atacante forja o seu endereço IP se passando por outra interface de rede.

tempo real) do sistema. Nesta abordagem, o sistema apenas envia um relatório (via *e-mail*) informando possíveis problemas detectados naquele período, repassando ao administrador da rede a responsabilidade de reação ao incidente ocorrido.

3.4: Abordagens para Coleta de Dados

As duas principais abordagens para a coleta de dados em IDSs são: baseada em *host* ou na rede. Do ponto de vista da fonte de dados um sistema de detecção de intrusão pode ser classificado em um dos três tipos básicos:

- baseado em *host*;
- baseado em rede;
- ou Híbrido.

Nas próximas seções apresentamos as características presentes nos IDSs de cada um dos tipos citados.

3.4.1: IDSs com coleta baseada em *Hosts*

Sobre os IDSs com coleta de dados em *host*, o TCSEC [11] (*Trusted Computer System Evaluation Criteria*), estabeleceu alguns critérios para auditoria, relacionando uma lista dos principais eventos e informações que devem ser auditados. Esta lista inclui o uso de identificação e autenticação, introdução de objetos dentro do espaço de endereçamento dos programas (abertura de arquivos ou execuções de programas), remoção de objetos, ações administrativas e outros eventos relevantes à segurança.

A coleta de dados baseada em *hosts* é uma forma de reunir dados em máquinas, através da análise de registros de eventos associados com processamentos locais (por exemplo, operações de arquivos, acessos a bibliotecas de sistema, etc.).

Em geral, a coleta de dados no *host* é feita de modo contínuo, durante a utilização do sistema operacional. Contudo, coletas periódicas de estados do sistema também podem ser úteis no registro de mudanças inesperadas. Os sistemas de coleta podem ser

um módulo específico de um IDS e não devem expor os dados coletados, necessitando portanto de certa medida de proteção. A seleção das informações e eventos a serem auditados e o nível de detalhes a ser registrado são apenas algumas das questões que precisam ser contempladas por um IDS baseado em *host* ou simplesmente HIDS[11]. Exemplos típicos de dados auditados são chamadas de sistema (*system calls*, seus parâmetros e a ordem em que aparecem), estatísticas de utilização de recursos e/ou *logs* do sistema.

Em sistemas que realizam auditorias para a detecção de padrões não usuais, existe uma relação direta entre o nível de detalhes de uma coleta de dados com o desempenho do sistema e o espaço de armazenamento necessário para guardar os dados coletados.

Análises posteriores (*off-line*) também serão afetadas pela quantidade de dados coletados. No entanto, realizar a coleta de poucos dados aumentará o risco de existência de manifestações ou ataques sem serem detectados pelo HIDS.

3.4.2: IDSs com coleta baseada em Rede

Um sistema de detecção de intrusão baseado em rede (ou simplesmente NIDS[12]), monitora o segmento de rede e analisa o tráfego dos fluxos de dados transmitidos.

Na coleta de dados baseada em rede, o IDS observa o tráfego de rede e procura por sinais de intrusão nos pacotes analisados. Esta abordagem apresenta a vantagem de um único sensor poder monitorar várias máquinas em uma rede e com facilidade detectar ataques que tentam explorar brechas de segurança ou que visam a degradação nos serviços de rede.

As técnicas de detecção de um NIDS podem analisar diferentes dados: *streams* de *bytes* sendo transmitidos, propriedades de cada conexão, dados de cabeçalho (*header*) que incluem endereços de origem e de destino, bytes transmitidos entre pontos da rede, tempo de conexão ou analisar o conteúdo (*payload*) ou carga útil de cada pacote do tráfego da rede.

[11] Host based Intrusion Detection System
[12] Network Intrusion Detection System

Um NIDS é considerado uma barreira de defesa eficaz contra ataques diretos a computadores e redes. Devido ao aumento da criticidade e possibilidade de ocorrência de ataques na Internet os NIDSs têm sido amplamente utilizados em quase todas as infra-estruturas de TI de larga escala [1].

A principal vantagem da abordagem baseada em rede é a possibilidade de monitoramento de dados e eventos sem afetar significativamente o desempenho de um único *host*.

Contudo, estes sistemas também apresentam desvantagens entre as quais podemos mencionar a dificuldade de detecção de ataques feitos a partir de um console ligado ao *host* e de ataques a serviços que utilizam criptografia (por exemplo, SSH e SSL) no canal de comunicação.

Apesar desses problemas, o monitoramento de rede é a abordagem escolhida pela maioria dos sistemas de detecção de intrusão. Os motivos desta escolha se baseiam na facilidade de construção de uma plataforma dedicada que combina sensores de rede com redução de dados e análise. A infra-estrutura necessária se resume a uma interface de rede para captura de tráfego de rede e alguns mecanismos para capturar o tráfego ou parte dele através do uso de filtros de pacotes. A biblioteca *libpcap* [36] pode ser utilizada para a construção do módulo de captura de dados na interface de rede.

Nos últimos anos, o volume de tráfego nas redes está aumentando muito rápido e a capacidade de transmissão também. Tanto que a capacidade de transmissão nas redes passou de 10 megabits/s para 100 megabits/s e muitas já estão operando a 1000 megabits/s. Considerando que a velocidade para a análise dos dados (CPU dos computadores) não aumentou na mesma proporção, a análise detalhada do tráfego em uma rede gigabit pode causar um atraso considerável. É uma prática comum realizar a coleta e a análise do tráfego em uma única plataforma, o que torna o NIDS vulnerável. Ainda outro ponto é o fato de que os sensores de rede também são alvos de ataques.

Quando falamos de redes de comunicação entre máquinas também nos referimos aos protocolos utilizados para este fim. Por exemplo, o protocolo TCP/IP possui algumas ambiguidades que são resolvidas de maneira diferente em cada sistema operacional. Com isso, uma análise de intrusão feita por um NIDS pode apresentar resultados

divergentes quando comparada com a mesma análise em uma plataforma distinta. Todas estas limitações resultam em ataques chamados de evasão e inserção, conforme formalizado por Ptacek e Newsham [62].

3.5: Abordagens para Detecção de Intrusão

A seguir apresentamos as abordagens de detecção de intrusão mais usadas, que são basicamente divididas em detecção baseada em assinaturas e a baseada em anomalias.

3.5.1: Detecção de Intrusão baseada em Assinaturas

Uma forma de detecção de intrusão bastante utilizada é a baseada em assinaturas. Nesta abordagem o IDS possui uma base de dados com registros de atividades intrusivas conhecidas chamadas de assinaturas. Durante a análise das informações do sistema (por exemplo, o tráfego da rede ou registros de log do sistema operacional), o IDS compara as informações com a base de assinaturas a fim de identificar a atividade atual com algum registro existente em sua base. Quando encontra alguma assinatura que seja similar a atividade detectada no ambiente monitorado, o sistema gera um alerta.

Esta abordagem geralmente apresenta bons resultados em termos de baixas taxas de falsos positivos, mas também apresenta limitações.

Em muitos sistemas, novos ataques ou variações pequenas dos ataques conhecidos podem passar sem detecção pelo IDS até que sua base de assinaturas seja atualizada com um registro sobre o novo tipo de ataque. Um sistema de detecção de intrusão baseado em assinaturas precisa então ter sua base de assinaturas atualizada regularmente, toda vez que novos ataques forem descobertos.

3.5.2: Definição de Assinaturas

Além das dificuldades abordadas, desenvolver uma assinatura para um ataque não é uma tarefa trivial. Uma vez que um ataque se torna conhecido, é necessária uma análise cuidadosa por algum especialista. O ataque pode explorar uma vulnerabilidade

bem conhecida ou uma nova vulnerabilidade ainda não explorada. Uma assinatura tem o objetivo de descrever a maneira como um ataque explora determinada vulnerabilidade ao invés de apenas analisar o efeito de um ataque. Por exemplo, um ataque de *buffer overflow* pode ser realizado usando-se diferentes vetores de ataque, mas estes vetores deverão apresentar um tamanho mínimo em bytes. Modificar o conteúdo do *payload* do ataque é algo relativamente simples para um atacante, embora isso não altere o efeito do ataque. Um ataque desse tipo pode ser detectado com uma (ou algumas) assinaturas.

Porém, abstrair o ataque nem sempre é possível e geralmente exige uma boa dose de trabalho: é difícil encontrar o equilíbrio certo entre uma assinatura muito específica (que não será capaz de detectar uma variação simples do ataque) e uma sobreposição de uma assinatura mais geral (que pode classificar tráfego legítimo como uma tentativa de ataque).

Para detectar intrusões, um IDS baseado em assinaturas precisa então manter um registro de descrições dos ataques que busca detectar. Estas descrições podem ser simples e associar um padrão específico a determinado ataque ou complexas quando mapeiam o estado de uma máquina ou a descrição de uma rede neural a múltiplos sensores contendo representações abstratas [51].

Quando determinada assinatura é associada com uma abstração de ataque conhecido, passa a ser relativamente simples para um detector baseado em assinaturas ligar atividades a tipos conhecidos de ataques (um exemplo, é o Ping da morte) [4]. Os detectores STAT [82], utilizam uma representação abstrata de algumas classes de ataques na tentativa de reconhecer novos ataques e gerar suas respectivas assinaturas.

Embora a desvantagem em tais sistemas seja a regularidade de atualização da base de assinaturas (sempre que novos ataques são descobertos), grande parte dos sistemas comerciais segue esta categoria e são frequentemente atualizados.

3.5.3: Detecção de Intrusão baseada em Anomalias

Um IDS baseado em anomalias constrói um modelo que descreve o comportamento normal do sistema/rede monitorados. Um sistema baseado em anomalias fun-

ciona com base no seu treinamento e capacidade de reconhecer padrões aceitáveis como comportamentos próximos do comportamento normal modelado e então dispara alertas para qualquer comportamento que difere do modelo normal.

A principal vantagem da abordagem baseada em anomalias é que o sistema é capaz de detectar novos ataques e suas respectivas variações quando estas ocorrem no ambiente monitorado pelo IDS.

Por outro lado, a principal desvantagem da abordagem baseada em anomalias é que, devido à natureza estatística de seu modelo, o sistema pode disparar muitos falsos positivos.

A detecção de intrusão baseada em anomalias assume que intrusões serão acompanhadas de manifestações que não são usuais e assim permitem a sua detecção. Infelizmente, nem sempre é isto o que acontece. De acordo com os estudos apresentados por Anderson [3] e Myers [54] é necessário estar ciente das altas habilidades que um intruso pode ter e que este tentará entender o sistema e evitar a sua detecção.

Os sistemas de detecção por anomalias podem fazer uso de diversas técnicas: séries temporais, regras ou ainda podem usar estatísticas descritivas. Os sistemas que utilizam séries temporais podem empregar técnicas de Inteligência Artificial (IA), como as de redes neurais, máquinas de vetores suporte e as técnicas bio-inspiradas de sistemas imunológicos.

O grande desafio em IDSs baseado em anomalias é a dificuldade de definição e representação das atividades normais de um principal. Esta abordagem consome diversos recursos computacionais para guardar e verificar as atividades no sistema que está sendo monitorado.

Antes de passarmos a consideração da classificação dos IDSs baseados em anomalias, temos a Tabela 3 que resume as vantagens e desvantagens da detecção por anomalias e da baseada em assinaturas.

Abordagens de análise em IDSs		
Abordagem de Detecção	Vantagens	Desvantagens
Baseada em Assinaturas	Baixa taxa de falsos positivos. Não necessita de treinamento para classificar alertas.	Não detecta novos ataques. Necessita de atualizações constantes. As atualizações podem se caracterizar como tarefas não triviais.
Baseada em Anomalias	Pode detectar novos ataques. Aprendizado automático.	Propenso a altas taxas de falsos positivos. Necessita de treinamento inicial.

Table 3: Vantagens e desvantagens da análise baseada em Assinaturas versus Anomalias - conforme apresentado em [8].

3.5.4: Classificação de IDSs baseados em anomalias

Depois de abordarmos os NIDSs temos também os ANIDSs. Um *Anomaly Network Intrusion Detection System* – ANIDS pode extrair informações para detecção de ataques das seguintes fontes: cabeçalhos de pacotes, conteúdo (*payload*) de pacotes ou ambos.

As informações do cabeçalho de pacotes são especialmente úteis no reconhecimento de ataques que visam explorar vulnerabilidades existentes nas implementações das pilhas de protocolos por diferentes plataformas.

IDSs baseados em anomalias que utilizam técnicas de análise do *payload* possuem certa relevância, pois são particularmente adequados para detecção de ataques específicos (por exemplo, ataques de *worms* e ataques baseados no protocolo HTTP). Para uma discusão detalhada consulte [83].

Um ANIDS pode ser classificado de acordo com:

- (a) o algoritmo subjacente que utiliza;
- (b) se analisa as características de cada pacote separadamente ou todo o pacote (*header* e *payload*), e como estes dados são correlacionados;

Com relação ao algoritmo subjacente, em [22] e [21] definem-se quatro abordagens diferentes, das quais apenas duas têm sido empregadas com bons resultados na última década: algoritmos baseados em modelos estatísticos e algoritmos baseados em redes neurais. A primeira é a mais utilizada: ainda de acordo com [22] mais de 50% dos ANIDSs existentes utilizam modelos estatísticos. Um ANIDS baseado em redes neurais funciona de maneira similar, porém em vez de construir um modelo estatístico a rede neural é treinada e depois utilizada para reconhecer entre tráfego normal e ataques.

Referente ao item (b), deve-se fazer a distinção entre sistemas com análise orientada a pacotes ou orientada a conexões. Um sistema orientado a pacotes usa cada pacote como fonte mínima de informação, enquanto que um sistema orientado a conexão considera as características de toda a comunicação para detectar se esta é um ataque ou comportamento normal. Teoricamente, um sistema orientado a conexões pode usar como entrada o conteúdo (*payload*) de toda a comunicação (possibilitando em princípio, uma análise mais precisa), mas isto exige maior esforço computacional (CPU e memória) e tempo de processamento, podendo limitar o sistema e introduzir alguma latência extra.

Na prática, um sistema orientado a conexão leva em conta o número de bytes enviados/recebidos, a duração da conexão e o protocolo utilizado. Os resultados obtidos por Wang e Stolfo em termos de desempenho [71] revelam que um sistema baseado em análise de *payload* não apresenta um aumento significativo de desempenho quando se analisa todas as conexões ao invés de analisar apenas pacotes isoladamente. Na prática, muitos ANIDSs são orientados a pacotes.

A última e praticamente a mais relevante distinção que podemos fazer é se o ANIDS é *header-based* ou *payload-based*. Um ANIDS baseado no *header* analisa o cabeçalho para detectar atividade maliciosa na rede. Um ANIDS baseado no *payload* analisa o conteúdo carregado pelos protocolos de transporte da camada de rede. Além

desses dois tipos existem sistemas híbridos que realizam uma análise mista tanto do *header* quanto do *payload* de dados dos pacotes observados na rede.

A próxima seção apresenta brevemente a diferença dos NIDSs com análise baseada no *payload* ou apenas no *header* dos pacotes de tráfego de rede que monitoram.

3.5.5: IDSs com análise de *Payload* vs *Header*

Existem ataques que somente os NIDSs que analisam o *payload* são capazes de detectar. Um exemplo de ataque que só pode ser detectado através da análise do *payload* é o de *SQL Injection*, o qual explora vulnerabilidades de uma aplicação web que realiza interface direta com um banco de dados. Neste tipo de ataque, pode-se injetar uma consulta SQL para forçar o retorno de informações confidenciais, tais como senhas, nomes de usuários ou executar comandos arbitrários (*drop* no banco de dados, *delete*, *update*, etc.) com privilégios de administrador.

Os sistemas SSAD, PAYL e POSEIDON são sistemas capazes de detectar ataques conduzidos via *payload*.

Por outro lado, alguns IDSs existentes os quais podemos citar: iSOM, IntelligentIDS, PHAD e SSAD, são capazes de analisar e detectar ataques ao nível do *header* das mensagens trocadas entre os *hosts* em uma rede. Exemplos de ataques realizados através do *header* dos pacotes são: *teardrop*, um ataque de *Denial of Service* que explora uma falha na implementação do protocolo TCP/IP em algumas plataformas. Estas implementações não tratam adequadamente de fragmentos IP que se sobrepoem. O atacante envia pacotes forjados e fragmentados que se sobrepoem e impedem que o *host* que os recebe consiga remontá-los do outro lado. Se o *host* não verifica os limites propriamente, ele tentará alocar um bloco de memória com um tamanho negativo, causando um *kernel panic* e/ou *crash* em determinados sistemas operacionais. Um IDS pode detectar este tipo de ataque apenas por analisar o *header* de dois datagramas IP fragmentados. Este ataque explora uma vulnerabilidade da camada do protocolo IP da pilha de protocolos de rede.

Após analisarmos brevemente as diferenças e a aplicação dos IDSs baseados

em *header* e *payload*, vamos abordar a seguir exemplos de IDSs utilizados e em fase de aprimoramentos.

3.6: Exemplos de IDSs

Muitos IDSs atualmente em uso, são baseados em rede e utilizam análise de detecção empregando assinaturas. Administradores de sistemas muitas vezes preferem IDSs de rede baseados em assinaturas (SNIDS[13]) ao invés dos NIDSs baseados em anomalias (ANIDS[14]) pois segundo alguns autores - são de fácil desenvolvimento, simples configuração e manutenção, apesar do fato de um SNIDS poder gerar muitos falsos negativos.

Contudo, a cada dia novos ataques são planejados com uma crescente frequência, o que torna a abordagem de uso de ANIDS bastante atrativa. A seguir, passamos a abordar aspectos relevantes de alguns dos sistemas IDSs mais utilizados.

Os sistemas analisados nas próximas subseções foram divididos de acordo com sua abordagem de detecção (assinaturas ou anomalias) e fonte de informação (host, rede ou híbrida).

3.6.1: Snort

O **Snort** [64] é um *software open source* para detecção de intrusão baseada em assinaturas, eficiente e que funciona em qualquer sistema *Unix* e utiliza a abordagem baseada em rede.

O Snort pode ser usado para análise de protocolos de rede e também para detectar uma variedade de ataques ou tentativas de ataques de estouro de buffer, varredura de portas, ataques de código CGI[15], tentativas de identificação da versão do sistema operacional ou ataques aos serviços de rede. Este sistema tem a capacidade de enviar alertas em tempo real. Tais alertas podem ser registrados no servidor de registros *syslog*, em um arquivo de alerta ou enviados ao administrador.

[13] Signature Network Intrusion Detection System
[14] Anomaly Network Intrusion Detection System
[15] Common Gateway Interface

O Snort NIDS é uma ferramenta baseada em uma linguagem orientada a regras e capaz de executar análises de tráfego e registro de pacotes em tempo real, combinando os benefícios dos métodos de inspeção de tráfego baseados em assinaturas e em anomalias.

Quando os pacotes chegam na interface de rede, existe uma série de estágios internos do Snort que estes percorrerão até que o sistema possa gerar um alerta, *log* ou descarte. O Snort pode assumir três modalidades de funcionamento:

1. *Sniffer*: captura de pacotes e exibição contínua em console;
2. *Packet logger*: arquivamento dos pacotes em disco para análise posterior;
3. *Network intrusion detection system*: análise do tráfego da rede comparando com as regras definidas pelo administrador, executando diversas verificações.

Quando o sistema detecta uma atividade suspeita, outros componentes (módulos de saída) irão gerar alertas e registrar as ocorrências em arquivos de *logs*. Do contrário o pacote será descartado pelo sistema.

Os sensores do *Snort* são capazes de comunicar-se facilmente com outros IDSs e sistemas de gerência pois as mensagens de alertas são geradas usando o padrão IDMEF [20].

3.6.2: Panacea

Conforme apresentado em [8], o Panaceia é um ANIDS que analisa o *payload* através do uso de técnicas de *machine learning* para classificar de maneira automática ataques detectados e gerar alertas de modo automático.

A idéia principal é a seguinte: ataques em geral compartilham traços comuns, ou seja, algumas seqüências de bytes do *payload* são geralmente encontradas em ataques pertencentes a mesma classe. Assim, ao extrair seqüências de bytes dos *payloads* que geram alertas pode-se comparar estas seqüências com dados previamente coletados usando-se um algoritmo apropriado na busca de *payloads* de alertas similares, e então inferir a classe a que o ataque pertence. O sistema Panacea é um ANIDS capaz de:

- Classificar automaticamente ataques detectados por um IDS baseado em anomalias, sem usar heurísticas pré-definidas;
- O sistema não necessita da assistência de um especialista humano para classificar os ataques.

No entanto, o sistema Panacea necessita de uma fase de treinamento para construir seu motor de classificação automática de ataques. Após a fase de treinamento, o sistema classifica ataques de maneira automática como qualquer IDS baseado em anomalias.

Com isso, qualquer alerta gerado por atividades/ataques que não envolva diretamente o *payload* (por exemplo, *port scan*, ou um Distributed Denial of Service – DDoS) não poderá ser classificado automáticamente pelo sistema. Esta não chega a ser uma grande limitação, uma vez que os ataques mais prejudiciais são aqueles que injetam dados nos sistemas alvos (aplicações). Contudo, o Panacea não consegue operar com um IDS baseado em anomalias que não analisa o *payload* dos pacotes coletados.

A arquitetura do Panacea é composta de dois componentes que interagem entre si: o Extrator de Informações de Alertas (*Alert Information Extractor* – AIE) e o Motor de Classificação de Ataques (*Attack Classification Engine* – ACE). O AIE recebe alertas de um ou mais IDSs, analisa os *payloads* e extrai informações relevantes, gerando alertas de saída na forma de meta-informações. Estas são repassadas para o ACE, que determina automaticamente a classe do ataque.

O processo de classificação é executado em dois estágios. No primeiro, o ACE é treinado com diversos tipos de meta-informações de alertas usadas para construir o modelo de classificação. O módulo ACE é alimentado com meta-informações e sua classe correspondente (no caso de IDSs baseados em assinaturas). No segundo estágio, quando o treinamento tiver sido finalizado, o ACE está preparado para classificar automaticamente os novos alertas que recebe.

A implementação do protótipo do sistema Panacea foi realizada utilizando-se a linguagem Java e a ferramenta Weka[16], a qual possui uma ampla coleção de algoritmos

[16] `http://www.cs.waikato.ac.nz/ml/weka/`

de aprendizado de máquina e contém uma implementação dos algoritmos de Support Vector Machines[17] e do algoritmo RIPPER utilizado para geração de regras.

3.6.3: ATLANTIDES

O ATLANTIDES (*Architecture for Alert verification in Network Intrusion Detection Systems*) é uma arquitetura de gerência de qualquer NIDS (seja baseado em assinaturas ou em anomalias) para ajudar a reduzir, de modo automático, o número de falsos positivos gerados por um NIDS. A idéia principal do ATLANTIDES é simples: um ataque bem sucedido causa uma anomalia na saída do serviço comprometido, modificando a saída normal esperada. Detectar este tipo de anomalia, pode ajudar a reduzir a taxa de falsos alertas. Por exemplo, um ataque de *SQL Injection* bem sucedido contra uma aplicação web pode causar a saída de uma tabela de conteúdo SQL (por exemplo, com credenciais usuário/administrador) em vez do conteúdo esperado para o serviço web.

O ATLANTIDES é totalmente baseado em rede e verifica o tráfego utilizando a análise n-gram para modelar os *payloads* de saída normal dos serviços disponíveis na rede monitorada que deverão ser observados em resposta às solicitações de um ou mais clientes. A saída normal é específica de cada site, portanto os modelos derivados refletem - de alguma forma - o contexto do serviço ou da rede. Por correlacionar as anomalias detectadas na saída com os alertas gerados pelo NIDS com base no tráfego de entrada, o ATLANTIDES pode descartar um número considerável de falsos alertas. Isto ajuda o sistema a obter menos falsos positivos que em uma abordagem de IDS sem este mecanismo de correlação de anomalias.

Em um trabalho anterior a este, a correlação simples entre o tráfego de entrada e o de saída foi utilizada para identificar possíveis ataques de *worms* [71].

O ATLANTIDES é a primeira solução de verificação de alertas que trabalha com uma combinação de análise baseada tanto em assinaturas quanto em anomalias e que opera de modo completamente automático após uma fase inicial de configuração, sem qualquer envolvimento humano, melhorando a usabilidade e facilitando a gerência do NIDS.

[17] Apresentado em detalhes no capítulo 5 deste trabalho.

Nos testes realizados com o ATLANTIDES e o Snort aplicando-se os dados do KDD 1999, conforme podem ser consultados em [8], as taxas de falsos positivos puderam ser reduzidas numa faixa de 50% a praticamente 100%.

Entre as principais limitações do ATLANTIDES, encontra-se o fato de ele se basear na análise do *payload* de saída com uma arquitetura projetada para analisar serviços cliente/servidor sobre TPC (por exemplo, HTTP). Como toda análise é baseada em *payload*, o ATLANTIDES não é capaz de operar normalmente com dados criptografados.

A arquitetura de verificação de alertas do ATLANTIDES é composta de um componente externo e dois componentes internos. O componente externo é um NIDS que monitora o tráfego de chegada no sistema.

O primeiro componente interno é um detector de anomalias na saída (*Output Anomaly Detector* - OAD), que é na verdade um NIDS baseado em anomalias que monitora o tráfego de saída. O OAD emprega um modelo estatístico para descrever a saída normal dos serviços de rede, flags e quaisquer comportamentos que variem significativamente do normal, sendo estes apontados como possíveis ataques. O segundo componente interno, é o mecanismo de correlação (*Correlation Engine* – CE) que rastreia conexões de rede usando inspeção baseada no estado da conexão [76] e faz a correlação de alertas no tráfego de chegada analisado pelo NIDS com a saída produzida pelo módulo OAD.

3.6.4: POSEIDON

O POSEIDON (*Payl Over Som for Intrusion DetectiON*) [10] é um ANIDS de duas camadas. A primeira camada é composta de um Self-organizing Map[18] (SOM), utilizado exclusivamente para classificar os dados do *payload*. A segunda camada é composta de uma pequena modificação no bem conhecido algoritmo PAYL [71]. O algoritmo PAYL utiliza informações de tamanho do *payload* para classificar cada pacote e definir modelos de detecção.

[18] Apresentado em detalhes no capítulo 4

Este sistema é baseado em análise do *payload* e utiliza apenas o endereço de destino e o número da porta do serviço para construir o perfil de cada *host* monitorado, sem considerar outras características do cabeçalho.

