

# Uma abordagem para a correlação de eventos de segurança baseada em técnicas de aprendizado de máquina

Este exemplar corresponde à redação final da  
Dissertação devidamente corrigida e defendida  
por Kleber Stroeh e aprovada pela Banca Exa-  
minadora.

Campinas, 22 de Setembro de 2009.



Edmundo Roberto Mauro Madeira  
Instituto de Computação – Unicamp  
(Orientador)

Dissertação apresentada ao Instituto de Com-  
putação, UNICAMP, como requisito parcial  
para a obtenção do título de Mestre em Ciência  
da Computação.

**FICHA CATALOGRÁFICA ELABORADA PELA  
BIBLIOTECA DO IMECC DA UNICAMP  
Bibliotecária: Miriam Cristina Alves CRB8a / 5094**

Stroeh, Kleber

St87u Uma abordagem para a correlação de eventos de segurança baseada em técnicas de aprendizado de máquina / Kleber Stroeh – Campinas, [SP. :s.n.], 2009.

Orientador : Edmundo Roberto Mauro Madeira  
Dissertação (mestrado) - Universidade Estadual de Campinas, Instituto de Computação.

1. Redes de computadores - Medidas de segurança. 2. Tecnologia da informação - Medidas de segurança. 3. Inteligência artificial. 4. Aprendizado do computador. I. Madeira, Edmundo Roberto Mauro. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título.

Título em inglês: An approach to the correlation of security events based upon machine learning techniques.

Palavras-chave em inglês (Keywords): 1. Computer networks – Security measures. 2. Information technology – Security measures. 3. Artificial intelligence. 4. Machine learning.

Área de concentração: Redes de computadores.

Titulação: Mestre em Ciência da Computação

Banca examinadora: Prof. Dr. Edmundo Roberto Mauro Madeira (IC-UNICAMP)  
Prof. Dr. Edson dos Santos Moreira (ICMC – USP)  
Prof. Dr. Jacques Wainer (IC-UNICAMP)

Data da defesa: 03/08/2009

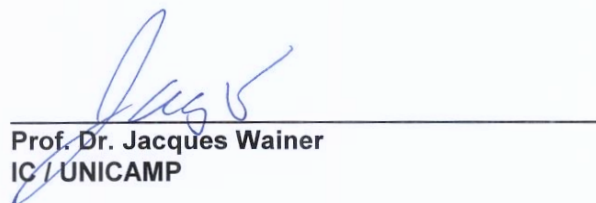
Programa de pós-graduação: Mestrado em Ciência da Computação

## TERMO DE APROVAÇÃO

Dissertação Defendida e Aprovada em 03 de agosto de 2009, pela Banca examinadora composta pelos Professores Doutores:



**Prof. Dr. Edson dos Santos Moreira**  
ICMC / USP



**Prof. Dr. Jacques Wainer**  
IC / UNICAMP



**Prof. Dr. Edmundo Roberto Mauro Madeira**  
IC / UNICAMP

# Uma abordagem para a correlação de eventos de segurança baseada em técnicas de aprendizado de máquina

**Kleber Stroeh**

Setembro de 2009

## **Banca Examinadora:**

- Edmundo Roberto Mauro Madeira  
Instituto de Computação – Unicamp (Orientador)
- Edson dos Santos Moreira  
Instituto de Ciências Matemáticas e de Computação – USP
- Jacques Wainer  
Instituto de Computação – Unicamp
- Maurício Ferreira Magalhães  
FEEC – Unicamp
- Ricardo Dahab  
Instituto de Computação – Unicamp

# Resumo

Organizações enfrentam o desafio crescente de garantir a segurança da informação junto às suas infraestruturas tecnológicas. Abordagens estáticas à segurança, como a defesa de perímetros, têm se mostrado pouco eficazes num novo cenário marcado pelo aumento da complexidade dos sistemas — e conseqüentemente de suas vulnerabilidades — e pela evolução e automatização de ataques. Por outro lado, a detecção dinâmica de ataques por meio de *IDSs* (*Intrusion Detection Systems*) apresenta um número demasiadamente elevado de falsos positivos. Este trabalho propõe uma abordagem para coleta e normalização, e fusão e classificação de alertas de segurança. Tal abordagem envolve a coleta de alertas de diferentes fontes, e sua normalização segundo modelo de representação padronizado — *IDMEF* (*Intrusion Detection Message Exchange Format*). Os alertas normalizados são agrupados em meta-alertas (fusão ou agrupamento), os quais são classificados — através de técnicas de aprendizado de máquina — entre ataques e alarmes falsos. Uma implementação desta abordagem foi testada junto aos dados do desafio DARPA e Scan of the Month, contando com três implementações distintas de classificadores (SVM – *Support Vector Machine* –, Rede *Bayesiana* e Árvore de Decisão), bem como uma coletânea (*ensemble*) de SVM com Rede *Bayesiana*, atingindo resultados bastante relevantes.

# Abstract

Organizations face the ever growing challenge of providing security within their IT infrastructures. Static approaches to security, such as perimetral defense, have proven less than effective in a new scenario characterized by increasingly complex systems — and, therefore, more vulnerable — and by the evolution and automation of cyber attacks. Moreover, dynamic detection of attacks through *IDSs* (*Intrusion Detection Systems*) presents too many false positives to be effective. This work presents an approach to collect and normalize, as well as to fuse and classify security alerts. This approach involves collecting alerts from different sources and normalizing them according to standardized structures — IDMEF (*Intrusion Detection Message Exchange Format*). The normalized alerts are grouped into meta-alerts (fusion or clustering), which are later classified — through machine learning techniques — into attacks or false alarms. An implementation of this approach is tested against *DARPA Challenge* and *Scan of the Month*, using three different classification techniques, as well as an ensemble of SVM and Bayesian Network, having achieved very relevant results.

# Agradecimentos

À minha amada esposa Luciana e minhas lindas filhas, Letícia e Gabriela, pelo suporte, amor, paciência e compreensão. Vocês iluminam meus dias; sempre.

Aos meus pais, Sandra e Werner, pela excelente educação que me proporcionaram.

Ao professor Edmundo Roberto Mauro Madeira, por acreditar neste projeto e por me motivar nos momentos mais difíceis.

Ao professor Siome Klein Goldenstein, pelo apoio técnico e empolgação com o projeto.

# Conteúdo

<b>Resumo</b>	<b>vii</b>
<b>Abstract</b>	<b>ix</b>
<b>Agradecimentos</b>	<b>xi</b>
<b>1 Introdução</b>	<b>1</b>
1.1 Analogias e Intuição: a Segunda Guerra Mundial . . . . .	4
1.2 Folhas, Árvores e Florestas . . . . .	8
1.3 Contribuições . . . . .	8
1.4 Organização do Documento . . . . .	9
<b>2 Fundamentação Teórica</b>	<b>11</b>
2.1 Segurança . . . . .	11
2.1.1 Tipos de Ataques . . . . .	12
2.1.2 Dispositivos de Segurança . . . . .	13
2.2 Aprendizado de Máquina . . . . .	19
2.2.1 Support Vector Machines . . . . .	20
2.2.2 Redes <i>Bayesianas</i> . . . . .	25
2.2.3 Árvores de Decisão . . . . .	29
2.2.4 Coletâneas ( <i>ensembles</i> ) . . . . .	31
2.2.5 Validação Cruzada ( <i>Cross-validation</i> ) . . . . .	31
2.2.6 Curva ROC . . . . .	32
<b>3 Trabalhos Correlatos</b>	<b>37</b>
3.1 Sistemas de Detecção de Mau-uso . . . . .	37
3.2 Sistemas de Detecção de Anomalias . . . . .	39
3.3 Sistemas de Detecção Híbridos . . . . .	40
3.4 Detecção de Ataques Multiestágio . . . . .	41
3.5 Agrupamento de Alertas de Segurança . . . . .	41



3.6	Novas abordagens . . . . .	42
3.6.1	Novas técnicas de aprendizado de máquina aplicadas . . . . .	44
<b>4</b>	<b>Abordagem Proposta</b>	<b>45</b>
4.1	Uma Abordagem em Camadas Hierárquicas . . . . .	45
4.2	Coleta e Normalização . . . . .	47
4.2.1	Coleta . . . . .	47
4.2.2	Normalização . . . . .	48
4.2.3	Taxonomia . . . . .	51
4.3	Agrupamento ou Fusão . . . . .	56
4.4	Classificação . . . . .	61
<b>5</b>	<b>Implementação e Experimentos</b>	<b>65</b>
5.1	DARPA . . . . .	65
5.2	SotM . . . . .	67
5.3	Instanciação do Modelo . . . . .	68
5.3.1	Instanciação de Coleta e Normalização . . . . .	70
5.3.2	Fusão de Alertas . . . . .	75
5.3.3	Métodos de Classificação . . . . .	76
5.4	Experimentos . . . . .	80
5.4.1	Experimento 1: DARPA 3x2 . . . . .	81
5.4.2	Experimento 2: DARPA kFold . . . . .	92
5.4.3	Experimento 3: SotM kFold . . . . .	102
5.4.4	Experimento 4: Alertas x Meta-Alertas . . . . .	112
5.4.5	Uma Análise Comparativa . . . . .	116
<b>6</b>	<b>Conclusão</b>	<b>119</b>
	<b>Referências Bibliográficas</b>	<b>123</b>
<b>A</b>	<b>Ataques no Desafio DARPA</b>	<b>129</b>
<b>B</b>	<b>Ataques no SotM</b>	<b>133</b>
<b>C</b>	<b>Tipos de Alertas no Desafio DARPA</b>	<b>135</b>
<b>D</b>	<b>Tipos de Alertas no SotM</b>	<b>139</b>

# Lista de Tabelas

2.1	Taxonomia no TSOM . . . . .	18
2.2	Matriz de confusão para classificador binário . . . . .	32
2.3	Matriz de observação experimental para classificador em <i>K-fold Cross-validation</i> . . . . .	34
4.1	Registro de Alerta . . . . .	50
4.2	Registro de Meta-Alerta . . . . .	58
4.3	Atributos para Classificação . . . . .	62
5.1	Fontes de Eventos DARPA . . . . .	81
5.2	Redução de Eventos . . . . .	82
5.3	Resultados para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	83
5.4	Resultados para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	85
5.5	Resultados para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	87
5.6	Resultados para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	89
5.7	Matriz de ataques para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	91
5.8	Resultados para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	92
5.9	Resultados para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	95

5.10	Resultados para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	97
5.11	Resultados para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	99
5.12	Matriz de ataques para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	101
5.13	Fontes de Eventos SotM . . . . .	102
5.14	Redução de Eventos em SotM . . . . .	103
5.15	Resultados para classificador SVM aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	104
5.16	Resultados para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	106
5.17	Resultados para classificador Árvore de Decisão aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	108
5.18	Resultados para classificador Ensemble aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	110
5.19	Resultados para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em alertas . . . . .	113
5.20	Resultados para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em meta-alertas . . . . .	114
A.1	Ataques no Desafio DARPA . . . . .	129
B.1	Ataques no SotM . . . . .	133
C.1	Tipos de Alertas no Desafio DARPA . . . . .	135
D.1	Tipos de Alertas no SotM . . . . .	139

# Lista de Figuras

1.1	Incidentes Reportados ao CERT.br de 1999 a 2008 . . . . .	2
1.2	Incidentes Reportados ao CERT.br de Janeiro a Dezembro de 2008 . . . .	3
1.3	Linha <i>Maginot</i> . . . . .	5
1.4	Sistema <i>Dowding</i> . . . . .	7
2.1	Separador Linear . . . . .	21
2.2	Exemplo de Rede <i>Bayesiana</i> . . . . .	26
2.3	Fragmento de Árvore de Decisão de Ataques para Classificação em Segurança	30
2.4	Exemplo de curva ROC . . . . .	33
4.1	Abordagem em camadas hierárquicas . . . . .	46
4.2	Hierarquia de Objetos . . . . .	53
4.3	Hierarquia de Ações . . . . .	54
4.4	Hierarquia de Condições . . . . .	55
4.5	Hierarquia de Suspeitas . . . . .	56
5.1	Topologia da Rede do desafio DARPA . . . . .	66
5.2	Instanciação do Modelo . . . . .	69
5.3	Uma Topologia de Rede <i>Bayesiana</i> Refletindo a Anatomia de um Ataque .	79
5.4	Curva ROC para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	83
5.5	Detecção por tipo de ataque para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	84
5.6	Curva ROC para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	85
5.7	Detecção por tipo de ataque para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	86

5.8	Curva ROC para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	87
5.9	Detecção por tipo de ataque para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	88
5.10	Curva ROC para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	90
5.11	Detecção por tipo de ataque para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes . . . . .	90
5.12	Curva ROC para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança .	93
5.13	Detecção por tipo de ataque para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	94
5.14	Curva ROC para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança . . . . .	95
5.15	Detecção por tipo de ataque para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	96
5.16	Curva ROC para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança . . . . .	97
5.17	Detecção por tipo de ataque para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	98
5.18	Curva ROC para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança . . . . .	99
5.19	Detecção por tipo de ataque para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se <i>K-fold Cross-validation</i> . . . . .	100
5.20	Curva ROC para classificador SVM aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança .	104
5.21	Detecção por tipo de ataque para classificador SVM aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	105
5.22	Curva ROC para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança . . . . .	106

5.23	Detecção por tipo de ataque para classificador Rede <i>Bayesiana</i> aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	107
5.24	Curva ROC para classificador Árvore de Decisão aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança . . . . .	108
5.25	Detecção por tipo de ataque para classificador Árvore de Decisão aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	109
5.26	Curva ROC para classificador Ensemble aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> e indicação de intervalo de confiança . . . . .	110
5.27	Detecção por tipo de ataque para classificador Ensemble aplicado a meta-alertas da base SotM utilizando-se <i>K-fold Cross-validation</i> . . . . .	111
5.28	Curva ROC para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em alertas . . . . .	113
5.29	Curva ROC para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em meta-alertas . . . . .	115
5.30	Detecção por tipo de ataque para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em meta-alertas . . . . .	115

# Capítulo 1

## Introdução

A sociedade da informação é cada vez mais dependente da infraestrutura tecnológica que a suporta. Dados sensíveis como contas bancárias, declarações de imposto de renda, prontuários médicos, números de cartões de crédito, estratégias políticas, e informações de defesa - entre tantos outros - são hoje armazenados de forma digital junto a sistemas computacionais.

O crescimento da importância destes ativos digitais e o avanço das tecnologias de redes e telecomunicações - com especial destaque para a Internet - têm provocado um aumento do número dos ataques a estas infraestruturas.

Os dados mais recentes publicados pelo CERT.br [7] mostram expressivo crescimento do número de incidentes reportados no ano de 2008 (222.528 contra apenas 3.107 em 1999), conforme ilustrado na Figura 1.1.

Relevante também é o crescimento deste número nos meses de novembro e dezembro de 2008, delineando um aumento do gradiente associado, conforme ilustrado na Figura 1.2.

Estudo da McAfee [38] apresentado no último Fórum Econômico Mundial, em Davos, afirma que “o roubo de dados e cibercrimes custaram às empresas, em todo o mundo, cerca de US\$ 1 trilhão.” Um único episódio em 2009, expôs dados de cartões de créditos, que especialistas estimam ter afetado 50 milhões de pessoas, o que está sendo intitulado de “a maior violação de dados da história” [18].

Os impactos não se restringem ao campo econômico. Relatos recentes dão conta de ações de piratas virtuais e ciberterroristas contra governos centrais de países e instituições militares, como os casos dos ataques à Presidência do Uruguai [19] ou aqueles disparados contra a Estônia e Geórgia, vindos da Rússia [17]. Desta feita, a segurança da informação ganha conotações de soberania nacional.

A matriz de gestão de risco indica um crescimento tanto na dimensão da probabilidade da ocorrência (como parecem indicar os números de aumentos de ataques) quanto na do impacto (perdas financeiras mais elevadas, questões de estado, entre outros). Tal fato

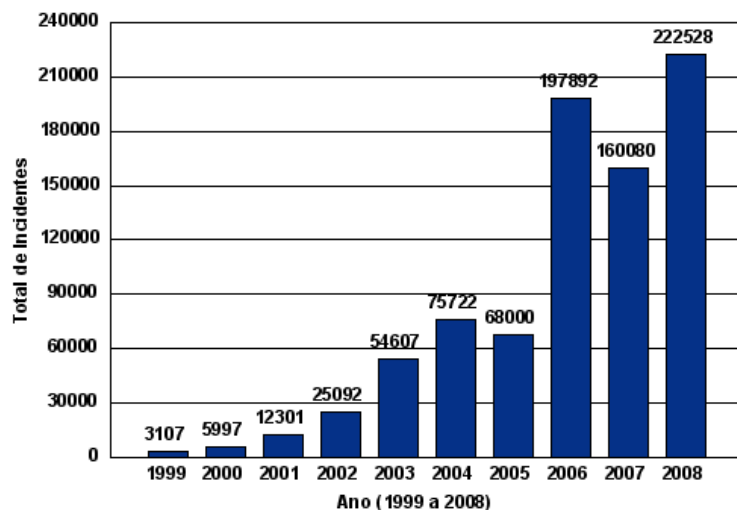


Figura 1.1: Incidentes Reportados ao CERT.br de 1999 a 2008

ocorre em meio à dificuldade prática de se prover real proteção a estes ativos digitais, num cenário de maior integração global.

Segundo relato da norma NBR ISO/IEC 17799 [25], alguns fatores contribuem para a dificuldade em se prover segurança nas redes e sistemas modernos: (1) as infraestruturas de TI têm se tornado mais complexas com o advento de novas tecnologias (como redes sem fio, protocolos P2P, dispositivos miniaturizados de armazenamento — *memory keys* —, câmeras digitais, entre outros); (2) ataques de grande complexidade tecnológica têm sido automatizados e disponibilizados de forma irrestrita na Internet (vide também Allen et al [1]); (3) e negócios exigem mudanças cada vez mais rápidas junto à infraestrutura de TI para suportar serviços mais complexos e mais interconectados.

A necessidade de mudanças rápidas de infraestrutura de TI e seu impacto na segurança também são observados por Hale e Brusil [23]. Eles afirmam que a segurança ganha conotações de viabilizador de negócios e fonte de vantagens competitivas, em complemento à visão tradicional de uma estratégia para gerência de riscos.

Dentro deste contexto de crescente dependência de uma infraestrutura em evolução, Sabata e Ornes [51] observam que a natureza crítica da infraestrutura torna muito cara uma resposta incorreta ou a ausência de uma resposta a um ataque.