A arquitetura do POSEIDON combina uma rede SOM com os modelos usados no algoritmo PAYL conforme a abordagem apresentada em [8]. Primeiro, a rede SOM pré-processa cada pacote sendo alimentada com os dados de *payload*. Todo o conteúdo de cada pacote é analisado pela rede SOM através da comparação com o vetor de pesos de cada neurônio. Uma vez que o tamanho do *frame* de rede no padrão Ethernet de uma LAN[19] pode conter até 1460 bytes onde cada neurônio tem um vetor de pesos com este tamanho fixo. No caso dos dados de um *payload* ser menor que este tamanho, os bytes restantes são marcados com valor (nulo) para não serem incluídos na análise.

A rede SOM retorna o valor do neurônio mais similar (*winning neuron*). Depois da classificação pela rede SOM, na segunda camada, o algoritmo PAYL seleciona o modelo usando o valor do neurônio da rede SOM ao invés de usar o tamanho do *payload* (como era realizado no algoritmo PAYL original).

Ao invés de usar o modelo matricial $\mathrm{M_{ijl}}$ onde l indica o tamanho do *payload* analisado, nesse caso o algoritmo PAYL modificado usa o modelo matricial $\mathrm{M_{ijn}}$ onde i e j são o endereço IP de destino e a porta, com n sendo a classificação derivada da rede neural. Então a freqüência relativa em bytes e a derivação padrão dos valores são computadas normalmente.

O treinamento do SOM e do módulo PAYL podem ser realizados de modo *off-line*, usando-se os mesmos dados para treinar a rede SOM e o módulo PAYL ou usando-se dados de treinamento diferentes para os compontentes separadamente. Além disso, é possível treinar o sistema de modo *on-line*, através da captura do tráfego de rede. Neste caso (treinamento *on-line*), o SOM e o PAYL são treinados com dados diferentes. Em relação ao consumo de memória exigído pelo POSEIDON, deve-se levar em consideração que a rede SOM precisa ser armazenada seguindo a seguinte fórmula: $\mathrm{n} * \mathrm{l} * \mathrm{k}$, onde n é o número de neurônios da rede, l é a média do tamanho dos *payloads* e a constante k representa o espaço necessário em memória física para armazenar um número

[19] Local Area Network

de ponto flutuante. Assim, o montante total de memória requerida pelo POSEIDON é calculado com: $n * l * k + i * j * n * 2 * 256 * k$.

3.6.5: Sphinx

O Sphinx é um ANIDS que utiliza uma abordagem de detecção de anomalias por analisar os fluxos de dados trocados entre o cliente e o servidor que disponibiliza alguma aplicação web.

A arquitetura do Sphinx explora o fato de que em geral, muitos parâmetros presentes nas requisições HTTP seguem padrões regulares ou irregulares (vide figura 4). Por considerar estas regularidades, seus projetistas dividiram os parâmetros em regulares ou irregulares. Padrões regulares são aqueles que não variam com muita freqüência. Por outro lado, os padrões irregulares sempre variam. Neste sistema, a construção dos modelos de detecção de intrusão é feita com base na exploração dos parâmetros regulares.

O Sphinx aplica um conceito de ”assinatura positiva”, que indica solicitações de serviço legítimas. O Sphinx implementa um algoritmo de inferência e geração de assinaturas positivas através do uso de autômatos que criam um gerador de expressões regulares.

Estas assinaturas são usadas depois pelo Sphinx para construir modelos de detecção automáticos que auxiliam na tarefa de detecção de anomalias nos parâmetros considerados regulares.

Diferentemente dos modelos matemáticos e estatísticos, uma assinatura positiva não utiliza nenhuma medida de freqüência, presença, observação ou limiar para gerar o modelo de detecção.

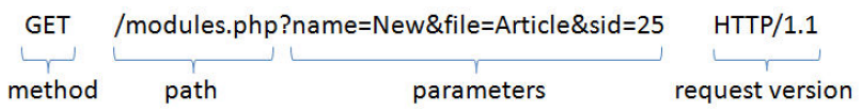


Figure 4: Uma requisição GET HTTP típica contendo parâmetros e seus respectivos valores.

Na Figura 4 os parâmetros são: name, file e sid e seus respectivos valores são: *New*, *Article* e *25*. O conjunto de parâmetros é finito. Um valor pode ser qualquer string, contudo, nem todas as strings possíveis são válidas. Uma vez que o tipo não é pré-definido, a semântica de cada parâmetro é implicitamente definida dentro do contexto da aplicação web e tais parâmetros são em geral usados de maneira simples e consistente (ou seja, sua sintaxe e ordem é fixada). Com isso em mente, o Sphinx utiliza uma função de mapeamento definida da seguinte maneira: dada uma entrada $i = path?p_1 = v_1\&p_2 = v_2\&...\&p_n = v_n$, onde $p_n(i) = v_n$ é a função que extrai o valor do parâmetro p_n do padrão de entrada i.

No caso dos parâmetros considerados irregulares, o Sphinx analisa seu conteúdo usando uma versão adaptada do NIDS POSEIDON.

Os resultados apresentados em [8] mostram que a abordagem de assinaturas positivas é capaz de detectar ataques com uma baixa taxa de falsos positivos ao analisar os parâmetros regulares.

3.6.6: SilentDefense

O SilentDefense é um IDS composto por vários módulos os quais também são IDSs individuais. Os módulos do SilentDefense podem ser formados a partir dos IDSs: ATLANTIDES, POSEIDON, Panacea e o Sphinx. O sistema SilentDefense utiliza o IDS ATLANTIDES para reduzir a taxa de falsos positivos[8]. Utiliza o sistema POSEIDON para melhorar a taxa de detecção e o sistema Panacea para realizar a classificação dos alertas gerados de maneira automática.

O SilentDefense inclui ainda um módulo que nada mais é do que uma extensão do sistema Sphinx que possui um motor de detecção de ataques a aplicações Web e que pode ser facilmente configurado através de interfaces com usuário.

O SilentDefense é uma iniciativa para melhorar o uso dos IDSs baseados em anomalias, tornando o mesmo mais adequado ao que esperam os administradores de redes. Este sistema utiliza uma versão melhorada do algoritmo PAYL (apresentando um aumento na taxa de detecção de 90% para 100% e uma diminuição da taxa de falsos

positivos de 0.17% para 0.0016%) através do uso de uma rede neural que pré-processa o tráfego de rede analisado.

Este sistema apresenta taxas de falsos positivos com valores entre 50% e 100% menores quando comparados com sistemas como o Snort e POSEIDON devido a utilização do ATLANTIDES, IDS que cruza os alertas gerados com o tráfego de entrada e de saída monitorado na rede.

Além disso, o SilentDefense possui um módulo capaz de detectar ataques a aplicações web através da geração de expressões regulares, que podem ser editadas pelos administradores para ajustar o mecanismo de detecção de anomalias e validar os valores das requisições HTTP recebidas.

Por fim, o SilentDefense utiliza também um módulo de classificação automática de alertas, extrai informações do *payload* e classifica os dados usando taxonomias pré-definidas pelo sistema ou pelo administrador da rede. Uma vantagem deste sistema é que cada componente pode operar de modo separado, ou em cooperação com os demais componentes, sendo que o IDS POSEIDON é o componente central do SilentDefense. Uma das principais capacidades do SilentDefense é a alta taxa de detecção de novos ataques e identificação de *hosts* atingidos por estes ataques enquanto que abordagens baseadas em assinaturas funcionam melhor para aplicações e sistemas bem conhecidos e amplamente utilizados.

3.6.7: Tripwire

Um exemplo de IDS baseado em *host* é o Tripwire[20] que é um *software* de verificação dos efeitos resultantes de uma atividade de intrusão em um sistema.

Após a instalação do *Tripwire*, uma base de dados é criada com informações dos arquivos que devem ser monitorados periodicamente. A base de dados criada é criptografada e armazena informações críticas ao sistema, as quais incluem os tamanhos dos arquivos e seus respectivos *checksums*.

Durante a análise realizada pelo sistema *tripwire*, ele compara a informação cor-

[20] http://www.tripwire.com

rente com a gerada anteriormente e identifica as mudanças nos arquivos.

Ao final da análise, o sistema gera um relatório dos arquivos que foram modificados, porém o administrador é quem deve decidir quais as modificações devem ter sido causadas por intrusões.

3.6.8: OSSEC

O OSSEC[21] é um outro exemplo de *Host* IDS usado para análise de *logs*, detecção de *rootkits*, verificação de integridade do sistema, geração de alertas e respostas pró-ativas. O OSSEC é capaz de executar estas operações e gerar alertas e respostas ativas em tempo real. Este sistema possui três modos de operação:

1. Modo local (*standalone*) – opera apenas na máquina onde o OSSEC está instalado.

2. Modo Agente – funciona como cliente, enviando informações para o servidor processar e analisar. Serve para centralização dos *logs* no servidor, sendo estes monitorados pelo OSSEC. A comunicação entre o agente e o servidor é feita com mensagens criptografadas sobre o protocolo UDP[22].

3. Modo Servidor – analisa e une os *logs* e informes de vários agentes. Neste modo, os múltiplos pontos de gerenciamento de *logs* são centralizados em um único ponto de análise de alertas.

A comunicação entre os agentes e o servidor pode ser feita via texto plano ou no modo criptografado usando chave simétrica e única com gerenciamento próprio das chaves do OSSEC.

O OSSEC suporta vários tipos de *logs*. Possui um módulo detector de *rootkits*[23] (*syscheckd*) que é um *scanner* que utiliza tanto identificação baseada em assinaturas quanto baseada em anomalias para identificar *rootkits*.

[21] `http://www.ossec.org`

[22] User Datagram Protocol

[23] Um rootkit é um tipo de *malware* que busca se esconder de *softwares* de segurança e do usuário utilizando diversas técnicas avançadas de programação.

Para a identificação baseada em assinaturas este módulo usa arquivos onde são detalhadas as características únicas de vários tipos de *rootkits* e *Trojan horses*.

Já a identificação baseada em anomalias deste módulo, utiliza um enfoque mais elaborado, pois não busca *rootkits* conhecidos, mas faz uma varredura em diretórios na tentativa de detecção de anomalias no sistema. Exemplos de anomalias podem ser: falta de arquivos em determinados diretórios do sistema, procura por arquivos com anomalias de permissão ou busca de processos e portas escondidas.

Uma vantagem deste sistema é que ele pode ser instalado em diversos sistemas operacionais. O OSSEC HIDS visa diminuir o tempo dispendido pelo administrador com o monitoramento do sistema, gerando avisos sobre ataques sem a necessidade de análise total dos *logs* em tempo real.

3.6.9: Real Secure

O Real Secure da ISS[24] é um IDS híbrido e sua arquitetura é dividida em três partes:

1. Um motor de reconhecimento baseado na rede;
2. Um motor de reconhecimento baseado no *host*;
3. Um módulo de controle administrativo.

O motor de reconhecimento baseado em rede é executado em uma máquina dedicada para prover detecção de intrusão e respostas. O monitoramento de um segmento de rede envolve a busca por padrões que combinem com as assinaturas de ataques existentes em sua base de dados. Quando este motor detecta uma atividade intrusiva na rede, ele pode gerar algum tipo de resposta que vai desde o término da conexão, envio de alertas, salvamento da sessão até a reconfiguração de regras de *firewall*.

O motor de reconhecimento baseado em *host* analisa os registros de eventos para reconhecer ataques. O sistema examina o *host* na busca de evidências de intrusões. Este motor pode prevenir a finalização súbita de processos de usuários ou

[24] http://www.iss.net

suspender contas de usuários. O módulo de controle administrativo realiza a gerência dos motores de rede e de host do sistema.

3.6.10: KIDS

O KIDS[25] é um IDS baseado em rede cuja função principal de capturar e comparar pacotes com regras pré-existentes é exercida no kernel de sistemas operacionais em forma de módulo.

A idéia básica de funcionamento do sistema KIDS é exercer as funcionalidades básicas de um IDS, porém sem a utilização da biblioteca libpcap, tratando os pacotes no *kernelspace* do sistema operacional, pretendendo com isso obter um ganho na desempenho do sistema como um todo.

A arquitetura do KIDS foi projetada para que a que a comparação dos pacotes de rede com as regras existentes em uma base, seja feita da forma simples e o mais rápido possível, executando esta ação no *kernelspace*. As outras atividades, como manipulação das regras, gerenciamento de logs, sistema de alertas, entre outras, serão todas manipuladas no userspace.

O KIDS é constituído de:

- um módulo para o kernel, chamado de kids mod, responsável pelo processamento dos pacotes que passam pelo framework netfilter[26];

- um daemon servidor, kidsd, responsável pelo gerenciamento das regras, estabelecimento de conexões (para gerenciamento das regras, estatísticas, etc.) e principalmente para fazer a comunicação com o módulo;

- uma aplicação cliente no userspace, o kids manager (kidsm), responsável pela comunicação com o servidor, sendo esta a que será utilizada pelo administrador do sistema para gerir as regras.

[25] Kernel Intrusion Detection System - `http://sourceforge.net/projects/ids-kids/`
[26] `http://www.netfilter.org/`

3.7: Exemplos de IDSs Distribuídos - DIDSs

Nesta seção, serão abordados alguns dos principais IDSs distribuídos conforme citados na literatura. Neste contexto, Spafford e Zamboni [5] definem um sistema de deteção de intrusão distribuído como: "um sistema onde a análise dos dados é executada em um número de locais proporcional ao número de máquinas que estão sendo monitoradas".

Esta definição sugere que apenas ter uma coleção de dados sendo analisados de maneira distribuída, não é suficiente para classificar um IDS como sendo um *Distributed Intrusion Detection System* ou simplesmente DIDSs.

O número de componentes de análise (do modelo básico de um IDS: E-boxes, A-boxes, D-boxes, C-boxes), deve ser proporcional ao número de *hosts* monitorados e estes devem estar em um ambiente distribuído.

Os componentes de detecção e de análise de dados são análogos aos E-boxes e A-boxes respectivamente.

Algumas das características que diferenciam um DIDS de outros são as seguintes:

- Número e localização de E-boxes;
- Número e localização de A-boxes;
- Coordenação entre componentes;
- *Framework de comunicação*.

O mecanismo de comunicação entre diferentes componentes de um sistema de detecção de intrusão distribuída constitui uma parte essencial para o funcionamento do sistema. Através da comunicação entre seus componentes, os DIDSs são capazes de obter uma visão global do sistema. Qualquer interrupção da comunicação pode causar um crash do sistema ou uma eventual falha.

Nesta seção abordamos alguns sistemas de detecção de intrusão que operam de forma distribuída, também conhecidos como *Distributed Intrusion Detection Systems* – (DIDSs).

A necessidade de um sistema de detecção de intrusão distribuído surgiu devido ao crescimento das redes nas organizações. Os IDSs comerciais geralmente concentram a detecção de intrusão dentro de uma única organização. Poucos projetos realizam a integração ou comunicação entre vários detectores, distribuídos em diversas organizações (envolvendo ambientes heterogêneos e de larga escala).

Conforme apresentado em [5], existem algumas características desejáveis nos DIDSs:

- executar continuamente sem intervenção humana;
- ser tolerante a faltas (capaz de se recuperar na ocorrência de *crashs* e reinicializações de componentes);
- monitorar a si mesmo e detectar ataques;
- causar um *overhead* mínimo no sistema onde o IDS está sendo executado;
- ser capaz de se adaptar às políticas de segurança locais (onde está sendo executado);
- ser capaz de se adaptar às mudanças de comportamento do sistema e dos usuários;
- ter escalabilidade para monitorar uma grande quantidade de *hosts*;
- ter baixa degradação, no caso de algum componente parar de funcionar;
- permitir a habilidade de reconfiguração dinâmica sem a necessidade de reinicialização do sistema.

Nos últimos anos, surgiram novos projetos de pesquisa de IDSs distribuídos que visam a utilização de técnicas (protocolos) para a comunicação e buscam a replicação de seus componentes para evitar a indisponibilidade dos serviços do sistema.

3.7.1: EMERALD

O EMERALD – *Event Monitoring Enabling Responses to Anomalous Live Disturbances*[27] usa coleta de dados no *host* e na rede, utilizando um sistema baseado no comportamento de usuários e em padrões baseados em assinaturas para a detecção de intrusão [61]. O objetivo inicial deste projeto foi prover detecção de intrusão em redes empresariais grandes e pouco acopladas.

O EMERALD utiliza pequenos componentes distribuídos em diferentes domínios para prover monitoramento, detecção e resposta a incidentes de segurança.

A principal estratégia de atuação do EMERALD é a de utilizar monitores distribuídos para analisar e responder a atividades maliciosas nos locais em que elas ocorrem e que tais monitores possam interoperar para formar uma hierarquia de análise.

Esta hierarquia é dividida em três níveis (camadas de monitoramento) e fornece um *framework* para reconhecimento de ameaças globais, incluíndo tentativas coordenadas para infiltrar ou destruir a conectividade em redes. A hierarquia de camadas de monitoramento inclui a análise de serviços abrangendo desde o mau uso de componentes individuais e serviços de rede dentro de um domínio até o mau uso de múltiplos serviços e componentes entre diferentes domínios.

A camada mais básica é a de monitoramento dos serviços de rede dentro de domínios. A camada intermediária é a de análise em escala de domínio. O monitor de domínio é responsável pelo monitoramento de todas as partes daquele domínio. Os monitores de domínio correlacionam alertas enviados por monitores locais, fornecendo uma perspectiva da atividade (ou padrões de atividade) em nível de domínio. O monitor de domínio é responsável pela reconfiguração dos parâmetros do sistema, trocando informações com outros monitores de domínios e reportando ameaças ao administrador do domínio a que pertence. A terceira camada é a análise em nível empresarial (em ambientes de larga escala), fornecendo uma abstração global das atividades nos domínios.

Com esta camada o EMERALD ameaça agentes externos que tentam subverter ou ignorar as interfaces de rede e controla o acesso não autorizado aos recursos do

[27] `http://www.csl.sri.com/projects/emerald/`

domínio. Além disso, o EMERALD fornece um *framework* para correlacionar os resultados de suas análises distribuídas e proporcionar detecção global e capacidade de resposta coordenada aos ataques em toda a rede.

As informações correlacionadas por um monitor de serviços são disseminadas para outros monitores através de um modelo de comunicação *publish/subscribe*, disseminando os resultados de forma assíncrona. Com isso, os monitores do EMERALD distribuídos em um domínio são capazes de disseminar relatórios de atividades maliciosas de maneira mais eficiente do que nos modelos síncronos. Estes monitores possuem a mesma arquitetura básica nas três camadas, sendo compostos por um conjunto de perfis para detecção de anomalias, um conjunto de assinaturas e um componente para integrar o resultado gerado. Existe também a possibilidade de integração de módulos desenvolvidos por terceiros.

3.7.2: AAFID

O AAFID[28] é um sistema de detecção de intrusão que utiliza agentes autônomos para tentar resolver os problemas de configuração, escalabilidade e eficiência encontrados em outros IDS [5].

A arquitetura do AAFID é composta de três componentes: agentes, *transceivers* e monitores. Comparando com o modelo de IDS desenvolvido pelo IETF, os agentes seriam os sensores, os *transceivers* seriam os analisadores e os monitores seriam os gerentes.

Neste modelo, os agentes podem prover mecanismos para auto reconfiguração sem precisar reiniciá-los e também podem ser testados antes de serem introduzidos em um ambiente mais complexo. Estes agentes realizam a coleta e a análise de dados na rede baseada em assinaturas, pelo uso dos *transceivers*. Se um agente pára de funcionar, duas coisas podem acontecer:

- Se os dados do agente forem usados por outros agentes, também podem ocorrer problemas com estes agentes;

[28] Autonomous Agents for Intrusion Detection

- Se o agente for independente, apenas os seus dados coletados serão perdidos.

Os agentes do modelo executam de maneira independente e fazem o monitoramento dos *hosts*, reportando anormalidades apenas aos *transceivers*, não podendo comunicar-se com outro agente. Os *transceivers* do modelo formam a interface externa de comunicação de cada *host*. Sua função é processar e controlar os dados. Eles podem inicializar ou desativar agentes, receber relatórios gerados pelos agentes e distribuir informações para outros agentes ou para um monitor, podendo também reportar para mais de um monitor, provendo redundância e resistência a falhas.

Entre as principais vantagens do modelo AAFID, podemos citar:

- Fácil configuração: uma vez que é possível ter uma série de pequenos agentes especializados em tarefas específicas de detecção, o sistema de detecção pode ser configurado da forma mais adequada para cada caso; a adição e remoção de agentes do sistema são facilitadas;

- Eficiência: agentes podem ser treinados previamente e otimizados para que realizem suas tarefas de maneira a gerar a menor sobrecarga possível no sistema;

- Distribuição da vigilância: um sistema de agentes pode ser facilmente modificado para operar em rede e permitir migração para rastrear comportamentos anômalos através da rede, ou mover para máquinas onde eles possam ser mais úteis.

- Escalabilidade: para atuar em sistemas maiores, basta adicionar mais agentes e aumentar sua diversidade.

Algumas desvantagens da arquitetura do AAFID são:

- O monitor no maior nível hierárquico é um ponto único de falha;

- A arquitetura não especifica mecanismos para controle de acesso, para permitir que diferentes usuários possam ter diferentes níveis de acesso ao IDS.

3.7.3: DOMINO

O DOMINO [29] [89] é um sistema de detecção de intrusão heterogêneo, escalar e robusto contra ataques e falhas.

O projeto do DOMINO, usa uma combinação de interação P2P (*peer-to-peer*) e componentes organizados em uma hierarquia distribuída para prover vantagens significativas sobre uma arquitetura puramente hierárquica. Estas vantagens incluem compartilhamento simplificado de informações, escalabilidade e tolerância a falhas.

O objetivo do DOMINO é disponibilizar um *framework* de compartilhamento de informação a fim de prover a capacidade de detecção de intrusão para todos os participantes da rede de compartilhamento [90].

Entre as principais características deste *framework* destacam-se: disponibilidade, descentralização, heterogeneidade e privacidade.

O DOMINO utiliza uma rede *overlay* para detecção de intrusão de forma cooperativa, sendo organizado em duas camadas: um pequeno centro de nós confiáveis e uma grande coleção de nós conectados ao centro.

Uma rede DOMINO é uma infra-estrutura dinâmica composta por uma coleção de diversos nós localizados através da Internet em um sistema distribuído em várias organizações ou em uma organização virtual.

Esta rede é composta por três conjuntos de participantes: nós concentradores, comunidades satélites e nós terrestres.

- Nós concentradores – são os componentes centrais da arquitetura do DOMINO. Eles são responsáveis pelo cruzamento e correlação dos dados usando comunicação segura para troca de informações entre cada nó concentrador;

- Comunidades satélites – são pequenas redes de nós satélites que implementam uma versão local do protocolo DOMINO. Os nós satélites estão organizados em uma hierarquia de sensores. Cada nó possui comunicação com seu pai.

[29] Distributed Overlay for Monitoring InterNet Outbreaks

- Nós terrestres – são os mais confiáveis e possuem uma grande quantidade de dados. Estes nós não implementam o protocolo DOMINO, eles servem apenas para expansão da cobertura, incluíndo conjuntos de dados sobre intrusões para fora da infra-estrutura.

As mensagens do DOMINO são representadas por XML[30].

Em seu funcionamento, o DOMINO usa uma abordagem de coordenação centralizada onde um centro de nós confiáveis coordenam o resto dos nós distribuídos em diversas redes. Desta forma é possível gerar alarmes rapidamente quando ataques de larga escala estiverem em andamento, além de reduzir o número de alarmes falsos. Contudo, com esta centralização pode haver uma inundação destes nós centrais, no caso de ataques por vírus ou outras pragas digitais que atuam de maneira uniforme gerando muitos alertas para estes nós centrais. Além disso, o DOMINO não produz respostas aos ataques, apenas detecta e gera os alertas, deixando aos administradores a tarefa de conter ou eliminar os ataques.

3.7.4: INDRA

O INDRA *Intrusion Detection and Rapid Action* é um sistema IDS distribuído, baseado no compartilhamento de informações entre pares confiáveis em uma rede, para proteger a rede como um todo contra tentativas de intrusão.

O INDRA coleta as informações na rede e utiliza uma base de assinaturas para realizar a análise e detecção de intrusão.

O principal objetivo do INDRA é distribuir as informações sobre tentativas de intrusão em uma rede P2P (*peer-to-peer*) e atuar de forma colaborativa nesta rede [37]. Uma vantagem significativa dos sistemas distribuídos como o INDRA é que eles podem ser usados para balanceamento de carga, realizando a distribuição das informações de alertas em vários *hosts* da rede. O funcionamento do sistema é fundamentado na posse de certificados digitais.

[30] eXtensible Markup Languagem

3.7.5: IDF

O IDF (*Intrusion Detection Force*) [77] é uma infra-estrutura virtual que permite a detecção de intrusão na escala da Internet. O objetivo do IDF é defender redes de computadores, por meio do compartilhamento de informações inter-organizações de forma segura, provendo inteligência na resposta e análise de dados.

O IDF define quatro propriedades básicas: compartilhamento da informação, segurança, escalabilidade e sobrevivência. Apresenta outras propriedades secundárias como: interoperabilidade, extensibilidade, integração do IDF com outros sistemas, prover uma plataforma segura de desenvolvimento. A infra-estrutura do IDF tem entidades e grupos de entidades distribuídas que trabalham em diferentes áreas e sistemas de um ambiente de larga escala.

O IDF é um IDS distribuído que realiza a coleta e o processamento dos dados de forma distribuída, coletando dados no *host* e na rede. O mesmo realiza sua análise dos dados com base em assinaturas de vulnerabilidades conhecidas e sua resposta pode ser ativa ou passiva. Este sistema possui a vantagem de atuar em tempo real, possibilitando respostas aos ataques no momento que eles acontecem. Desse modo é possível evitar que invasões ou danos maiores ocorram no sistema atacado. Uma desvantagem do IDF é que ele utiliza um formato próprio para troca de mensagens. Assim, não é possível realizar sua integração com outras ferramentas de segurança.

3.7.6: Prelude

O Prelude-IDS[31] é um IDS híbrido que agrega funcionalidades de análise de tráfego de rede e realiza auditoria em logs da máquina na qual opera.

Sistemas híbridos agregam as vantagens de sistemas baseados em *host* e em rede, possibilitando um índice maior de acerto e uma melhor identificação de incidentes de segurança quando comparados com as abordagens tradicionais [91].

Este sistema possui alguns módulos que usam mecanismos de detecção baseados em assinaturas e outros que não usam assinaturas para realizar a detecção de

[31] http://www.prelude-ids.com/en/welcome/index.html

intrusões.

Os módulos baseados em assinaturas utilizam uma base de dados de vulnerabilidades conhecidas e permite a utilização de assinaturas desenvolvidas por IDS de terceiros (por exemplo, pode utilizar uma base de assinaturas do Snort, além de outros sensores disponíveis no mercado). Tais assinaturas identificam determinado tipo de ataque conhecido.

O Prelude envia os dados coletados a um ou mais de seus gerenciadores, onde um gerenciador pode compartilhar alertas com os demais gerenciadores do sistema.

Para facilitar a comunicação entre os gerenciadores do sistema, o Prelude implementa um agente de *software* independente para cada sensor. Este *software* recebe os dados originais, formata os mesmos e os transmite ao Prelude. Contudo, cada agente não tem a capacidade de gerenciamento dos sensores e o gerenciador do Prelude não interfere nos agentes. Assim, a configuração e ativação dos agentes precisa ser executada manualmente. Na ocorrência de falhas, a recuperação também é manual.

A troca de mensagens entre os componentes do Prelude é feita sobre SSL (*Secure Socket Layer*), usando uma variação do formato IDMEF, que não é baseada em XML.

O Prelude funciona de maneira distribuída e permite o uso de diversas aplicações de segurança (analisadores de registros, IDSs de rede, analisadores de aplicações) que reportam seus eventos detectados ao próprio Prelude. O Prelude permite a utilização de vários sensores como o *snort*, *honeyd*, *nessus*, *samhain*, e mais de 30 tipos de sistemas de *logs*.

A arquitetura do Prelude segue um modelo hierárquico com um gerente centralizado. Neste modelo, os sensores realizam a coleta e análise dos dados e enviam os resultados para seus gerentes. Os gerentes podem ser replicados, evitando assim a ocorrência de faltas. A comunicação de eventos é feita pelo uso de mensagens no padrão IDMEF, um padrão IETF que permite a geração e o entedimento entre diferentes sensores através do uso de uma mesma linguagem de comunicação.

3.8: Comparação entre IDSs apresentados

Nesta seção apresentamos uma breve comparação entre os sistemas de detecção apresentados nas seções anteriores. A comparação não pode ser feita com base nas características próprias de cada IDS (parâmetros de configuração, por exemplo), mas podemos comparar as características comuns entre estes: modo de coleta de dados, método de detecção, algoritmo de deteção, suporte ao padrão IDMEF e outras características relevantes para este trabalho.

Conforme pode ser observado na Tabela 4, os IDSs foram agrupados de acordo com a forma de coleta de dados, sistema operacional, suporte ao padrão IDMEF de troca de mensagens e se o IDS atua em ambientes distribuídos e heterôgeneos.

Dentre os IDSs apresentados na Tabela 4, o POSEIDON é um dos ANIDSs que utiliza uma arquitetura em duas camadas, em que são aplicadas redes SOM (*Self Organizing Maps*) para melhorar as taxas de detecção e o número de falsos positivos. O POSEIDON foi desenvolvido para operar com base no *payload* e para análise de tráfego TCP.

Outro IDS que utiliza redes neurais é o sistema iSOM [56], desenvolvido para análise dos *headers* de protocolos específicos.

Dentre os sistemas analisados, o SilentDefense é um dos mais interessantes devido à integração e utilização de IDSs existentes para análise e detecção de intrusão. O sistema SilentDefense utiliza ao todo quatro IDS existentes, os quais podem trabalhar em cooperação ou em modo *stand-alone*. O SilentDefense usa o IDS POSEIDON (usa redes SOM) para melhorar as taxas de detecção, o ATLANTIDES para reduzir as taxas de falsos positivos, o Panacea (que usa SVMs para classificação e o algoritmo RIPPER[32] para geração de regras [17]) na classificação automática de alertas e o Sphinx como mecanismo de detecção de ataques em aplicações *Web*.

No caso dos sistemas que realizam a coleta de dados na rede em geral apresentam dois problemas: altas taxas de falsos positivos e uma visão limitada de redes

[32] *Repeated Incremental Pruning Produce Error Reduction* – É um método para extração de regras com base nos dados analisados.

grandes e heterogêneas, restringindo sua capacidade de deteção de ataques. Algo similiar ocorre nos sistemas que coletam dados no *host*, que em geral acabam tendo apenas uma visão das informações contidas em um *host* específico, deixando de identificar possíveis ataques ou tentativas de ataques através da rede.

O uso de sistemas de detecção de intrusão híbridos (que coletam dados tanto na rede quanto no *host*) e que utilizam técnicas de aprendizado de máquina e/ou algoritmos de classificação de dados têm se mostrado mais eficazes na detecção de intrusão do que as abordagens baseadas regras. Na tentativa de redução das taxas de falsos positivos e aprimoramento das taxas de detecção os diversos ANIDSs propostos na literatura tem empregado diferentes algoritmos de detecção conforme apresentados brevemente na próxima seção.

3.8.1: Algoritmos de detecção utilizados em ANIDSs

Alguns exemplos de ANIDSs que utilizam algoritmos de classificação de dados e/ou técnicas de aprendizado de máquina para análise e detecção de intrusão são:

- O sistema iSOM [56], que usa uma arquitetura de camada única constituída de um *Self-organizing Map*, utilizado para detectar ataques a serviços SMTP e FTP;
- O sistema IntelligentIDS [25], que extrai a informação a partir dos metadados de conexão assim que são remontados;
- O sistema PHAD [50], que combina 34 valores diferentes conforme extraídos dos cabeçalhos dos pacotes.
- O sistema MADAM ID [47], que extrai informações do tráfego monitorado e gera modelos de classificação (projetado especificamente para detectar certos tipos de intrusão) usando técnicas de mineração de dados.
- O sistema SSAD (*Service Specific Anomaly Detection*) combina diferentes informações, tais como: tipo, comprimento e freqüência de distribuição do *payload*, classificando o payload em seis categorias distintas com base nas ocorrências de caracteres nas solicitações.

- O sistema PAYL [71] e POSEIDON [9] que realizam a detectação de anomalias por meio da análise de *payload*.

A Tabela 5 a seguir, apresenta um resumo das principais características destes sistemas.

3.9: Conclusões do Capítulo

Neste capítulo, apresentamos a evolução dos sistemas de detecção de intrusão em termos de técnicas utilizadas na detecção. Analisamos o padrão definido pelo IETF, que serve de modelo para a construção de IDSs. Depois, abordamos as diferentes formas de coleta de dados (na rede, no *host*, ou ambas) que podem ser utilizadas por um IDS, bem como as abordagens de para a detecção de intrusão, sendo: baseada em assinaturas ou em anomalias. Ao detalhar algumas das técnicas de análise e detecção empregadas nos sistemas de detecção por anomalias, constatamos que tais IDSs podem analisar o *header*, o*payload* ou ambos para detectar ataques.

Na seqüência, citamos vários exemplos de IDSs comerciais e IDSs em fase de experimentação. Alguns dos sistemas vistos realizam análise baseada em assinaturas, outros por anomalia e ainda outros utilizam as duas abordagens. O que se pode concluir da análise realizada neste capítulo é que tanto a abordagem Apresentamos exemplos de sistemas que operam em ambiente distribuído e por fim realizamos um comparativo entre os IDSs apresentados que utilizam algoritmos de classificação e/ou técnicas de aprendizado de máquina na deteção de intrusão.

Comparação de características comuns entre IDSs						
IDS	Coleta	Método Detecção	Padrão ID-MEF	Ambiente Distribuído	Algoritmo de Detecção	Nível
Real Secure	rede	assinaturas	não	não	regras	P
Tripwire	host	anomalias	não	não	regras	–
OSSEC	host	híbrido	não	não	regras	–
Snort	rede	assinaturas	sim	não	regras	H + P
EMERALD	híbrido	misto	não	sim	regras	H + P
Prelude	híbrido	assinaturas	não	sim	regras	H + P
Sphinx	rede	híbrido	não	não	Autômatos	P
ATLANTIDES	rede	híbrido	não	não	análise n-gram	H + P
AAFID	rede	assinaturas	não	sim	regras	H + P
DOMINO	rede	assinaturas	não	sim	regras	H + P
Panacea	rede	anomalias	não	não	aprendizado de máquina	H + P
INDRA	rede	assinaturas	não	sim	regras	H + P
IDF	rede	assinaturas	não	sim	regras	H + P
POSEIDON	rede	anomalias	não	não	PAYL e redes neurais	H + P
KIDS	rede	assinaturas	sim	não	redes neurais	H + P
iSOM	rede	anomalias	não	não	redes neurais	H
IntelligentIDS	rede	anomalias	não	não	redes neurais	H
PHAD	rede	anomalias	não	não	método estatístico	H
MADAM ID	rede	anomalias	não	não	método estatístico	H + P
SSAD	rede	anomalias	não	não	método estatístico	H + P
PAYL	rede	anomalias	não	não	método estatístico	P
SilentDefense	rede	híbrido	não	não	híbrido	P + H

Table 4: Comparação entre IDSs. Onde P indica análise de *Payload* e H análise de *Header*.

Características comuns em IDSs			
IDS	Algoritmo Detecção	Nível Análise	Dados Analisados
iSOM	RNA	OP + OC	Meta dados
IntelligentIDS	RNA	OC	Meta dados
PHAD	E	OP	C
MADAM ID	E	OC	Meta dados
SSAD	E	OP	C + P
PAYL	E	OP	P
POSEIDON	RNA + E	OP	P
Panacea	SVM e RIPPER	OP	P

Table 5: Principais ANIDs: RNA indica redes neurais artificiais, E para modelo estatístico, OP para orientado a pacotes enquanto que OC indica orientado a cabeçalho, C indica cabeçalho e P pacote - adaptado de [8].

4. Redes Neurais Artificiais

"Tudo deve ser apresentado da maneira mais simples possível, porém não mais simples do que isso."

Albert Einstein

4.1: Introdução

A Inteligência Artificial (IA) estuda os possíveis modos de como fazer a máquina executar ações próprias da inteligência humana. De acordo com [63] e [85], a IA representa o campo na Ciência da Computação destinado ao desenvolvimento de sistemas que simulam algum aspecto da cognição humana. Segundo [27], a IA é o campo da ciência que tenta explicar a origem da natureza do conhecimento. A IA desenvolve modelos computacionais que possam imitar o comportamento humano. Esses modelos não somente contribuem para o aperfeiçoamento das máquinas, mas representam uma motivação para se conhecer melhor as atividades mentais do homem [55].

São discernidas duas orientações a partir desta definição; a tecnológica, associada ao desenvolvimento de programas inteligentes; e o científico, que trata dos aspectos teóricos relacionados com a cognição humana.

Existem ainda na área de IA várias abordagens, as quais se diferenciam na manipulação do conhecimento, na maneira de adquiri-lo, armazená-lo e empregá-lo. Classificando a IA quanto ao método de solução de problemas tem-se a IA Simbólica (IAS), a Conexionista (IAC), a Evolucionária (IAE) e a Híbrida (IAH). Em relação à localização espacial tem-se ainda a IA Monolítica (IAM) e a IA Distribuída (IAD). Na IA conexionista apresentam-se os estudos das redes neurais artificiais (RNAs).

As redes neurais são uma técnica amplamente utilizada nas tarefas de aproximação de funções, previsão de séries temporais, classificação e reconhecimento de padrões. Conforme [23], entre todos os domínios de utilização de modelos de redes neurais, o reconhecimento de padrões possui maior potencial.

Neste capítulo abordamos as arquiteturas de RNAs avaliadas nesta dissertação com o objetivo de facilitar a compreensão dos métodos de classificação, detecção de novidades e reconhecimento de padrões empregados no desenvolvimento de IDSs através do uso das redes neurais artificiais. Neste sentido, este capítulo apresenta uma breve análise da classificação e do funcionamento das arquiteturas de redes neurais freqüentemente utilizadas no desenvolvimento de IDSs.

4.2: Classificação

As redes neurais artificiais – RNAs, são constituídas de camadas compostas de neurônios e de pesos para representar as interconexões entre os neurônios de cada camada.

As RNAs podem ser classificadas de acordo com o método de aprendizado que utilizam e estes podem ser divididos em: **supervisionados** e **não-supervisionados**.

No aprendizado supervisionado, cada entrada apresentada vem acompanhada de uma resposta ou saída desejada. Por outro lado, no aprendizado não-supervisionado padrões e características estatísticas do espaço de entrada são utilizados para construir uma representação compacta do espaço de entrada. Existem diversas arquiteturas baseadas nestes dois modos de aprendizado. Nas próximas seções, abordamos algumas das principais arquiteturas de redes neurais.

4.2.1: Arquiteturas

As arquiteturas de redes neurais variam de acordo com o número de camadas e de neurônios existentes em cada camada. Em geral, uma rede neural possui alguns neurônios na camada de entrada, pesos sinápticos para a ligação dos neurônios da camada de entrada com os da camada de saída, podendo ter camadas ocultas (*hiden*), conforme a arquitetura, modo de aprendizado e de treinamento (*on-line* ou *off-line*), conforme visto em 5.

Existem diversas arquiteturas de redes neurais conforme propostas na literatura por diferentes autores, mas por razões de escopo as arquiteturas de redes neurais abor-

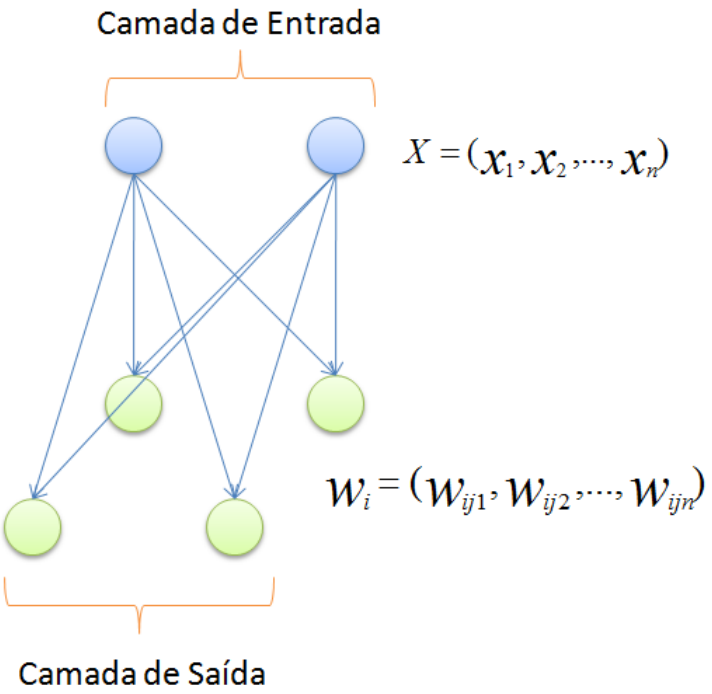


Figure 5: Arquitetura geral de uma Rede Neural simplificada.

dadas neste capítulo serão as seguintes:

- **Redes não-supervisionadas:** *Winner-Take-All*(WTA), *Self Organizing Map*(SOM) propostas por Tuevo Kohonen.
- **Redes supervisionadas:** *Multilayer Perceptron*(MLP), *Radial Basis Functions*(RBF) e Filtro Linear Detector de Novidades.

Quando o treinamento de uma RNA é realizado com o modo de aprendizado **supervisionado**, o mapeamento entrada-saída é conduzido por um professor, que possui informações a *priori* das classes que devem ser escolhidas para os vetores de entrada. Em contrapartida, quando o treinamento utiliza aprendizado **não-supervisionado** ou **auto-organizado**, a rede resultante deve ser capaz de fazer representações internas do espaço de entrada somente pela apresentação sucessiva dos padrões. O modo de aprendizado auto-organizado segue a quatro princípios [57]:

- As **modificações dos pesos sinápticos**, em geral, tendem a se auto-amplificar.

- Existência de um **processo de competição** entre as sinapses devido à limitação de recursos e concentração da informação em neurônios específicos.

- Existência de um **processo de cooperação** entre neurônio vencedor e seus vizinhos, ao longo da atualização dos pesos. A atualização dos pesos, tem como objetivo relacionar o valor dos pesos com as características dos padrões de entrada. A cooperação, por sua vez, modela a hipótese de maior excitação na vizinhança de um neurônio que está disparando, conduzindo a um processo de seleção que, ao seu final, ativará apenas um neurônio ou um grupo específico de neurônios.

- Mapas ou redes auto-organizadas aprendem a **extrair informação** a partir da redundância presente na ordem e na estrutura dos padrões de entrada.

Neste trabalho, as arquiteturas de redes neurais não-supervisionadas receberão um enfoque maior do que as supervisionadas. Algumas das vantagens das RNAs baseadas em auto-organização são:

- Produção natural de quantização (ou codificação) vetorial do espaço de entrada;

- Habilidade de produzir classificação não-paramétrica dos padrões de entrada;

- Compressão da informação presente no espaço de entrada para o espaço de saída resultante.

Como desvantagens e limitações das RNAs baseadas em auto-organização podemos citar:

- Problemas de estabilização e convergência;

- Definição de taxas de aprendizado e número de épocas de treinamento precisa ser definido empíricamente;

- Tempo de treinamento é outro fator crítico e cresce proporcionalmente ao tamanho da entrada.

Tendo em vista a utilização das redes SOM neste trabalho para a proposta de um classificador de dados de tráfego de intrusão, nas próximas seções apresentamos suas principais características.

4.3: Redes Neurais Artificiais Não-Supervisionadas

O objetivo das redes neurais não-supervisionadas é extrair características estatísticas predominantes nos dados de entrada e construir um modelo (de forma auto-organizada, sem o auxílio externo e sem conhecimento prévio), ou uma representação reduzida do espaço de entrada, codificando-a em seus pesos.

Uma rede neural não-supervisionada recebe como entrada um número finito de N exemplos de treinamento, cada um deles sendo representado como um vetor $x(t) \in \Re^n$, conforme:

$$x(t) = (x_1(t) \ldots x_n(t)) \tag{1}$$

em que $t = 1, 2, ..., N$, indica o instante de apresentação deste vetor à rede neural durante a fase de treinamento. Cada componente $x_i(t)$ carrega alguma informação relevante para a análise em questão, sendo denominado normalmente de característica ou atributo. Assim, um vetor $x(t)$ é chamado de vetor de características ou vetor de atributos no contexto de reconhecimento de padrões [84].

A seguir, apresentamos a rede *Winner-take-all* que aplica um algoritmo simples de competição entre seus neurônios.

4.3.1: Winner-Take-All

A rede WTA implementa um algoritmo de competição simples e na fase de treinamento apenas o neurônio vencedor tem seu vetor de pesos $w_{i*}(t)$ atualizados para cada vetor de entrada $x(t)$. O treinamento desta rede pode ser dividido em três passos:

1. Selecionar um exemplo aleatóriamente do conjunto de treinamento $x(t)$ e utilizar como vetor de entrada atual.

2. Realizar uma busca pelo neurônio vencedor, $i_*(t)$, para o vetor de entrada $x(t)$ usando a distância euclidiana.

3. Atualizar o vetor de pesos do neurônio vencedor, $w_i^*(t)$, pela equação:

$$w_{i*}(t+1) = w_{i*}(t) + \eta\,[x(t) - w_{i*}(t)], \tag{2}$$

em que $0 < \eta < 1$ denota o passo de aprendizagem.

É comum inicializar os valores dos pesos de forma aleatória dentro do intervalo $[0,1]$ ou os vetores de pesos iniciais podem também ser selecionados sorteando-se algum vetor no conjunto de vetores de treinamento.

Segundo [57], é possível mostrar que o vetor de pesos de um determinado neurônio i converge para o centróide (centro de gravidade) do conjunto de vetores de treinamento no qual o neurônio i é selecionado como o vencedor. Isto ocorre devido ao resultado de $w_i(t+1)$ e $w_i(t)$ serem iguais quando $t \to \infty$. Sendo assim, podemos marcar o último vetor de pesos w_i calculado como sendo w_i^u e obtemos a seguinte expressão:

$$C\{\eta\,[x - w_i^u]\} = 0 \Rightarrow w_i^u = \frac{\int_{V_i} x p(x)\,dx}{\int_{V_i} p(x)\,dx} \tag{3}$$

onde V_i é o conjunto de vetores de treinamento onde o neurônio i foi selecionado vencedor. É comum fazer com que a taxa de aprendizagem decresça com o tempo, para aumentar a probabilidade de convergência do algoritmo para um mínimo global. O decaimento pode ser exponencial conforme a equação 4 ou respeitar outra função.

$$\eta(t) = \eta_0 \left(\frac{\eta_T}{\eta_0}\right)^{t/T} \tag{4}$$

em que η_0 e $\eta_T (\eta_T \ll \eta_0)$ são os valores inicial e final de η. O parâmetro T representa o número máximo de iterações de treinamento e controla a velocidade do decaimento da taxa de aprendizado.

Devido a simplicidade da rede WTA, algumas questões comprometem seu desempenho. A primeira tem a ver com a escolha dos valores iniciais dos pesos da rede,

já que com base nestes valores, alguns neurônios podem dominar o treinamento e ser sempre selecionados como vencedores, enquanto outros nunca vencem. As unidades não selecionadas são conhecidas como unidades mortas (*dead units*). A segunda limitação da rede WTA refere-se ao fato de haver uma valorização excessiva da informação contida na entrada $\mathrm{x(t)}$ mais recente, pois devido à natureza do algoritmo, as entradas apresentadas à rede no início do treinamento têm menos influência no valor final dos pesos dos neurônios que aquelas apresentadas ao final do treinamento.

Diversas modificações no algoritmo da rede WTA podem ser feitas para minimizar estes e outros problemas. Neste sentido existem variantes deste algoritmo, que alteram as equações apresentadas nesta seção. Uma das variações mais conhecidas deste algoritmo é a rede SOM que é apresentada a seguir.

4.3.2: Self-Organizing Maps

As redes de *Kohonen* foram introduzidas em 1982 pelo pesquisador Tuevo Kohonen e são um tipo de *self-organizing map* – SOM, que representam uma classe especial de rede neural (competitiva). Seus neurônios ficam dispostos em uma grade fixa uni- ou bi-dimensional, permitindo a definição de uma relação de vizinhança espacial entre os neurônios desta grade.

O objetivo de um mapa auto organizável (*self-organizing map*) é converter um sinal de entrada de um espaço dimensional elevado para um mapa discreto num espaço dimensional inferior. Assim, SOMs são indicados para aplicações de reconhecimento e classificação de padrões, formação de agrupamentos (*clustering*) de dados e quantização vetorial. Neste tipo de aplicações, o vetor de pesos associado ao neurônio vencedor, também chamado de *Best Match Unit* – BMU, é visto como um *protótipo* representativo de um determinado grupo de vetores de entrada.

Os *Self-organizing maps* são baseados em aprendizado competitivo, onde os neurônios da camada de saída da rede competem entre si para ser o neurônio vencedor (ativado) de acordo com cada observação particular informada na entrada da rede neural.

As RNAs competitivas têm um único neurônio ou um pequeno grupo deles, denominados de *neurônios vencedores*, ativados de acordo com o grau de proximidade entre o vetor de pesos e o vetor de entrada atual, onde o grau de proximidade é medido segundo alguma métrica pré-estabelecida [57].

Do mesmo modo que as redes neurais tradicionais, as redes SOMs são *feed-forward*: não permitem *loops* ou ciclos e são completamente conectadas, indicando que cada neurônio de uma camada possui conexões com cada neurônio da próxima camada, contudo, nunca com outros neurônios de sua camada. Cada conexão entre neurônios tem um peso (*weight*) ou valor de ponderação associado, que durante a inicialização é ajustado com valores aleatórios entre zero e 1. O ajuste de pesos representa o fator chave do processo de treinamento dos *self-organizing maps*.

Os valores das variáveis de entrada da rede devem ser normalizados ou (*standardized*), de modo que algumas variáveis não venham a causar *overfitting*[33] em relação a outras no algoritmo de aprendizagem.

Uma característica peculiar das redes SOM é que elas não possuem camadas ocultas. Os dados recebidos pelos neurônios da camada de entrada são repassados diretamente para os da camada de saída. A camada de saída da rede SOM é representada na forma de uma grade (*lattice*), de uma ou duas dimensões e usa uma estrutura de vizinhança tipicamente no formato retangular ou hexagonal conforme apresentado na figura 6.

Nesse sentido, a equação 2 deve ser modificada para contemplar o conceito de *vizinhança* espacial entre os neurônios na grade. A vizinhança é o conjunto de neurônios que estão em torno do neurônio vencedor $\mathrm{i}^*(\mathrm{t})$, onde t representa a época em que o neurônio foi definido como vencedor. Durante o processo de treinamento da rede, os vetores de pesos dos neurônios vencedores também são ajustados, obedecendo à regra de aprendizado descrita na equação 5:

[33] Overfitting ocorre quando um modelo estatístico descreve o erro aleatório ou ruído ao invés da relação subjacente desejada. Ocorre geralmente quando um modelo é excessivamente complexo, ou quando tem muitos graus de liberdade em relação à quantidade de dados disponíveis. Um modelo que sofre de overfitting pode ter seu desempenho preditivo comprometido, pois pode exagerar pequenas flutuações nos dados.

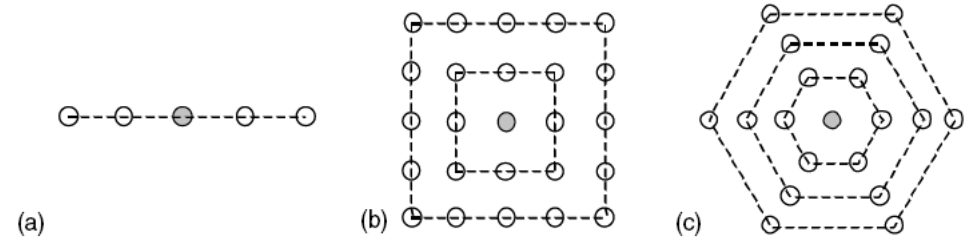


Figure 6: **Definição de vizinhança:** (a) linear (b) quadrada e (c) hexagonal a um neurônio vencedor (círculos sólidos indicam um neurônio vencedor e os círculos vazios indicam os seus vizinhos)

$$w_i(t+1) = w_i(t) + \eta(t)h(i^*, i; t)\,[x(t) - w_i(t)], \tag{5}$$

onde $h(i^*, i; t)$ é a função de vizinhança, que em geral é gaussiana conforme a equação 6:

$$h(i^*, i; t) = exp\left(-\frac{\|r_i(t) - r_{i^*}(t)\|^2}{\vartheta^2(t)}\right), \tag{6}$$

onde $\vartheta(t)$ define o raio de influência da função de vizinhança e $r_i(t)$ e $r_{i^*}(t)$ representam as posições respectivas dos neurônios i e i^* na grade da rede.