A dificuldade em se promover segurança real aos ativos digitais advém do uso de técnicas e tecnologias que não mais são suficientes para tal. Faour et al [14] ressaltam que as tecnologias clássicas de proteção de redes do tipo filtragem de pacotes (*firewalls*) são ineficazes contra a maior parte dos ataques atuais. Abordagens do tipo defesa de



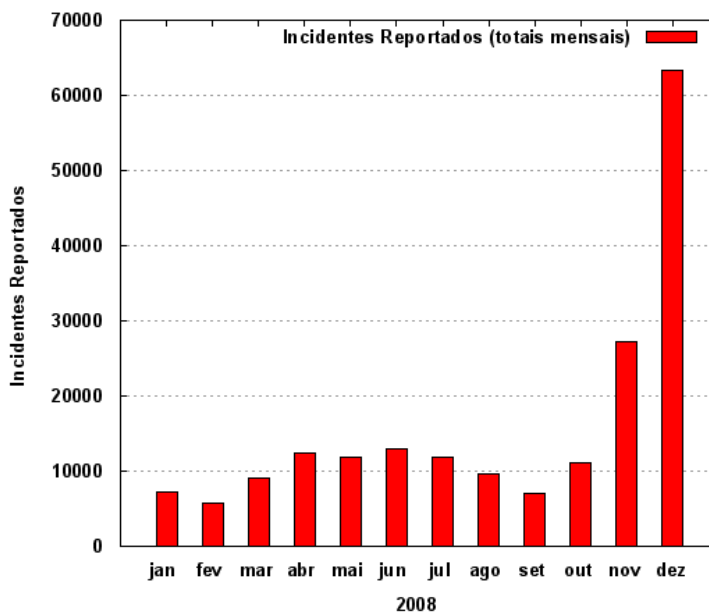


Figura 1.2: Incidentes Reportados ao CERT.br de Janeiro a Dezembro de 2008

perímetro, embora essenciais em qualquer arquitetura de segurança da informação, não podem promover de forma isolada a garantia da segurança da infraestrutura associada.

Assim surgiram novas tecnologias, em especial, os *IDSs* (*Intrusion Detection Systems*) cujo objetivo é detectar um ataque (intrusão) que um *firewall* não pode barrar. Numa analogia rápida, se um *firewall* pode ser comparado a um grande muro que cerca a propriedade, os IDSs seriam os sensores de alarmes espalhados em pontos chaves desta mesma propriedade (sensores de presença, infravermelhos, movimento, etc).

Infelizmente, os IDSs geram uma alta quantidade de eventos a serem tratados, tornando difícil a determinação dos ataques reais que geraram aqueles eventos (ou um subconjunto dos mesmos). Este problema é ainda agravado pelo volume de falsos positivos gerados, ou seja, o número de eventos identificados pelo IDS que não correspondem a qualquer tipo de ataque.

Gerenciar todos os alarmes emitidos por tais dispositivos demanda um nível de esforço que transcende a capacidade das equipes de segurança da maior parte das instituições, dificuldade esta também relatada por Ning et al [45] e Liu e Zang [31].

Ma et al [39] observam que este cenário é ainda agravado pela falta de cooperação entre as fontes de eventos de segurança, a saber, *firewall*, IDSs, roteadores, servidores, etc.

Tal cooperação pode ser traduzida pela correlação de eventos de segurança destas diferentes fontes. Conforme observado recentemente por Martin-Flatin et al [36], as técnicas de correlação cada vez mais são usadas para detecção de intrusões; também ratificam que a correlação de eventos não pode mais ser considerada um problema resolvido.

Desta forma propomos uma abordagem para gerenciamento dos alertas de segurança provenientes de sistemas de IDS e outros dispositivos de segurança, objetivando correlacionar tais alertas para separar ataques reais de alarmes falsos. Sua essência reside na percepção de que um sistema que possa agregar dados de diferentes fontes tem a potencialidade de tomar decisões embasadas em mais informações e formar uma visão mais completa da realidade da segurança do ambiente de TI, o que pode ser verificado nos testes executados sobre uma implementação desta abordagem.

Sistemas que integram e correlacionam alertas vindos de múltiplos dispositivos e IDSs são conhecidos como sistemas de IDS distribuídos (*DIDS — Distributed Intrusion Detectors Systems*) ou *SIEM — Security Information Event Management*.

A aplicação adequada de sistemas de *SIEM* aliada a uma sólida política de segurança — amplamente galgada na conscientização e educação dos profissionais envolvidos — parece ser a melhor resposta para os desafios apresentados à segurança da informação.

## 1.1 Analogias e Intuição: a Segunda Guerra Mundial

Analogias envolvendo estratégias militares são lugar comum em várias áreas do conhecimento como planejamento estratégico, modelos de concorrência e, também, segurança da informação. O uso de analogias permite que desenvolvamos intuições acerca dos campos de estudo que estamos explorando.

Em especial duas batalhas da Segunda Guerra Mundial, a Batalha da França e a Batalha da Inglaterra, permitem um desenvolvimento muito interessante de intuições sobre estratégias de segurança da informação.

### A Batalha da França

Após o final da Primeira Guerra Mundial, a França, traumatizada por uma guerra em seu território e preocupada com a possibilidade de um novo confronto com a Alemanha, investiu 3 bilhões de francos na construção de uma linha fortificada que separava seu território da Alemanha e Itália, a chamada Linha *Maginot*. Seu conceito era fruto da experiência angariada na Primeira Guerra, a qual fora marcada por uma guerra de trincheiras. Tratava-se, portanto, de uma linha fortificada estática, composta por um complexo de *bunkers* para milhares de homens, 108 fortes principais (a 15km de espaçamento uns dos outros), fortes menores e casamatas, e mais de 100km de túneis, os quais separavam as

fronteiras da França com Alemanha e Itália, conforme mostra a Figura 1.3.

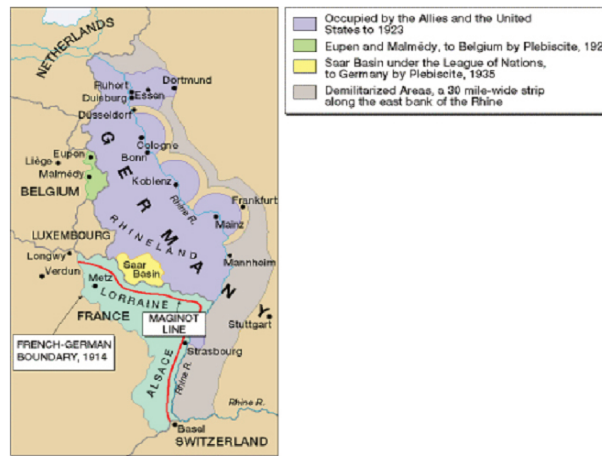


Figura 1.3: Linha *Maginot*

Com tamanho investimento em sua defesa de perímetro, a França julgou-se segura com relação a uma possível invasão alemã ou italiana.

Poucos meses após o início da Segunda Guerra Mundial, em maio de 1940, Hitler deu início à invasão da França. Sua estratégia foi simples. Ao invés de confrontar a Linha *Maginot*, suas tropas invadiram os Países Baixos, e o território francês foi adentrado por meio da fronteira com a Bélgica. Novamente usando de sua *Blitzkrieg* (guerra relâmpago), os alemães combinaram alto poder de mobilização e suporte aéreo, e, em junho de 1940, menos de dois meses após o início da Batalha da França, Paris capitulou.

## A Batalha da Inglaterra

Tomada a França, as forças do eixo dominavam a maior parte da Europa continental. A Inglaterra viu-se isolada. Suas tropas comissionadas no continente tiveram de bater em retirada pelo porto de Dunquerque sob pesado fogo inimigo. A ilha britânica, que outrora estivera fora do raio de ação da aviação de caça alemã, agora encontrava-se a um Canal da Mancha de distância da mesma.

Analistas militares da época debatiam, e, em geral, concluíam que a invasão da Grã-Bretanha pela Alemanha seria um processo natural, uma vez que os nazistas estabelecessem domínio aéreo sobre a mesma. Com tal domínio, o desembarque de paraquedistas e tropas seria possível, e uma nova *Blitzkrieg* seria observável em território bretão. Embora haja divergência sobre números, é de consenso entre historiadores que a força aérea alemã contava com um contingente de aeronaves substancialmente superior ao contingente

inglês. Ademais, os pilotos alemães eram, em geral, veteranos da Guerra Civil Espanhola, tendo, portanto, mais experiência prática de guerra que os ingleses.

Em agosto de 1940 começa a operação Leão-Marinho para tomada da Inglaterra pela Alemanha.

Embora inferiorizada numericamente em aeronaves, a Inglaterra desenvolveu um sistema de proteção aérea denominado Sistema *Dowding* (uma homenagem a seu idealizador, Sir Hugh Dowding, comandante da aviação de caça da RAF durante a Batalha da Inglaterra). Segundo este sistema, aviões inimigos eram detectados por uma nova tecnologia, denominada radar. Como esta tecnologia estava em sua infância, seu funcionamento era complementado pelo corpo de observadores, que, a exemplo das torres de radar, encontravam-se estrategicamente dispostas na costa e no território inglês. Falsos positivos gerados pelos radares (por deficiências da tecnologia) podiam ser contra-checados pelo corpo de observadores. Por semelhante modo, falsos negativos de um dos grupos podia ser identificado pelo outro. Em suma, temos um sistema de correlação de eventos de segurança anterior ao advento da computação digital como a conhecemos. Um centro de comando, abastecido pelas informações dos radares e dos observadores por um sistema de rádio e um protocolo de comunicação claramente definido, fazia a correlação dos dados e definia a vetorização da interceptação. Ou seja, o centro de comando otimizava o uso das poucas aeronaves de caça disponíveis ao definirem claramente quais grupos aéreos deveriam enviar quantos aviões em quais coordenadas para confrontar o inimigo (a Figura 1.4 ilustra a disposição dos grupos aéreos disponíveis para a vetorização).

À luz dos alemães, os recursos ingleses pareciam multiplicarem-se e seus caças serem quase onipresentes na defesa de seu território. Com poucos aviões e pilotos, os ingleses resistiram por várias semanas aos ataques aéreos e por fim, em maio de 1941, Hitler desiste de invadir a Grã-Bretanha e tem fim a Batalha da Inglaterra. Trata-se da primeira derrota nazista na Segunda Guerra; um triunfo de um modelo de inteligência de segurança e otimização de recursos, retratado pela célebre frase de Sir Winston Churchill: “*Never was so much owed by so many to so few (Nunca tanto foi devido a tão poucos)*”.

## Sistema *Dowding* e Segurança da Informação

Um paralelo pode ser traçado entre as Batalhas da França e da Inglaterra, e as técnicas de defesa da segurança da informação. A defesa de perímetros, sem uma gestão de eventos de segurança associada, em muito se assemelha à Linha *Maginot*. Embora por vezes segura em sua construção, ela não é capaz de impedir todos os tipos de ataques e sua baixa flexibilidade torna-a inadequada num cenário de constante evolução das infraestruturas tecnológicas. Por outro lado, o Sistema *Dowding*, focado na detecção e correlação de eventos oriundos de diferentes fontes, contempla questões como falsos positivos e falsos negativos, permitindo que poucos recursos provejam mais segurança a uma



Figura 1.4: Sistema *Dowding*

dada infraestrutura. No mundo de TI, podemos encarar nossos IDSs, arquivos de *logs* de segurança, eventos de *firewall* e outras fontes de eventos como radares e grupos de observadores. Cada evento não significa necessariamente um ataque; na verdade, em geral, um evento não implica num ataque. Porém, a correlação destes eventos permite a detecção de ataques reais, e, conseqüentemente, o emprego mais otimizado dos recursos humanos e tecnológicos na garantia da segurança. Dentro deste contexto, propomos uma abordagem baseada na correlação de eventos de segurança de diferentes fontes, mediante usos de técnicas de aprendizado de máquina, para determinarmos de uma forma mais assertiva a ocorrência de ataques a uma infraestrutura tecnológica. Tal metodologia deve ser aplicável a diferentes cenários como Centros de Processamentos de Dados (CPDs), redes de telecomunicações (fixas e móveis), redes de ATMs (*Automated Teller Machines*), *Data Centers*, *SOCs* (*Security Operation Centers*), e correlatos.

## 1.2 Folhas, Árvores e Florestas

A enxurrada de eventos de segurança oriundos dos diversos dispositivos da rede impõem um desafio à equipe de segurança de qualquer instituição. Volumes da ordem de centenas de milhares a milhões de eventos por dia são comuns em infraestruturas corporativas e governamentais de grande porte.

Em meio a esta miríade de dados, vemos frequentemente profissionais de segurança perdidos investigando cada evento isoladamente, sem conseguir compreender o cenário que se delineia ao entorno do mesmo. A este fenômeno denominamos “miopia em segurança”.

A analogia mais adequada, aqui, é aquela da pessoa que se perde observando as folhas sem conseguir enxergar a floresta.

É imperativo que consigamos associar folhas a árvores, e a partir destas, entender a floresta. Em outras palavras: diferentemente da maior parte dos trabalhos em detecção de intrusão, que se esmeram em processar eventos individuais, propomos uma abordagem que agrupa alertas (ou folhas) em meta-alertas (árvores), segundo critérios de afinidade. Estes meta-alertas serão o insumo para as técnicas de aprendizado de máquina que auxiliarão na decisão de identificar os ataques (árvores especiais) em meio ao ruído de alertas gerado pela infraestrutura (restante da floresta).

## 1.3 Contribuições

Este trabalho apresenta uma abordagem para o processamento de eventos (alertas) da segurança da informação. Tal abordagem objetiva indicar reais ameaças à infraestrutura de segurança, reduzindo-se o número de falsos alarmes.

Outra contribuição reside na implementação desta abordagem. Apresentamos uma implementação em camadas funcionais: coleta e normalização, fusão, e classificação. Cada camada é detalhadamente apresentada, discutindo-se suas premissas e principais decisões implementacionais.

Em especial, a camada de classificação é implementada de quatro maneiras distintas, utilizando-se, para tal, de quatro técnicas de aprendizado de máquina.

Tais implementações são testadas contra duas bases de dados de segurança distintas, e os resultados obtidos são analisados e contrastados entre si.

Ao final, consolidamos uma abordagem que permite às equipes de segurança da informação processar os alertas de segurança de uma forma eficiente, reduzindo o volume de informações a serem analisadas e priorizando aquelas que apresentam menor possibilidade de serem falsos positivos.

Tal abordagem é sustentada pela fusão de alertas de segurança, pela definição de uma taxonomia para eventos de segurança e pelo uso de modernas técnicas de aprendizado de

máquina sobre os meta-alertas identificados.

## 1.4 Organização do Documento

Este documento está organizado da forma a seguir.

O Capítulo 2 introduz os fundamentos teóricos associados ao trabalho. Apresentam-se os principais conceitos associados a segurança da informação (Seção 2.1) e aprendizado de máquina (Seção 2.2).

No Capítulo 3 destacamos alguns trabalhos correlatos. O interesse reside em contribuições na área de processamento e correlações de eventos de segurança, bem como áreas afins.

A abordagem proposta é detalhada no Capítulo 4. Nele abordamos cada uma das camadas do modelo, bem como suas especificidades e desafios.

A implementação do modelo e os experimentos executados são apresentados no Capítulo 5. Neste capítulo detalhamos a implementação da abordagem e as bases de dados usados nos testes. Por fim, analisamos detalhadamente os resultados obtidos nos experimentos realizados.

O Capítulo 6 resume as principais conclusões e contribuições oriundas deste trabalho.

# Capítulo 2

## Fundamentação Teórica

A correlação de eventos de segurança emerge como um campo de pesquisa multidisciplinar, que orbita sobre a segurança da informação e técnicas de processamento de grandes volumes de dados, dentre os quais, neste trabalho, destaca-se o aprendizado de máquina. Nas próximas seções apresentamos um resumo dos principais conceitos associados a estas duas disciplinas dentro de sua interface com nosso universo de pesquisa.

### 2.1 Segurança

Segundo Garfinkel e Spafford [20], um computador é considerado seguro se um usuário pode depender deste e de seus programas. Sistemas seguros devem satisfazer algumas características, comumente denominada tríade de CIA, a saber:

- *Confidencialidade (C)*: informação não pode ser lida ou copiada por alguém que não tenha sido explicitamente autorizado a fazê-lo pelo proprietário da informação;
- *Integridade (I)*: informação não deve ser apagada ou alterada sem a permissão de seu proprietário;
- *Disponibilidade (A - do inglês, Availability)*: garantir que serviços não são degradados ou colocados indisponíveis sem autorização.

A segurança da informação pode também ser abordada pela ótica das entidades e usuários, e seus acessos e ações junto aos sistemas. Uma nova tríade, denominada AAA, descreve tais conceitos:

- *Autenticação (A)*: garantir a real identidade de um usuário ou entidade. Permite a garantia da relação entre uma pessoa ou sistema do mundo físico com sua respectiva identidade digital.



- *Autorização (A)*: determinar quais ações podem ser executadas por um usuário ou entidade, subsidiando o controle de acesso aos recursos computacionais.
- *Irretratabilidade (A - do inglês, Accounting)*: associar ações e acessos a usuários e entidades de forma inquestionável. Um usuário não deve poder repudiar uma ação de sua autoria.

Quando a segurança de um sistema computacional está em risco, tal informação é comumente veiculada sob a forma de um evento de segurança.

Segundo a norma ABNT NBR ISO/IEC 17799:2005 [25], um *evento* de segurança da informação é uma ocorrência identificada de um sistema, serviço ou rede, que indica uma possível violação da política de segurança da informação ou falha de controles, ou uma situação previamente desconhecida, que possa ser relevante para a segurança da informação. Eventos de segurança por vezes são denominados também *alertas*, e ambos os termos serão empregados de forma intercambiável neste trabalho.

Eventos ou alertas de segurança são comuns nas infraestruturas tecnológicas atuais, podendo ser gerados por uma série de dispositivos: *firewalls*, sistemas de detecção de intrusão, *logs* de sistemas operacionais, *logs* de servidores web, sistemas de autenticação de usuários, entre muitos outros. A presença de um evento ou alerta não implica necessariamente no comprometimento da segurança do sistema associado. Todavia, ela pode ser um indicativo deste comprometimento.

Seguindo a escala de probabilidade de comprometimento da segurança, define-se que um *incidente* de segurança da informação corresponde a um ou mais alertas indesejados ou inesperados que tenham grande probabilidade de comprometer as operações do negócio e ameaçar a segurança da informação.

Incidentes de segurança costumam estar associados à exploração de *vulnerabilidades*. Uma vulnerabilidade corresponde à fragilidade de um ativo ou grupo de ativos que pode ser explorada por ameaças que constituem causas potenciais de incidentes indesejados.

A exploração de vulnerabilidades é uma das principais fontes de ataques à segurança da informação. O comprometimento da confidencialidade, integridade e disponibilidade tipicamente se dá através da exploração de uma fragilidade conhecida, como um *bug* de software, uma fraqueza criptográfica, uma falha de implementação de um protocolo, ou mesmo explorando a confiança de um usuário através da engenharia social.