Desse modo, para cada época de treinamento, a *função de vizinhança* atua como uma janela de ponderação, atualizando os pesos dos neurônios vizinhos ao vencedor, mais intensamente do que os demais neurônios que não se encontram na vizinhança do vencedor. Além disso, o neurônio vencedor tem seus pesos reajustados com maior intensidade, uma vez que ele tem $h(i^*, i; t) = 1$ e no caso dos demais neurônios tem-se que $h(i^*, i; t) < 1$. A função de vizinhança deve decrescer seu valor com o passar do tempo, para permitir a convergência e estabilidade do aprendizado da rede neural. Em outras palavras, o raio de influência $\vartheta(t)$ decai ao longo do treinamento obedecendo à equação 7, que faz com que a vizinhança diminua com o passar das iterações de treinamento.

$$\vartheta(t) = \vartheta_0 \left(\frac{\vartheta_T}{\vartheta_0} \right)^{t/T} \tag{7}$$

em que ϑ_0 e $\vartheta_T (\vartheta_T \ll \vartheta_0)$ são os valores inicial e final de ϑ.

A rede SOM implementa uma projeção não-linear Φ do espaço de entrada contínuo $X \subset \Re^n$ (espaço de dados), em um espaço de saída discreto A, sendo representado pelo espaço das coordenadas dos neurônios na grade, tal que $dim(A) \ll n$ o que conduz a seguinte função de projeção: $\Phi X \rightarrow A$.

Entre as principais utilizações da rede SOM estão as aplicações de classificação de padrões. A atuação dos processos de competição e cooperação é implementada como uma projeção Φ que preserva relações de proximidade espacial (ou topologia) entre os dados do espaço de entrada no espaço de saída.

As redes SOM apresentam uma propriedade denominada de **preservação de topologia** que pode ser expressa, utilizando dois vetores arbitrários x_1 e x_2 no espaço de entrada X, sendo respectivamente, $r_{i_1^*}$ e $r_{i_2^*}$, as coordenadas dos neurônios vencedores para x_1 e x_2. Se a rede tiver sido treinada corretamente, ela preservará a topologia do espaço de entrada por observar a seguinte relação:

$$\|x_1 - x_2\| \Rightarrow \left\| r_{i_1^*} - r_{i_2^*} \right\| \rightarrow 0, \tag{8}$$

ou seja, caso quaisquer dois vetores encontrem-se fisicamente próximos no espaço de entrada, então os neurônios vencedores estarão espacialmente próximos na rede.

A capacidade de preservação de topologia da rede SOM, permite a construção de uma aproximação **discreta** do espaço de entrada, onde cada neurônio da rede representa uma determinada região do espaço de entrada que define sua **região de atração** ou campo receptivo, também conhecido como **célula de Voronoi**, conforme pode ser visto na figura 7.

Portanto, esta propriedade possibilita a aplicação da rede SOM na tarefa de classificação de dados não-rotulados em agrupamentos (*clusters*) e após o treinamento, na classificação de vetores de características ausentes durante a fase de treinamento.

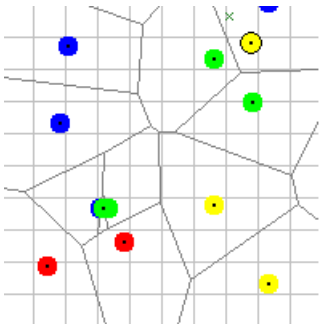


Figure 7: Regiões de atração e células de Voronoi.

A preservação de topologia permite ainda à rede SOM realizar uma estimação pontual da função densidade de probabilidade, indicando que o mapeamento da rede SOM reflete variações na distribuição estatística do espaço de entrada. Assim, regiões no espaço de entrada X onde as observações x têm grande probabilidade de ocorrência são povoadas com um maior número de neurônios, possuindo uma melhor resolução do que regiões em X onde as observações x são retiradas com baixa probabilidade de ocorrência.

Os *Self-organizing maps* possuem três processos característicos: (i) Competição, (ii) Cooperação e (iii) Adaptação.

O processo de competição define o neurônio vencedor para cada vetor de entrada apresentado à rede SOM. Este processo utiliza o vetor de pesos de todos os neurônios da rede e realiza uma comparação com o vetor de entrada. Essa comparação utiliza uma medida de distância entre cada vetor de entrada com os vetores de pesos. O neurônio cujo vetor de pesos estiver mais próximo do vetor de entrada é considerado o vencedor. O processo de **competição** pode ser implementado em termos da distância euclidiana na forma da equação 9:

$$i^*(x(t)) = \arg \overset{\text{minimizar}}{i \in A} \|x(t) - w_i(t)\| \tag{9}$$

em que $i^*(x(t))$ é o índice que representa o neurônio vencedor para o vetor de entrada $x(t)$. A norma euclidiana $\|\cdot\|$ é definida como:

$$\|x(t) - w_i(t)\| = \sqrt{[x(t) - w_i(t)]^T [x(t) - w_i(t)]} = \sqrt{\sum_{j=1}^{p} [x_j(t) - w_{ij}(t)]^2} \quad (10)$$

onde $(\cdot)^T$ denota o vetor transposto. A equação 10 permite a seguinte observação: "O espaço contínuo dos vetores de entrada é mapeado em um espaço de saída discreto através de um processo de competição entre os neurônios."

A **cooperação** entre os neurônios leva a um contínuo aumento de sensitividade de um determinado grupo de neurônios a um padrão de entrada específico, de modo que o neurônio vencedor $i^*(t)$ determina o centro da vizinhança i^* de um grupo espacialmente localizado no mapa de neurônios. Na cooperação a idéia é especializar regiões contíguas de neurônios tornado-as seletivas a padrões de entrada com características similares. A especialização de grupos de neurônios cuja ativação torna-se sensível às propriedades do espaço de entrada, resulta na formação de um mapa topologicamente ordenado. A idéia de ordenação topologica é apresentada na figura 8.

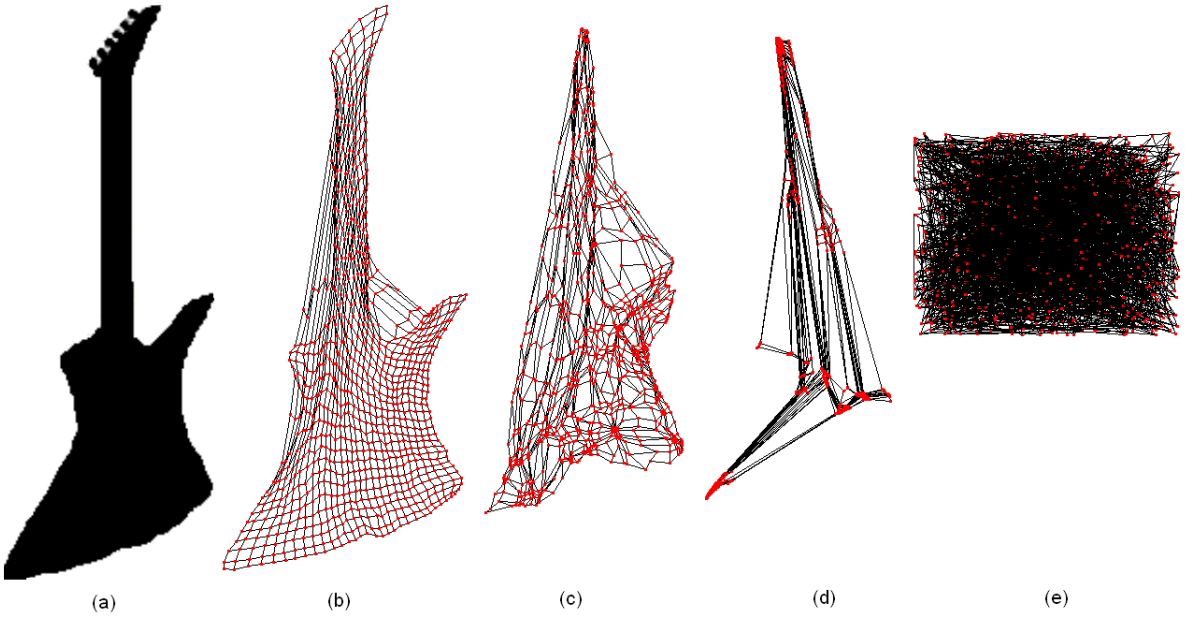


Figure 8: Mapa de Kohonen contendo regiões topológicamente ordenadas. Sendo (a) Padrão a reconhecer, (b) Rede de Kohonen após treinamento, (c) e (d) passos intermediários de treinamento (ajustes de pesos sinápticos) e (e) inicialização da rede neural com valores aleatórios.

O neurônio vencedor e sua vizinhança interagem lateralmente de forma cooperativa. Segundo Tuevo Kohonen, existe evidência neurobiológica para interação lateral entre os neurônios excitados [41]. Um neurônio que vence sucessivamente tende a excitar mais intensamente aos neurônios de sua vizinhança imediata.

A intensidade da interação lateral entre o vencedor i^* e um neurônio i qualquer é descrita por uma função matemática de vizinhança do tipo $h(i^*, i; t)$, definindo a vizinhança topológica no neurônio vencedor $i^*(x(t))$. Os neurônios da rede SOM podem assumir um arranjo discreto em pelo menos dois modelos de grade bastante utilizados [41]: (i) unidimensional e, (ii) bidimensional. No caso em que os neurônios estejam dispostos em uma grade unidimensional, tem-se que $r_i(t) \in \Re$, onde a posição de um neurônio i qualquer coincide com seu próprio índice, sendo $r_i(t) = i$ e tendo apenas vizinhos à direita ou à esquerda, conforme a figura 6 item (a).

Por outro lado, se os neurônios da rede SOM estão dispostos em uma grade bidimensional, tem-se que $r_i(t) \in \Re^2$, onde a posição de um neurônio i na grade é dada pelas coordenadas (x_i, y_i) em relação a uma origem pré-fixada. Nesta abordagem, cada neurônio pode ter vizinhos à esquerda, à direita, acima, abaixo e diagonalmente, conforme figura 6 item (b).

Durante o processo de **adaptação** ou ajuste dos pesos sinápticos que ocorre o último passo da formação de um mapa auto-organizável. Nesta fase ocorre o aprendizado da rede ou retenção da informação. Uma vez que a rede SOM utiliza o aprendizado não-supervisionado, o vetor de pesos de um determinado neurônio i deve ser modificado apenas em função do estímulo de entrada. Assim, tanto o processo de competição quanto o de cooperação atuam em conjunto durante a adaptação sináptica buscando extrair algum tipo de regularidade presente no espaço de entrada X. A regra utilizada para aprendizagem do algoritmo SOM utiliza suposições feitas por Hebb (1949), visando relacionar a alteração estrutural de sinapses reais com a memória e, conseqüentemente, com aprendizagem ou experiência. Mais tarde porém, Kohonen propos uma abstração matemática das suposições de Hebb como uma regra recursiva para a adaptação ou ajuste de pesos [41]:

$$w_i(t+1) = w_i(t) + \alpha h(i^*, i; t)\,[x(t) - w_i(t)], \tag{11}$$

onde o parâmetro $0 < \alpha < 1$, é a *taxa de aprendizagem* que controla a intensidade com que os pesos sinápticos são modificados. Assim como a largura da vizinhança topológica pode diminuir com o transcorrer do treinamento, a taxa de aprendizagem α também diminui, de modo a garantir convergência e estabilidade (manutenção da memória previamente aprendida quando novos dados são apresentados à rede). Os algoritmos utilizados neste trabalho aplicam uma taxa de aprendizado variável.

4.3.3: Estabilidade, Ordenamento e Convergência

Após a inicialização dos vetores de pesos da rede SOM, o algoritmo modifica maximamente os valores dos pesos em direção a uma representação que melhor modele a estrutura (topologia) do espaço de entrada. Para obter uma configuração organizada e estável do mapa é necessária uma seleção criteriosa dos parâmetros (taxa de aprendizado e número de épocas, fator de decaimento da largura da vizinhança). Se a escolha destes parâmetros for adequada, uma configuração organizada pode ser obtida e neste ponto diz-se que o algoritmo convergiu ou atingiu um estado final.

O fator de decaimento da largura da vizinhança é extremamente importante para o processo de convergência da rede. No caso de uma função vizinhança gaussiana, esta largura é expressa em termos do valor do parâmetro $\vartheta(t)$. Inicialmente, a largura da vizinhança deve ser alta para promover um rápido ordenamento dos pesos e decrescer ao longo das épocas de treinamento de forma a garantir a convergência do algoritmo.

A estabilidade da rede SOM é obtida quando se consegue determinar em que condições a rede converge. Em geral, as funções custo empregadas em sistemas não-lineares possuem pontos de mínimos locais e quando os vetores de pesos são direcionados para estes pontos durante o processo de treinamento é provável que o estado final da rede seja indesejável.

Agora que abordamos brevemente as redes com aprendizado não-supervisionado a próxima seção abordará as redes neurais com aprendizado supervisionado.

4.4: Redes Neurais Artificiais Supervisionadas

No caso das RNAs supervisionadas a principal exigência está relacionada a saída esperada. O processo de treinamento só é finalizado quando é alcançado um nível aceitável de semelhança entre a saída atual e a desejada. Para o treinamento, essas redes utilizam informação externa para indicar a saída desejada de cada vetor de entrada. Esta arquitetura apresenta várias aplicações práticas e em geral produz bons resultados em tarefas de aproximação de funções e classificação de padrões. O restante deste capítulo apresentará as Redes *Perceptron* Multicamada e a rede *Radial Basis Function* .

4.4.1: Rede Perceptron Multicamada

Uma rede *Multilayer Perceptron* – MLP é formada por uma camada de entrada, uma ou mais camadas ocultas intermediárias, compostas de neurônios somadores que aplicam alguma função de ativação não-linear e uma camada de saída, contendo neurônios somadores (os quais podem ser lineares).

As camadas ocultas tornam a rede MLP capaz de extrair progressivamente as características mais significativas do espaço de entrada. Nesse tipo de rede, existe um alto grau de conectividade. O processo de treinamento utiliza um algoritmo de retropropagação do erro (*Error Backpropagation*).

Uma vez que o escopo deste trabalho assume a utilização de redes não-supervisionadas para a construção de classificadores e detectores de novidades, as redes neurais supervisionadas são abordadas de modo menos abrangente. Além da rede MLP padrão, existem algumas variações que também podem ser utilizadas em aplicações de detecção de novidades. Dentre estas, encontra-se a rede MLP Gaussiana (*Gaussian* MLP, GMLP) e a rede MLP Autoassociativa (*Autoassociative*MLP, AAMLP).

4.4.2: Radial Basis Functions - RBF

Outro tipo de RNA são as redes de Funções de Base Radial (*Radial Basis Function, RBF*), compostas por uma camada de entrada, apenas uma camada intermediária

onde os neurônios geralmente tem funções de ativação do tipo gaussiana e uma camada de saída com ativação linear destes neurônios.

Existem muitas maneiras de se projetar uma rede RBF e seu processo de treinamento pode ser realizado em mais de uma etapa. Pode-se inicialmente treinar a primeira camada, que pode ser formada por uma rede SOM e depois, os parâmetros da segunda camada de neurônios lineares, sendo calculada usando-se o método dos mínimos quadrados.

Quando este esquema é seguido, a cada apresentação de um vetor de entrada x na iteração t, a saída do i-ésimo neurônio da camada intermediária pode ser calculada obedecendo a equação:

$$\nu_i(t) = \varphi_i[x(t)] = exp\left[-\frac{\|x(t) - c_i\|^2}{2\gamma_i^2}\right], \tag{12}$$

em que o vetor c_i, constante para o neurônio i, define o centro do i-ésimo neurônio, enquanto a constante $\gamma_i > 0$ define a largura da função de ativação gaussiana do neurônio em questão.

4.5: Conclusões do Capítulo

Neste capítulo, apresentamos uma visão geral das RNAs e abordamos sua a classificação visando a compreensão dos modelos das redes utilizadas no desenvolvimento do protótipo de IDS conforme apresentado no capítulo 6 deste trabalho.

5. Máquinas de Vetores de Suporte

"Em tempos de crise, somente a imaginação é mais importante que o conhecimento. O conhecimento é limitado. A imaginação envolve o mundo."

Albert Einstein

5.1: Introdução

As Máquinas de Vetores de Suporte (*Support Vector Machine* – SVM) são um método de aprendizagem computacional proposto por Vapnik e Chervonenkis nas décadas de 1960 e 1970 [81], revisado posteriormente [79], para aplicação em problemas de classificação e regressão. Este método baseia-se no princípio da minimização do risco estrutural[34], o qual estabelece que para se obter um bom desempenho de generalização, um algoritmo de aprendizado de máquina deve minimizar o risco estrutural ao invés do risco empírico [69].

O risco empírico é o erro no conjunto de treinamento e o risco estrutural engloba tanto o erro no conjunto de treinamento quanto a complexidade das classes de funções que podem ser ajustadas aos dados em questão. Neste capítulo apresentamos uma visão geral das SVMs e dos métodos baseados em funções *Kernel* e por fim abordamos suas principais aplicações.

5.2: Motivação

Nos últimos anos, principalmente depois da década de 90, as SVMs receberam crescente destaque nas pesquisas da área de Aprendizado de Máquina [52], devido aos resultados obtidos com a aplicação desta técnica, muitas vezes superiores aos obtidos por outros algoritmos de aprendizado incluíndo as Redes Neurais Artificiais (RNAs).

As SVMs têm sido amplamente utilizadas na construção de aplicações de reconhecimento de padrões [26], classificação de imagens e detecção facial [59], reco-

[34] *Structural Risk Minimization* – SRM

nhecimento de escrita [78], análises de sequências biológicas [58] entre outras [80] [84], e são também uma das ferramentas amplamente utilizadas nas aplicações que requerem aprendizagem computacional desenvolvidas nos últimos anos [19]. Nas próximas seções, serão abordados conceitos básicos de aprendizagem computacional, incluíndo aprendizagem estatística e métodos baseados em funções Kernel e por fim as SVMs e suas principais aplicações.

5.3: O Problema SVM

O principal objetivo de uma SVM é a construção de um hiperplano ótimo, ou seja, através de hiperplanos que maximizam a margem de separação das classes, possibilitando a separação dos padrões de treinamento em classes distintas no conjunto de dados. Desse modo, um classificador construído usando o método SVM busca minimizar a equação 13 apresentada a seguir com base nas restrições impostas na equação 14.

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i \tag{13}$$

$$y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \xi_i \geq 0 \tag{14}$$

Satisfazendo-se ambas as equações, os padrões dos dados de treinamento são mapeados em um espaço de alta dimensão pela função Kernel utilizada. Com isso, o classificador SVM encontra um hiperplano capaz de separar o espaço linearmente com a máxima margem nesse espaço de alta dimensão. Esse mapeamento é realizado através da utilização de uma função *Kernel*, definida como $K(\vec{x}, \vec{y}) = \phi^T$.

Para exemplificar, algumas funções *Kernel* incluem funções polinomiais, RBF e Perceptron entre outras.

Vamos considerar inicialmente algumas das principais técnicas utilizadas para o treinamento de máquinas de vetores suportes, comumente aplicadas a solução do problema de otimização quadrática na forma dual de Wolfe, também conhecido como problema SVM [18], e descrito como:

$$\begin{aligned} &\text{maximizar } \Delta \cdot 1 - \tfrac{1}{2} \cdot \Delta^T \cdot H \cdot \Delta \\ &\text{Sujeito a:} \\ &\Delta^T \cdot Y = 0 \\ &0 \leq \Delta \leq C \end{aligned} \tag{15}$$

onde o vetor de multiplicadores é $\Delta = (\alpha_1, \alpha_2, ..., \alpha_m)$ e o vetor de rótulos de valores binários é $Y = (y_1, y_2, ..., y_m)$ e H é a matriz Hessiana simétrica positiva semi-definida com todos os autovalores não negativos, na forma:

$$H = [h_{i,j}] \text{ onde } h_{i,j} = y_i \cdot y_j \cdot K_{i,j}$$

associada a um conjunto de treinamento $Z = \{(x_i, y_i)\}$ e a uma função *Kernel* $k : \Re^d \times \Re^d \rightarrow \Re$, onde $K_{i,j} = k(x_i, x_j)$.

Se a matriz H for positiva definida, a função objetiva do problema tem a forma estritamente convexa e sua solução ótima global relativa a um ponto de máximo que satisfaça as condições KKT é única, podendo ser obtida segundo [30] pelo uso de algum método de otimização quadrática convexa. Se a matriz Hessiana for positiva semi-definida a solução obtida pode ser global e única.

No caso mais geral, a solução não será única se dado alguma solução Δ, escolhemos um vetor $\acute{\Delta}$ pertencente ao espaço nulo da matriz, sendo o vetor $\acute{\Delta}$ ortogonal ao vetor unitário, derivando uma solução $\Delta + \acute{\Delta}$ também ótima. Contudo, a solução encontrada será sempre uma solução ótima global, em contraste com as técnicas de Redes Neurais Artificiais que empregam o algoritmo *backpropagation*, onde podem existir muitas soluções de máximos locais.

5.4: A Dimensão Vapnik-Chervonenkis

Para a escolha da função que melhor se ajusta ao conjunto de treinamento, deve-se usar uma medida de perda ou discrepância (dimensão VC) que indica à máquina quando ocorrem erros ou acertos durante a aprendizagem [80].

No caso de problemas de classificação binária, a função de perda utilizada é:

$$P(y, f(x,z)) = \begin{cases} 1 & \text{se } f(x,z) \neq y \\ 0 & \text{se } f(x,z) = y \end{cases} \quad (1)$$

em que x é a entrada da máquina, z é parâmetro da função indicadora e $f(x,z)$ é a saída da máquina.

Um princípio indutivo geralmente empregado pelas máquinas de aprendizagem existentes é a **minimização do risco empírico** calculado usando (1) sendo obtido pela equação:

$$R_{empirico} = \frac{1}{N}\left[\sum_{I=1}^{N} P(y, f(x,z))\right] \quad (16)$$

A minimização de $R_{empirico}$ não garante a obtenção de resultados adequados, pois ela não considera a complexidade das funções indicadoras. Quando a complexidade das funções é superior à necessidade do problema, ocorre o *overfitting*[35] da função em relação ao conjunto de treinamento. Quando é inferior, ocorre o sub-ajuste (*underfitting*). Nos dois casos tem-se a redução da capacidade de generalização.

Usando o conceito de dimensão VC [79], foi desenvolvida uma expressão, com probabilidade $(1-n)$ de ocorrer, indicando que o valor do limite superior do risco funcional é:

$$R_{funcional} \leq R_{empirico} + R_{bound}(h, \eta, N) \quad (3)$$

dado um valor de $\eta \in [0,1]$. A dimensão VC é h e N indica o número de exemplos de treinamento.

A minimização do risco estrutural tem como objetivo minimizar $R_{bound}(h, \eta, N)$, o fator bound somado ao risco empírico em (3). Este princípio usa a dimensão VC para controlar a complexidade das funções indicadoras, de forma a adequá-las a cada

[35] Ocorre quando um modelo estatístico descreve o erro aleatório ou ruído ao invés da relação subjacente. *Overfitting* geralmente ocorre quando um modelo é excessivamente complexo, ten muitos graus de liberdade em relação à quantidade de dados disponíveis. Um modelo que apresenta *overfitting* tende a ter seu desempenho preditivo comprometido.

problema.

5.5: Máquinas de Vetores Suporte

SVM é um método de obtenção do limite ótimo de separação de dois conjuntos em um espaço-vetor independente da distribuição probabilística dos dados do conjunto de treinamento. A idéia fundamental é bastante simples: localizar os limites que estão mais distantes dos vetores mais próximos da fronteira de separação em ambos os conjuntos. Esta idéia simples e tradicional, nos últimos anos tem atraído muito a atenção de pesquisadores da área devido à introdução do método do Kernel, o que equivale a uma transformação do vetor para a localização de uma fronteira no espaço não-linear resultante onde os dados podem ser separados.

A técnica de SVM foi inicialmente desenvolvida para classificação binária (duas classes) ótima, e mais tarde foi estendida para suportar regressão e problemas de classificação de múltiplas classes. SVM representa ainda um caso particular de métodos baseados em Kernels, pois realiza o mapeamento de vetores de características em um espaço de alta dimensão usando funções previamente escolhidas e para o novo espaço linear resultante, cria-se um hiperplano ótimo de separação das classes.

Como resultado da aplicação desta técnica, obtém-se uma solução dita ótima, onde a margem entre o hiperplano e os vetores de características mais próximos das duas classes é máxima. Os vetores de características que encontram-se mais próximos do hiperplano são chamados de vetores de suporte (*support vectors*).

Entre as principais vantagens da técnica de SVM sobre outros métodos de classificação estão:

- Não possuem mínimos locais
- Fase de testes é rápida
- Capacidade de generalização alta, evitando sobretreinamento (*overfitting*).
- Robustez para categorização de dados com dimensões altas, que tendem a ser sobretreinados em outros classificadores pois muitas microcaracterísticas podem

discriminar muito pouco.

- Convexidade da função objetivo pois esta é uma função quadrática com apenas um ótimo global.
- Teoria bem estabelecida nas áreas de matemática e estatística.
- A etapa de treinamento das SVMs pode ser supervisionado ou não-supervisionado.

Como principais desvantagens da técnica de SVM sobre outros métodos de classificação temos:

- Pouca robustez com dados de grande dimensão;
- Precisão é afetada por atributos pouco relevantes;
- Dificuldade em lidar com dados contínuos;

As máquinas de vetores de suporte são utilizadas para designar todos os métodos de separação através do hiperplano ótimo, contudo a definição original do termo conforme proposta por [79], designa a construção que utiliza núcleos ou **funções *Kernel*** para aumentar a dimensionalidade.

5.6: Dados Linearmente Separáveis

Vamos considerar a utilização do hiperplano de separação para o caso em que os dados são linearmente separáveis.

Um exemplo da idéia de um hiperplano de separação pode ser visualizado na figura 9 apresentada a seguir:

Nosso objetivo é encontrar o hiperplano ótimo de separação capaz de separar de maneira exata um conjunto do outro. Note que o hiperplano ótimo de separação deve classificar tanto os vetores de treinamento quanto os vetores novos, os quais não estão inicialmente presentes nos dados de treinamento.

Assumindo-se uma abordagem simplificada deste problema, sem qualquer estimativa da distribuição probabilística dos dados, o hiperplano ótimo é definido como

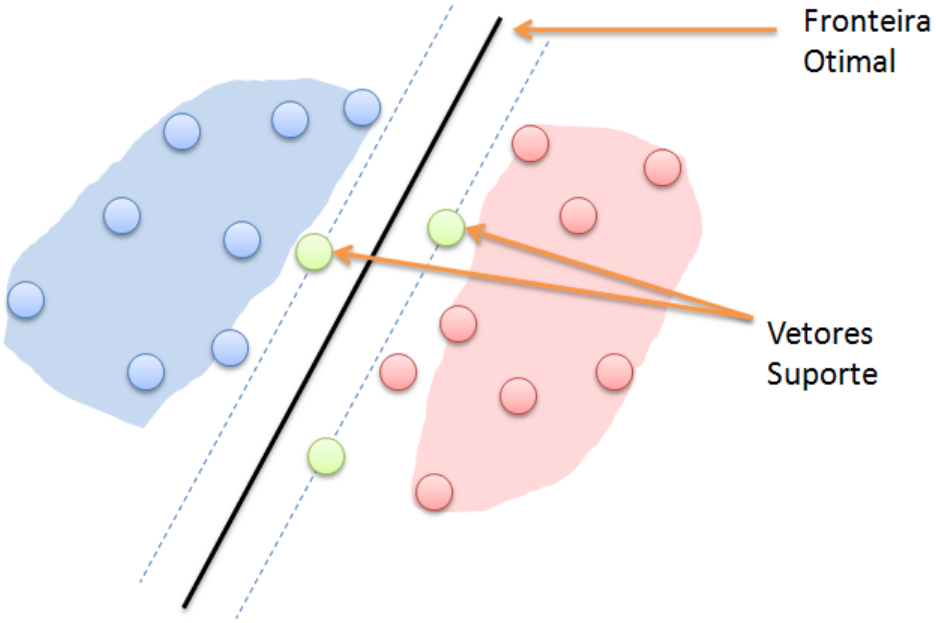


Figure 9: Hiperplano de separação SVM.

sendo o mais distante possível de ambos os conjuntos. Em termos simples, este hiperplano está no ponto médio da distância entre estes conjuntos. Uma vez que a distribuição de cada conjunto é desconhecida, esta fronteira deve classificar os conjuntos de maneira otimal, já que representa a fronteira mais distante de ambos os conjuntos. Os vetores de treinamento próximos a fronteira definida pelo hiperplano ótimo são chamados de vetores de suporte (*support vectors*).

Assim, seja x um vetor em um espaço vetorial. O hiperplano separador é expresso como sendo um dos hiperplanos definidos na equação 17:

$$w^T x + b = 0 \tag{17}$$

onde w é o peso e b é um bias. A distância entre um vetor de treinamento x e a fronteira, chamada de margem pode ser expressa como em 18:

$$\frac{\left|w^T x_i + b\right|}{\|w\|} \tag{18}$$

Uma vez que os hiperplanos são expressos pela equação 18, onde w e b são

multiplicados por um fator constante comum, introduzimos uma restrição a esta expressão conforme 19:

$$\min_i \left| w^T x_i + b \right| = 1 \tag{19}$$

A fronteira otimal maximiza a equação 18. Aplicando a restrição da equação 19, isto pode ser reduzido a maximização de 20:

$$1/\|w\|^2 = 1/w^T w \tag{20}$$

Consequentemente, a otimização é formalizada como sendo:

$$\text{minimizar } w^T w \tag{21}$$

Sendo a equação 21 sujeita as restrições:

$$y_i(w^T x_i + b) \geq 1 \tag{22}$$

onde y_i é 1 se x_i pertence a um conjunto e -1 se x_i pertence ao outro conjunto. Se a fronteira estabelecida pelo hiperplano de separação for capaz de classificar os vetores corretamente, $y_i(w^T x_i + b) \geq 0$ então esta será a margem de separação entre os dois conjuntos. Com isso, a otimização condicional é obtida pelo coeficiente intermediário do método Lagrangeano. Formalmente, definimos a função:

$$L(w, b, \alpha_i) = \frac{1}{2} w^T w - \sum_i \alpha_i \left[y_i(w^T x_i + b) - 1 \right] \tag{23}$$

onde $\alpha_i \geq 0$ são os coeficientes intermediários. Se w e b tiverem o valor ótimo, suas derivadas parciais:

$$\begin{aligned} \frac{\partial L}{\partial w} &= w - \textstyle\sum_i \alpha_i y_i x_i \\ \frac{\partial L}{\partial b} &= -\textstyle\sum_i \alpha_i y_i \end{aligned} \tag{24}$$

serão igual a zero. Quando as derivadas parciais da equação 24 são iguais a zero, nós temos:

$$w = \sum_i \alpha_i y_i x_i \tag{25}$$

$$\sum_i \alpha_i y_i = 0 \tag{26}$$

Reescrevendo-se a equação 23 obtemos a equação 27:

$$L(w, b, \alpha_i) = \frac{1}{2} w^T w - \sum_i \alpha_i y_i w^T x_i - b \sum_i \alpha_i y_i + \sum_i \alpha_i, \tag{27}$$

realizando as substituições necessárias nas equações 25, 26, 27 obtemos a equação 28:

$$\begin{aligned} L(w, b, \alpha_i) &= \tfrac{1}{2} \left(\textstyle\sum_i \alpha_i y_i x_i\right)^T \left(\textstyle\sum_j \alpha_j y_j x_j\right) - \textstyle\sum_i \alpha_i y_i \left(\textstyle\sum_j \alpha_j y_j x_j\right)^T x_i + \textstyle\sum_i \alpha_i \\ &= -\tfrac{1}{2} \textstyle\sum_i \textstyle\sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \textstyle\sum_i \alpha_i. \end{aligned} \tag{28}$$

Neste caso, a contribuição do segundo termo da equação 23 deve ser mínima e o valor de L deve ser maximizado sendo sujeito a α. Consequentemente, a otimização é reduzida a um problema de programação com complexidade computacional quadrática conforme:

$$\begin{aligned} &\text{maximizar } -\tfrac{1}{2} \textstyle\sum_i \textstyle\sum_j \alpha_i \alpha_j y_i y_j x_i^T x_j + \textstyle\sum_i \alpha_i, \\ &\text{onde } \textstyle\sum_i \alpha_i y_i = 0, \alpha_i \geq 0. \end{aligned} \tag{29}$$

Nesta seção foram discutidos os aspectos matemáticos para o caso dos dados linearmente separáveis. A próxima seção abordará o caso em que os dados são não-linearmente separáveis.

5.7: Dados Não-Linearmente Separáveis – Método Soft Margin

Para os casos em que os conjuntos de pontos de dados são não-linearmente separáveis, não existe um hiperplano capaz de resolver um problema de classificação binária de maneira exata. Nestes casos podem existir *overlaps* entre as classes, provo-

cando violações de restrições de classificação dos dados do sistema. Assim, o método chamado de *Soft Margin* é a solução comumente aplicada para a correção deste tipo de problema. Este método consiste na introdução de variáveis de folga, não negativas segundo [79], as quais permitirão que o conjunto de treinamento seja separado linearmente com um número mínimo de erros relacionado ao controle da capacidade do classificador.

O efeito da utilização das variáveis de folga na formulação matemática das equações das SVMs pode ser visualizado na figura 10:

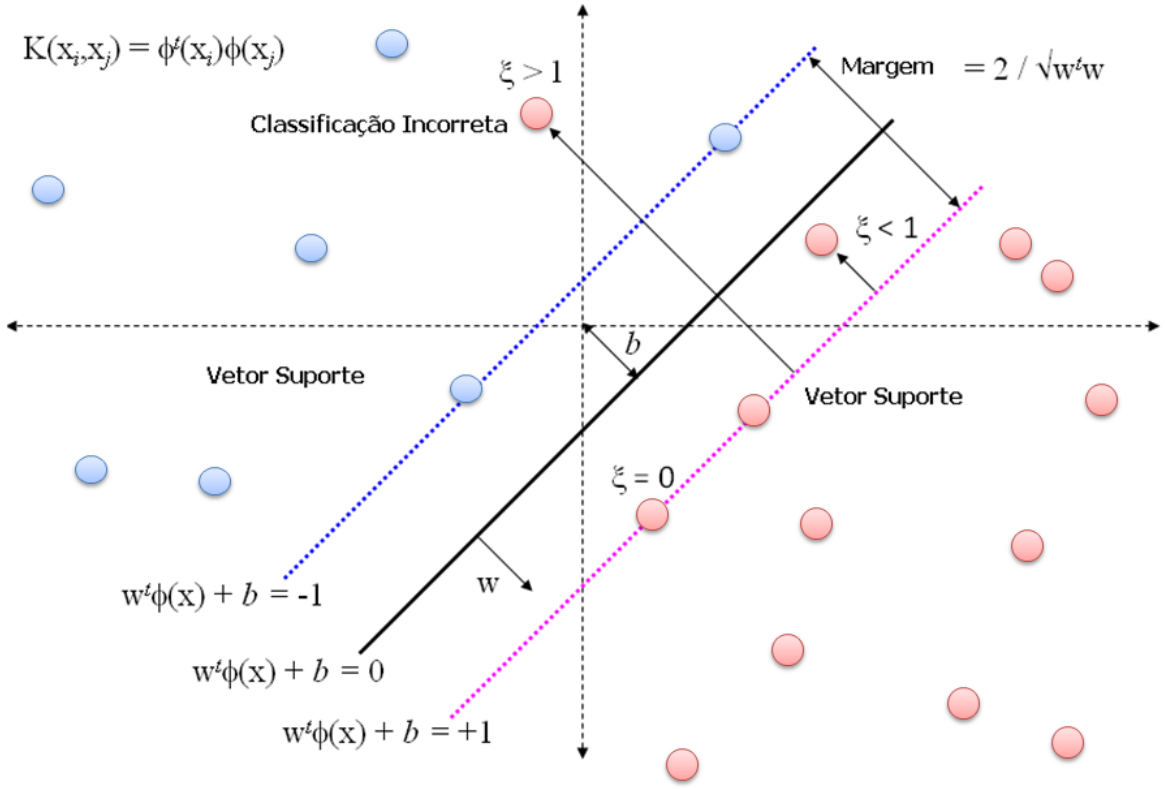


Figure 10: Introdução de variáveis de folga.

Aplicando o conceito do método *soft margin*, subtituímos as restrições impostas à equação 21 com as seguintes:

$$\text{sujeito a } y_i(w^T x_i + b) \geq 1 - \xi_i, \tag{30}$$

onde ξ_i, é chamado de variável de folga, sendo um valor positivo que indica a tolerância de classificações errôneas. Esta substituição indica que podem existir vetores ou pontos

de treinamento em uma região limitada no lado errado ao longo da fronteira de separação conforme mostrado na figura 11.

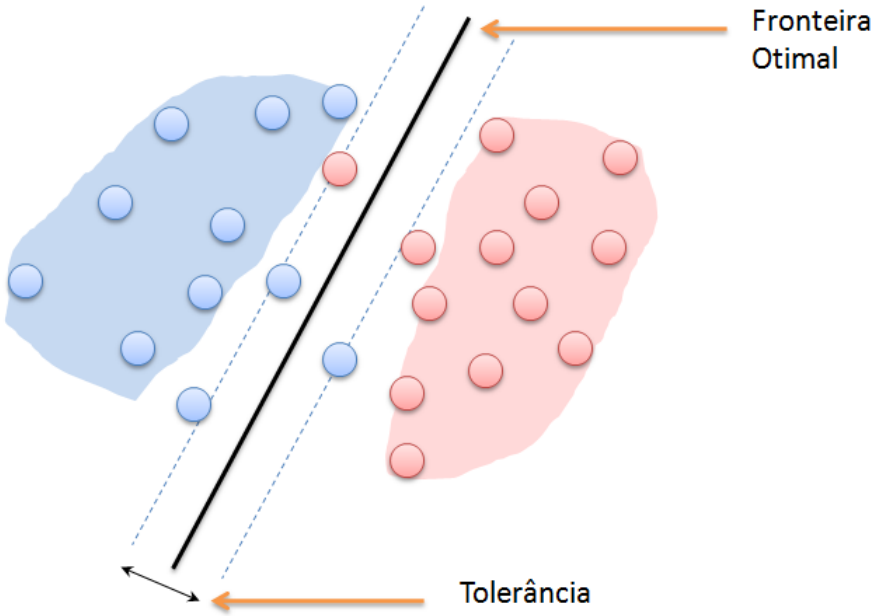


Figure 11: Vetores ou pontos não são perfeitamente linearmente separáveis.

Diversas funções de otimização são propostas para este caso. Por exemplo:

$$\text{minimizar } w^T w + C \sum_i \xi_i, \tag{31}$$

em que o segundo termo da expressão em 31 é uma penalidade para o erro de classificação e a constante C determina o grau de contribuição deste segundo termo. E o vetor w se refere ao vetor normal do hiperplano separador.

A definição da margem sobre o plano de separação pode apresentar o aspecto apresentado na figura[36] 12:

Outra formulação alternativa válida seria tomarmos $\Phi(u) = u$ e $\sigma = 1$, em que segundo [18], a primeira solução do classificador com *soft margin* definida em 31 pode ser reescrita em termos da minimização da norma linear ou normal L_1 conforme a equação 32:

[36] Adaptada de http://journals.iucr.org/d/issues/2008/08/00/hv5107/index.html

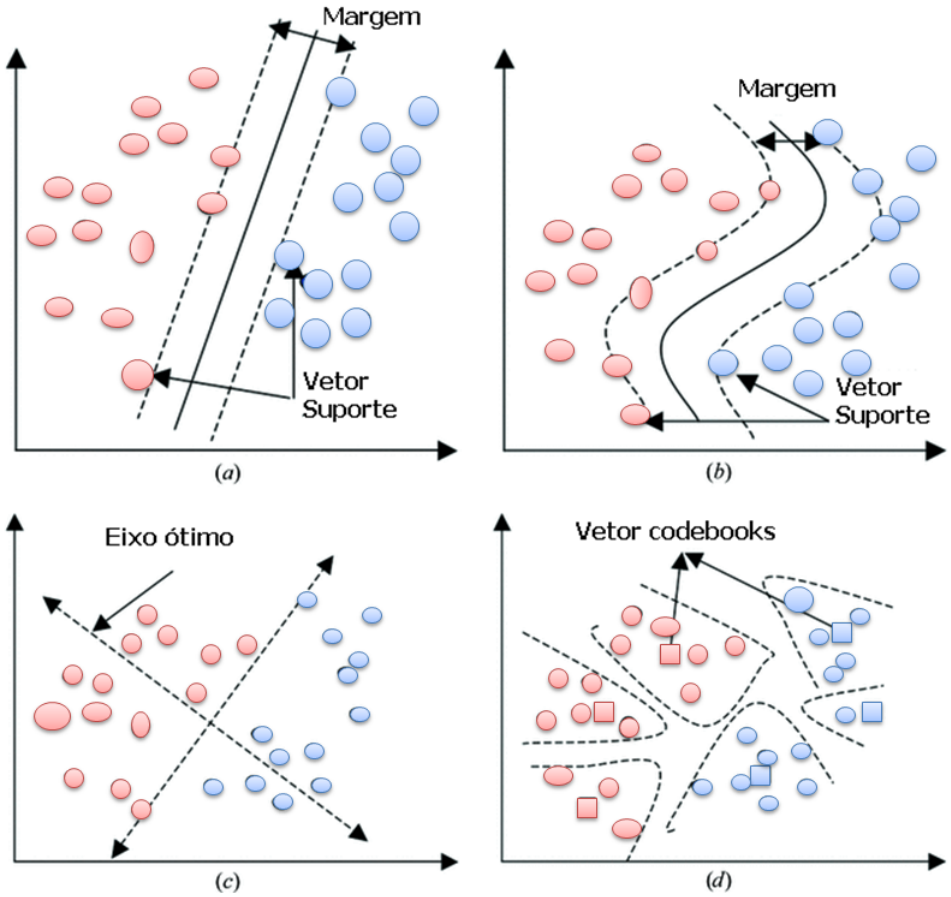


Figure 12: Margem entre hiperplano de separação linear e não linear.

$$\text{minimizar } \frac{1}{2}w^T w + C\sum_i \xi_i, \tag{32}$$

$$\begin{gathered}\text{Sujeito a } y_i \cdot (w \cdot x_i + b) + \xi_i - 1 \geq 0, para \\ i = 1, ..., m \\ \xi_i \geq 0\end{gathered} \tag{33}$$

Após introduzirmos os multiplicadores α e μ, obtemos:

$$\begin{gathered}
\text{minimizar } -\frac{1}{2}w^t w + C\sum \xi_i - \sum \alpha_i y_i (w x_i + b) + \sum \alpha_i - \sum \alpha_i \xi_i - \sum \mu_i \xi_i \\
\text{ou} \\
\text{minimizar } -\frac{1}{2}w^t w - \sum \alpha_i y_i (w x_i + b) + \sum \alpha_i - \sum \xi_i - \sum \xi_i (C - \alpha_i - \mu_i) \\
\text{onde } \alpha_i, \mu_i \geq 0.
\end{gathered} \tag{34}$$

Aplicando o gradiente da função lagrangeana em relação a w, b e ξ igualando a zero para satisfazer as condições de otimalidade de primeira ordem temos:

$$\begin{aligned}
\frac{\partial L}{\partial w} = 0 \Rightarrow w - \sum \alpha_i y_i x_i = 0 \\
\frac{\partial L}{\partial b} = 0 \Rightarrow \alpha_i y_i = 0 \\
\frac{\partial L}{\partial \xi_i} = 0 \Rightarrow C - \alpha_i - \mu_i = 0
\end{aligned} \tag{35}$$

Realizando uma substituição no valor de w na função lagrangeana e introduzindo as demais equações como restrições do problema, anulamos a dependência da função objetivo em relação aos parâmetros w, b e ξ, estabelecendo o problema na forma dual de Wolfe:

$$\begin{gathered}
\text{maximizar } L(\alpha) = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \\
\text{Sujeito a: } \sum_i y_i \alpha_i = 0, \\
C - \alpha_i - \mu_i = 0 \\
\mu_i \geq 0, \alpha_i \geq 0.
\end{gathered} \tag{36}$$

Considerando $\mu_i = C - \alpha_i$ e $\mu_i \geq 0$, temos $\alpha_i \geq C$, tornando possível reescrevermos o problema SVM na forma quadrática conforme apresentado a seguir:

$$\begin{gathered}
\text{maximizar } L(\alpha) = \sum_i \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle, \\
\text{Sujeito a: } \sum_i y_i \alpha_i = 0, \\
0 \geq \alpha_i \geq C
\end{gathered} \tag{37}$$

Desse modo a resolução do problema primal e dual, associada às equações de complementaridade, estabelecem as seguintes condições de otimalidade, conhecidas como condições de KKT (*Karush-Kuhn-Tucker*) definidas em 38:

$$\begin{gathered} y_i \cdot (w \cdot x_i + b) + \xi_i - 1 \geq 0, \\ \xi_i, \alpha_i, \mu_i \geq 0 \\ \alpha_i(y_i(wx_i + b) + \mu_i - 1) = 0 \\ \mu_i, \xi_i = 0 \end{gathered} \tag{38}$$

Podendo ser feita a seguinte análise de viabilidade considerando a flexibilização na classificação dos dados em 39:

$$\begin{gathered} \text{Caso (a): O vetor encontra-se fora das margens.} \\ \text{Se } \alpha_i = 0 \Rightarrow \mu_i = C \Rightarrow \xi_i = 0 \\ \text{derivando: } y_i \cdot (w \cdot x_i + b \sum) - 1 \sum \geq 0, \\ \text{Caso (b): O vetor ultrapassa as margens.} \\ \text{Se } \alpha_i = C \Rightarrow \mu_i = 0 \Rightarrow \xi_i \geq 0 \\ \text{derivando: } y_i \cdot (w \cdot x_i + b) - 1 \leq 0, \\ \text{Caso (c): O vetor está sobre a margem.} \\ \text{Se } 0 < \alpha_i < C \Rightarrow \mu_i > 0 \Rightarrow \xi_i = 0 \\ \text{derivando: } y_i \cdot (w \cdot x_i + b) = 1. \end{gathered} \tag{39}$$

Na próxima seção será apresentado o principal artifício utilizado para a separação de dados, nos casos em que um hiperplano separador linear não é capaz de separar totalmente os dados em questão.

5.8: Métodos baseados em Kernel

Para o caso em que o conjunto de dados não é linearmente separável, existem algumas funções baseadas em *Kernels* que podem ser usadas para a construção de classificadores usando SVMs. As SVMs aprendem um classificador da forma:

$$f(x) = sgn\left(\frac{1}{l}\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b\right) \tag{40}$$

que é um hiperplano em algum espaço característico definido implicitamente por uma função *Kernel* K.

O objetivo das funções *Kernel* é realizar a projeção dos vetores de características de entrada em um espaço de características com alta dimensão para permitir a classificação em espaços não-linearmente separáveis. Considerando ainda o caso dos dados não-linearmente separáveis, conforme a figura 11, se o espaço bidimensional for convertido ou transformado em um espaço tri-dimensional, usando-se uma função *kernel* apropriada neste espaço de dimensionalidade superior, é possível realizar a separação entre os vetores nos dados de treinamento.

Seja Φ uma transformação para um espaço dimensional superior. Essa transformação pode ser obtida através do uso de várias funções de mapeamento. Após a transformação, os dados são separados de forma linear no espaço de características através da construção de um hiperplano separador por um classificador SVM.

O espaço transformado deve satisfazer a condição de que a distância definida neste novo espaço e a do espaço original tenham uma relação entre si. Neste caso, a função *Kernel* $K(x, x^{'})$ é apresentada para satisfazer a estas condições conforme expresso na equação 41:

$$K(x, x^{'}) = \Phi(x)^{T}\Phi(x^{'}). \tag{41}$$

A equação 41 indica que a função *Kernel* é equivalente a distância entre x e $x^{'}$ medida no espaço dimensional superior conforme transformado por Φ. Assim, se medirmos a margem com esta função *Kernel* e executarmos uma otimização, obtemos uma fronteira não-linear que pode ser expressa na forma da equação 42:

$$w^{T}\Phi(x) + b = 0 \tag{42}$$

Realizando ainda uma substituição da equação 25 na 42, trocando x por $\Phi(x)$ temos a equação 43:

$$\sum_{i} \alpha_i y_i \Phi(x_i^{T})\Phi(x) + b = \sum_{i} \alpha_i y_i K(x_i, x) + b = 0 \tag{43}$$

A otimização da função da equação 29 no espaço transformado é também obtida

pela substituição:

$$x_i^T x_j \text{ com } K(x_i, x_j) \tag{44}$$

Este resultado significa que todo o cálculo pode ser obtido apenas usando-se $K(x_i, x_j)$ e não sendo necessário conhecer a função Φ ou qual é na verdade o espaço transformado resultante. Uma condição suficiente para satisfazer a equação 41 é que K seja definido como sendo um valor positivo. Existem diversos exemplos de funções *Kernel* bem conhecidas, tais como:

$$\begin{aligned} K(x,z) &= \langle x,z\rangle \text{ Kernel Linear discriminant} \\ K(x,z) &= (\langle x,z\rangle + 1)^d \text{ Kernel Polinomial} \\ K(x_i,x') &= (x^T x' + 1)^P \\ K(x_i,x') &= \exp\left(-\frac{\|x-x'\|^2}{\sigma^2}\right) \text{ Kernel Gaussiano} \end{aligned} \tag{45}$$

Um exemplo simples é o seguinte: $\Phi : \Re^2 \longrightarrow \Re^3$ em que $(x_1, x_2) \longmapsto (z_1, z_2, z_3) \Leftrightarrow (x_1^2, \sqrt{2x_1x_2}, x_2^2)$, calculando o produto escalar dos valores mapeados temos os resultados descritos em 46:

$$\begin{aligned} (\Phi(x)\Phi(y)) &= (x_1x_2, \sqrt{2x_1x_2}, x_2^2)^T (y_1y_2, \sqrt{2y_1y_2}, y_2^2) \\ &= ((x_1,x_2)^T (y_1,y_1)^2) \\ &= (x\cdot y)^2 \\ &= K(x,y) \end{aligned} \tag{46}$$

5.9: Detecção de Novidade usando SVM

Em diversas aplicações reais a tarefa principal não é classificar dados, antes detectar novidades ou instâncias que não estavam presentes no conjunto de treinamento. Detecção de novidades ou anomalias tem tido aplicações em muitos domínios envolvendo monitoramento e diagnóstico médico. Neste sentido, uma abordagem é modelar o *support* da distribuição dos dados (ao invés de tentar encontrar uma função real para estimar a densidade dos dados analisados). Assim, de maneira simples o objetivo é criar

uma função binária que seja positiva nas regiões do espaço de entrada onde os dados estão presentes e negativa em outras regiões onde eles não estão presentes.

Uma abordagem é encontrar uma hiper-esfera com o valor mínimo de raio R e centro a que contenha grande parte dos dados: os pontos de teste de novidades ficam fora dos limites impostos pela hiper-esfera. A técnica descrita a seguir foi originalmente sugerida por Tax e Duin [75] [74] e usada por estes autores em aplicações reais. O efeito dos resultados é reduzido pelo uso de variáveis de folga z para permitir pontos de dados fora da esfera demarcatória. Em termos práticos, a principal tarefa é minimizar o volume da esfera e a distância em relação aos pontos do lado de fora da esfera. Isto pode ser formalizado pela equação 47:

$$\underset{R,z,a}{\text{minimizar}} \; R^2 + \frac{1}{m\nu}\sum_{i=1}^{m} z_i (x_i - a)^T (x_i - a) \leq R^2 + z_i \tag{47}$$

$$\begin{gathered}\text{Sujeito a: } z_i \geq 0 \\ i = 1, ..., m\end{gathered} \tag{48}$$

Usando a mesma metodologia de resolução do problema SVM de classificação, o problema dual Lagrangeano é formulado e as funções Kernel são substituídas para produzir a seguinte tarefa dual QP para detecção de novidades:

$$\underset{\alpha}{\text{minimizar}} \; -\sum_{i=1}^{m} \alpha_i K(x_i, x_i) + \sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i, x_j) \sum_{i=1}^{m} \alpha_i = 1 \tag{49}$$

$$\begin{gathered}\text{Sujeito a: } \tfrac{1}{m\nu} \geq \alpha_i \geq 0 \\ i = 1, ..., m\end{gathered} \tag{50}$$

Se $m\nu > 1$, então os exemplos sobre os limites demarcados pela hiper-esfera (conforme a figura 13) ocorrerão com $\alpha_i = 1/m\nu$ e estes correspondem ao resultado obtido com o processo de treinamento. Após completar o processo de treinamento um ponto de dados de teste ν é declarado como sendo novo se:

$$K(v,v) - 2\sum_{i=1}^{m} \alpha_i K(v,x_i) + \sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i,x_j) - R^2 \geq 0 \tag{51}$$

onde R^2 é calculado primero por encontrar um exemplo que não esteja dentro da esfera e a inegualdade é ajustada para uma igualdade com a equação 51 sendo zero.