### 2.1.1 Tipos de Ataques

Conforme observado por Zwicky et al [63], há vários tipos de ataques a uma infraestrutura e várias formas de se categorizar tais ataques. Neste trabalho, utilizaremos uma versão estendida daquela apresentada por Zwicky, constituída pelos seguintes tipos de ataques:

- *Negação de Serviço (Denial of Service, ou, simplesmente, DoS)*: ataque que impede que um usuário ou conjunto de usuários seja plenamente atendido pelos sistemas que lhe prestam serviços. Tais ataques podem ocorrer de diversas formas, como a derrubada de sistemas, exaustão de seus recursos computacionais (memória, capacidade de processamento, banda de rede), interrupção da comunicação, entre outros. Quando o ataque tem múltiplas origens, denomina-se negação de serviço distribuída (em inglês, *Distributed Denial of Service (DDoS)*). Trata-se de um ataque à disponibilidade ou integridade do sistema e/ou serviço;
- *Exploração (Probe)*: ataque que busca vulnerabilidades a serem exploradas junto ao sistema alvo. Alguns autores, como Scambray et al [52] subdividem este ataque em duas partes: varredura e enumeração. Um ataque de exploração visa mapear pontos de invasão para futuros ataques de outros tipos. Trata-se de um ataque à confidencialidade da configuração do sistema;
- *Acesso Remoto (Remote to Local, ou, simplesmente, R2L)*: atacante obtém acesso ao sistema alvo, sem que ele tivesse credenciais prévias para tal. Dentre as instâncias deste tipo de ataque destacam-se o roubo de senhas de usuários locais, a exploração de vulnerabilidades de serviços ou *daemons* em Unix, entre outros. Trata-se de um ataque que pode promover perdas de confidencialidade ou integridade de um sistema;
- *Acesso Superusuário (User to Root, ou, simplesmente, U2R)*: a partir de uma conta de usuário comum, atacante consegue explorar vulnerabilidades do sistema e escalar seus privilégios a superusuário. Trata-se de um ataque que pode implicar em perdas de confidencialidade ou integridade de um sistema;
- *Roubo de Dados (Data Theft, ou, simplesmente, Data)*: dados confidenciais são obtidos por entidade não autorizada a acessar os mesmos. Trata-se do cenário mais clássico de ataque à confidencialidade da informação.

Esta classificação está em linha com aquela utilizada por Lippmann et al [22] e outros trabalhos correlatos a este, e servirá de base para análises da abordagem a ser apresentada adiante.

### 2.1.2 Dispositivos de Segurança

Alertas de segurança são provenientes de diversos dispositivos de uma infraestrutura de TI. Todavia, dois destes dispositivos possuem especial relevância no contexto da segurança da informação: *firewalls* e sistemas de detecção de intrusão.

Nas próximas subseções abordaremos as funcionalidades e características principais destes dispositivos.

### ***Firewalls***

Segundo Zwicky et al [63], *firewall* é um componente ou conjunto de componentes que restringe o acesso entre partes protegidas da rede e a Internet, ou entre diferentes grupos de redes.

Conforme observado por Garfinkel e Spafford [20], o termo *firewall* faz alusão à indústria de construção. Da mesma forma que as paredes corta-fogo impedem que o fogo se alastre de um dormitório a outro, *firewalls* contêm o avanço de invasores. Devido às suas características, *firewalls* são usados para diferentes propósitos, dentre eles:

- Bloquear acessos a determinados sítios da Internet ou evitar que usuários e computadores acessem determinados servidores ou serviços;
- Monitorar comunicações entre redes internas e externas;
- Estabelecer redes privadas virtuais (VPNs) entre diferentes localidades ou pontos remotos de uma rede corporativa;
- Inspeccionar pacotes de redes e impedir o tráfego de conteúdo sensível;
- Segregar redes de forma a criar uma arquitetura de exposição seletiva a riscos, como a construção de uma zona desmilitarizada (DMZ); entre outros.

Tipicamente *firewalls* são localizados em pontos estratégicos de uma infraestrutura de TI. Tal fato torna-os especialmente relevantes para a gerência da segurança de uma rede, atuando como sensores que detectam a violação de políticas de segurança, ou ameaças à segurança da infraestrutura.

### **Sistemas de Detecção de Intrusão (IDSs)**

Um IDS é um sistema voltado para detecção de intrusão num computador — HIDS (*Host Intrusion Detection System*) —, ou numa rede — NIDS (*Network Intrusion Detection System*). Um exemplo de NIDS de grande profusão é o Snort [54].

Sistemas de detecção de intrusão do tipo HIDS baseiam seu funcionamento no monitoramento de diferentes tipos de entidades dentro de um sistema computacional, tais como: sequência de chamadas ao sistema operacional (*system calls*), alterações ao sistema de arquivos, arquivos de *log* e de configuração do sistema operacional, entre outros.

Por sua vez, sistemas de detecção de intrusão de rede tipicamente operam sobre a análise de pacotes que trafegam num segmento de rede. Sua análise pode envolver tanto o cabeçalho (*header*) quanto o conteúdo (*payload*) dos pacotes monitorados.

Outra categorização de IDSs envolve o princípio pelo qual a intrusão é determinada. Neste sentido, há dois tipos de IDSs: sistemas de detecção de anomalia e os sistemas de detecção de uso indevido.

Segundo Lee e Stolfo [29], sistemas de detecção de anomalia estabelecem um padrão de uso normal (ou perfil) usando medidas estatísticas sobre o sistema e o contrastam com seu comportamento atual. Mudanças percebidas em relação ao padrão são traduzidas em alertas de intrusão, como o que ocorre com os sistemas PHAD E ALAD de Mahoney e Chan [33].

Sistemas de detecção de uso indevido codificam sequências de ações como assinaturas de ataques. Se tais sequências são percebidas, alertas de intrusão são gerados.

Um IPS é uma extensão do conceito de IDS, onde a ação tomada pelo sensor transcende a simples detecção da intrusão, e volta-se para sua prevenção, através de técnicas como a proteção de processos, a interceptação de sessões de rede, entre outras [11].

No intuito de viabilizar a interação e integração de diferentes IDSs e IPSs, o IETF publicou um padrão denominado IDMEF (*Intrusion Detection Message Exchange Format*) [12], o qual define formatos de dados e procedimentos para trocas de informações entre sistemas de detecção de intrusão e sistemas de gerenciamento de segurança.

## IDMEF

Segundo relatado na própria RFC 4765 [12], o IDMEF (*Intrusion Detection Message Exchange Format*) tem o objetivo de “definir formatos e procedimentos para troca de dados que sejam de interesse para sistemas de detecção e resposta a intrusões bem como sistemas de gerenciamento que necessitem interagir com os mesmos”.

O IDGW (grupo associado ao IETF) escolheu o uso de XML (*eXtensible Markup Language*) como forma de representar as mensagens de intrusão. A classe mensagem (*IDMEF-Message*) é subdividida em duas possíveis subclasses: *Alert* e *Heartbeat*.

A subclasse *Alert* é de especial interesse, pois consiste na forma padrão de se representar um evento ou alerta de segurança detectado por um IDS. Ela é composta pelos seguintes atributos:

- *Analyzer*: identificação do sistema que gerou o alerta em questão;
- *CreateTime*: hora e data da criação do alerta;
- *DetectTime*: datas e horas dos eventos que deram origem ao alerta;

- *AnalyzerTime*: data e hora corrente do sistema que gerou o alerta;
- *Source*: as origens dos eventos que originaram o alerta. Este atributo pode, por sua vez, ser composto pelas seguintes entidades: *Node* (sistema computacional identificado a partir de seu endereço de rede), *User* (usuário pertencente a um sistema computacional ou domínio), *Process* (processo), e *Service* (serviço);
- *Target*: os destinos dos eventos que originaram o alerta. Este atributo pode, por sua vez, ser composto pelas seguintes entidades: *Node* (sistema computacional identificado a partir de seu endereço de rede), *User* (usuário pertencente a um sistema computacional ou domínio), *Process* (processo), *Service* (serviço), e *File* (arquivo);
- *Classification*: “nome” do alerta ou outra forma de identificação. Comumente preenchido com identificações de vulnerabilidades como CVE, *bugtraqid*, entre outros;
- *Assessment*: informações sobre o impacto do evento, ações tomadas em resposta, e confiança na avaliação;
- *AdditionalData*: informações adicionais incluídas pelo analisador que não se encaixam nas demais estruturas do modelo.

Sua representação em DTD é a seguinte:

```
<!ELEMENT Alert
    (
        Analyzer, CreateTime, DetectTime?, AnalyzerTime?,
        Source*, Target*, Classification, Assessment?, (ToolAlert |
        OverflowAlert | CorrelationAlert)?, AdditionalData*
    )>
<!ATTLIST Alert
    messageid          CDATA          '0'
    %attlist.global;
>
```

A subclasse *Alert* pode ser derivada em subclasses mais especializadas, segundo o tipo de alerta que se quer representar. Exemplos de classes de alertas derivadas são: *ToolAlert*, *OverflowAlert* e *CorrelationAlert*.

A subclasse *Heartbeat* é utilizada por analisadores (*Analyzers*) para indicar seu estado para os sistemas de gerenciamento. Ela é composta pelos seguintes atributos:

- *Analyzer*: identificação do sistema que gerou o *heartbeat* em questão;

- *CreateTime*: hora e data da criação do *heartbeat*;
- *HeartbeatInterval*: o intervalo (em segundos) segundo o qual *heartbeats* são gerados;
- *AnalyzerTime*: data e hora corrente do sistema que gerou o *heartbeat*;
- *AdditionalData*: informações adicionais incluídas pelo sistema de detecção que não se encaixam nas demais estruturas do modelo.

Em DTD, esta classe é descrita da seguinte forma:

```
<!ELEMENT Heartbeat
    (
        Analyzer, CreateTime, HeartbeatInterval?, AnalyzerTime?,
        AdditionalData*
    )>
<!ATTLIST Heartbeat
    messageid          CDATA          '0'
    %attlist.global;
>
```

Conforme observado por Martimiano [34], a identificação de tipos de alertas ocorre através de seus nomes (chaves ou códigos) junto a bases de dados de vulnerabilidades, como os projetos CVE ou CERT/CC. Tal identificação apresenta algumas limitações:

- Nomes não apresentam uma estrutura padrão que possa ser usada em sistema de correlação de alertas;
- Não há suporte para correlação entre vulnerabilidades ou extração de valor semântico;
- Não há mecanismos diretos para correlação de vulnerabilidades e incidentes.

A falta de uma ontologia comum no modelo proposto pelo IETF também é observada por Valdes e Skinner ([58]).

Desta feita, faz-se necessário o desenvolvimento de mecanismos de semântica mais forte para suporte a sistemas de processamento de alertas. Tais mecanismos podem ser alcançados através de técnicas como taxonomias e ontologias de segurança.

## Taxonomias e Ontologias

No desenvolvimento de sistemas de gerenciamento de segurança há especial interesse no estudo de *taxonomias*, ou seja, classificação dos alertas e princípios subjacentes.

É importante que alertas possam ser classificados de forma que sua representação e identificação ocorram de forma independente das fontes de sua detecção ou processamento. Logo, se um pacote é filtrado por um *firewall* ou IPS, o alerta associado deve ser classificado de uma forma homogênea.

Quanto ao formato do alerta, o padrão IDMEF apresenta-se como uma boa alternativa de representação. Todavia, o atributo *Classification* não favorece o desenvolvimento de correlações. Para tal, faz-se necessário o estabelecimento de uma taxonomia de eventos de segurança.

Algumas iniciativas de estabelecer tal taxonomia existem na iniciativa privada, como, por exemplo, no produto TSOM (*Tivoli Security Operations Manager*) abordado por Buecker et al [5].

A Tabela 2.1 mostra alguns exemplos da taxonomia usada pelo TSOM.

Tabela 2.1: Taxonomia no TSOM

Class Id	Event Class
10004	os.log.audit.success
20001	app.virus.detect
20002	app.shutdown
40002	neu.auth.config.group
40001	neu.auth.config.account
50001	fw.accept
50003	fw.reject
60002	ids.host.reset
60003	ids.host.banner
60011	honeypot.detect

Contudo, não há padrões amplamente adotados ou reconhecidos, e suas estruturas carecem de maior rigor técnico, inclusive para sua promoção.

Do aprofundamento de estudo de taxonomias derivam-se *ontologias* de taxonomias, conforme relatado por Martimiano e Moreira [35]. Segundo Gruber [21], ontologia é “a especificação formal e explícita de uma abstração, uma visão simplificada de um domínio de conhecimento”.

Taxonomias e ontologias são relevantes no apoio à correlação de alertas de segurança, pois permitem a normalização e a representação. Sua estrutura oferece a base sobre a qual correlações de diferentes tipos podem ser desenvolvidas de forma independente da

camada de coleta de eventos e alertas.

### Potes de Mel (*Honeypots*)

Potes de Mel (*honeypots*) consistem em um ou mais sistemas configurados de forma a atrair a atenção de invasores de redes e permitir o monitoramento de suas atividades.

Dois são os usos mais comuns para potes de mel no dia-a-dia da segurança da informação:

- desviar a atenção de invasores para sistemas que não contêm informações sigilosas ou serviços de natureza crítica, poupando, portanto, os servidores mais sensíveis da abordagem dos invasores;
- aprender sobre o comportamento de invasores e vulnerabilidades de sistemas.

Derivado do conceito de pote de mel advém a rede de mel (*honeynet*). Esta última consiste numa ferramenta de pesquisa do comportamento de invasores. Trata-se de uma rede criada especificamente com o propósito de ser invadida. Seus componentes são sistemas de produção padrão, com sistemas e aplicativos reais, conforme encontrados na Internet. Desta feita, os riscos e vulnerabilidade descobertos com o uso de redes de mel são os mesmos aos quais sistemas estão expostos no mundo real.

Um exemplo de um rede de mel foi construído pelo Projeto *Honeynet* [48]. Trata-se de uma organização internacional, sem fins lucrativos, dedicada à melhoria da segurança na Internet. Utilizando-se de *honeynets* como ferramentas de trabalho, publicam informações acerca de segurança em formatos que variam de livros [48] e informações *on-line* (<http://www.honeynet.org>) a desafios de detecção e invasão de computadores (*The Scan of the Month*), entre outros.

Neste trabalho, estamos interessados nos dados disponibilizadas por redes de mel (em especial na forma de desafios do *Scan of the Month*) como massas de teste em nossa abordagem para a detecção de intrusão.

## 2.2 Aprendizado de Máquina

*Aprendizado de Máquina* é o estudo de algoritmos computacionais cujo desempenho melhora com a experiência.

Dentro desta disciplina estamos especialmente interessados no funcionamento de agrupadores e classificadores.

Um *agrupador* ou *clusterer* é uma ferramenta que permite o particionamento de dados em conjuntos cujos elementos compartilham características comuns.



Um *classificador* provê o mapeamento entre um espaço de características, ou dados de entrada,  $X$  para um conjunto discreto de rótulos  $Y$ .

Três técnicas têm merecido especial atenção pela comunidade científica no endereçamento de questões de classificação: Support Vector Machines (SVMs) [53], Redes *Bayesianas* (BNs) [49][9][43] e Árvores de Decisão.

Nas próximas subseções descreveremos seus funcionamentos juntamente com técnicas de agregação de classificadores (*ensembles*). Também abordaremos métodos de verificação de eficácia em classificação e uma forma de representação do desempenho de um classificador, a curva ROC (*Receiver Operating Characteristic*).

### 2.2.1 Support Vector Machines

*Support Vector Machines* têm sua origem em métodos de *Kernel* da estatística, e constituem uma importante ferramenta de classificação, através de uma estratégia de simultaneamente maximizar a margem de separação entre dois grupos e de minimizar os erros de tal separação.

Como dados de treinamento, sejam os pontos

$$D = \{(x_i, y_i) | x_i \in \mathcal{H}, y_i \in \{-1, +1\}\}, \forall i = 1, \dots, n \quad (2.1)$$

onde  $y_i$  corresponde a  $+1$  ou  $-1$ , indicando a que classe corresponde a instância  $x_i$ ;  $\mathcal{H}$  é um espaço com produto interno.

Pode-se pensar em  $x_i$  como o conjunto dos alertas ou meta-alertas a serem classificados, e  $y_i$  como a definição se aquele alerta ou meta-alerta representa um ataque ( $+1$ , por exemplo), ou um falso alarme ( $-1$ , neste caso).

Queremos determinar o hiperplano que divide os pontos com  $y_i = +1$  dos pontos com  $y_i = -1$ . Tal hiperplano é descrito como o conjunto de pontos  $x$  que satisfaz

$$\langle w, x \rangle + b = 0 \quad (2.2)$$

onde  $\langle w, x \rangle$  representa o produto interno de  $w$  e  $x$ ; desta feita,  $w$  é o vetor perpendicular ao hiperplano.

A Figura 2.1 mostra uma representação gráfica dos conjuntos de treinamento e do hiperplano separador.

A distância de um ponto qualquer  $x$  ao hiperplano é dado por

$$\frac{\langle w, x \rangle + b}{\|w\|} \quad (2.3)$$

onde  $\|w\|$  é a norma de  $w$  definida por

$$\|w\| = \sqrt{\langle w, w \rangle} \quad (2.4)$$

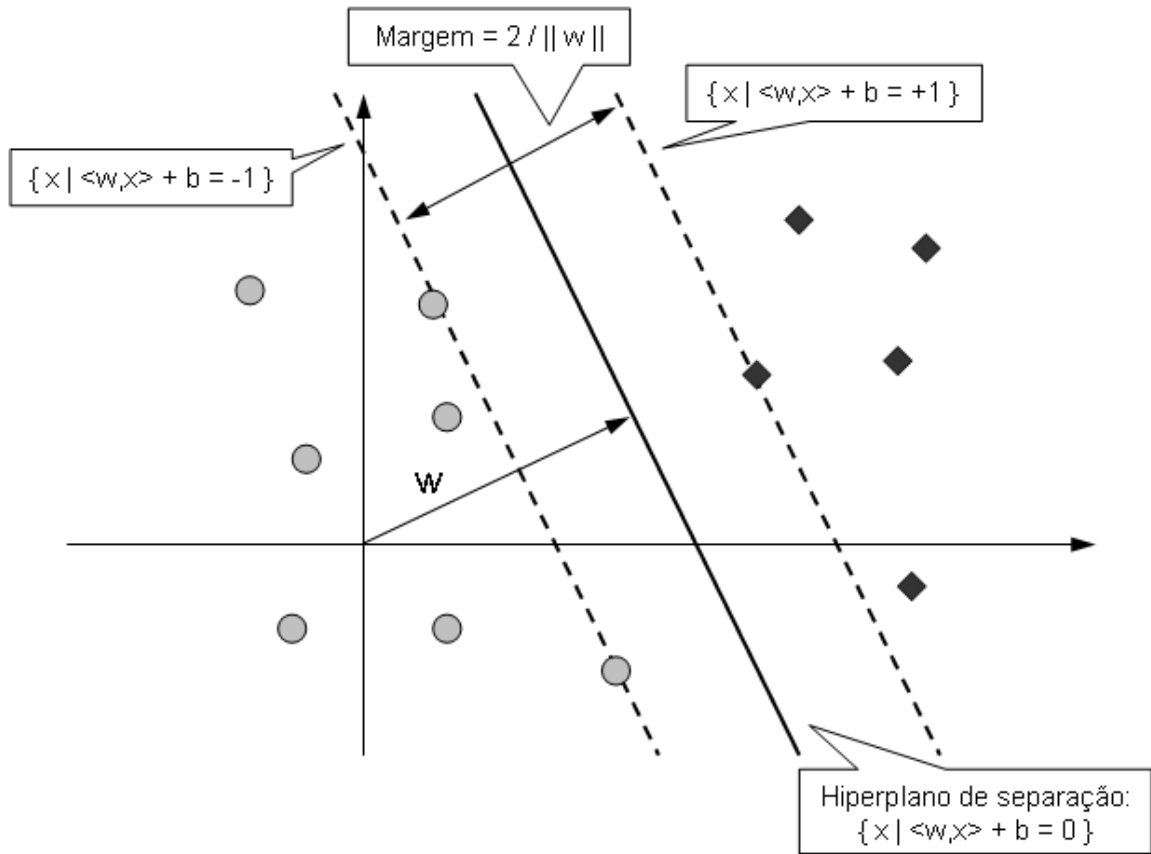


Figura 2.1: Separador Linear

Um SVM pode ser dividido em três categorias: linearmente separável, linearmente não-separável e não-linear.