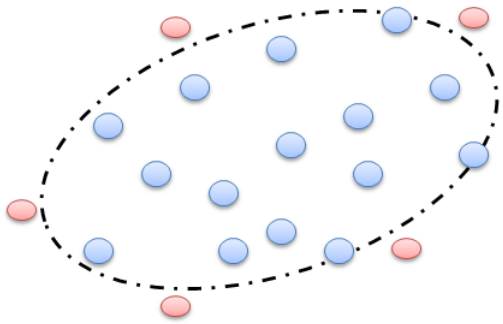


Figure 13: Detecção de pontos fora dos limites da hiper-esfera são vistos como novidade nos dados.

Uma abordagem alternativa vem sendo desenvolvida por Schölkopf [66] [67]. Suponha que façamos uso de Kernels RBF: neste caso os dados ficam numa região da superficie da hiper-esfera no espaço característico onde $\theta(x) \cdot \theta(x) = K(x,x) = 1$. O objetivo portanto é separar esta região da superfície que não contém dados. Isto é obtido pela construção de um hiperplano que possua a distância máxima da origem com todos os pontos de dados situados no lado oposto da origem, onde $w \cdot x_i - b \geq 0$. Após a substituição de Kernel da formulação dual a tarefa de aprendizado envolve a minimização da equação 52:

$$\underset{\alpha}{\text{minimizar}} \; -\sum_{i,j=1}^{m} \alpha_i \alpha_j K(x_i,x_j) \sum_{i=1}^{m} \alpha_i = 1 \tag{52}$$

$$\begin{gathered} \text{Sujeito a: } \frac{1}{mv} \geq \alpha_i \geq 0 \\ i = 1,...,m \end{gathered} \tag{53}$$

Para determinar b procuramos um exemplo k que não esteja nos limites (α_i e β_i são diferente de zero e $0 < \alpha_i < 1/mv$) e obtemos b calculando:

$$b = \sum_{j=1}^{m} \alpha_i K(x_i, x_k) \tag{54}$$

O *support* da distribuição é modelado pela função de decisão 55:

$$f(z) = sgn\left(\sum_{j=1}^{m} \alpha_j K(x_j, v) - b\right) \tag{55}$$

O parâmetro v tem uma interpretação clara sendo um limite superior em parte dos pontos fora e um limite inferior de parte dos padrões que são vetores suporte. Schölkopf em [66] apresenta uma boa evidencia experimental em favor desta abordagem incluindo reconhecimento de novos dígitos de caracteres manuscritos. Para o modelo de Schölkopf a origem espacial característica desempenha um papel especial. Ela aparece efetivamente como uma prévia para a origem da classe de ocorrências anormais. Ao invés de repelir da origem poderíamos considerar atrair o hiperplano para os pontos de dados no espaço característico. No espaço de entrada, isto corresponde a uma superfície que envolve os dados agregados e pode ser obtida através da seguinte tarefa de programação linear [12]:

$$\underset{w, b, z}{minimizar} \sum_{i=1}^{m} \left(\sum_{j=1}^{m} \alpha_j K(x_i, x_j) - b\right) + \lambda \sum_{i=1}^{l} z_i \tag{56}$$

$$\text{Sujeito a: } \left(\sum_{j=1}^{m} \alpha_j K(x_i, x_j) - b\right) + z_i \geq 1 z_i \geq 0, \alpha_i \geq 0, i = 1, ..., m \tag{57}$$

O parâmetro b é tratado apenas como um parâmetro adicional no processo de minimização. Ruídos e novos pontos são alcançados pela introdução de uma margem suave com erro z. Este método tem sido utilizado com sucesso para a detecção de anomalias em amostras de sangue e detecção de falhas em aplicações de monitoramento [12].

5.10: Implementações de SVMs

Atualmente, diversos *softwares* comerciais para resolução de problemas desta natureza encontram-se disponíveis na forma de ferramentas comerciais e/ou bibliotecas de software de código fonte aberto. Entre as ferramentas utilizadas na implementação e uso de SVMs, durante o desenvolvimento deste trabalho encontram-se: a ferramenta Weka - [86] e a biblioteca Libsvm - [14], sendo que outros softwares que implementam SVMs e suas extensões podem ser consultados em `http://www.support-vector-machines.org/SVM_soft.html`.

5.11: Conclusões do Capítulo

Neste capítulo, apresentamos as definições dos conceitos envolvidos na operação das SVMs para facilitar o entendimento das técnicas disponíveis para classificação e reconhecimento de padrões utilizando as SVMs.

Abordamos de forma resumida o tratamento que as SVMs dispensam aos dados linearmente separáveis e em seguida, analisamos os casos em que os dados não-linearmente separáveis são processados utilizando funções Kernel. Por fim, encerramos o capítulo indicando algumas das implementações de SVMs de código aberto utilizadas neste trabalho.

6. Modelo Proposto de Sistema de Detecção de Intrusão Inteligente

"A mente que se abre a uma nova idéia jamais volta ao seu tamanho original."

Albert Einstein

6.1: Introdução

O objetivo deste capítulo é apresentar a arquitetura de um sistema de detecção de intrusão conforme proposto por este trabalho. Esta arquitetura utiliza técnicas de inteligência artificial e de aprendizado de máquina, visando alta taxa de detecção e baixa taxa de falsos positivos, mesmo quando aplicada a dados de tráfego real de rede. Além disso, testes do sistema desenvolvido foram executados para verificação dos resultados em comparação com outros IDSs propostos por outros autores na literatura.

Sendo assim, este capítulo está organizado da seguinte maneira: começamos o capítulo abordando os dados de entrada do sistema. Depois tratamos das técnicas utilizadas na construção do protótipo. Na seção 6.4 apresentamos a arquitetura do IDS desenvolvido o qual chamamos de Polvo-IIDS. Em seguida detalhamos o funcionamento do protótipo e na seqüência, na seção 6.6, apresentamos os testes realizados com o sistema desenvolvido e encerramos o capítulo analisando os resultados obtidos.

6.2: Dados de Treinamento

Os dados utilizados nos experimentos realizados com o protótipo construído do Polvo-IIDS foram os do DARPA 98, utilizados no KDD Cup 99[37]. Utilizamos estes dados para validar o protótipo desenvolvido em razão de sua freqüente escolha em testes de outros IDSs conforme publicados na literatura. Essa escolha torna possível a comparação com sistemas que empregam arquiteturas e técnicas similares às do Polvo-IIDS.

[37] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

No entanto, estes dados apresentam sérias limitações, uma vez que a taxa de ataques não é natural. Cerca de 80% das instâncias correspondem a ataques com apenas uma conexão. Os dados para tráfego normal são gerados por simuladores, sem a presença de pacotes fragmentados ou desordenados o que não corresponde aos dados reais da Internet.

Para utilização dos dados do DARPA 98 no protótipo foram realizadas converções para um formato de arquivo padrão chamado de ARFF[38], devido a simplicidade de utilização por ferramentas de mineração de dados, tais como a ferramenta *Weka*[39], utilizada como *framework* para a implementação do protótipo desenvolvido. Na próxima seção apresentamos as características das técnicas utilizadas no desenvolvimento da arquitetura do IDS proposto.

6.3: Técnicas Utilizadas

Os sistemas de detecção de intrusão inteligentes encontrados na literatura, geralmente aplicam apenas uma arquitetura específica de rede neural e nem sempre obtêm taxas de detecção satisfatórias. Tais sistemas enfrentam dificuldades no treinamento (problemas de convergência e baixa capacidade de generalização) das redes neurais devido à alta variância de comportamento do tráfego presente nas redes que compõe a Internet.

No modelo de IDS proposto neste trabalho utilizamos uma rede SOM como classificador de ataques e usamos verificadores SVMs para determinar se os exemplos classificados como ataque pela rede SOM são de fato ataques, ou se são apenas tráfego normal. O objetivo da criação deste modelo é a verificação do desempenho e da precisão da abordagem de IDS em duas camadas.

6.3.1: Uso da rede SOM

Ao analisarmos as características da rede neural de Kohonen (*Self-organizing Maps*), destacamos sua habilidade de classificar dados de maneira genérica e auto-

[38] *Attribute-Relation File Format*
[39] *Waikato Environment for Knowledge Analysis*

organizada. Adicionalmente, as redes SOM utilizam treinamento não-supervisionado, o que desobriga a utilização de um especialista para a validação de cada instância de treinamento da rede neural. Este tipo de rede neural tem a habilidade de representar um conjunto grande de vetores de entrada por meio de um conjunto menor de vetores localizados em um espaço de dimensão mais baixa. Ou seja, seu objetivo ao processar os dados que recebe é realizar a quantização do espaço de entrada e a redução de dimensão no espaço de saída resultante.

Na arquitetura de IDS desenvolvida, a rede SOM exerce o papel de classificador dos dados de tráfego de rede analisado.

6.3.2: Uso de SVMs

As Máquinas de Vetores Suporte – SVMs são outra técnica amplamente utilizada com bons resultados de verificação por que possuem boa capacidade de generalização quando treinadas para separar dados em duas classes.

As SVMs são um conjunto de métodos de aprendizagem supervisionada que podem ser utilizados em tarefas de classificação e regressão. Em termos simples, dado um conjunto de exemplos de treinamento e duas categorias possíveis, um algoritmo de treinamento SVM gera um modelo que prevê se um novo exemplo cai em uma categoria ou em outra.

Intuitivamente, um modelo SVM é uma representação dos exemplos de treinamento como pontos no espaço $\Re^2$ dividos em categorias. Neste modelo, cada ponto é mapeado de tal forma que cada elemento pertencente a uma categoria específica seja separado por uma lacuna entre as categorias existentes. No caso da apresentação de novos exemplos, estes são mapeados para o mesmo espaço (mesma categoria) e a previsão do SVM revela se estes pertencem a uma ou a outra categoria.

No IDS proposto neste trabalho, as SVMs desempenham a função de verificadores dos dados de tráfego de rede conforme classificado pelo classificador de ataques do sistema. A seguir apresentamos a arquitetura do IDSs desenvolvido.

6.4: Arquitetura do Polvo-IIDS

Com base nas características da rede SOM e das SVMs, foi desenvolvido um IDS que aplica um modelo multicamada denominado de Polvo-IIDS, um sistema de detecção de intrusão inteligente que realiza a coleta de dados na rede (NIDS) e que aplica técnicas de inteligência artificial e de aprendizado estatístico em sua construção.

O Polvo-IIDS faz uso de um SOM (*self-organizing Map*) e de alguns SVMs (*Support Vector Machine*) para realizar a classificação do tráfego de rede e determinar se uma instância observada no tráfego de rede é ou não um ataque.

Uma visão simplificada da arquitetura e da forma de atuação do Polvo-IIDS sobre os pacotes de tráfego da rede pode ser vista na figura 14.

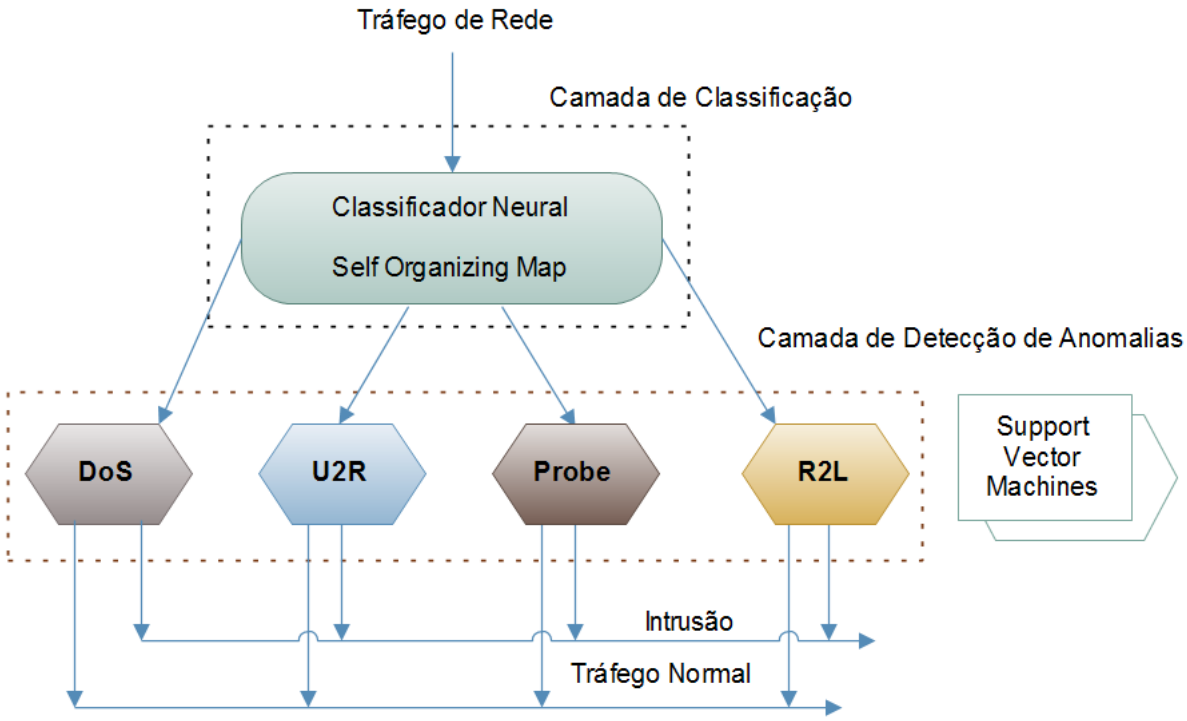


Figure 14: Arquitetura do Polvo-IIDS

O objetivo primário do Polvo-IIDS é prover um sistema de detecção inteligente modular, preciso (com baixa taxa de falsos positivos e falsos negativos), capaz de detectar variações de ataques conhecidos, sendo adaptativo às mudanças de características

do tráfego de rede e atuando em tempo real. A seguir, passamos a detalhar o funcionamento do Polvo-IIDS.

6.4.1: Funcionamento do Sistema

Em sua operação, o Polvo-IIDS analisa os padrões de comportamento da rede, gera mapeamentos de transmissão de pacotes entre *hosts* e executa a detecção de intrusões com base nestes mapeamentos.

O sistema é composto de duas camadas. Na primeira camada, existe um classificador SOM responsável pela coleta do tráfego de rede, análise de pacotes e classificação em quatro categorias: **DoS**, **U2R**, **Probe** or **R2L**, conforme definidas pelo DARPA. Onde a categoria de DoS (*Denial of service*), refere-se aos ataques de negação de serviço. A categoria de U2R (*User to root*), inclui tentativas de acesso a recursos que somente usuários administradores possuem. Na categoria de Probe (*Probing*) estão os ataques que representam fraudes. E em R2L (*Remote to local*) temos os ataques que incluem acessos remotos a recursos locais.

Na segunda camada do sistema (para detecção de anomalias), os dados classificados são apresentados a detectores específicos de acordo com sua classificação inicial realizada pela rede SOM da primeira camada. Estes detectores são responsáveis por indicar a qual classe específica os padrões observados pertencem, indicando, por exemplo, se um determinado tráfego pertence ou não a alguma das classes de ataques.

6.4.2: Classificador

O classificador SOM da primeira camada realiza uma pré-seleção dos padrões de tráfego de entrada, por meio da análise das características encontradas nos pacotes coletados em determinado período de tempo.

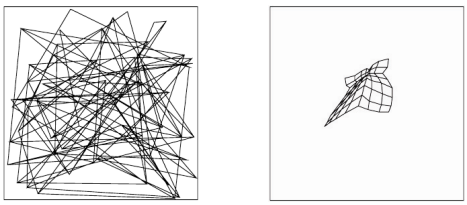
Este classificador é similar ao de alguns IDSs baseados em anomalias presentes em trabalhos relacionados pela literatura, em [88]) por exemplo, que também utiliza um classificador neural para realizar a detecção de intrusos.

Assim, para cada entrada o objetivo do sistema é identificar em qual das classes

de ataque o tráfego deve ser classificado. Porém, usando apenas o classificador da primeira camada o sistema apresenta altas taxas de falsos positivos (alarmes falsos). Para reduzir a taxa de falsos positivos, ajustamos o limiar de detecção do algoritmo de classificação visando a redução da taxa de falsos negativos (ataques não detectados) e em nossos testes verificamos as saídas obtidas pelo sistema.

No modelo de IDS desenvolvido o tráfego considerado normal também foi utilizado na fase de treinamento do classificador e pode ser agrupado em uma destas classes de ataque. A saída do classificador SOM é enviada para um dos detectores SVMs, que é especializado em apenas uma das quatro classes de ataque. Deste modo é delegada a cada verificador a tarefa de analisar os dados de entrada e identificar com maior precisão o que é um ataque e o que pode ser considerado tráfego normal.

A escolha do classificador SOM se justifica pela sua forma de treinamento não-supervisionado, facilidade de separação de padrões já conhecidos (treinados) e pela generalização de padrões ainda não conhecidos, ou novos nas entradas do sistema. As redes SOM podem ser utilizadas na detecção de novidades e fornecem um método eficaz na detecção de variações de ataques.



(a) Rede de Kohonen sem treinamento

(b) Rede de Kohonen, iniciando o treinamento

Figure 15: Ajuste dos pesos da Rede de Kohonen ao longo do tempo.

A figura 15 apresenta a organização dos neurônios de uma rede neural ao longo do tempo. Na figura 15(a) pode-se observar os pesos de uma rede SOM que ainda não foi treinada. A figura 15(b) mostra a rede após a fase inicial de treinamento em um período após algumas iterações de treinamento [42]. Pode-se observar as mudanças das distâncias entre os neurônios e a formação de picos. Em um gráfico 3D, após a fase de treinamento, cada padrão é identificado como um pico no plano resultante (con-

forme a figura 16). Este plano caracteriza um *self-organizing map* (SOM), contendo uma representação da localização espacial das coordenadas dos neurônios e representa as características contidas nos padrões dos dados de entrada apresentados à rede durante o processo de treinamento.

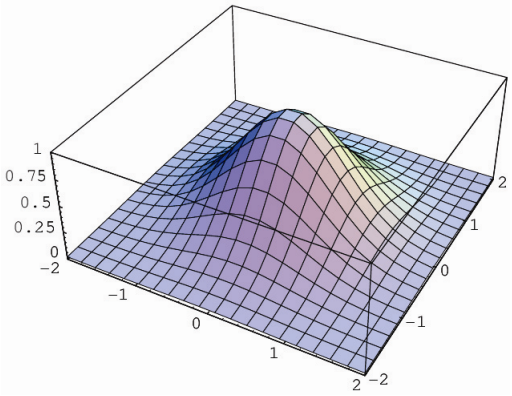


Figure 16: Rede de Kohonen após a etapa de treinamento

Para treinar o classificador SOM, foram utilizados padrões pertencentes as categorias (DoS, U2R, Probe e R2L) com o objetivo de verificar as taxas de detecção. Após a fase inicial de treinamento, o classificador SOM é capaz de separar tráfego de rede suspeito de ser um ataque em uma destas categorias. No caso da implementação do protótipo, o classificador neural foi desenvolvido com uma rede SOM contendo 41 neurônios de entrada e 4 neurônios de saída.

6.4.3: Cálculo da Vizinhança

Para funcionar de maneira adequada, o classificador SOM deve calcular a distância entre os neurônios utilizando alguma métrica, que pode ser a distância de *Manhatan* ou a Euclidiana. No caso do classificador SOM desenvolvido, o cálculo da distância entre os neurônios foi realizado com o método da distância Euclidiana, em que $\mathrm{d} = \sqrt{\sum_{\mathrm{i}} (v_{\mathrm{i}} - w_{\mathrm{i}})^2}$, onde v_{i} é o vetor de entradas e w_{i} é o vetor de pesos aplicados e obtidos durante o processo de treinamento. Utilizamos a distância Euclidiana por ser ideal para dados de entrada aleatórios, como é o caso dos padrões de tráfego de rede analisados.

Durante a fase de treinamento, foi utilizado o método de aprendizado por reforço (*reinforcement*), onde os padrões de entrada da rede foram aplicados diversas vezes auxiliando no processo de ajuste de pesos sinápticos da rede. Nos testes realizados, os padrões de entrada foram apresentados 5000 vezes na primeira rodada de testes e 15000 na segunda.

6.4.4: Limiar de Ativação

Para cada padrão de entrada, o classificador neural ativa apenas um dos neurônios de saída. O valor de cada neurônio de saída pode variar de 0 a 1. Quando o valor de saída de determinado neurônio é maior ou igual a 0.8, então o neurônio é considerado ativo. Se o valor de saída do neurônio for menor ou igual a 0.2, então o neurônio é considerado inativo. Quando nenhum neurônio é definido como ativo (com um valor maior ou igual a 0.8), uma função força um dos neurônios de saída cujo valor esteja mais próximo de 0.8 a ser o vencedor. Se houver empate entre os neurônios, um deles será eleito o vencedor por uma função aleatória.

Em geral, os limiares inferiores e superiores utilizados para se considerar um neurônio ativo ou inativo são de 0.1 para inativo e 0.9 para ativo. No entanto, no caso da classificação dos padrões de tráfego de rede analisados, com esta faixa de valores a rede SOM tende a ser mais restritiva, identificando um conjunto menor de ataques.

A faixa de valores foi estabelecida empiricamente entre 0.2 e 0.8 no classificador para tentar reduzir os erros de classificação. Para exemplificar, quando um padrão observado é direcionado pelo classificador ao detector de ataques **DoS**, este detector verificará se o padrão é de fato um ataque DoS ou se é tráfego normal.

6.4.5: Taxa de Aprendizado e Número de Épocas

Nos primeiros testes realizados, os valores de taxa de aprendizado utilizados ficaram entre 0.3 e 0.7 com 15000 e 30000 épocas de treinamento, mas os resultados ficaram abaixo do esperado com taxas de classificação entre 35 a 40%, classificando um grande número de ataques em classes erradas.

Novos testes foram realizados e através de experimentação arbitrária chegamos aos valores de 0.5 para 5000 épocas e a 0.6 para 15000 épocas.

6.4.6: Detector de Anomalias

O detector de anomalias do modelo compõe-se de quatro SVMs, que recebem os dados de tráfego em um formato padrão utilizado pelo classificador SOM.

As SVMs são uma poderosa técnica para resolver problemas relacionados a aprendizagem, classificação e predição [53]. As SVMs do modelo proposto são treinadas em paralelo ao treinamento do classificador SOM e trabalham cooperativamente para detectar ataques conforme as seguintes características:

- **DoS**: Detector SVM responsável por identificar ataques de DoS (*denial of service*), caracterizados pelo recebimento de múltiplas requisições a uma determinada porta por parte de um mesmo *host* em um curto período de tempo.
- **U2R**: Detector SVM usado para identificar ataques caracterizados pelas tentativas de um usuário local (sem privilégios de administrador), realizar logon como usuário *root*, ou executar ações que somente um administrador pode acessar.
- **Probe**: Detector SVM destinado a identificação de ataques caracterizados por tentativas de abertura de conexões e varreduras em diversas portas de um mesmo *host* de destino, na tentativa de descoberta de serviços e versões de sistema que estão em uso no *host* de destino.
- **R2L**: SVM que recebe fluxos de tráfego que podem ser identificados como sendo ataques remotos a serviços específicos.

Para as quatro categorias de ataque que o sistema analisa, existe um detector SVM especializado para cada categoria correspondente e estes apresentam duas opções de saída: identificar a entrada recebida como tráfego normal ou ataque.

As SVM foram adotadas como detectores na arquitetura do Polvo-IIDS, devido aos resultados obtidos em [15] e [34] que analisam os detectores SVM como sendo

mais eficientes do que as redes neurais quando utilizadas em tarefas de identificação de anomalias.

Na seleção dos parâmetros de configuração, as SVMs são menos complexas que alguns modelos de redes neurais, *Multi-Layer Perceptron*, por exemplo, onde é necessário definir o número de camadas ocultas, número de neurônios em cada camada e as funções de transferência mais adequadas.

A escolha errada de alguns destes parâmetros pode causar desde a degradação de desempenho durante o processo de treinamento até dificuldades de estabilização e convergência da rede.

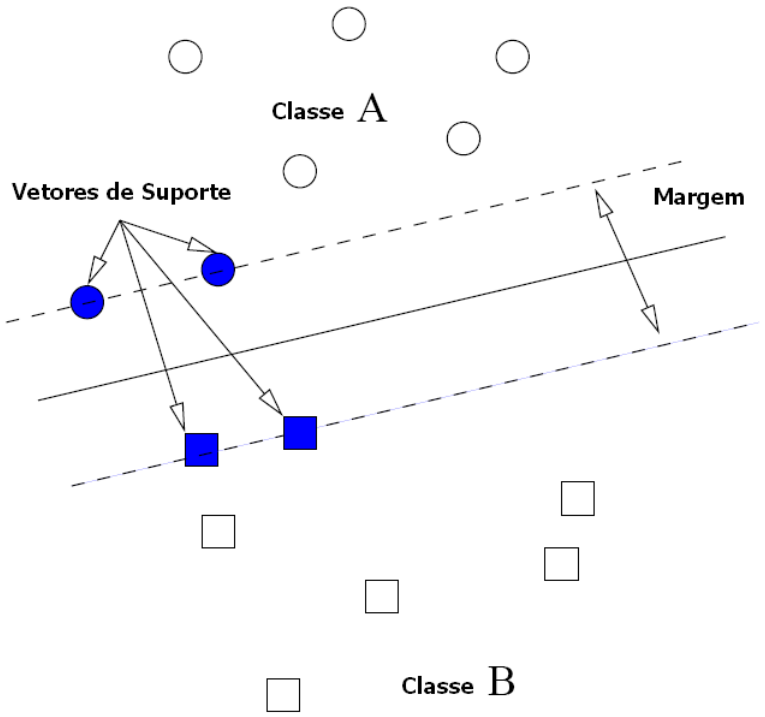


Figure 17: Separação de duas classes por uma SVM

Ao utilizar as SVMs, cada categoria de ataque passa a ser representada por um hiperplano, definida por um número de vetores suporte (*support vectors*), onde os dados de treinamento são separados em duas classes: uma para tráfego normal e outra para ataques (classe A representando ataques e a classe B da figura 17, representando tráfego normal). Os vetores de suporte de um sub-conjunto dos dados de treinamento

são usados para definir limites entre as duas classes (vetores preenchidos na figura). Os limites podem ser expressos matematicamente como sendo: $w^{T}x + b = 0$, onde w é o vetor de pesos, b é o *bias* e x é um vetor de entrada (padrão de entrada).

Após a análise dos dados por cada SVM, esta informação é utilizada para alimentar o sistema (ajustando os pesos dos neurônios do classificador neural e do respectivo SVM), permitindo a atualização do sistema.

Para os dados de treinamento das SVMs, foram utilizadas baixas taxas de aprendizado, que ficaram em valores de 0.01 e 0.02, de modo a permitir que o sistema atualize-se ao longo do tempo.

6.5: Protótipo

O protótipo desenvolvido é baseado na idéia do classificador SOM e nos detectores de anomalias utilizando as SVMs, ambos implementados com o uso da linguagem Java. A rede SOM e os detectores SVMs foram programados utilizando uma extensão ou *plugin* chamado de *WEKA Classification Algorithms* do *framework Weka* (*Waikato Environment for Knowledge Analysis*) [40].

No protótipo, os dados de entrada consistem em 41 variáveis apresentadas ao classificador SOM e encaminhadas ao SVM indicado pelo classificador conforme a análise de cada padrão de entrada. Estas variáveis representam características relevantes das conexões entre os *hosts* monitorados, tais como endereço de origem e destino, tempo de duração da conexão, tipo de protocolo, total de bytes transmitidos, entre outras.

Nesta versão do Polvo-IIDS, o sistema é executado a partir de entradas de linhas de comando em um *shell* de execução padrão. Um exemplo de linha de comando para execução do treinamento da rede SOM é o seguinte: $-W41 - H4 - M1 - R123 - I3 - N1 - L1 - K12 - P0.5 - F5000 - Sfalse - Vfalse - tfile - Train.arff$

Cada parâmetro informado é relevante para a construção do modelo gerado durante a fase de treinamento. Por exemplo, o parâmetro:

[40] *Software* que implementa diversos algoritmos de *Machine Learning* desenvolvido em Java. Ambos disponíveis nos endereços: `http://weka.wiki.sourceforge.net` e `http://wekaclassalgos.sourceforge.net`

- W - representa a largura (*width*) da rede;
- H - representa a altura (*height*);
- M - topologia da rede;
- R - se realiza sorteio de instâncias (random seed) para treinar a rede;
- I - modo de inicialização;
- N - função de atualização de neurônios vizinhos (*neighbor function*);
- L - função de treinamento (*learning function*);
- K - tamanho da vizinhança (*neighbor size*);
- P - taxa de aprendizado (*learn rate*);
- F - número de épocas ou iterações;
- S - se utiliza aprendizado supervisionado;
- V - se utiliza (*voting*).

A mesma idéia de configuração é aplicada aos detectores SVM para indicação de seus parâmetros de treinamento na prática é algo parecido com:

$-S0-K2-D3-G0.0-R0.0-N0.5-M40.0-C1.0-E0.0010-P0.1file-Train.arff$ Sendo que alguns destes parâmetros significam o seguinte:

- S - Seleciona o tipo de SVM onde:
  - 0 - C-SVC;
  - 1 - nu-SVC;
  - 2 - one-class SVM;
  - 3 - epsilon-SVR;
  - 4 - nu-SVR;
- K - Indica o tipo da função Kernel, onde:

- $0 - linear : u * v$
- $1 - polynomial : (gamma * u * v + coef0)^{d}egree$
- $2 - radialbasisfunction : exp(-gamma * |u - v|^2)$
- $3 - sigmoid : tanh(gamma * u * v + coef0)$
- $4 - precomputedkernel(kernelvaluesintraining\_set\_file)$

- D - Define o Grau da função Kernel selecionada;

A escolha dos parâmetros de configuração da rede SOM e dos detectores SVM foi realizada de modo empírico, através de testes sucessivos para verificação de estabilidade e convergência da rede SOM.

Após executar as fases de treinamento da rede SOM e dos detectores SVMs o sistema apresenta quatro telas construídas com a ferramenta Weka, uma para cada classe de ataque (DoS, U2R, R2L e Probe), semelhantes às apresentadas nas figuras 18 e 19 respectivamente.

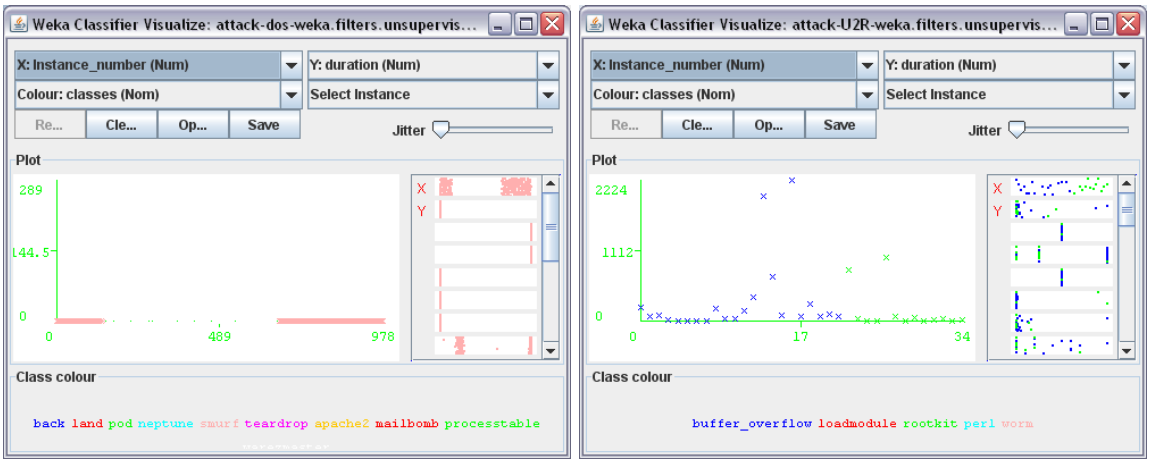


Figure 18: Exemplo de resultados dos detectores SVM das categorias Dos e U2R.

Cada tela desta representa o resultado da execução dos detectores SVM de cada categoria de ataques, após terem recebido cada entrada de dados separada pelo classificador SOM. A distribuição dos pontos nos gráficos depende dos dados recebidos como entrada a cada iteração de treinamento.

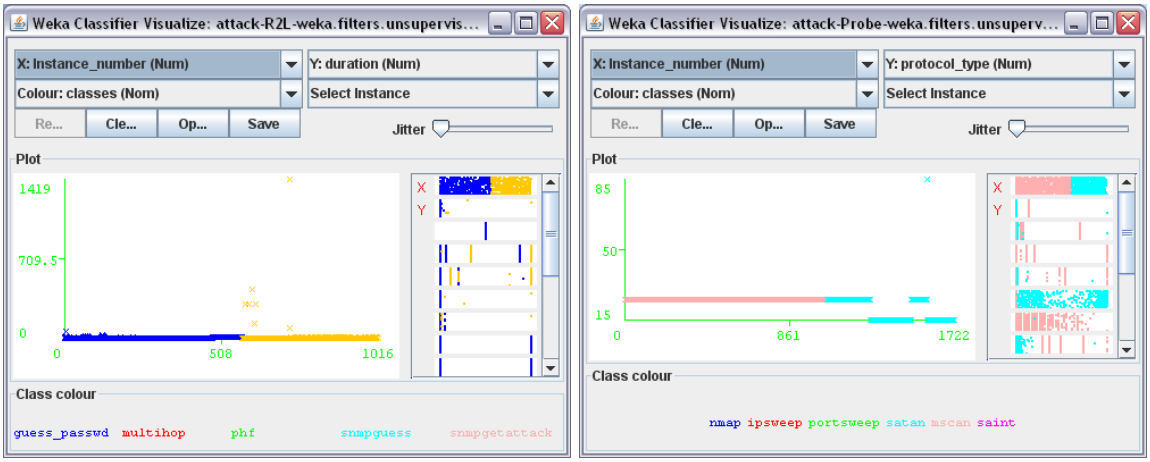


Figure 19: Exemplo de resultados dos detectores SVM das categorias R2L e Probe.

No protótipo do Polvo-IIDS, foi desenvolvido um conversor de formato para adaptar os dados de entrada que estavam no formato padrão pcap[41] para valores que podem ser mapeados nas variáveis de entrada do classificador SOM. O download da versão utilizada nos testes apresentados neste trabalho pode ser encontrado no endereço `http://www.das.ufsc.br/~vmoll/polvo0.1.tar.gz`.

6.5.1: Desafios e Limitações do Protótipo

A versão atual do protótipo do sistema possui algumas limitações e pontos de melhorias cuja aplicação ao modelo exigiriam um pouco mais de tempo e esforço do que o previsto no início do desenvolvimento deste trabalho. Uma melhoria importante seria a inclusão de um módulo de captura de tráfego de rede possibilitando ao Polvo-IIDS realizar a coleta dos dados e a análise em tempo real. Outra melhoria interessante seria a construção de um console para monitoramento dos alertas gerados pelo sistema e a adequação dos alertas gerados no formato IDMEF.

Além disso, novos testes podem ser realizados para verificar o desempenho e a utilização de outros algoritmos de classificação e de seleção de atributos.

Um dos grandes desafios para os IDSs que utilizam redes neurais é a dificuldade de treinamento com dados de tráfego real. Isso ocorre por que as amostras de tráfego

[41] Formato padrão para captura de pacotes de rede. `http://www.tcpdump.org/pcap/pcap.html`

real podem apresentar algum tipo de tráfego malicioso (ruído embutido), prejudicando o treinamento da rede neural. Classificar grandes quantidades de tráfego real identificando todos os ataques existentes, pacotes mal formados e fragmentados não é uma tarefa trivial.

6.5.2: Análise de características nos dados do KDD

Visando a melhoria do modelo proposto foram realizadas algumas modificações no funcionamento dos detectores SVM durante o desenvolvimento do protótipo.

Uma vez que cada classe de ataque tem seu próprio conjunto distinto de características relevantes, modificamos cada detector SVM para analisar apenas as características mais importantes e realizamos um mapeamento das 41 características disponíveis [72] de acordo com sua relevância.

Em [39], apresentam-se os graus de importância das 41 características em cada tipo específico de ataque de cada categoria. Nos primeiros testes realizados neste trabalho, foram utilizadas todas as 41 características. Verificamos que os ataques evoluem ao longo do tempo, porém, as classes de ataques continuam as mesmas.

Na implementação do protótipo, utilizamos algoritmos de seleção de atributos (J48, Id3, etc) fornecidos pela ferramenta Weka. Com a seleção de atributos por classe de ataques foi possível simplificar o modelo e utilizar apenas características relevantes para o treinamento dos detectores SVM, reduzindo a quantidade de memória e o tempo de processamento necessário para treinar o sistema. Na tabela 6.5.2 apresentamos um resumo das características relevantes obtidas com a execução dos testes realizados.

Para ilustrar utilizamos o algoritmo J48 com dados de ataque DoS, onde dos 41 atributos o algoritmo selecionou 10 sendo estes os seguites: 1,3,4,5,6,23,31,32,33,40. Na prática isso significa a utilização apenas dos seguintes atributos para o treinamento do detector SVM de ataques DoS: duration, service, flag, src_bytes, dst_bytes, count, srv_diff_host_rate, dst_host_count, dst_host_srv_count e dst_host_rerror_rate.

Na seção 6.6, apresentamos o desempenho do modelo proposto para detecção de intrusão, fazendo testes comparativos do modelo que usa todas as características com o modelo simplificado tendo apenas as características escolhidas pelo algoritmo de seleção de atributos. A seguir, na seção de testes e resultados temos um comparativo dos resultados obtidos com outras abordagens presentes na literatura.

6.6: Testes e Resultados

Os testes do protótipo foram executados em um computador com 4GB de memória RAM e dois processadores *Quad Core Intel Xeon* de 1.6GHz. Os dados de treinamento foram obtidos no *web site* do KDD Cup 1999 Data traffic, disponível na Internet [72], e foram usados para testar o modelo do sistema Polvo-IIDS.

De acordo com os experimentos realizados, a configuração mínima para a execução dos testes deve ser um computador com um processador de 1.6GHz, 1GB de memória e 4GB ou mais de espaço em disco disponível.

Os resultados obtidos na execução dos testes mostram que a detecção de intrusão, utilizando-se das duas camadas seqüenciais, uma para a classificação e outra para a tomada de decisão é factível e passível de utilização na construção de sistemas de detecção de intrusão inteligente.

Esta abordagem é capaz de produzir um aumento na taxa de detecção e a redução nas taxas de falsos positivos quando comparados com outros sistemas IDSs baseados em anomalias presentes na literatura atual. Também pode-se usar o Polvo-IIDS na análise de tráfego real.

6.6.1: Treinamento do Classificador neural

O classificador neural foi treinado com ataques das quatro categorias: (DoS, User to Root (U2R), Probe, e Remote to Local (R2L)). Depois, quatro detectores SVM foram treinados com tráfego específico de cada classe de ataque com os dados tendo sido separados pelo classificador SOM.

Os vetores de pesos da rede treinada foram armazenados em arquivos para uso futuro, evitando a necessidade de treinamentos posteriores. Um parâmetro importante a ser escolhido foi a taxa de treinamento do classificador. Se o valor for muito baixo, serão necessárias muitas épocas e instâncias para treinar a rede neural com boa capacidade de generalização. Se o valor for muito alto, a rede pode não convergir. Foram executados testes com diversos valores da taxa de aprendizado da rede SOM e diferentes épocas (conforme a Tabela 7 e as figuras 20 e 21) mas os valores que se mostraram mais adequados foram a taxa de aprendizado entre 0.5 e 0.6 e 15000 épocas.

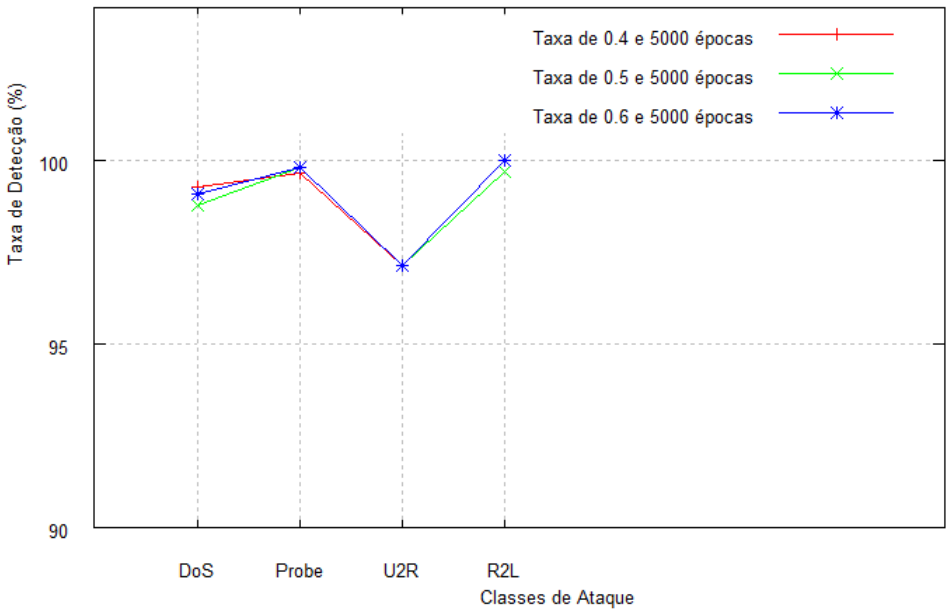


Figure 20: Taxas de 0.4, 0.5 e 0.6 para 5000 épocas de treinamento.

Após o período de treinamento do sistema, foram realizados testes para medir a indicação correta do classificador para os detectores de anomalias SVMs.

Os testes foram executados com 19.808 registros de entrada escolhidos aleatoriamente dos dados do KDD Cup 1999 (13.208 instâncias no conjunto de treinamento e 6.600 instâncias para testes). A taxa de aprendizado usada foi de 0.5 e 0.6 com 5.000 e 15.000 épocas de treinamento.

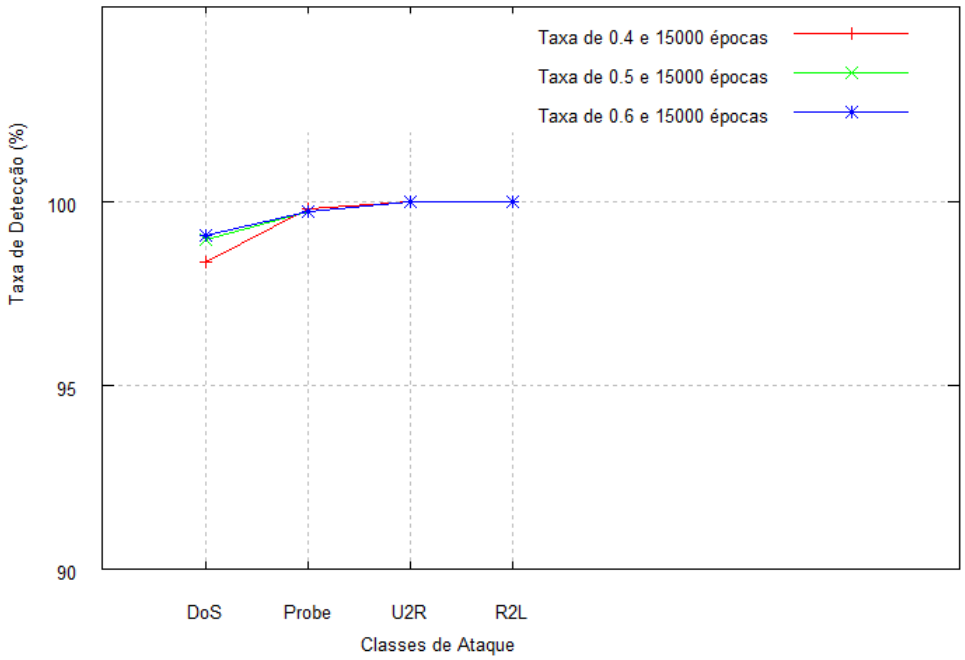


Figure 21: Taxas de 0.4, 0.5 e 0.6 para 15000 épocas de treinamento.

Com base nos resultados obtidos, pode-se observar que o treinamento do classificador SOM com uma taxa de aprendizado de 0.6 é melhor do que o de 0.5.

O número de épocas de treinamento (número de vezes que os registros de dados são apresentados a rede neural) também mostrou-se importante para a consolidação de uma classe de ataques.

6.6.2: Treinamento dos Detectores SVM

Para o treinamento dos detectores de anomalias SVM, foram executados alguns testes com os seguintes algoritmos: SMO Polykernel, SMO Normalized PolyKernel, SMO Pukkernel, LibSVM and SMO RBFKernel [42]. O algoritmo SMO (*Sequential Minimal Optimization*) apresentado no **Apêndice ??** e em [60, 40], é um tipo de SVM eficiente que busca resolver o problema de otimização de modo incremental. O tamanho mínimo do problema de otimização envolve dois multiplicadores de Lagrange para encontrar o máximo / mínimo de uma função.

[42] A documentação da implementação destes algoritmos está disponível em `http://nlp.stanford.edu/nlp/javadoc/weka-3-2/weka.classifiers.SMO.html`

A vantagem do algoritmo SMO reside no fato de que a solução para os dois multiplicadores de Lagrange pode ser feita analiticamente. A diferença entre os algoritmos testados é a função kernel utilizada para a classificação. Por exemplo, a SMO Kernel RBF é uma função de base radial. Os testes com estes algoritmos foram executados com as 41 características e com a seleção de características para cada classe de ataque.

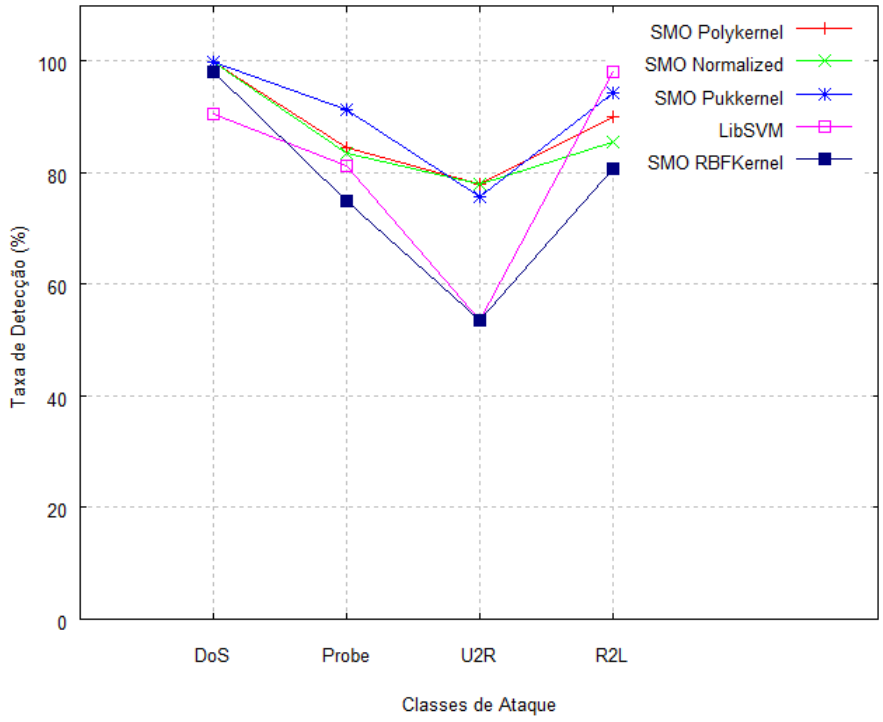


Figure 22: Taxas de Detecção x algoritmo sem seleção de atributos x classe de ataque.

A Tabela 8 e a Figura 22 apresentam uma comparação das taxas de detecção para cada algoritmo usado no treinamento das SVMs utilisadas no protótipo com todas as 41 características dos dados analisados.

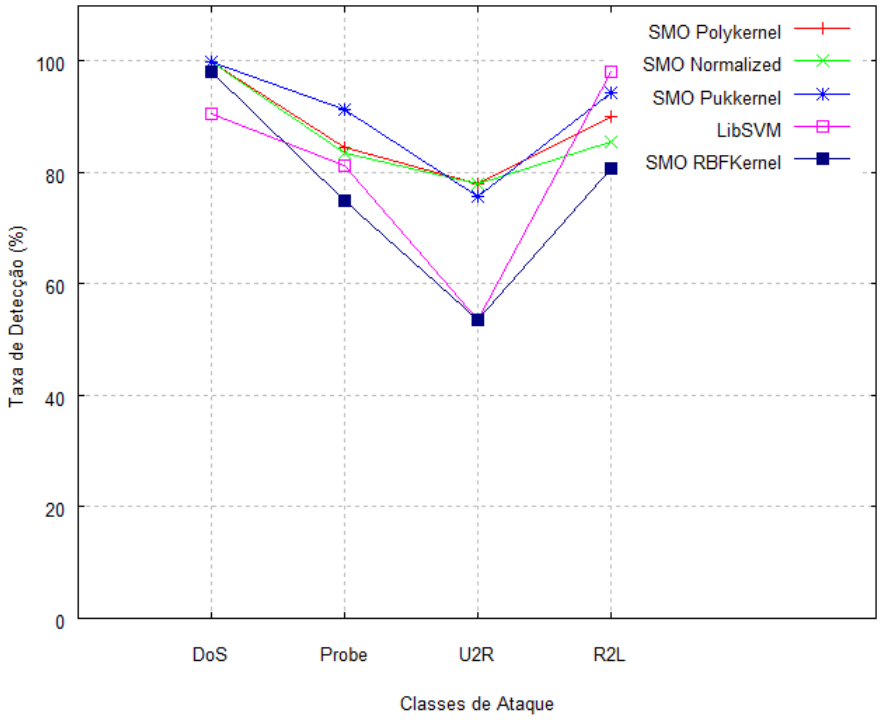


Figure 23: Taxas de Detecção x algoritmo com seleção de atributos x classe de ataque.

A fim de comparar o percentual de classificação correta dos algoritmos selecionados para cada classe de ataque, realizamos testes analisando apenas as características relevantes. A Tabela 9 e a Figura 23 resumem os resultados obtidos. Existe uma melhoria da taxa de detecção utilizando apenas as características relevantes. Isso pode ser explicado porque, com um número menor de entradas para analisar, a SVM resultante é menos complexa, sem a interferência de outras características que não auxiliariam na detecção.

Com base neste resultado, os detectores SVM do sistema foram modificados para utilizar uma heurística e escolher a melhor função kernel de acordo com a classe de ataque que deve detectar. Com essa modificação o sistema passou a empregar as seguintes funções kernel: SMOPolyKernel para DOS, U2R e R2L, SMOPukKernel para Probe.

Os mesmos dados de treinamento foram testados com outros algoritmos de classificação com o objetivo de compararmos seu desempenho. Os resultados podem ser visualizados na Tabela 10.

6.6.3: Comparação com outros sistemas

Para verificar os resultados de nossos experimentos com o Polvo-IIDS, realizamos uma comparação com outros IDSs presentes na literatura.

Na tabela 6.6.3 e na figura 24, apresentamos os resultados obtidos pelo sistema Polvo-IIDS e por outros IDSs que utilizam alguma técnica de IA. Todos os sistemas citados nesta tabela utilizam os dados do KDD CUP 99. Podemos ver que o Polvo-IIDS obteve um bom desempenho em todas as categorias (desvio máximo baixo). Além disso, os sistemas citados não apresentam algumas informações como o número de registros de dados utilizados durante o período de treinamento e detalhes sobre os ajustes nos valores de ativação dos neurônios em suas redes.

Em outro teste, capturamos o tráfego de rede real a partir da Internet e aplicamos à entrada do Polvo-IIDS, a fim de verificar a sua eficiência com o tráfego de rede real.

Foram coletadas 126.772 entradas de tráfego normal em um servidor de Internet

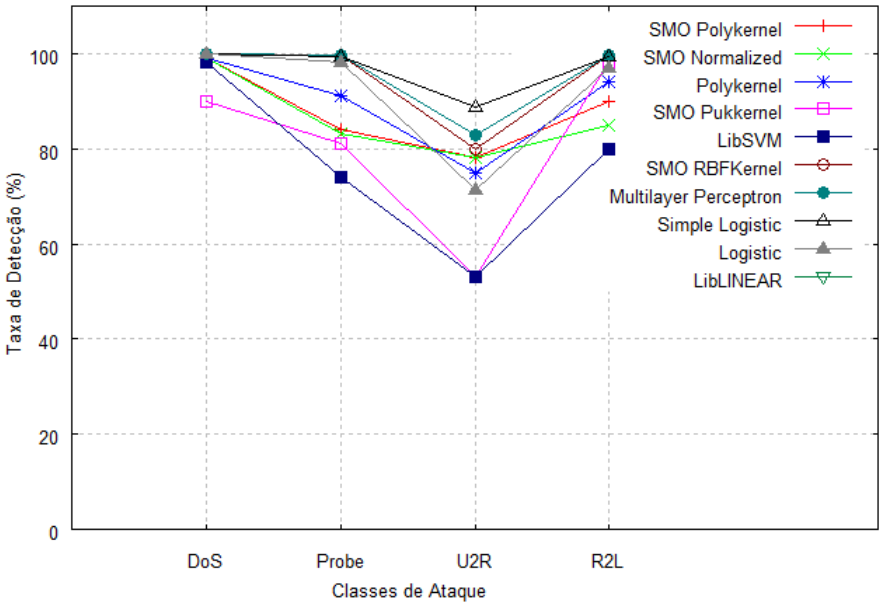


Figure 24: Taxas de Detecção x algoritmo x classe de ataque.

e 31.758 entradas de ataques de um *honeypot* configurado para aceitar qualquer tipo de ataque a partir da Internet. O tráfego normal foi distribuído nas seguintes portas: 25, 993 e 995 para e-mail; 80 e 443 para páginas web, e 53 para o sistema de nome de domínio – DNS.