### SVM linearmente separável

Como os pontos são linearmente separáveis, temos que:

$$y_i(\langle w, x_i \rangle + b) \geq 0 \quad (2.5)$$

Pode-se ajustar as escalas de  $w$  e  $b$  de forma que o(s) ponto(s) mais próximo(s) ao hiperplano satisfaça(m)

$$y_i(\langle w, x_i \rangle + b) = 1, \quad (2.6)$$

obtendo-se a forma canônica do hiperplano:

$$y_i(\langle w, x_i \rangle + b) \geq 1, \quad (2.7)$$

Neste caso, a distância do ponto mais próximo ao hiperplano é igual a  $1/\|w\|$ . Isto resulta numa margem de  $2/\|w\|$ .

Logo, existe um hiperplano ótimo de separação, que maximiza a margem entre os pontos de treinamento e o próprio hiperplano. Ele é a solução de

$$\min_{w \in \mathcal{H}, b \in \mathcal{R}} \mathcal{T}(w) = \frac{1}{2} \|w\|^2 \quad (2.8)$$

sujeito a

$$y_i(\langle w, x_i \rangle + b) \geq 1, \forall i = 1, \dots, m. \quad (2.9)$$

Trata-se de um problema de otimização com restrições, que pode ser resolvido com multiplicadores de Lagrange  $\alpha_i \geq 0$  e o *Lagrangiano*

$$L(w, b, \alpha) = \frac{1}{2} \|w\|^2 - \sum_{i=1}^m \alpha_i (y_i(\langle x_i, w \rangle + b) - 1) \quad (2.10)$$

Precisamos minimizar o *Lagrangiano* com respeito às variáveis  $w$  e  $b$  e maximizar com relação a  $\alpha_i$ . Buscamos um ponto de sela, no qual as derivadas parciais com relação a estas variáveis são iguais a zero:

$$\frac{\partial}{\partial b} L(w, b, \alpha) = 0 \Rightarrow \sum_{i=1}^N \alpha_i y_i = 0; \quad (2.11)$$

e

$$\frac{\partial}{\partial w} L(w, b, \alpha) = 0 \Rightarrow w = \sum_{i=1}^N \alpha_i x_i y_i \quad (2.12)$$

Substituindo 2.11 e 2.12 em 2.10, eliminam-se as variáveis  $w$  e  $b$ , e obtém-se o problema dual

$$\max_{\alpha \in \mathcal{R}^m} W(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \quad (2.13)$$

sujeito a

$$\alpha_i \geq 0 \forall i = 1, \dots, m \quad (2.14)$$

e

$$\sum_{i=1}^m \alpha_i y_i = 0 \quad (2.15)$$

O problema da maximização de  $W$  é mais simples que o de  $L$  e computacionalmente menos custoso.

Portanto, a função de classificação para um novo ponto  $z$  em função do hiperplano de decisão é dada por:

$$\text{sgn}(\langle w, z \rangle + b) \quad (2.16)$$

onde  $\text{sgn}$  corresponde ao sinal, positivo (+1) ou negativo (-1), do valor entre parênteses.

### SVM linearmente não-separável

Um hiperplano de separação pode não existir devido à superposição das classes. Tal situação pode ser contornada através da introdução de variáveis de folga

$$\xi_i \geq 0, \forall i = 1, \dots, m \quad (2.17)$$

que relaxam as restrições da equação do hiperplano 2.9 da seguinte forma

$$y_i(\langle w, x_i \rangle + b) \geq 1 - \xi_i, \forall i = 1, \dots, m \quad (2.18)$$

O novo classificador deve tanto maximizar a margem (minimizando, para isto,  $\|w\|$ ) quanto minimizar os erros  $\sum_i \xi_i$ . Um possível classificador pode ser obtido pela minimização da função objetivo

$$\tau(w, \xi) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \quad (2.19)$$

onde a constante  $C > 0$  determina o *trade-off* entre a maximização da margem e a minimização dos erros. Tipicamente esta constante é definida pelo usuário no estabelecimento do comportamento desejado do classificador.

Introduzindo as variáveis de folga e a constante  $C$  nos desenvolvimentos do SVM linearmente separável, obtemos o mesmo problema dual de otimização descrito em 2.13, sujeito a 2.15. A única diferença reside na adição uma nova restrição

$$0 \leq \alpha_i \leq C, \forall i = 1, \dots, m \quad (2.20)$$

Esta restrição define um limite para os multiplicadores do Lagrangiano  $\alpha_i$ , reduzindo a influência de um dado particular de treinamento sobre o classificador.

### SVM não-linear

A classificação pode não ser possível com um SVM linear. Isto ocorre quando um hiperplano linear no espaço de origem não permite a adequada separação dos dados.

Nestes casos, pode-se lançar mão de SVMs não-lineares. Nestes, os dados de entrada são projetados num espaço de dimensões maiores através de uma função de mapeamento:

$$\phi : \mathcal{H} \rightarrow \mathcal{H}' \quad (2.21)$$

Para cada elemento  $x_i$  de nosso conjunto de treinamento, aplicamos  $\phi(x_i)$ , determinando um novo espaço  $\mathcal{H}'$  que contém, tipicamente, mais dimensões que o espaço original  $\mathcal{H}$ .

Pode não ser conveniente executarmos a otimização da SVM neste novo espaço dimensional  $\mathcal{H}'$ , dado o custo computacional envolvido.

Deve-se notar que a otimização 2.13 envolve apenas operações sobre produtos internos dos dados de treinamento. Logo, pode-se definir uma função de *kernel*  $k$  que tenha a seguinte propriedade:

$$k(x, x') = \langle \phi(x), \phi(x') \rangle \quad (2.22)$$

Com o auxílio desta função  $k$  pode-se realizar a referida otimização sem a necessidade da projeção de todo o conjunto de treinamento no espaço  $\mathcal{H}'$ .

Dentre as funções de *kernel* mais comuns pode-se destacar:

- Polinomial:  $k(x, x') = (\gamma \langle x, x' \rangle + r)^d$
- Função Radial de Base:  $k(x, x') = e^{-\sigma \|x - x'\|^2}$
- Sigmoidal:  $k(x, x') = \tanh(\gamma \langle x, x' \rangle + r)$

Portanto, a função de classificação para um novo ponto  $z$  é dada por

$$\text{sgn}(\langle w, \phi(z) \rangle + b) \quad (2.23)$$

que pode ser reescrita como

$$\text{sgn}\left(\sum_{i=1}^m \alpha_i k(x_i, z) y_i + b\right) \quad (2.24)$$

onde  $\text{sgn}$  corresponde ao sinal, positivo (+1) ou negativo (-1), do valor entre parênteses;  $k$  é a função de *kernel* escolhida.

### 2.2.2 Redes *Bayesianas*

Redes Bayesianas (*Bayesian Networks* ou, simplesmente, BNs) são grafos que apresentam as seguintes propriedades:

1. Vértices representam variáveis randômicas.
2. Arestas direcionadas conectam pares de vértices. Se uma aresta direcionada liga um vértice  $A$  (sua origem) a um vértice  $B$  (seu destino), dizemos que  $A$  é pai de  $B$ , e  $B$  é filho de  $A$ . A intuição é que  $A$  tem alguma influência em  $B$ .
3. Cada vértice contém uma tabela de probabilidades condicionais (*Conditional Probability Table* ou CPT) que quantifica os efeitos dos pais sobre o dito vértice.
4. Dadas as direções das arestas, o grafo não possui ciclos — grafo direcionado acíclico (*Directed Acyclic Graph* ou, simplesmente, DAG).

Mais formalmente, Redes *Bayesianas* consistem em grafos direcionados acíclicos (DAGs) que representam um conjunto de variáveis e suas independências probabilísticas, segundo a *condição de Markov* [43].

A *condição de Markov* estabelece que um vértice é condicionalmente independente de todos seus não-descendentes dado o conjunto de todos os seus pais.

O funcionamento de uma BN apóia-se no teorema de Bayes:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \quad (2.25)$$

Uma BN permite que restrinjamos uma tabela completa de distribuições conjuntas de probabilidades (*Joint Probability Distributions* ou JPD) — que representa as probabilidades de todas as combinações possíveis dos valores das variáveis observadas — a uma estrutura de dados menor e de processamento menos custoso. Isto é de especial valia quando se considera que uma JPD cresce exponencialmente com o número de variáveis envolvidas (e seus valores).

Dada uma BN, pode-se reescrever sua JPD da seguinte forma:

$$P(X_1, \dots, X_n) = \prod_{i=1}^n P(X_i | \text{pais}(X_i)) \quad (2.26)$$

Logo, cada entrada da JPD é representada pelo produto dos elementos apropriados nas CPTs da Rede *Bayesiana*.

### Intuição sobre Rede *Bayesiana*

Para ganhar intuição sobre o funcionamento de uma BN, e dado o tema envolvendo alarmes e segurança, utilizemos o mesmo exemplo retratado por Russel e Norvig [49] para ilustrar uma BN:

Você instalou um sistema de segurança residencial para sua mansão em Los Angeles. Ele é bastante preciso em detectar tentativas de invasão a sua casa. Contudo, ele eventualmente dispara com pequenos terremotos. Seus dois vizinhos, João e Maria, prometeram ligar sempre que ouvirem o alarme disparar. João sempre liga quando o alarme dispara, mas, por vezes, confunde o toque do telefone com o alarme, e liga de qualquer modo. Maria gosta de ouvir música em alto volume e, por vezes, não escuta o alarme tocar. Dadas as evidências de quem ligou, quer-se determinar a probabilidade de ter ocorrido uma invasão.

A topologia da BN está descrita na Figura 2.2, juntamente com as CPTs dos vértices associados. Pode-se observar que as arestas do grafo parecem traduzir relações de causa-efeito. Isto não é obrigatório em BNs; todavia, trata-se de uma prática comum que agrega semântica a sua estrutura.

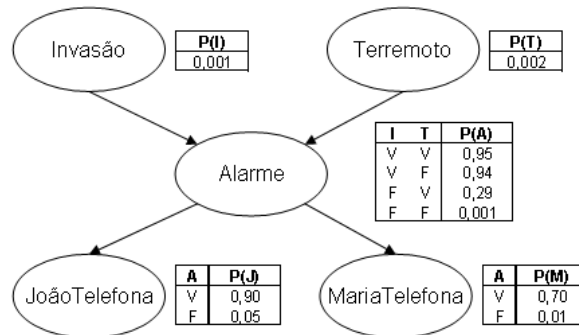


Figura 2.2: Exemplo de Rede *Bayesiana*

Através da equação 2.26 podemos calcular a probabilidade do alarme ter disparado devido a uma invasão — não um terremoto — e João e Maria terem telefonado:

$$P(J \wedge M \wedge A \wedge I \wedge \neg T) = P(J|A)P(M|A)P(A|I \wedge \neg T)P(I)P(\neg T) \quad (2.27)$$

Logo,

$$P(J \wedge M \wedge A \wedge I \wedge \neg T) = (0,9).(0,7).(0,94).(0,001).(0,998) = 0,0006 \quad (2.28)$$

### Inferência em Redes *Bayesianas*

Redes *Bayesianas* permitem o cálculo de inferências com base em evidências.

No mesmo exemplo da Figura 2.2, pode-se calcular a probabilidade de uma invasão à casa dado que João telefonou. Pelo teorema de Bayes:

$$P(I|J) = \frac{P(J|I)P(I)}{P(J)} \quad (2.29)$$

Definimos os *priors* ( $P(I)$  e  $P(J)$ ), ou seja, as probabilidades das variáveis sem qualquer evidência adicional.

Diretamente da CPT de I e T, extraímos:

$$P(I) = 0,001 \quad (2.30)$$

e

$$P(T) = 0,002 \quad (2.31)$$

(Somente em Los Angeles terremotos são mais frequentes que invasões a casas!)

Seguimos a topologia da rede e usamos os *priors* dos pais para calcular os filhos. Por este motivo, este algoritmo é também conhecido como de passagem de mensagem. Calculamos o *prior* de A:

$$P(A) = P(A|I, T)P(I, T) + P(A|\neg I, T)P(\neg I, T) + P(A|I, \neg T)P(I, \neg T) + P(A|\neg I, \neg T)P(\neg I, \neg T) \quad (2.32)$$

$$P(A) = (0,95).(0,001).(0,002) + (0,29).(0,999).(0,002) + (0,94).(0,001).(0,998) + (0,001).(0,999).(0,998) \quad (2.33)$$

$$P(A) = 0,0025 \quad (2.34)$$

Seguindo no mesmo algoritmo, calcula-se o *prior* de J:

$$P(J) = P(J|A)P(A) + P(J|\neg A)P(\neg A) \quad (2.35)$$

$$P(J) = (0,9).(0,0025) + (0,05).(0,9975) \quad (2.36)$$

$$P(J) = 0,0521 \quad (2.37)$$

Falta calcular  $P(J|I)$ :

$$P(J|I) = P(J|A)P(A|I) + P(J|\neg A)P(\neg A|I) \quad (2.38)$$

Calcular  $P(A|I)$  requer um pouco mais de cálculos:

$$P(A|I) = P(A|I, T)P(I, T) + P(A|I, \neg T)P(I, \neg T) \quad (2.39)$$



$$P(A|I) = (0,95).(1).(0,002) + (0,94).(1).(0,998) \quad (2.40)$$

$$P(A|I) = 0,94 \quad (2.41)$$

Aplicando a equação 2.41 em 2.38, tem-se:

$$P(J|I) = (0,9).(0,94) + (0,05).(0,06) \quad (2.42)$$

$$P(J|I) = 0,849 \quad (2.43)$$

Por fim, aplicando as equações 2.43, 2.30 e 2.37 em 2.29, tem-se:

$$P(I|J) = \frac{(0,849).(0,001)}{0,0521} \quad (2.44)$$

$$P(I|J) = 0,0163 \quad (2.45)$$

Ou seja, a probabilidade de ter ocorrido uma invasão, dado que João ligou, é de apenas 0,0163. Se Maria ligar em seguida, a probabilidade de ocorrência de invasão sobe para 0,29. De uma forma muito intuitiva, quanto mais evidências apontam para a invasão, maior a probabilidade da mesma ter ocorrido. Este será o tipo de inferência que utilizaremos em nossos classificadores, para determinar se, dado um conjunto de eventos, houve um ataque.

Diferentes tipos de inferências são possíveis a partir de uma BN, a saber:

1. Inferência Causal: da causa para o efeito.

Exemplo:  $P(JoaoTelefona|Invasao) = 0,86$ .

2. Inferência de Diagnóstico: do efeito para a causa.

Exemplo:  $P(Invasao|JoaoTelefona) = 0,0163$  (mesmo cenário que elaboramos anteriormente).

3. Inferência Intercausal: entre causas de um mesmo efeito.

Exemplo:  $P(Invasao|Alarme \wedge Terremoto) = 0.003$ .

4. Inferência Mixta: combinações dos anteriores.

Exemplo:  $P(Alarme|JoaoTelefona \wedge \neg Terremoto) = 0,03$ .

O cálculo de inferências dá-se através de algoritmos especializados, dos quais destacam-se: Algoritmo de Passagem de Mensagem de Pearl que explora as independências condicionais definidas pelo DAG através de um método de passagem de mensagens; e o SPI (*Symbolic Probabilistic Inference*) que aproxima a descoberta de uma caminho ótimo que calcula a distribuição marginal de interesse junto a JPD.

### Aprendizado em Redes *Bayesianas*

O processo de aprendizado em Redes *Bayesianas* implica na construção de um DAG e na definição dos CPTs dos vértices, de forma que a BN resultante represente consistentemente a JPD observável junto aos dados de treinamento.

O aprendizado pode ocorrer de quatro maneiras diferentes, variando segundo a existência ou não de uma topologia da rede, e segundo haver apenas variáveis plenamente observáveis ou haver variáveis ocultas:

1. Topologia conhecida, variáveis observáveis: o aprendizado restringe-se ao cálculo das CPTs de cada vértice. Este cálculo é feito a partir da observação estatística dos dados de treinamento.
2. Topologia conhecida, variáveis ocultas: o aprendizado incorpora as técnicas de descensão de gradiente existentes em redes neurais. Através dela pode-se ajustar os valores das CPTs aos dados observáveis.
3. Topologia desconhecida, variáveis observáveis: o aprendizado incorpora a definição de uma topologia e o cálculo das CPTs dos vértices. A construção da estrutura da rede baseia-se num processo de busca de topologias candidatas e seleção destas.
4. Topologia desconhecida, variáveis ocultas: o aprendizado incorpora a definição da topologia e das CPTs dos vértices. A construção da estrutura é computacionalmente intensiva e métodos de aproximação são indicados.

Os casos de topologia conhecida refletem, por exemplo, os cenários em que a estrutura da rede foi estabelecida por relações causais definidas por um especialista. Nestes casos, o aprendizado restringe-se à definição das CPTs dos vértices segundo a observação dos dados de entrada.

Os processos de definição de topologias tipicamente apóiam-se em buscas heurísticas no espaço de topologias candidatas.

### 2.2.3 Árvores de Decisão

*Árvores de decisão* são a implementação da estratégia de dividir-para-conquistar aplicada ao problema de aprendizado [62].

Uma árvore de decisão toma por entrada um objeto descrito por um conjunto de propriedades e retorna uma decisão do tipo Sim/Não. Outras saídas são possíveis, porém a configuração mais típica de uma árvore de decisão é de uma função *booleana*.

Cada nó da árvore de decisão corresponde a um teste aplicado sobre uma propriedade do objeto de entrada, sendo que as arestas que ligam aos outros nós são rotulados com os

possíveis resultados do teste. As folhas das árvores contêm os valores *booleanos* a serem retornados quando as mesmas são alcançadas.

Desta forma, cada nó provoca o particionamento do conjunto de entrada segundo o atributo testado.

A Figura 2.3 ilustra uma árvore de decisão aplicada à detecção de ataques.