Durante o período de treinamento, foram escolhidas aleatoriamente 84.937 entradas de tráfego normal e 21.278 entradas de ataques. Na fase de testes, foram aplicadas 41.835 entradas de tráfego normal e 10.480 entradas de ataques (estas entradas são diferentes do tráfego de treinamento). Nesse teste foi obtida uma taxa de detecção de 83.90%, com 9.72% de desvio máximo.

6.7: Considerações finais

Este capítulo descreveu o desenvolvimento do modelo do Polvo-IIDS, um sistema de detecção de intrusão inteligente baseado em anomalias, que utiliza uma rede neural artificial SOM e e quatro detectores SVMs.

A utilização dessas técnicas foi importante e decisiva para identificar atividades maliciosas através da análise de tráfego de rede, reduzindo a taxa de falsos positivos e melhorando as taxas de detecção em comparação com IDSs existentes na literatura.

Os sistemas baseados em anomalia são caracterizados pelo rápido tempo de resposta (tempo linear para o tamanho dos dados de entrada). Tais sistemas não funcionam adequadamente na análise de tráfego criptografado e o Polvo-IIDS, para a classe R2L de ataque, não é uma exceção. Os resultados iniciais com o Polvo-IIDS demonstram a viabilidade do modelo e uma melhoria considerável na taxa de detecção, quando comparado com outros sistemas existentes.

Os testes realizados com tráfego real mostraram a viabilidade do modelo apresentado, embora existam ainda algumas dificuldades em compararmos os resultados obtidos com outras abordagens.

No modelo desenvolvido, o processo de treinamento pode ser realizado em uma base contínua, com taxa de aprendizagem de 0.01, mesmo na presença de algum tráfego classificado de modo equivocado. Desse modo é possível manter o Polvo-IIDS atualizado, mesmo com a evolução das várias aplicações que fazem uso da Internet.

Classe do Ataque	Características relevantes
DoS (10)	duration, service, flag, src_bytes, dst_bytes, count, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_rerror_rate
Probe (12)	protocol_type, service, flag, src_bytes, count, srv_count, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate
U2R (32)	duration, protocol_type, service, flag, src_bytes, dst_bytes, hot, logged_in, num_compromised, root_shell, num_root, num_file_creations, is_guest_login, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, diff_srv_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_diff_srv_rate, dst_host_same_src_port_rate, dst_host_srv_diff_host_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_rerror_rate, dst_host_srv_rerror_rate
R2L (15)	duration, protocol_type, service, flag, src_bytes, dst_bytes, logged_in, is_guest_login, count, srv_rerror_rate, srv_diff_host_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_srv_rerror_rate

Table 6: Características relevantes para cada classe de ataque.

Taxa de Aprendizado	Épocas	DoS	Probe	U2R	R2L
0,4	5000	99.285%	99.6518%	97.1429%	100%
0,5	5000	98.7743%	99.8259%	97.1429%	99.705%
0,6	5000	99.0807%	99.8259%	97.1429%	100%
0,4	15000	98.3657%	99.8259%	100%	100%
0,5	15000	98.9785%	99.7678%	100%	100%
0,6	15000	99.0807%	99.7678%	100%	100%

Table 7: Testes do classificador SOM com diferentes taxas de aprendizado e épocas por classe de ataque.

Função Kernel	DoS	Probe	U2R	R2L
SMO Polykernel	99,95%	84,61%	78,05%	90,02%
SMO Normalized Polykernel	99,85%	83,46%	78,05%	85,45%
SMO Pukkernel	99,75%	91,29%	75,61%	94,35%
LibSVM	90,52%	81,30%	53,66%	98,02%
SMO RBFKernel	98,19%	74,94%	53,66%	80,63%

Table 8: Desempenho das SVMs por algoritmo analisado sem a seleção de características relevantes por classe de ataque.

Função Kernel	DoS	Probe	U2R	R2L
SMO Polykernel	100%	99,54%	91,43%	99,71%
SMO Normalized Polykernel	100%	99,42%	88,57%	99,13%
SMO Pukkernel	100%	99,59%	82,86%	99,61%
LibSVM	99,59%	88,10%	62,86%	94,10%
SMO RBFKernel	100%	98,72%	62,86%	97,05%

Table 9: Desempenho das SVMs por algoritmo analisado com seleção de características relevantes por classe de ataque.

Função Kernel	DoS	Probe	U2R	R2L
SMO Polykernel	99,95%	84,61%	78,05%	90,02%
SMO Normalized Polykernel	99,85%	83,46%	78,05%	85,45%
SMO Pukkernel	99,75%	91,29%	75,61%	94,35 %
LibSVM	90,52%	81,30%	53,66%	98,02%
SMO RBFKernel	98,19%	74,94%	53,66%	80,63%
Multi-layer Perceptron	99.89%	99.65%	80.00%	99.70%
Simple Logistic	100%	99.59%	82.85%	99.70%
Logistic	100%	99.36%	88.57%	99.31%
LibLINEAR	100%	98.08%	71.42%	96.95%

Table 10: Comparativo com outros algoritmos de Classificação.

IIDS	Média Detecção	Desvio Máx
Anomalous Payload-based IDS [9]	58,80%	41,20%
HPCANN [48]	77,49%	22,53%
MADAM ID [47]	77,97%	17,97%
Multi-level Hybrid Classifier [88]	89,19%	22,52%
Polvo-IIDS	97,40%	8,57%

Table 11: Comparação de resultados entre IIDSs

7. Conclusões e Perspectivas

"A formulação de um problema é de longe mais essencial que sua solução, a qual pode ser meramente uma questão de habilidades matemáticas ou experimentais.

Para levantar novas perguntas, novas possibilidades, ver velhos problemas sobre um novo ponto de vista requer imaginação criativa e marca os reais avanços na ciência."

Albert Einstein

7.1: Conclusões e Trabalhos Futuros

Manter a disponibilidade de serviços e aplicações seguras via Internet exige a utilização de abordagens elaboradas de identificação e autenticação devido à explosão de usuários na rede mundial observada nos últimos anos. O aumento significativo de fraudes e de outras atividades maliciosas neste ambiente aberto, vem motivando o uso de ações e controles preventivos nos serviços e aplicações. Contudo, nem sempre estes controles se mostram efetivos. A elaboração de métodos de detecção e de análises para minimizar as perdas e fortalecer as defesas contra atividades maliciosas nos sistemas tem sido o tema de diversas pesquisas.

Nos últimos anos, o desenvolvimento e o uso de IDSs continua a se intensificar à medida em que o projeto destes sistemas vem recebendo novas abordagens para identificação e geração de alertas de violações à segurança nos sistemas computacionais.

Um IDS pode aplicar uma abordagem de análise baseada em assinaturas e operar de forma similiar à um *software* anti-vírus. Porém, um IDS baseado em assinaturas não é capaz de detectar novos ataques ou suas variações em vista da necessidade de atualização de seu banco de assinaturas ou ainda pela inexistência de uma assinatura que aborde ao novo ataque.

Para superar essas e outras limitações, uma abordagem diferente vem sendo

adotada pelos pesquisadores, através do desenvolvimento dos IDSs baseados em análises de anomalias.

Um IDS baseado em anomalias cria um modelo de comportamento normal durante a fase de treinamento, e em seguida, compara as novas observações com o modelo gerado usando alguma métrica de similaridade. O IDS gera alertas quando detecta que uma observação no ambiente monitorado apresenta um desvio significativo em relação ao modelo de comportamento gerado.

Neste sentido, esta dissertação descreveu o desenvolvimento de um modelo de sistema de detecção de intrusão baseado em anomalias, através do uso de redes neurais artificiais e de máquinas de vetores suporte. O uso destas técnicas visa a identificação de atividades maliciosas através da análise do tráfego de rede, alcançando alta taxa de detecção e baixa taxa de falsos positivos.

De maneira geral, os IDSs baseados em anomalias geralmente possuem um índice menor de falsos negativos porém um índice maior de falsos positivos do que IDSs baseados em assinaturas. Os sistemas baseados em anomalias são caracterizados pela rapidez nas consultas, em tempo linear ao tamanho do *payload* dos pacotes. Tais sistemas não funcionam corretamente na análise de tráfego cifrado. Outra limitação presente na maioria dos IDSs baseados em anomalias está nos dados utilizados na fase de treinamento do sistema. Essa limitação é decorrente da dificuldade de identificar todo o tráfego malicioso presente em amostras de tráfego real. Muitos trabalhos presentes na literatura usam os dados do KDD 1999 [72]. Porém estes dados não refletem o tráfego real existente atualmente na Internet.

Os resultados iniciais com o Polvo-IIDS demonstram a viabilidade do modelo e uma considerável melhora na taxa de detecção e redução na taxa de falsos positivos. Durante o desenvolvimento do protótipo, enfrentamos etapas de esforço extra no pré-processamento dos dados do DARPA, coleta de dados de tráfego real e com os ajustes dos parâmetros para a rede SOM e para os SVMs.

No modelo desenvolvido, o treinamento da rede e das SVMs pode acontecer de maneira constante, através do uso de dados de tráfego real previamente capturados

e pré-processados por alguma ferramenta de captura de pacotes (Ethereal[43], ou Wireshark[44], por exemplo).

Durante o desenvolvimento deste trabalho, tivemos um artigo aceito e publicado [49] nos anais do (SBSeg 2008)[45], onde a proposta desta arquitetura também foi avaliada.

Existem diversas formas de continuidade deste trabalho. Trabalhos futuros poderiam ser a expansão do modelo de análise de tráfego, para a análise de arquivos de log nos sistemas alvos aos quais o Polvo-IIDS estiver em operação.

Uma atividade imediata seria o aprimoramento do protótipo desenvolvido, com a implementação de uma interface de usuário (*Graphic User Interface*) mais intuitiva e amigável que auxiliaria na configuração e utilização do Polvo-IIDS. Outra melhoria significativa seria a inclusão de um módulo de captura de pacotes no próprio IDS desenvolvido, facilitando o acesso e a coleta de dados para treinamento do sistema. Além disso, outra perspectiva interessante seria a utilização de novos dados de treinamento para verificação de desempenho do modelo proposto.

Aprimoramentos do modelo de IDS proposto neste trabalho, execução de novos testes e a proposta de um novo modelo para utilização em ambientes diversificados, tais como as redes móveis, serão objetos de trabalhos futuros.

[43] http://www.ethereal.com/

[44] http://www.wireshark.org/

[45] VIII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais - http://sbseg2008.inf.ufrgs.br/

References

[1] ALLEN, J., CHRISTIE, A., FITHEN, W., MCHUGH, J., AND PICKEL, J. State of the practice of intrusion detection technologies. In *CMU/SEI-99-TR-028* (Carnegie Mellon Software Engineering Institute, 2000).

[2] ANDERSON, J. P. Computer Security Technology Planning Study. In *Technical Report ESC-TR-73-51* (1972).

[3] ANDERSON, J. P. Computer Security Threat Monitoring and Surveillance. In *Technical Report* (1980).

[4] AXELSSON, S. Intrusion Detection Systems: A Survey and Taxonomy. Tech. Rep. 99-15, Chalmers Univ., Mar. 2000.

[5] BALASUBRAMANIYAN, J. S., FERNANDES, J. G., ISACOFF, D., SPAFFORD, E., AND ZAMBONI, D. An Architeture for Intrusion Detection using Autonomous Agents. In *14th IEEE Computer Security Applications Conference* (1998), pp. 13–24.

[6] BELL, E. D., AND LAPADULA, J. L. Secure Computer Systems: Unified Exposition and Multics Interpretation. Tech. rep., MITRE Corporation, Bedford, MA, 1976.

[7] BISHOP, M. *Computer Security: Art and Science*. Pearson Education, Boston, MA, 2003.

[8] BOLZONI, D. *Revisiting anomaly-based network intrusion detection systems*. PhD thesis, Enschede, June 2009.

[9] BOLZONI, D., ETALLE, S., AND HARTEL, P. Poseidon: a 2-tier anomaly-based network intrusion detection system. In *Fourth IEEE International Workshop on Information Assurance* (April 2006), pp. 220–237.

[10] BOLZONI, D., ETALLE, S., HARTEL, P., AND ZAMBON, E. Poseidon: a 2-tier anomaly-based network intrusion detection system. *Innovative Architecture for Future Generation High-Performance Processors and Systems, International Workshop on 0* (2006), 144–156.

[11] BRAND, S. *Trusted Computer System Evaluation Criteria DoD 5200.28-STD*. US Department of Defense, 1985.

[12] CAMPBELL, C., AND BENNETT, K. *A Linear Programming Approach to Novelty Detection.*, 2001.

[13] CERT.BR. Incidentes reportados ao cert.br – julho a setembro de 2009, 2009.

[14] CHANG, C.-C., AND LIN, C.-J. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[15] CHEN, W.-H., HSU, S.-H., AND SHEN, H.-P. Application of svm and ann for intrusion detection. *Comput. Oper. Res. 32*, 10 (2005), 2617–2634.

[16] CLARK, A., AND LIMITED, S. Cryptographic controls the eternal triangle. In *Proceedings of the COMPSEC World Conference on Computer Security* (September 1996).

[17] COHEN, W. W. Fast effective rule induction. In *In Proceedings of the Twelfth International Conference on Machine Learning* (1995), Morgan Kaufmann, pp. 115–123.

[18] CORTES, C., V.-V. *Support Vector Networks*. Machine Learning, 273-297, 1995.

[19] CRISTIANINI, N., AND SHAWE-TAYLOR, J. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

[20] DEBAR, H., CURRY, D., AND FEINSTEIN, B. The Intrusion Detection Message Exchange Format. technical report draft-ietf-idwg-idmef-xml-16, 2006.

[21] DEBAR, H., DACIER, M., AND WESPI, A. Towards a taxonomy of intrusion-detection systems. In *Computer Networks 31* (1999), pp. 805–822.

[22] DEBAR, H., DACIER, M., AND WESPI, A. A revised taxonomy for intrusion detection systems. In *Annales des Telecommunications volume 55* (2000), pp. 361–378.

[23] DELIANG, W. Pattern recognition: neural networks in perspective. in ieee expert„ Aug. 1993.

[24] DENNING, D. An Intrusion Detection Model. In *Proceedings of the 1986 IEEE Symposium on Security and Privacy* (1986).

[25] DEPREN, M., TOPALLAR, M., ANARIM, E., AND CILIZ, K. Network-based anomaly intrusion detection system using soms. In *Signal Processing and Communications Applications Conference, 2004. Proceedings of the IEEE 12th* (April 2004), vol. 1, pp. 76–79.

[26] DUDA, R. O., HART, P. E., AND STORK, D. G. *Pattern Classification*. Wiley-Interscience, second edition, 2000.

[27] DURKIN, J. *Expert Systems Design and Development*, 1 ed., vol. 1. Prentice Hall, 1994.

[28] FERRAIOLO, D., AND KUHN, R. Role-based Access Controls. In *NIST-NCSC National Computer Security Conference* (October 1992), pp. 554–563.

[29] FERRAIOLO, D. F., GILBERTO, D. M., AND LYNCH, N. An Examination of Federal and Commercial Access Control Policy Needs. In *NIST-NCSC National Computer Security Conference* (September 1993), pp. 107–116.

[30] FLETCHER, R. *Practical Methods of Optimization*. John Wiley, and Sons, 1987.

[31] FRAGA, J., AND POWELL, D. A fault– and intrusion–tolerant file system. In *In Proceedings of the 3rd International Conference on Computer Security* (1985), pp. 203–218.

[32] GARFINKEL, S., SPAFFORD, G., AND SCHWARTZ, A. *Practical Unix and Internet Security*. O'Reilly, 2003.

[33] GUTTMAN, B. *An Introduction to Computer Security: The NIST Handbook.* U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology, 1995.

[34] HAIJUN, X., FANG, P., LING, W., AND HONGWEI, L. Ad hoc-based feature selection and support vector machine classifier for intrusion detection. In *Proceedings of*

2007 IEEE Conference on Grey Systems and Intelligent Services (Nanjing, China, November 2007).

[35] HOCHBERG, J., JACKSON, K., STALLINGS, C., MCCLARY, J., DUBOIS, D., AND FORD, J. NADIR, an automated system for detecting network intrusion and misuse. In *8th National Information Systems Security Conference* (1993), Computer Security 12(3):235-248.

[36] JACOBSON, V., LERES, C., AND MCCANNE, S. Tcpdump, 2005.

[37] JANAKIRAMAN, R., WALDVOGEL, M., AND ZHANG, Q. Indra: A Peer-to-Peer Approach to Network Intrusion Detection and Prevention. In *Proceedings of IEEE WETICE 2003* (June 2003).

[38] JANSEN, W., MELL, P., KARYGIANNIS, T., AND MARKS, D. Applying Mobile Agents to Intrusion Detection and Response. Tech. Rep. 6416, National Institute of Standards and Technology, 1999.

[39] KAYACIK, H. G., ZINCIR-HEYWOOD, A. N., AND HEYWOOD, M. I. Selecting features for intrusion detection: A feature relevance analysis on kdd 99 intrusion detection datasets. In *Proceedings of the Third Annual Conference on Privacy, Security and Trust (PST-2005)* (October 2005).

[40] KEERTHI, S., SHEVADE, S., BHATTACHARYYA, C., AND MURTHY, K. Improvements to platt's smo algorithm for svm classifier design. Tech. rep., Dept of Mechanical and Production Engineering, National University of Singapore, 1999.

[41] KOHONEN, T. Self-organized formation of topologically correct feature maps. *Journal of the American Society for Information Science and Technology* (1988), 509–521.

[42] KRÖSE, B., AND VAN DER SMAGT, P. *An introduction to neural networks*. URL ftp://ftp.informatik.uni-freiburg.de/papers/neuro/ann_intro_smag.ps.gz, The University of Amsterdam, 1996.

[43] LAMPSON, B. Protection. In *Proceedings of 5th Princeton Symp. on Information Science and Systems* (1974), ACM, pp. 437–443.

[44] LANDWEHR, C. E. Formal Models for Computer Security. In *ACM Comput. Surv.* (september 1981), pp. 247–278.

[45] LANDWEHR, C. E. Hardware requirements for secure computer systems. In *Proceedings of IEEE Symposium on Security and Privacy* (april 1984), pp. 34–40.

[46] LANDWEHR, C. E. Computer Security. *International Journal of Information Security* (July 2001).

[47] LEE, W., AND STOLFO, S. A framework for constructing features and models for intrusion detection systems. 227–261.

[48] LIU, G., YI, Z., AND YANG, S. A hierarchical intrusion detection model based on the pca neural networks. *Journal of the American Society for Information Science and Technology* (December 2006), 1561–1568.

[49] MAFRA, P. M., FRAGA, J. S., MOLL, V., AND SANTIN, A. O. Polvo-iids: Um sistema de detecção de intrusão inteligente baseado em anomalias. In *Proceedings of the 2008 Simpósio Brasileiro em Segurança da Informação e de Sistemas (SBSeg)* (2008), pp. 201–214.

[50] MAHONEY, M. V., AND CHAN, P. K. Learning nonstationary models of normal network traffic for detecting novel attacks. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (2002), pp. 376–385.

[51] MCHUGH, J. Intrusion and Intrusion Detection. Tech. rep., Carnegie Mellon University, Pittsburg, PA, July 2001.

[52] MITCHELL, T. *Machine Learning*. McGraw Hill, 1997.

[53] MUKKAMALA, S., JANOSKI, G., AND SUNG, A. Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks, IJCNN '02* (2002), vol. 2, pp. 1702–1707.

[54] MYERS, P. The neglected aspect of computer security. Master's thesis, Naval Postgraduate School, Monterey, 1980.

[55] NASCIMENTO, J., CAIRO, L., AND YONEYAMA, T. *Inteligência Artificial em controle e automação.* FAPESP e Editora Edgard Blücher Ltda., São Paulo, 2000.

[56] NGUYEN, B. V. Self organizing map (som) for anomaly detection. Tech. rep., "Ohio University", 2002.

[57] NIE, J., AND HAYKIN, S. A dynamic channel assignment policy through q-learning. *IEEE Transactions on Neural Networks 10*, 6 (1999), 1443–1455.

[58] NOBLE, W. S. Support vector machine applications in computational biology. In *In B. Schölkopf, K. Tsuda, and J.-P. Vert, editors, Kernel Methods in Computational biology* (2004), MIT Press, pp. 71–92.

[59] OSUNA, E., FREUND, R., AND GIROSI, F. Training support vector machines: an application to face detection. *CVPR97* (1997).

[60] PLATT, J., SCHLKOPF, B., BURGES, C., AND SMOLA, A. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1998.

[61] PORRAS, P. A., AND NEUMANN, P. G. EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *1997 National Information Systems Security Conference* (oct 1997).

[62] PTACEK, T. H., AND NEWSHAM, T. N. Insertion, evasion, and denial of service: Eluding network intrusion detection. Secure networks, inc., University, 1998.

[63] RICH, E. *Inteligência artificial.* Makrom Books, São Paulo, 1993.

[64] ROESCH, M. Snort - Lightweight Intrusion Detection for Networks. In *Proceedings of USENIX LISA, Berkeley* (1999), pp. 229–238.

[65] SANDHU, R. S., AND SAMARATI, P. Authentication, Access Control and Intrusion Detection. In *ACM Computing Survey, Vol. 28* (1996), pp. 241–273.

[66] SCHÖLKOPF, B. SHAWE-TAYLOR, J. S. A., AND WILLIAMSON, R. C. *Estimating the support of a high-dimensional distribution.*, 1999. Microsoft Research Corporation Technical Report MSR-TR-99-87.

[67] SCHOLKOPF, B. SUNG, K. B. C. G. F. N. P. P. T., AND V., V. *Comparing Support Vector Machines with Gaussian Kernels to Radial Basis Function Classifiers.*, 1997. IEEE Transactions on Signal Processing, 45, p. 2758-2765.

[68] SEBRING, M., SHELLHOUSE, E., HANNA, M., AND WHITEHURST, R. Expert Systems in Intrusion Detection: A case study. In *Proceedings of the Eleventh National Computer Security Conference, Washington* (1988), pp. 74–81.

[69] SMOLA, A., AND SCHÖLKOPF, B. A tutorial on support vector regression. In *Technical Report NC2-TR-1998-030* (1998), NeuroCOLT2.

[70] STANIFORD-CHEN, S. Common Intrusion Detection Framework (CIDF), 1998.

[71] STOLFO, J., AND WANG, K. In raid 04 proc 7th symposium on recent advances in intrusion detection, volume 3224 of lncs, springer.

[72] STOLFO, J. S., WEI, F., LEE, W., PRODROMIDIS, A., AND CHAN, P. K. Kdd cup data - knowledge discovery and data mining competition (1999), 1999.

[73] STONEBURNER, G., GOGUEN, A., AND FERRINGA, A. *Risk Management Guide*. NIST Special Publication 800-30, 2001.

[74] TAX, D. YPMA, A., AND DUIN, R. Support vector data description applied to machine vibration analysis. In *In: M. Boasson, J. Kaandorp, J. Tonino, M. Vosselman (eds.), Proceedings of 5th Annual Conference of the Advanced School for Computing and Imaging* (1999), Heijen, NL, June 15-17, pp. 398–405.

[75] TAX, D., AND DUIN, R. Data domain description by support vectors. In *In Proceedings of ESANN99* (1999), Ed. M Verleysen, D. Facto Press, Brussels, pp. 251–256.

[76] TECHNOLOGIES., C. P. S. Stateful inspection technology, 2005.

[77] TEO, L., ZHENG, Y., AND AHN, G. Intrusion Detection Force: An infrastructure for internet-scale intrusion detection. In *Proceedings of the first IEEE International Workshop on Information Assurance* (2003).

[78] THORSTEN, J. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning* (1998), Springer.

[79] VAPNIK, V. *The Nature of Statistical Learning Theory*. Springer, New York, 1995.

[80] VAPNIK, V. An overview of statistical learning theory. 988–999.

[81] VAPNIK, V., AND CHERVONENKIS, A. *Theory of Pattern Recognition*. Nauka, Moscow, 1974.

[82] VIGNA, G., ECKMANN, S., AND KEMMERER, R. The STAT tool suite. In *Proceedings of DARPA Information Survivability Conference and Exposition, IEEE Press, Los Alamitos* (2000), pp. 46–55.

[83] WANG, H., HUANG, J. Z., QU, Y., AND XIE, J. Web services: problems and future directions. *J. Web Sem. 1*, 3 (2004), 309–320.

[84] WEBB, A. *Statistical Pattern Recognition*. John Wiley and Sons, second edition, 2002.

[85] WEBER, R. Raciocínio baseado em casos. [online], 1996. Disponível na Internet via WWW.

[86] WITTEN, I. H., AND FRANK, E. *Data Mining: Practical machine learning tools and techniques*, 2005. 2nd Edition, Morgan Kaufmann, San Francisco, Software available at `http://www.cs.waikato.ac.nz/~ml/index.html`.

[87] WOOD, M. Intrusion Detection Message Exchange Requirements, 2002.

[88] XIANG, C., AND LIM, S. M. Design of multiple-level hybrid classifier for intrusion detection system. In *Proceedings of 2005 IEEE Workshop on Machine Learning for Signal Processing* (September 2005), pp. 117–122.

[89] YEGNESWARAN, V., BARFORD, P., AND JHA, S. Global intrusion detection in the domino overlay system. In *In Proceedings of Network and Distributed System Security Symposium (NDSS* (2004).

[90] YEGNESWARAN, V., BARFORD, P., AND JHA, S. Global Intrusion Detection in the DOMINO Overlay System. In *Network and Distributed System Security Simposium* (2004).

[91] ZARASKA, K. Prelude IDS: Current state and development perspectives, 2003.