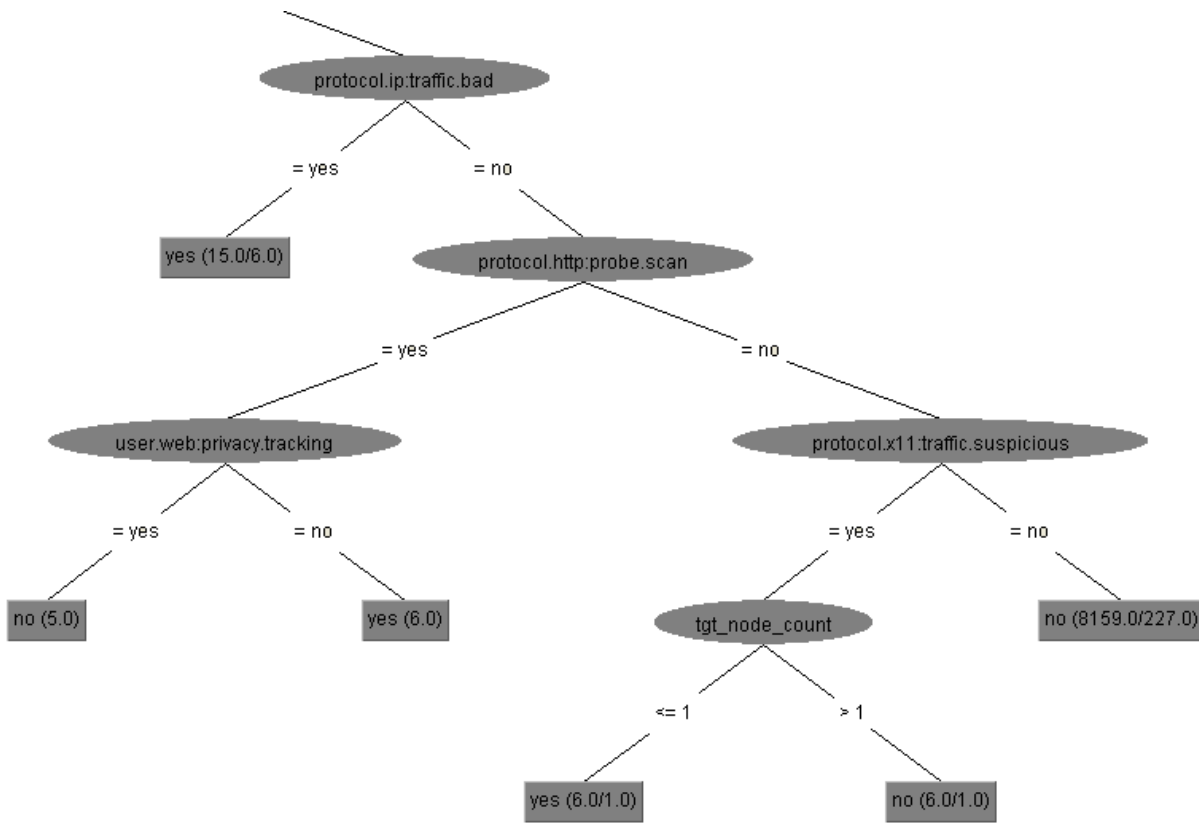


Figura 2.3: Fragmento de Árvore de Decisão de Ataques para Classificação em Segurança

Se um ataque apresenta eventos do tipo *protocol.http : probe.scan*, então deve-se verificar se há eventos do tipo *user.web : privacy.tracking*. Se houver, não se trata de um ataque, porém de um mecanismo de rastreamento de navegação, comum na Internet. Caso contrário, provavelmente trata-se de um ataque do tipo *Probe*.

### Aprendizado em Árvores de Decisão

O processo de aprendizado materializa-se na técnica usada na construção da árvore de decisão. A pergunta reside em qual propriedade do dado selecionar para estabelecer o

próximo nó da árvore.

Dado um nó  $i$ , seja  $y \in 1, \dots, m$  um dos valores possíveis para o mesmo, e  $f(i, j)$  a probabilidade de se obter o valor  $j$  no nó  $i$ . Portanto,  $f(i, j)$  corresponde à proporção dos registros associados ao nó  $i$  para os quais  $y = j$ .

Algoritmos, como o CART, utilizam a *impureza de Gini* (*Gini impurity*), para a seleção do próximo atributo a ser usado no particionamento dos dados de entrada:

$$I_G(i) = 1 - \sum_{j=1}^m f(i, j)^2, \quad (2.46)$$

Outros algoritmos, como ID3, C4.5 e C5.0 utilizam o conceito de ganho de informação (*information gain*):

$$I_E(i) = - \sum_{j=1}^m f(i, j) \log_2 f(i, j) \quad (2.47)$$

### 2.2.4 Coletâneas (*ensembles*)

Em aprendizado de máquina, uma coletânea (ou *ensemble*) consiste numa técnica que usa uma coleção de modelos para obter previsões melhores do que aquelas obtidas por um modelo isoladamente.

No caso de coletâneas de classificadores, alguns métodos são usados em sua construção:

- Diferentes subconjuntos de dados com a mesma técnica de aprendizado;
- Diferentes parâmetros de treinamento para uma mesma técnica de aprendizado;
- Diferentes técnicas de aprendizado.

Em todos estes casos, faz-se necessário um módulo de decisão que componha a resposta com base nas entradas providas por seus classificadores constituintes. Neste trabalho abordaremos o uso de uma coletânea composta por diferentes técnicas de aprendizado, como forma de cobrir padrões de ataques que alguma técnicas tenha dificuldade em detectar. Nosso módulo de decisão tomará a decisão conservadora. Se qualquer um dos classificadores constituintes indicar um ataque, a coletânea indicará o mesmo ataque.

### 2.2.5 Validação Cruzada (*Cross-validation*)

A validação cruzada consiste na prática de particionar os dados amostrais em subconjuntos. O primeiro subconjunto — denominado conjunto de treinamento — é utilizado para a realização de análises iniciais. Os demais subconjuntos — denominados conjuntos de

testes ou de validação —, potencialmente um único subconjunto, são usados para validar ou confirmar as análises iniciais.

Dentre as formas mais comuns de validação cruzada, destacam-se:

- *Holdout Validation*: dados amostrais são escolhidos aleatoriamente, sem repetições, para compor o conjunto de treinamento. Os demais dados constituem o conjunto de testes;
- *K-fold Cross-validation*: os dados amostrais são particionados em K subamostras. Destas, apenas uma subamostra compõe o subconjunto de testes. As demais K-1 subamostras são usadas como conjunto de treinamento. O processo de validação cruzada é executado K vezes (número de *folds*), com cada subamostra (das K existentes) sendo usada apenas uma vez como conjunto de validação.
- *Leave-one-out cross-validation*: neste caso uma única observação das amostras é usada para validação, sendo os demais dados usados como conjunto de treinamento. O processo é repetido de tal forma que cada amostra é utilizada uma vez como validação. Logo, o número de repetições é igual ao número de amostras.

### 2.2.6 Curva ROC

Em aprendizado de máquina, a curva ROC (*Receiver Operating Characteristic*) consiste na representação gráfica da *taxa de positivos verdadeiros* versus a *taxa de falsos positivos* de um classificador binário à medida que variamos seu limiar de decisão.

Neste trabalho empregamos uma curva ROC alternativa que é obtida pela variação da assimetria de custo no aprendizado do classificador, ao invés da alteração do limiar de decisão, conforme proposto por Bach et al [2]. Tal variação da curva ROC traduz-se num gráfico de conformação própria, onde não necessariamente se observa um gráfico monotonicamente crescente e suave, tradicional de uma curva ROC clássica.

Seja a matriz de confusão de um classificador binário que retorna Verdadeiro (T, *True*) ou Falso (F, *False*), representada na Tabela 2.2.

Tabela 2.2: Matriz de confusão para classificador binário

	T (Real)	F (Real)
T (Estimado)	TP	FP
F (Estimado)	FN	TN

Se o classificador estima que o resultado seja positivo (T), e na vida real ele realmente o é, tem-se um *positivo verdadeiro* (*true positive*, TP). Denomina-se taxa de positivos

verdadeiros (*True Positive Rate*, TPR) a razão entre TP e todos os positivos do mundo real:

$$TPR = \frac{TP}{TP + FN} \quad (2.48)$$

Onde FN são os *falsos negativos*, ou seja, aquelas ocorrências em que o classificador resultou em negativo (F), mas na vida real era positivo (P).

Por semelhante modo, se o classificador julga positivo (T), mas na vida real é negativo (N), tem-se um *falso positivo* (*false positive*, FP). Denomina-se taxa de falsos positivos (*False Positive Rate*, FPR) a razão entre FP e todos os negativos do mundo real:

$$FPR = \frac{FP}{FP + TN} \quad (2.49)$$

Onde TN são os *negativos verdadeiros*, ou seja, aquelas ocorrências que o classificador acertadamente classificou como negativas (N).

À medida que variamos o limiar de decisão (ou a assimetria de custos no aprendizado) e plotamos, obtemos um gráfico como o apresentado na Figura 2.4.

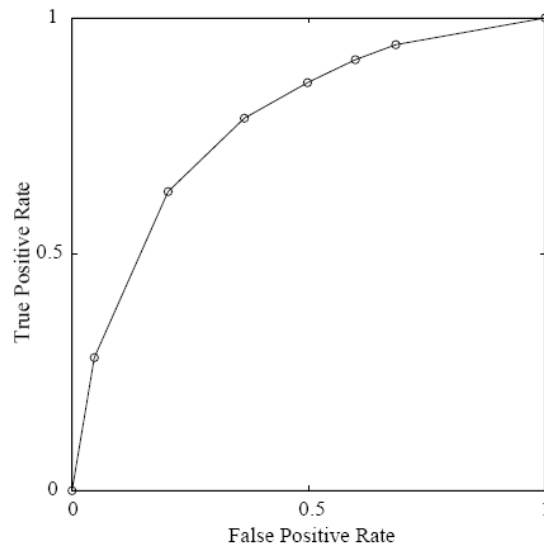


Figura 2.4: Exemplo de curva ROC

A melhor classificação possível é aquela que gera um ponto na extremidade superior esquerda do gráfico; o ponto (0,1). Esta é conhecida como a *classificação perfeita*.

A diagonal definida por  $x = y$  estabelece a chamada *linha da indiferença*, e corresponde à escolha aleatória dos valores de saída. Um classificador binário que desempenha na linha da indiferença é tão bom quanto o processo de atirar uma moeda.

Portanto, a performance de um classificador é tão melhor quanto maior for a área sob a curva ROC. Neste caso, a curva deve se estabelecer acima da linha da indiferença, preferencialmente aproximando-se do ponto da classificação perfeita.

### Intervalos de Confiança e *K-fold Cross-validation*

Curvas ROC são frequentemente derivadas de experimentos baseados em *K-fold Cross-validation*. Nestes casos, torna-se importante indicar o intervalo de confiança associado à curva, segundo as observações experimentais. Para tal, recorre-se à teoria de tratamento de resultados experimentais e propagação de erros.

Dada a matriz de confusão disposta em 2.2, e seja  $k$  o número de *folds* utilizados no experimento, pode-se, a partir dos experimentos, obter a matriz de observação experimental disposta em 2.3.

Tabela 2.3: Matriz de observação experimental para classificador em *K-fold Cross-validation*

Variável	Média	Desvio Padrão
TP	$\mu_{TP}$	$\sigma_{TP}$
FP	$\mu_{FP}$	$\sigma_{FP}$
FN	$\mu_{FN}$	$\sigma_{FN}$
TN	$\mu_{TN}$	$\sigma_{TN}$

Tem-se que  $\mu_{TP}$  é a média do valor de TP, dado por:

$$\mu_{TP} = \frac{\sum_i^k TP_i}{k} \quad (2.50)$$

onde  $TP_i$  corresponde ao valor de TP observado junto ao experimento associado ao *fold* de número  $i$ ;

e,  $\sigma_{TP}$  é o desvio padrão de TP, dado por:

$$\sigma_{TP} = \sqrt{\frac{(TP_i - \mu_{TP})^2}{k - 1}} \quad (2.51)$$

Desta feita, os valores de TPR e FPR a serem usados na curva ROC são calculados com base nas médias dos valores observados nos *folds*, ou seja

$$TPR = \frac{\mu_{TP}}{\mu_{TP} + \mu_{FN}} \quad (2.52)$$

e

$$FPR = \frac{\mu_{FP}}{\mu_{FP} + \mu_{TN}} \quad (2.53)$$

Tomando-se a fórmula geral de propagação de erros

$$\Delta f(x_1, x_2, \dots, x_n) = \sqrt{\sum_{i=1}^n \left( \frac{\partial f}{\partial x_i} \cdot \Delta x_i \right)^2} \quad (2.54)$$

e aplicando-a às equações 2.52 e 2.53, obtemos:

$$\Delta TPR = \sqrt{\left( \frac{\mu_{FN}}{(\mu_{FN} + \mu_{TP})^2} \cdot \sigma_{TP} \right)^2 + \left( \frac{\mu_{TP}}{(\mu_{TP} + \mu_{FN})^2} \cdot \sigma_{FN} \right)^2} \quad (2.55)$$

e,

$$\Delta FPR = \sqrt{\left( \frac{\mu_{TN}}{(\mu_{TN} + \mu_{FP})^2} \cdot \sigma_{FP} \right)^2 + \left( \frac{\mu_{FP}}{(\mu_{FP} + \mu_{TN})^2} \cdot \sigma_{TN} \right)^2} \quad (2.56)$$

Logo, para cada ponto da curva ROC, definido por TPR e FPR, deve-se considerar um intervalo de confiança definido por  $\Delta TPR$  e  $\Delta FPR$ , conforme as equações 2.55 e 2.56.



# Capítulo 3

## Trabalhos Correlatos

A pesquisa em sistemas de IDS pode ser dividida em duas linhas principais, segundo a estratégia de detecção utilizada: sistemas de detecção de mau-uso (Seção 3.1) e sistemas de detecção de anomalias (Seção 3.2). A combinação destas deu origem aos sistemas de detecção híbridos (Seção 3.3).

Dentro da pesquisa em detecção de intrusão, alguns temas despertam especial atenção. A detecção de ataques multiestágio (Seção 3.4) e o agrupamento de alertas de segurança (Seção 3.5) são dois destes temas.

Novas abordagens em detecção de intrusão (Seção 3.6) e novas técnicas de aprendizado de máquina aplicadas à segurança da informação (Seção 3.6.1) também constituem importante fonte consulta e inspiração para o desenvolvimento de nossa abordagem.

Nas próximas seções introduzimos alguns trabalhos ligados a estas linhas de pesquisa. Também discutimos suas influências neste trabalho.

### 3.1 Sistemas de Detecção de Mau-uso

Sistemas de detecção de mau-uso definem explicitamente as características de um uso indevido dos sistemas, como indicativos de potenciais ataques. Neste grupo estão sistemas marcados pelo determinismo de seu comportamento. Em geral, tais sistemas estão baseados em assinaturas de ataques representadas de alguma forma.

A detecção de mau-uso dá-se em geral através da codificação de regras acerca do que é aceitável e da determinação de eventos que não se encaixam nestas regras.

Dentre os sistemas de detecção de mau-uso encontra-se, por exemplo, a proposta de desenvolvimento de linguagens de especificação como a BMSL (*Behavioral Monitoring Specification Language*) [3]. Através da BMSL é possível especificar propriedades para detecção de intrusão. Suas correlações, definidas através de padrões, são compiladas e o código gerado é executado dentro do *kernel* do sistema operacional. Desta feita, sua

arquitetura demonstra boa performance, dando suporte à implementação de NIDSs e HIDSs que atuam diretamente sobre os pacotes e chamadas ao sistema, respectivamente.

Uma abordagem similar é apresentada pela STAT Tool Suite [60]. Neste, cenários de possíveis ataques são modelados através de uma linguagem de descrição de máquina de estados denominada *STATL*. Tais cenários são construídos de forma gráfica num editor chamado *Scenario Editor*, o qual gera arquivos *STATL*. Estes arquivos são lidos pelo *Parser* que gera um formato interno que é usado por outros dois módulos: *Analyzer* e *Translator*. Através do *Analyzer* é possível fazer modificações aos cenários. Finalmente, o *Translator* traduz os cenários (potencialmente modificados) e gera código C, que, compilado com bibliotecas do STAT, produz um executável que contém tanto a arquitetura da plataforma quanto as correlações de identificação de intrusão.

Com base nesta infraestrutura, foram gerados três diferentes detectores de intrusão:

- USTAT: HIDS para sistemas Unix, baseado no processamento de dados de auditoria providos pelo *Basic Security Module* (BSM) da Sun Microsystems;
- WinSTAT: HIDS para sistemas Windows (originalmente Windows NT), baseado no processamento de eventos do Sistema Operacional;
- NetSTAT: NIDS baseado no processamento de pacotes que trafegam no segmento de rede onde o módulo está instalado.

Boyer et al [4] também sugerem a definição de uma linguagem, SADL (*Security Assessment Declarative Language*), e do sistema Stellar para a construção de cenários e a análise de riscos em segurança. Seu sistema integra eventos oriundos de diferentes origens para a avaliação de prioridades de cenários.

A linguagem SADL permite a construção de bases de regras cujos objetivos são aumentar a prioridade de cenários de ataques considerados sérios em detrimento de outros cenários julgados de menor impacto. As regras são escritas com predicados de lógicas que assemelham-se a declarações em SQL. A primeira regra que é avaliada positivamente atribui a prioridade ao cenário em questão. O sistema é desenvolvido de forma a otimizar o processamento desta base de regras como forma de processar grandes volumes de eventos.

Em linhas gerais, sistemas baseados em detecção de mau-uso tendem a ser mais precisos na identificação de ataques específicos, para os quais estão programados, gerando, desta feita, menos falsos positivos. Por outro lado, apresentam dificuldades na detecção de novos padrões de ataques para os quais não havia sido previamente expostos ou programados. Assemelham-se ao paradigma do sistema de anti-vírus, onde uma assinatura de detecção de um vírus é desenvolvida segundo a observância do próprio vírus. Desta feita, torna-se difícil extrapolar o funcionamento do IDS para novos tipos de ataque, da mesma forma como o anti-vírus tem dificuldades em detectar novos tipos de vírus.

## 3.2 Sistemas de Detecção de Anomalias

Outra linha de desenvolvimento de IDSs baseia-se na detecção de anomalias. Ela busca determinar aquilo que foge do comum, do comportamento considerado normal de uma infraestrutura ou sistema.

No trabalho apresentado por Lee et al em [30], propõe-se o uso de técnicas e conceitos de mineração de dados para o desenvolvimento de sistemas de IDS. Os autores observam que “os IDSs tradicionais (baseados em mau-uso) falham na generalização com vistas à detecção de novos ataques ou ataques para os quais não há assinaturas”. Segundo Lee et al, o uso de técnicas de mineração de dados permite a generalização a partir de ataques conhecidos e comportamento normal como forma de detectar novos ataques. Todavia, os autores reconhecem que os seguintes desafios impõem-se no uso desta tecnologia para detecção de intrusões:

- Acurácia: número de falsos alarmes costuma ser mais alto que aquele gerado por IDSs baseados em assinaturas;
- Eficiência: custo do treinamento e avaliação costuma ser intensivo computacionalmente (embora algumas técnicas mais recentes, como SVM, possam reduzir tal custo);
- Usabilidade: estes sistemas requerem grandes quantidades de dados para treinamento e são mais complexos que sistemas de IDS tradicionais.

Para contemplar os desafios da acurácia, os autores reforçam a importância da extração de atributos, da detecção de anomalias e do uso conjunto de técnicas de detecção baseadas em mau-uso e anomalias. Já para tratar a eficiência, propõem modelagens sensíveis a custo (regras e estruturas que otimizam o processamento) e o uso de sistemas distribuídos. Por fim, para a questão de usabilidade, os autores mesclam o aprendizado adaptativo (com o uso de *ensembles* de classificadores) e técnicas de aprendizado não-supervisionado. A arquitetura proposta neste trabalho engloba o uso de IDMEF para representação de alertas, sensores e detectores para sua geração, e um par casado composto por um *data warehouse* e um gerador adaptativo de modelos.

Trabalho semelhante é desenvolvido por Tandon e Chan [55] para processamento de chamadas ao sistema operacional. Nele, um HIDS é desenvolvido a partir de técnicas de aprendizado de regras extraídas do processamento do *log* de um sistema BSM. Os autores avaliam seus resultados com os dados do desafio DARPA, a exemplo do que fazemos no presente trabalho.

### 3.3 Sistemas de Detecção Híbridos

Abordagens híbridas, que mesclam a detecção de mau-uso e de anomalias, sugeridas por Lee et al, têm sido propostas por vários autores. No projeto europeu Safeguard [10], propõe-se uma arquitetura para coleta e processamento de eventos de segurança com vistas a resolver os seguintes problemas junto ao mercado de telecomunicações: (1) redução do número de eventos a serem tratados por operadores; (2) redução de falsos alarmes; (3) coleta e correlação de eventos de diferentes fontes; e (4) avaliação da “saúde” da rede e predição para evitar colapsos do serviço.

Os autores apresentam um sequenciamento de passos (chamados por eles de métodos) do seguinte tipo: detecção (uso de detectores padrões de mercado como Snort e syslog), filtragem (estática e adaptativa), normalização, agregação e correlação. A camada de correlação faz uso de técnicas de aprendizado de máquina como redes neurais e k-vizinhos. Trata-se de um *framework* que mescla técnicas determinísticas com detecção de anomalias.

Também nesta linha de pesquisa temos o sistema EMERALD [44], onde uma arquitetura baseada em blocos (componentes) é usada para abrigar diferentes sistemas de correlação, inferência e raciocínio, cujas instanciações podem variar de máquinas de inferência baseadas em assinaturas de ataques a análises estatísticas usando redes *bayesianas*.

O uso de abordagens estatísticas deu origem a um dos trabalhos mais influentes na pesquisa de sistemas de IDS, o qual desenvolve o conceito de correlação probabilística de alertas de segurança [58]. Este trabalho, desenvolvido na SRI (Stanford Research Institute) International, propõe a criação de uma hierarquia de correlações. Alertas de segurança — detectados por sensores espalhados pela rede — são fundidos em *meta-alertas* (grupos de alertas que possuem características comuns e que provavelmente estão relacionados ao mesmo ataque) em três níveis distintos de processamento: (1) *intra-sensor* ou *threads sintéticas* — alertas provenientes de um único sensor são fundidos segundo um limiar de expectativa de similaridade entre a classe do ataque e endereços de rede envolvidos, tanto da origem quanto do alvo; (2) *incidentes de segurança* — alertas que pertencem à mesma classe de ataque e que confluem no mesmo destino de ataque são fundidos a despeito do sensor que os originou; e (3) *ataques correlacionados* — ataques multi-estágios são detectados pelo relaxamento da expectativa mínima de similaridade entre as classes de ataques dos alertas. A similaridade entre eventos é calculada com base em uma matriz de similaridade de classes de incidentes e no decaimento temporal da similaridade. A classificação de incidentes consiste, também, numa primeira tentativa de se promover uma taxonomia para eventos de segurança. Todavia, tal taxonomia é bastante restrita, e prevê a divisão dos eventos nas seguintes classes apenas: *invalid*, *privilege violation*, *user subversion*, *denial of service*, *probe*, *access violation*, *integrity violation*, *system environment corruption*, *user environment corruption*, *asset distress*,

*suspicious usage, connection violation, binary subversion e action logged.*

Embora rudimentar, sem tal taxonomia não seria possível desenvolver a matriz de similaridade, nem gerar meta-alertas. Outra contribuição importante do trabalho de Valdes e Skinner [58] reside justamente neste agrupamento de alertas em meta-alertas, o que implica numa razão de redução do número de elementos a serem analisados por operadores. Este ponto será importante dentro da abordagem de nosso trabalho.

### 3.4 Detecção de Ataques Multiestágio

A detecção de ataques multi-estágios, introduzida por Valdes, também é contemplada por Mathew et al [37], os quais utilizam uma linguagem de grafos — SGIF — para descrever tais ataques. Sua implementação é equivalente à de um sistema de detecção de mau-uso, porém apresenta suporte para a fusão de nós de cenários de ataque. Tal fusão provê uma representação para ataques que comprometem um elemento da rede e usam-no em ataques subsequentes. Trata-se de um sistema de tempo-real baseado em assinaturas de ataques. Sua limitação reside justamente em não generalizar bem para novos padrões de ataque, conforme observado pelos próprios autores.

Ning et al [45] utilizam predicados de lógica de primeira ordem para representar pré-requisitos e consequências de estágios de ataques. Pré-requisitos são as condições necessárias para que um ataque seja bem sucedido. Consequências são os possíveis resultados de um ataque. Quando um ataque anterior serve de preparação para um ataque subsequente, suas consequências devem satisfazer os pré-requisitos do segundo. Desta feita, alertas que representam diferentes estágios de um mesmo ataque são fundidos em um *hiperalerta*, algo equivalente ao meta-alerta de Valdes e Skinner[58].

A exemplo do que ocorre com Valdes e Skinner [58], a proposta de Ning et al [45] também traz o benefício de reduzir o número de alertas (no caso, hiperalertas) que precisam ser analisados pelo time de segurança.

### 3.5 Agrupamento de Alertas de Segurança

A redução de número de alertas com base em técnicas de agrupamento (*clustering*) também é proposta por Julisch [27]. Neste trabalho é notado que “*algumas poucas dúzias de causas raízes persistentes, em geral, são responsáveis por 90% dos alarmes gerados por IDSs*”. O autor propõe o uso de atributos generalizados para representar agrupamentos de alarmes. Cada alarme generalizado representa um grupo de alarmes que compartilham uma causa raiz comum. A detecção dos alarmes generalizados é feita *offline* usando-se técnicas de mineração de dados e heurísticas.

A identificação de causas raízes de tempestades de alertas permite ao grupo de segurança focar na fonte dos alertas (como equipamentos mal configurados, vulnerabilidades frequentemente exploradas, sensores defeituosos, entre outros), e, como consequência, reduzir o número de alarmes a ser observado futuramente. O princípio de buscar causas raízes de incidentes de segurança é análogo ao processo de gerência de problemas definido pelo itSMf na sua biblioteca de melhores práticas em TI, o ITIL [26].

Mesclar técnicas de fusão de alertas e técnicas estatísticas *Bayesianas* é proposto por Burroughs et al [6]. Neste, faz-se uso de BMHT (*Bayesian Multiple Hypothesis Tracking*) para identificar o comportamento de um atacante à medida que alertas são gerados pelos sistemas de IDS distribuídos numa rede. A modelagem dos ataques é realizada de forma a reduzir o número de atributos necessários para descrever um ataque e, portanto, aumentar a performance do sistema. Em especial, os seguintes atributos são utilizados: endereço de origem, endereço de destino, serviço sob ataque, tipo de ataque e data e hora do ataque, configurando, portanto, um subconjunto dos campos presentes do IDMEF. Segundo os autores, a fusão de dados oriundos de diferentes sensores associada ao rastreamento continuado dos alvos dos ataques resulta no aumento do nível de consciência situacional (*situational awareness*) do sistema de segurança.

Uma abordagem semelhante é proposta por Sabata [50]. Neste trabalho, eventos de sensores são indicativos de ações de ataques. Tais ações de ataques são agrupados em estágios de ataques que, por sua vez, são agrupados em ataques. Ataques, estágios de ataques e ações de ataques são representados por fragmentos de redes *bayesianas*. Estes fragmentos são unidos em uma rede *bayesiana* por meio de um processo de *reasoning*. Como resultado, observa-se a redução do número de alertas gerados, estabelecendo altas razões de redução em casos reais. Em seu trabalho subsequente, Sabata e Ornes [51] dão continuidade ao trabalho com fragmentos de redes *bayesianas*. Neste caso, o modelo proposto em seu trabalho anterior é refinado arquiteturalmente e testado contra bases de dados reais, dentre elas, o desafio DARPA.

## 3.6 Novas abordagens

Trabalhos recentes têm apresentado novas visões para o processamento de alertas de segurança. Martimiano e Moreira [35] sugerem a definição de uma ontologia de segurança — OntoSec — para o gerenciamento de incidentes de segurança de rede.

Segundo os autores, o conceito principal reside na habilidade de modelar relações do tipo um *Agente* executa um *Ataque*, podendo causar um *Incidente de Segurança*; tipicamente usa uma *Ferramenta* e explora uma *Vulnerabilidade*, adquirindo, então, *Acesso* ao sistema. *Incidentes* agem sobre *Ativos* e possuem uma *Pré-Condição* — tipicamente associada com uma vulnerabilidade. Incidentes de segurança implicam em *Consequências*,

e podem preceder ou suceder outros incidentes.

Tal abordagem possui elementos análogos aos propostos por Ning et al [45], especialmente no tocante às Pré-Condições e Consequências, que, no caso de Nin, são denominados de Pré-Requisitos e Consequências.

Liu e Zang [31] propõem o uso de Teoria dos Jogos [42] para modelar e inferir intenções, objetivos e estratégias de atacantes. Através deste formalismo, buscam dois tipos de inferência: (a) inferir estratégias de ataque (que mais provavelmente serão usadas pelo atacante); e, (b) inferir as intenções e objetivos de um atacante.

Trata-se de uma linha de pesquisa recente cuja aplicação prática ainda é incerta. O comportamento do modelo sofre forte impacto de falsos positivos gerados por IDSs, e do atraso entre a ocorrência de um evento e sua detecção pelo sistema. Todavia, um dos potenciais benefícios da tecnologia reside na otimização da postura defensiva de uma rede como forma de aumentar sua resiliência.

Abordagens distribuídas e colaborativas para detecção de intrusão, conforme proposto por Locasto et al [32], também visam compreender as intenções do atacante, modelar seu comportamento e obter uma visão global das atividades de ataque à rede. Os autores propõem um sistema composto por dois componentes:

- *Worminator*: sistema que extrai informações de fluxos de alertas e as codificam em *Bloom Filters*;
- *Whirlpool*: sistema para coordenação de correlações entre pares de nós.

*Bloom Filters* são estruturas de dados compactas para representação de alertas de segurança. Consistem de um vetor compacto de bits, associado a uma poderosa função de *hash* que suporta duas operações: inserção e verificação. Seu uso apresenta-se como uma alternativa ao uso do padrão IDMEF dentro de um contexto de busca por performance do processamento distribuído, aportando três benefícios:

- Compactação: *bloom filters* conseguem representar alertas com muito menos bits que sua contrapartida em XML dentro do padrão IDMEF;
- Resiliência: mesmo que se configure *bloom filters* de tamanho inferior ao necessário, a consequência desta atitude implica em falsos positivos, mas nunca em falsos negativos;
- Segurança: o fato da estrutura de dados ser binária e compacta permite que corporações troquem informações na forma de *bloom filters* sem que seus dados internos (endereços IP, por exemplo) sejam expostos nesta operação.

Os sistemas distribuídos publicam seu interesse em tipos de eventos através do uso de *watchlists*. Quando um evento, codificado na forma de um *bloom filter*, passa pelo critério de uma *watchlist*, o mesmo é enviado para o sistema requerente.

### 3.6.1 Novas técnicas de aprendizado de máquina aplicadas

Dentro das novas técnicas em aprendizado de máquina, algumas têm começado a ser aplicadas aos desafios de correlação de eventos de segurança.

Mukkamala et al [41] aplicam um *ensemble* de SVM e ANN (rede neural) sobre um subconjunto da base DARPA. Seu trabalho atesta a melhor performance de SVM sobre ANN, e enfoca a escolha dos eventos relevantes na detecção de ataques em detrimento de técnicas de fusão ou agrupamento como forma de obter consciência situacional.

Outra iniciativa que visa o uso de SVM na construção de um NIDS é o proposto por Tian et al [56]. Nele o autor usa SVMs para o processamento do tráfego de uma rede ou servidor, ao invés de processar eventos oriundos de outros sensores. Sua base de dados não é pública, consistindo de tráfego de uma subrede da Universidade de Aeronáutica e Astronáutica de Nanjing.

A questão da redução de falsos positivos em detecção de intrusão é abordada por Ohta et al [46]. Os autores observam que o uso de técnicas de aprendizado de máquina para construção de IDSs gera muitos falsos positivos, diminuindo sua eficácia. Dentro de um cenário que usa árvores de decisão como classificadores, este trabalho busca a escolha dos atributos que devem ser usados para a redução dos falsos positivos.

A avaliação de sua proposta é feita com base no desafio DARPA. Segundo os pesquisadores, “se o classificador desempenhar com sucesso na base DARPA, ele desempenhará com sucesso em intrusões reais ou ataques na Internet”.



# Capítulo 4

## Abordagem Proposta

Processar e correlacionar alertas de segurança é fundamental para o estabelecimento de uma gestão eficaz da segurança da informação. Para tal, buscamos uma abordagem que proveja a consciência situacional, ou seja, a capacidade de analisar os alertas de segurança dentro de um contexto amplo e holístico. Queremos analisar o maior número de evidências possíveis que nos permita julgar se a segurança da informação está em risco, ou não.

Visamos, portanto, integrar eventos de diferentes fontes. De uma forma bastante intuitiva, se mais fontes geram alertas acerca de um potencial incidente, maior a probabilidade deste incidente ser real. Logo, necessitamos coletar eventos de fontes distribuídas e, de alguma forma, provermos mecanismos para um processamento homogêneo dos mesmos.

A consciência situacional parece depender da capacidade de criarmos cenários de ataques. Tais cenários são frutos da contraposição de diferentes alertas, de, potencialmente, distintas fontes. Este processo de agrupamento, ou fusão, agrega alertas em entidades de maior hierarquia, de maior carga semântica. Tais elementos permitem que a classificação entre ataques reais e falsos alarmes seja realizada com base em dados mais completos, significativos.

Objetivamos também criar uma abordagem de fácil instaciação e alto desempenho. Necessitamos que uma implementação desta abordagem execute em tempo-real, indicando os ataques à medida que os mesmos ocorrem, propiciando à equipe de segurança o tempo para implementar as contra-medidas necessárias.

### 4.1 Uma Abordagem em Camadas Hierárquicas

Face a estes requisitos, nossa abordagem é composta por um processo de três camadas (Coleta e Normalização, Fusão, e Classificação), as quais são representadas na Figura 4.1.

Cada camada desempenha uma função do modelo, abstraindo complexidades à camada imediatamente superior.

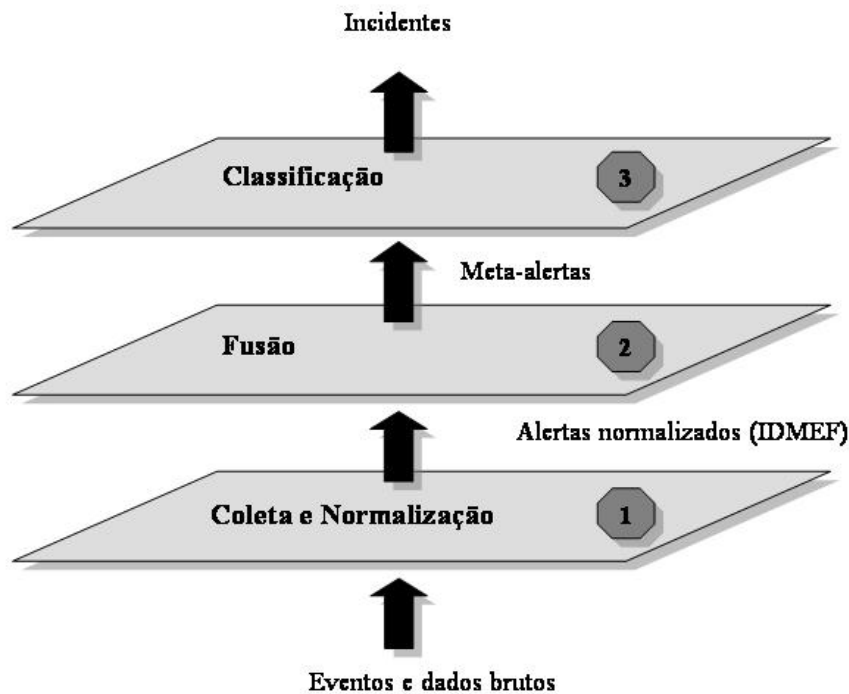


Figura 4.1: Abordagem em camadas hierárquicas

A camada de Coleta e Normalização é responsável por coletar — e em alguns casos, sintetizar também — alertas de segurança e normalizá-los (traduzir para uma representação única, independente de sua origem) para tratamento pelas camadas superiores.

O aspecto da coleta torna esta camada intrinsecamente ligada à fonte dos alertas. Logo, tem-se um cenário de instanciamento desta camada para cada fonte distinta de alertas de segurança.

A função de normalização toma os alertas gerados ou coletados e os traduz para um formato comum, no caso o IDMEF (ver Seção 2.1.2). Isto permite que particularidades das fontes de alertas sejam abstraídas das camadas subsequentes.

Todavia, o uso de IDMEF não esgota as questões de normalização. Em especial, a questão de classificação dos alertas de segurança requer uma extensão sobre o modelo proposto. Isto ocorre porque o tipo de um evento IDMEF está definido com base em identificações de vulnerabilidades, as quais carecem de um critério de classificação e/ou semântica mais fortes. Desta feita, propõe-se também a extensão da notação IDMEF para suporte ao modelo.

Os alertas vindos da camada de coleta e normalização, via de regra, são tão numerosos que excedem a capacidade de análise dos times de segurança. Não obstante, muitos destes alertas possuem interrelacionamentos que transcrevem um cenário mais abrangente do ataque, o qual é de grande relevância para as técnicas de classificação da última camada do modelo.

A camada de fusão, ou agrupamento, identifica alertas que podem ser agrupados segundo as informações transcritas em atributos normalizados. Cada grupo de alertas é representado numa estrutura denominada meta-alerta, a qual traduz os atributos comuns e características relevantes do grupo em questão.

O processo de agrupamento pode ocorrer de forma recursiva, onde meta-alertas são agrupados em outros meta-alertas de hierarquia superior. Tal estratégia permite, por exemplo, a detecção de ataques multi-estágio, de forma análoga à proposta por Ning et al [45].

Meta-alertas provenientes da camada de fusão são analisados e classificados entre incidentes (ataques reais) ou alarmes falsos, na camada de classificação. O processo de seleção ocorre a partir da análise de atributos dos meta-alertas, segundo o modelo de abstração definido pela arquitetura proposta.

## 4.2 Coleta e Normalização

### 4.2.1 Coleta

Conforme mencionado anteriormente, a coleta de alertas está acoplada à fonte destes eventos. Fontes possíveis de eventos são sistemas de IDS (HIDS ou NIDS), *firewalls*, *logs* de sistemas, chamadas ao sistema operacional, entre outros.

Elementos de coleta podem ser divididos em dois tipos básicos: passivos e ativos. Sistemas passivos obtêm as informações de eventos a partir da simples observação de evidências, acontecimentos ou estados de elementos. Exemplos são a recepção de *traps* SNMP ou o *parsing* de *logs* de sistemas. Por outro lado, sistemas ativos interagem com o ambiente para poderem determinar ocorrências que devam ser traduzidas em alertas. Exemplos de coletas ativas incluem *polling* SNMP, *probing* ICMP, entre outros.

A coleta também se distingue quanto à cardinalidade dos eventos gerados. Em alguns cenários, para cada elemento observado (entrada de *log*, *trap* SNMP, etc) corresponde um alerta a ser gerado. Em outros cenários, a camada de coleta é responsável pela filtragem, ou seja, a determinação de quais elementos observados devem ser traduzidos em alertas. Exemplos deste último são *logs* de chamadas ao sistema operacional. Nem toda entrada implica num problema. Em verdade, a maior parte das entradas representa atividades normais e corriqueiras do sistema. Entradas que divergem dos parâmetros da normalidade

são detectadas pela camada de coleta e transformadas em alertas.

Como suporte ao escopo deste trabalho, foram desenvolvidos softwares de coleta para processamento de eventos de um NIDS (Snort), chamadas ao sistema operacional Solaris (BSM), alertas do sistema Windows, entradas de *Syslog*, e *logs* de acesso a um sistema web.

Destes, apenas a coleta de eventos do Snort implica numa cardinalidade 1 x 1; ou seja, para cada alarme do Snort corresponde um evento em nosso sistema. Todos os outros elementos de coleta implementados exigiram o uso de mecanismos de filtragem, os quais detalharemos em seções subsequentes .

### 4.2.2 Normalização

Uma vez coletados os eventos, os mesmos são representados de uma forma padronizada, normalizada. Propomos o uso de uma representação baseada no IDMEF, ajustada para o contexto de nossa abordagem. A primeira adaptação reside em não utilizar XML como formato de persistência dos alertas. Em seu lugar, optou-se por uma versão linearizada de um subconjunto dos atributos introduzidos pelo IDMEF. A linearização privilegia a persistência dos alertas em bases de dados relacionais e otimiza seu processamento. Ela permite que estruturas de dados do tipo registro (*struct* em C ou *record* em Pascal) possam representar seu conteúdo sem a necessidade de *parsing* XML. A este subconjunto dos atributos do IDMEF propusemos uma extensão que contempla conceitos como taxonomia dos tipos de alertas e priorização de alertas. A Tabela 4.1 apresenta a estrutura utilizada para normalização dos alertas de segurança.

O campo *msg\_id* é a chave primária da estrutura. Consiste num contador com valor sempre crescente. Cada alerta tem um único *msg\_id* que o identifica.

Todos os campos começados por *<analyzer\_>* representam o sensor (ou analisador, no jargão do IDMEF) que deu origem ao evento. Exemplos de sensores são Snort, *firewall*, etc.

Todo sensor tem um identificador, *analyzer\_id*, único e um nome, *analyzer\_name*. Sensores constituem-se, tipicamente, de elementos de rede dispersos numa geografia; logo, informações como seu endereço de rede, *analyzer\_address*, e sua localização física, *analyzer\_location*, complementam sua descrição.

Todo alerta possui um registro de relógio, *timestamp*, que identifica o momento de sua ocorrência. O preenchimento deste campo é bastante sensível e requer algum nível de sincronia entre os relógios dos vários sensores envolvidos. Seu conteúdo é crítico para o funcionamento de correlações baseadas em janelas de ocorrência, comuns na detecção de intrusões. Neste trabalho, assumiremos que os relógios dos sensores estão sincronizados através de algum mecanismo, como protocolos específicos para tal.

Os campos prefixados por *<src\_>* referem-se às informações de origem dos alertas, tipicamente associados a potenciais atacantes à infraestrutura. Eles podem subdividir-se, de forma não exclusiva, em nós de rede (*node*), usuários (*user*) e processos (*proc*). Um alerta pode, portanto, ter a identificação do nó de onde partiu o ataque, do usuário envolvido e/ou do processo participante da ação. Sistemas de NIDS terão maior capacidade de determinar o nó de origem, e dificuldades em indicar usuários e procesos. Sistemas de HIDS terão o comportamento inverso. Outrossim, há casos em que todas as informações são determináveis para a origem do alerta.

Um nó de origem é identificado por seu nome, *src\_node\_name*, e seu endereço de rede, *src\_node\_address*. No caso do sensor conseguir identificar que o endereço de rede foi forjado, numa tentativa de iludir o sistema sob ataque, o campo *src\_node\_spoofed* pode ser usado para representar tal fato.

Usuários e processos são representados por seus nomes, *src\_user\_name* e *src\_proc\_name*, e identificadores, *src\_user\_id* e *src\_proc\_id*, respectivamente. Para o casos de processos, indica-se ainda o caminho do arquivo executável do processo, *src\_proc\_path*.

A representação do alvo ou destino, indicados pelo prefixo *<tgt\_>*, dos alertas (tipicamente vítimas dos ataques) ocorre de forma análoga à de sua origem. Neste caso, os alvos podem subdividir-se em: nós (*node*), serviços (*service* ou *svc*), processos (*process* ou *proc*), URLs, CGIs e arquivos (*file*).

Tabela 4.1: Registro de Alerta

<b>Atributo</b>	<b>Tipo</b>
msg_id	Integer
analyzer_id	String
analyzer_name	String
analyzer_addr	String
analyzer_location	String
create_time	Date
src_node_name	String
src_node_addr	String
src_node_port	Integer
src_node_spoofed	String
src_user_name	String
src_user_id	String
src_proc_name	String
src_proc_path	String
src_proc_id	String
tgt_node_name	String
tgt_node_addr	String
tgt_node_decoy	String
tgt_port_list	String
tgt_svc_name	String
tgt_user_name	String
tgt_user_id	String
tgt_proc_name	String
tgt_proc_path	String
tgt_proc_id	String
tgt_url	String
tgt_http_method	String
tgt_cgi	String
tgt_file_name	String
tgt_file_path	String
tgt_file_access	String
raw_text	LongText
ext_class	String
ext_taxonomy	String
ext_priority	Integer
ext_src_node_addr_type	String
ext_tgt_node_addr_type	String
meta_alert_id	Integer

Nós de destino são retratados por seu nome e endereço, *tgt\_node\_name* e *tgt\_node\_addr*, respectivamente. O atributo *tgt\_node\_decoy* auxilia a identificar quando o destino é um chamariz para atrair atacantes, como um pote de mel (ver Seção 2.1.2), por exemplo. Adicionalmente, pode-se identificar a porta ou conjunto de portas atingida(o) pelo alerta em questão. Tal informação estará contida no campo *tgt\_port\_list*.

Quando o alvo trata-se de usuário ou processo do sistema, sua representação segue extritamente aquela usada para origens de alerta.

Ataques remotos podem ocorrer contra serviços (*tgt\_srv\_name*), URLs (*tgt\_url*) ou CGIs (*tgt\_cgi*). Para URLs e CGIs pode-se ainda representar o método HTTP usado em sua invocação, *tgt\_http\_method*.

Um alerta pode ainda representar um ataque a um arquivo do sistema. Neste caso, os campos *tgt\_file\_name*, *tgt\_file\_path* e *tgt\_file\_access* representam, respectivamente, o nome, caminho de diretório e permissões de acesso ao arquivo em questão.

Para fins de auditoria e suporte a análise forense, decidiu-se manter um campo que armazenasse a origem do alerta em sua forma mais pura. A este campo denominamos *raw\_text*.

As extensões propostas ao modelo IDMEF estão identificadas pelo prefixo *<ext\_>*.

O campo *ext\_class* contém uma indicação da classe de ataque ao qual o alerta pode ser associado. Valores possíveis são: Negação de Serviço (Dos), Exploração (Probe), Acesso Remoto (R2L), Acesso Superusuário (U2R), e Roubo de Dados (Data). Mais adiante detalharemos como este campo pode auxiliar na implementação de mecanismos de fusão de eventos, em conjunto com os demais campos normalizados.

Os campos *ext\_src\_node\_addr\_type* e *ext\_tgt\_node\_addr\_type* representam a localização relativa dos nós de origem e de destino com relação à rede monitorada. Nós podem ser externos, internos ou pertencentes à DMZ (zona desmilitarizada) de uma rede.

A prioridade de um evento pode ser indicada pelo sensor ao sistema através do campo *ext\_priority*. Convencionou-se usar a mesma escala introduzida pelo Snort, variando de 1 a 3, sendo 1 o mais prioritário, e 3, o menos.

O campo *ext\_taxonomy* consiste, sem dúvida, na maior contribuição deste trabalho ao modelo iniciado pelo IDMEF. Seu conteúdo merece a seção seguinte.

### 4.2.3 Taxonomia

O uso de taxonomias em sistemas de detecção de intrusão é objeto de discussão há mais de uma década. Desde os trabalhos de Debar et al [13] e as propostas para o CIDF que culminaram no modelo do IDMEF, busca-se um formato comum para representação e identificação de intrusões.

O modelo do IDMEF endereçou parte substancial deste problema. Todavia, ele não

provê semântica aos tipos de eventos, conforme observado por Martimiano [34].

Necessitamos de uma forma de identificar tipos de alertas de uma forma simples e direta; uma forma que dê suporte a atividades de fusão e classificação de alertas e meta-alertas.

A partir da observação de alertas de diferentes fontes — NIDSs (Snort), *logs* de chamadas ao sistema operacional (BSM), *logs* de acesso a sistemas web, eventos de segurança do sistema Windows, e eventos de *firewalls* — propomos uma representação para tipos de alertas, a qual é dividida em três grandes grupos:

1. Uma ação que ocorre contra um objeto ( $\langle object \rangle : \langle action \rangle$ ).
2. Uma condição à qual um objeto está submetido ( $\langle object \rangle : \langle condition \rangle$ ).
3. Uma suspeita acerca do estado de um objeto ( $\langle object \rangle : \langle state \rangle$ ).

Relacionar-se a um  $\langle objeto \rangle$  é o ponto comum dos três grupos da representação.

Objetos são representados através de uma hierarquia multinível. No primeiro nível, um objeto consiste em um dos itens: sistema, elemento de rede, protocolo, usuário, arquivo ou segurança. Cada um destes itens subdivide-se em conceitos mais granulares. Um sistema pode consistir de um sistema operacional, um banco de dados, um servidor web, uma aplicação, uma ferramenta ou um serviço. Estes podem ainda ser subdivididos em níveis menores, aumentando a especificidade da identificação. A Figura 4.2 apresenta a hierarquia dos tipos de objetos tratados em nossos experimentos segundo esta taxonomia.

Uma *ação* assemelha-se a um método do objeto ao qual está associado. Desta feita, um alerta que representa o desligamento (*shutdown*) de um sistema operacional seria representado como: *system.os:shutdown*.

A exemplo do que ocorre com os elementos, ações também são representadas de forma hierárquica. Os níveis mais baixos da hierarquia de ações qualificam-nas. Qualificações podem ser advérbios associados à ação, o resultado de uma ação, ou equivalente. Para representar uma tentativa frustrada de login por erro de senha junto a um usuário de administração do sistema, usaríamos: *user.admin:login.fail.pwd*.



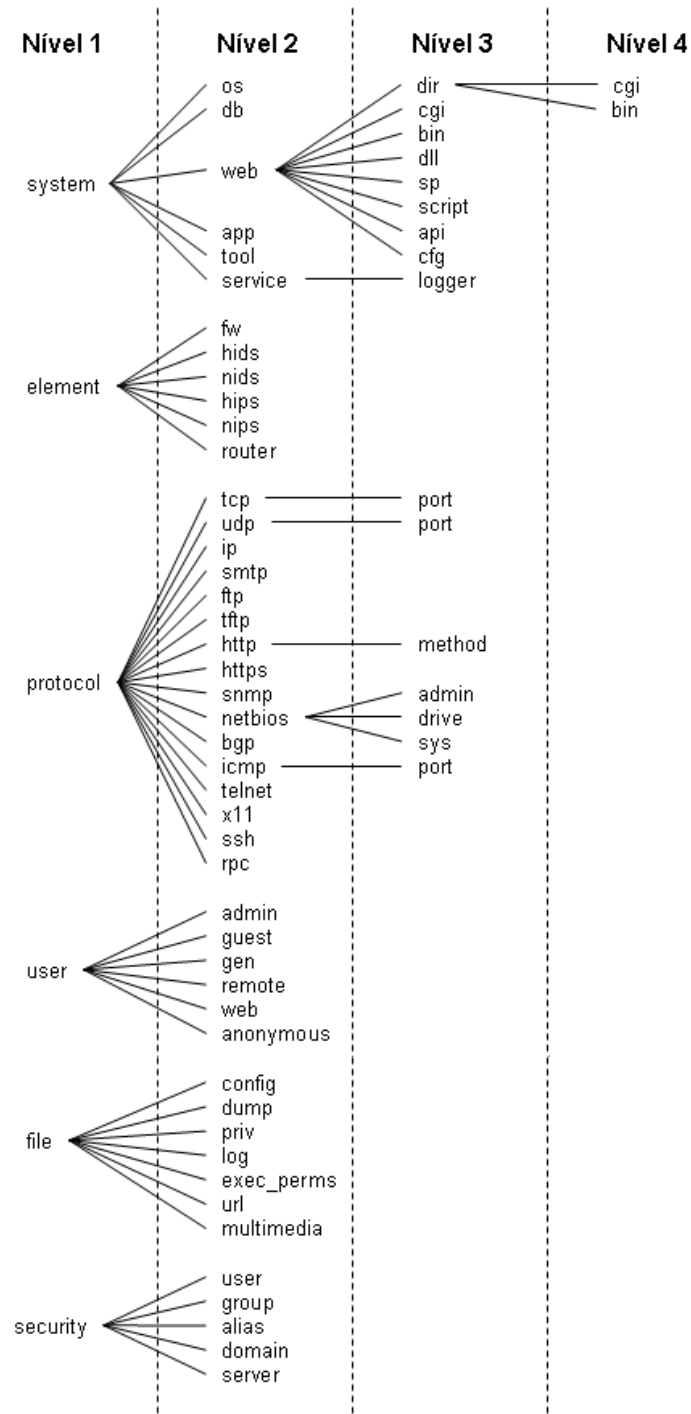


Figura 4.2: Hierarquia de Objetos

A Figura 4.3 apresenta os tipos de ações observáveis junto aos experimentos realizados.

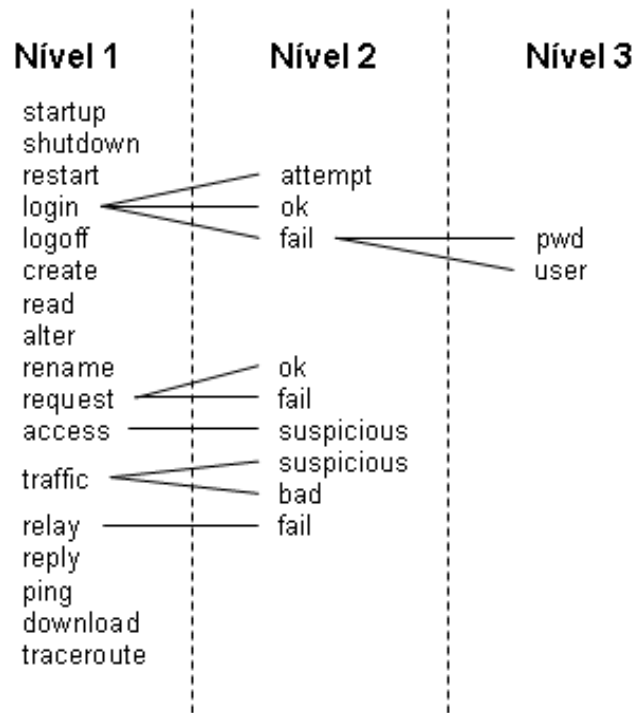


Figura 4.3: Hierarquia de Ações

O uso de hierarquias para representar ações permite que correlações atuem em diferentes níveis de refinamento das ações. Desta feita, se para uma correlação é indiferente o motivo pelo qual um *login* de administrador falhou, as seguintes taxonomias de eventos seriam equivalentes: *user.admin:login.fail*, *user.admin:login.fail.pwd* e *user.admin:login.fail.user*. O uso de níveis hierárquicos contempla esta questão de uma forma elegante, na medida que a correlação pode simplesmente desprezar o Nível 3 das ações, reduzindo todas as opções anteriores a uma única: *user.admin:login.fail*.

Ações são usadas tipicamente para representar sentenças que comporíamos com as seguintes construções gramaticais:

- Sujeito + Verbo Intransitivo (+ Adjunto Adverbial);
- Verbo Transitivo Direto + Objeto Direto (+ Adjunto Adverbial).

Deve-se notar que a distinção entre os dois casos é dependente da transitividade do verbo. Todavia, esta ambiguidade não impacta o uso prático da taxonomia, e não foi

necessária uma ação específica para sua resolução, a qual tornaria, invariavelmente, a notação mais complexa, sem ganhos reais para o domínio da solução.

Alguns alertas não representam ações executadas por elementos da infraestrutura, porém estados nos quais tais elementos se encontram.

Estes estados, ou *condições*, são representados tipicamente por expressões gramaticais da forma: Sujeito + Verbo de Ligação + Complemento Nominal.

Um exemplo deste cenário seria expresar que uma determinada porta TCP está inacessível. Segundo nossa taxonomia, teríamos: *protocol.tcp.port:unreachable*. Outro exemplo de condição seria a existência de um *log* de erro presente junto a um sistema web: *system.web:log.error*.

A Figura 4.4 apresenta os tipos de condições observados em nossos experimentos.

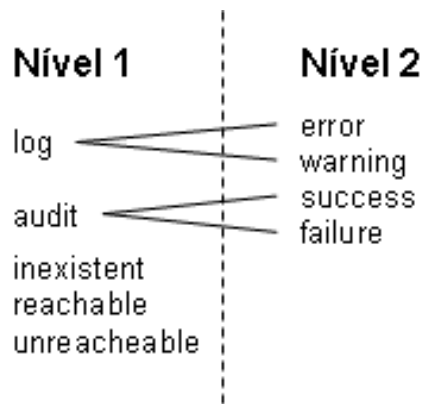


Figura 4.4: Hierarquia de Condições

Existe, todavia, uma categoria especial de alertas de segurança que representa a *suspeita* de um sensor acerca de uma atividade. Ela diverge dos dois casos anteriores na medida que não representa um fato concreto, real, senão uma dedução acerca de fatos existentes. Dada a incerteza inerentemente associada, suspeitas perfazem um conjunto a parte dentro da taxonomia proposta.

Se um NIDS detecta uma ação associada a uma tentativa de uso de uma porta dos fundos (*backdoor*), este alerta seria identificado da seguinte forma: *system.os:malware.backdoor.attempt*. Uma suspeita de injeção de código junto a um CGI seria representado por *system.web.cgi:exploit.code\_injection*.

A Figura 4.5 apresenta os tipos de suspeitas observadas em nossos experimentos.

A identificação de alertas que representam suspeitas apresenta uma particularidade acerca do objeto associado. O objeto representa a origem na qual a suspeita foi detectada.

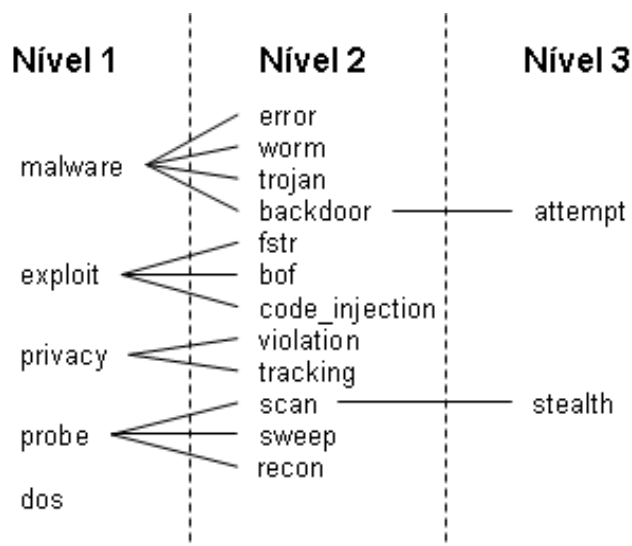


Figura 4.5: Hierarquia de Suspeitas

Objetos são associados a ações, condições ou suspeitas. Todos estes estão representados em estrutura multiníveis. As possibilidades de representação multiplicam-se. Todavia, dado um domínio de eventos bem definido, todos estes conjuntos terão tamanhos finitos, e o produto cartesiano de objetos por qualquer um dos outros conjuntos será também finito. Esta propriedade será de especial interesse na representação de cenários junto a meta-alertas e no melhor uso de classificadores, conforme veremos a seguir.

### 4.3 Agrupamento ou Fusão

Nesta camada, os alertas provenientes da camada de coleta e normalização são agrupados em meta-alertas. Propõe-se, para tal, estender os conceitos introduzidos por Valdes e Skinner [58], adicionando-se outros atributos na busca de verossimilhanças entre alertas.

A abordagem consiste em definir os critérios a serem usados no agrupamento com base na classificação (segundo a extensão proposta do modelo IDMEF) de cada evento constituinte.

O processo de agrupamento ou fusão deve preencher alguns requisitos básicos, associados à busca de performance em sua execução e sua utilidade prática, quais sejam:

1. Eficiência: operar sobre um subconjunto mínimo de atributos e limitar a busca de meta-alertas no tempo.

2. Inteligibilidade: produzir meta-alertas que sejam inteligíveis por seres humanos, ou seja, simples de compreender.
3. Coerência: adequar os critérios de fusão aos tipos de ataques previstos no modelo.

Tais requisitos impõem-se uma vez que o algoritmo de fusão deve executar em tempo real, agrupando alertas em meta-alertas à medida que os mesmos chegam ao sistema. Desta feita, a eficiência em seu processamento é chave para sua eficácia como gestor de eventos de segurança.

Também é imperativo que os meta-alertas gerados sejam de fácil compreensão. Afinal, são estes meta-alertas (pós classificação) que serão apresentados aos usuários como resultados do processo de correlação de alertas. Todo o esforço seria em vão se não houvesse coesão perceptível entre os alertas agrupados em um meta-alerta.

Após muito debate acerca da linha a ser seguida na fusão de meta-alertas, variando de um espectro que abrangia métodos totalmente determinísticos até técnicas de agrupamento (*clustering*) incremental, optou-se — com base nos requisitos apresentados — por um algoritmo determinístico simples.

Um algoritmo determinístico, minimalista e eficiente, parece ser a melhor resposta na busca do atendimento aos requisitos acima. Alertas são, portanto, fundidos a meta-alertas segundo critérios claros e objetivos, inteligíveis ao usuário comum.

Com base nestes conceitos, propomos uma estrutura de Meta-Alerta que possa suportar o algoritmo de fusão, a fase subsequente de classificação e a interpretação pelo usuário numa camada futura de apresentação. Tal estrutura encontra-se descrita na Tabela 4.2.

A exemplo do que ocorre com a estrutura para representação de alertas, propomos uma versão linearizada para representar meta-alertas, com viés para otimização do processamento da fusão e subsequente classificação.

Contadores (*analyzer\_count*, *src\_node\_count*, *tgt\_port\_count*, entre outros) são mantidos atualizados à medida que alertas são fundidos aos meta-alertas. Alguns deles são usados posteriormente na camada de classificação como discriminadores do meta-alerta. Eles têm especial importância em ataques do tipo *Probe* ou *DoS*, pois o número de alertas, e elementos de origem e destino envolvidos, são especialmente relevantes na detecção destes ataques.

Outro campo especialmente relevante é *ext\_max\_priority*. Meta-alertas que contêm alertas de alta prioridade podem se sobressair na multidão através deste atributo.

Sem sombra de dúvida, o campo mais relevante e inovador na estrutura é *alert\_taxonomy\_set*. Trata-se de um vetor de bits, onde cada bit representa uma das possíveis taxonomias de tipos de alertas suportados. Se um (ou mais) alerta(s) de um tipo está(ão) presente(s) no meta-alerta, o bit correspondente é ligado. Caso contrário, ele fica desligado.

Tabela 4.2: Registro de Meta-Alerta

Atributo	Tipo	Descrição
meta_alert_id	Integer	Identificação única de um meta-alerta
analyzer_id_list	String	Conjunto de analisadores que geraram os alertas fundidos neste meta-alerta
analyzer_count	Integer	Número de analisadores que detectaram os alertas
init_time	Date	Data e hora do alerta mais antigo
end_time	Date	Data e hora do alerta mais recente
time_window_len	Integer	Diferença de tempo (em segundos) entre end_time e init_time
src_network_addr	String	Endereço base da rede que origina os alertas
src_node_addr_list	LongText	Lista de endereços de origem dos alertas
src_node_count	Integer	Número de diferentes endereços de origem dos alertas
src_user_id_list	String	Lista de identificações de usuários de origem dos alertas
src_user_count	Integer	Número de usuários de origem
src_proc_id_list	String	Lista de identificações de processos de origem dos alertas
src_proc_count	Integer	Número de processos de origem
tgt_node_addr_list	LongText	Lista de endereços-alvos dos alertas
tgt_node_count	Integer	Número de diferentes endereços-alvos
tgt_port_list	LongText	Lista de portas-alvos dos alertas
tgt_port_count	Integer	Número de diferentes portas-alvos
tgt_user_id_list	String	Lista de identificações de usuários-alvos dos alertas
tgt_user_count	Integer	Número de usuários-alvos
tgt_proc_id_list	String	Lista de identificações de processos-alvos dos alertas
tgt_proc_count	Integer	Número de processos-alvos
tgt_file_name_list	LongText	Lista dos nomes dos arquivos-alvos
tgt_file_count	Integer	Número de arquivos-alvos
ext_class	String	Classe que define o tipo de ataque em questão
ext_max_priority	Integer	Maior prioridade dentre os alertas que compõem o meta-alerta
alert_count	Integer	Número de alertas que compõem este meta-alerta
alert_taxonomy_set	Bit Array	Deteção da presença ou ausência de cada um dos tipos de alertas possíveis na taxonomia

Podemos entender este campo como o conjunto de todas as pistas de um crime. Cenários de potenciais ataques, ou crimes, representados por um meta-alerta, apresentam um subconjunto destas pistas. Quanto mais estas pistas assemelham-se às de casos anteriores conhecidos, maior nossa suspeita de que efetivamente trata-se de um ataque.

O vetor de bits permite o uso de técnicas de produto interno para verificação da similaridade entre dois cenários de ataque na fase de classificação, dando especial suporte ao processo de classificação. Mas este é um assunto para a Seção 4.4.

O preenchimento da estrutura de meta-alerta deve ocorrer como consequência natural da fusão dos alertas no mesmo.

Da união dos requisitos para fusão e da estrutura de um meta-alerta, propomos o seguinte algoritmo de fusão determinística de alertas em meta alertas:

**Require:**  $j\_cache > 0$ ,  $j\_dos > 0$ ,  $j\_probe > 0$ ,  $j\_r2l > 0$ ,  $j\_u2r > 0$

**Require:**  $a \in \text{Alertas}$ ,  $M$  set of Meta-Alerts

**Ensure:**  $M$  updated with alert  $a$

1:  $f \leftarrow \emptyset$

```

2: for all  $m$  in  $M$  do
3:   if  $a.ext\_class \neq m.ext\_class$  then
4:     nop
5:   else if  $a.ext\_class = DoS$  then
6:     if ( $a.src\_node\_addr \in m.src\_network\_addr$ ) and
       ( $a.tgt\_node\_addr \in m.tgt\_node\_addr\_list$ ) and
       ( $a.create\_time \leq (m.end\_time + j\_dos)$ ) and
       ( $a.create\_time \geq (m.init\_time - j\_dos)$ ) then
7:        $f \leftarrow m$ 
8:     end if
9:   else if  $a.ext\_class = Probe$  then
10:    if ( $a.src\_node\_addr \in m.src\_node\_addr\_list$ ) and
      ( $a.create\_time \leq (m.end\_time + j\_probe)$ ) and
      ( $a.create\_time \geq (m.init\_time - j\_probe)$ ) then
11:       $f \leftarrow m$ 
12:    end if
13:  else if  $a.ext\_class = R2L$  then
14:    if ( $a.src\_node\_addr \in m.src\_node\_addr\_list$ ) and
      ( $a.tgt\_node\_addr \in m.tgt\_node\_addr\_list$ ) and
      ( $a.create\_time \leq (m.end\_time + j\_r2l)$ ) and
      ( $a.create\_time \geq (m.init\_time - j\_r2l)$ ) then
15:       $f \leftarrow m$ 
16:    end if
17:  else if  $a.ext\_class = U2R$  then
18:    if ( $a.src\_node\_addr \in m.src\_node\_addr\_list$ ) and
      ( $a.tgt\_node\_addr \in m.tgt\_node\_addr\_list$ ) and
      ( $a.create\_time \leq (m.end\_time + j\_u2r)$ ) and
      ( $a.create\_time \geq (m.init\_time - j\_u2r)$ ) then
19:       $f \leftarrow m$ 
20:    end if
21:  end if
22: end for
23: if  $f = \emptyset$  then
24:    $n = \text{new Meta-Alert}$ 
25:   init  $n$  with data from  $a$ 
26:    $a.meta\_alert = n$ 
27:    $M = M \cup \{n\}$ 
28: else

```

```

29:   a.meta_alert = f
30:   Update f with data from a
31: end if

```

A primeira linha de requeridos (*Require*) define as janelas de tempo usadas para fusões das diferentes classes de alertas. Alertas são agrupados apenas se seus *timestamps* estiverem separados por menos de uma janela de tempo. Tratam-se de parâmetros de configuração do algoritmo que definem os tamanhos das janelas deslizantes para associação dos alertas. A definição destes deve ser cuidadosa para garantir um bom desempenho do algoritmo. Janelas deslizantes muito grandes podem provocar o agrupamento de alertas não afins. Por outro lado, janelas muito estreitas podem ocasionar o fracionamento de eventos de um único ataque em múltiplos meta-alertas.

Quando um novo alerta é coletado e enviado para fusão, o algoritmo itera sobre o *cache* de meta-alertas (linha 2) buscando candidatos para a fusão.

Uma condição para que a fusão ocorra é a coincidência entre a classe do alerta e a classe do respectivo meta-alerta, conforme observa-se na linha 3.

Caso trate-se de um evento de negação de serviço (DoS), linhas 5 e 6, então o destino do ataque deve ser o mesmo para todos os alertas. Aceitamos, todavia, que o endereço de origem possa variar dentro de uma subrede. Para fins deste trabalho, aceitamos subredes de classe C (255.255.255.0) como fontes de alertas de um mesmo ataque distribuído (DDoS). Logo, um ataque de negação de serviço distribuída constitui um caso especial da classe de ataques de negação de serviço.

Em se tratando de um evento de exploração (Probe), então o destino pode variar livremente, porém a origem deve ser sempre a mesma (linha 10). Afinal, a exploração impede o atacante de usar técnicas de *spoofing* para mascarar seu endereço de origem, sob a penalidade de não receber de volta os pacotes de retorno da exploração.

Eventos referentes a possíveis ataques remotos (R2L) apenas são fundidos se origem e destino coincidirem com o respectivo meta-alerta (linha 14). O mesmo vale para ataques de ganho de privilégios (U2R), conforme observável na linha 18.

Não é difícil concluir que o campo de Meta-Aleta *src\_node\_addr\_list* contém apenas um endereço em ataques do Probe, R2L e U2R; de forma análoga, o campo *tgt\_node\_addr\_list* contém apenas um endereço em ataques do tipo DoS, R2L e U2R.

Em todos os casos, deve-se respeitar janelas de tempo de fusão que variam conforme os tipos de ataque. Neste sentido, ataques de Probe podem ter janelas de fusão mais longas, permitindo a detecção de ataques mais cadenciados, lentos. Em contrapartida, ataques de DoS costumam ser mais incisivos, curtos; pedem, portanto, janelas mais estreitas para fusão de seus alertas. Estas condições são observáveis nas linhas 6, 10, 14 e 18.

Quando um meta-alerta para fusão não é encontrado para um dado alerta, este dá



início a um novo meta-alerta que contém apenas a si próprio (linhas 24 a 27).

Uma vez identificado um meta-alerta que preencha as pré-condições para fusão, o alerta é associado ao respectivo meta-alerta, conforme as linhas de 29 a 30.

Com este algoritmo — simples e intuitivo — é bastante fácil compreender porque determinados alertas foram fundidos ou não. Ademais, o processamento pode ser otimizado com uso de estruturas de *cache* de meta-alertas cujas entradas são gradualmente invalidadas segundo critérios temporais.

A simplicidade tem, porém, seu custo. Atributos mais refinados de fusão como sessões, usuários, processos, e arquivos, estão de fora deste algoritmo. A razão é bastante simples. Nem todos os alertas contém tais informações, ao passo que endereços de rede estão presentes em todos os eventos — mesmo aqueles oriundos de sistemas de HIDS (neste caso, pode-se sempre assumir como valores *default* para endereços de origem e destino aqueles do próprio sistema monitorado em questão).

Outra limitação reside na incapacidade de prover a fusão entre alertas de ataques híbridos (que mesclam dois ou mais tipos de ataques) ou ataques multiestágios (que tomam um sistema e usam-no de “trampolim” para atacar outros sistema). Deixamos esta funcionalidade como uma sugestão de pesquisa em trabalhos futuros.

Não obstante, este algoritmo simples apresentou excelentes resultados na taxa de redução de dados em informação (DIR, *Data to Information Ratio*), conforme observaremos mais adiante. Além disto, a estrutura de meta-alerta resultante provê importante suporte para a camada subsequente do modelo: a classificação.

## 4.4 Classificação

Conforme nossa analogia inicial (ver Seção 1.2), após a Fusão, as folhas (alertas) estão devidamente associadas a suas árvores (meta-alertas). Cabe-nos agora separar as “boas árvores” das “más árvores”.

Separar ataques reais de falsos alarmes é alcançado com técnicas de aprendizado de máquina. A idéia reside em usar meta-alertas como caracterizações de cenários de ataques, permitindo a comparação de novos cenários com outros já conhecidos.

Propomos o uso de modernas técnicas de classificação baseadas em aprendizado assistido. Nosso interesse reside em entender melhor como técnicas novas como SVMs, redes *bayesianas*, árvores de decisão, e coletâneas destas, comportam-se dentro de nossa camada de classificação.

A escolha dos atributos dos meta-alertas a serem usados por estas técnicas é fator chave para o sucesso. Deve-se evitar o uso excessivo de atributos, com vistas a prevenir fenômenos como *overfitting*. Neste momento, as definições das taxonomias (vetor de bits) e a estrutura de um meta-evento confluem na escolha de uma boa caracterização de um

cenário de ataque.

Da análise dos atributos dos meta-eventos, e após algumas seções de experimentações, reduzimos a caracterização a ser usada na classificação para aquela presente na Tabela 4.3.

Tabela 4.3: Atributos para Classificação

Atributo	Descrição
alert_count	Número de alertas que constituem o meta-alerta
ext_max_priority	Maior prioridade dentre os alertas pertencentes ao meta-alerta
tgt_node_count	Número de nós-alvos do ataque
tgt_port_count	Número de portas-alvos envolvidas no ataque
analyzer_id_list	Indica se alerta foi identificado por um NIDS, HIDS Unix e/ou HIDS Windows
ext_class	Classe do ataque ao qual o meta-alert está associado
alert_type_set	Vetor de bits indicando os tipos de alertas que constituem o meta-alerta

Alguns atributos exercem papel importante na separação entre ataques e falsos alarmes. O número de alertas, nós e portas envolvidos auxiliam na detecção de ataques do tipo exploração (*Probe*) e negação de serviço (DoS).

O campo de prioridade auxilia a separar meta-alertas que contenham alertas de baixa prioridades daqueles que já foram julgados como relevantes pela camada de sensores, coleta e normalização.

A lista de tipos de sensores envolvidos (*analyzer\_id\_list*) reforça a noção de consciência situacional. Quanto mais sensores diferentes tiverem percebido um ataque, maior a probabilidade dele ser real.

Todavia, nenhum dos critérios anteriores deve ser tomado como absoluto. Todos são relativos ao tipo de ataque em questão, o qual representamos pelo atributo *ext\_class*, e por outras características do ataque, como os tipos de alertas envolvidos.

O vetor de bits de tipos de alertas merece uma menção especial. Ele é o melhor caracterizador do cenário de ataque. Conforme já mencionado, ele descreve o conjunto das evidências do cenário sendo avaliado; as provas de um eventual crime. As semelhanças entre o conjunto de evidências de um crime e outro é a base do critério de verossimilhança que utilizamos no aprendizado supervisionado.

A idéia de trabalhar com um vetor de bits representando as entradas da taxonomia adveio da analogia com o problema da comparação de documentos de texto. O vetor de bits desempenha o mesmo papel do *bag of words* daquele. Desta feita, o produto interno dos vetores resulta no grau de proximidade entre dois documentos, ou, em nosso caso, dois meta-alertas.

O conceito é especialmente adequado para classificadores como SVMs. Dada a propriedade do *kernel* no qual se apoiam, o produto interno usado pela SVM reflete nativamente a noção de distância (ou semelhança) entre conjuntos de meta-alertas. Note-se que

cada bit do vetor de bits é traduzido numa dimensão (atributo ou *feature*) do dado de entrada da SVM. Tal fato, aliado à propriedade do *kernel* de projetar o espaço de entrada num espaço de maior dimensão, onde, tipicamente, combinações dos atributos de entrada são verificáveis, torna o uso do conceito do *bag of words* bastante adequado à solução do problema de classificação.

Como veremos na parte experimental, os atributos de classificação escolhidos também apresentaram empiricamente grande aderência a técnicas *bayesianas*.

Também é relevante notar que o uso de uma taxonomia padrão e identificação de cenários (a partir de conjuntos da mesma) torna a proposta mais flexível e resiliente na detecção de novos ataques. Obviamente isto apenas é possível mediante o uso da granularidade correta na definição da taxonomia. Se ela for muito específica, o classificador termina por decorar os cenários de ataque conhecidos, e obter resultados ruins na extrapolação para novos cenários. Por outro lado, se ela for muito genérica, a capacidade de classificação é impactada pela falta de informação para tomada de decisão.

Conforme mostraremos na Seção 5, o modelo sugerido, amparado pela taxonomia proposta, trouxe bons resultados junto às bases de teste utilizadas, corroborando os elementos-chaves de nossa abordagem: normalização, taxonomia, agrupamento, descrição de cenários e classificação.

# Capítulo 5

## Implementação e Experimentos

O modelo proposto necessita ser validado, com o objetivo de identificarmos seus pontos fortes e suas limitações.

Precisamos de uma implementação desta abordagem e de massas de dados que possam exercitar os vários conceitos do modelo.

Como massas de dados utilizamos duas fontes: (a) desafio DARPA [22], o padrão *de facto* para testes de sistema de intrusão; e (b) para complementação desta base, lançamos mão dos dados publicados por outro desafio, o *Scan of the Month* [47]. Duas bases de dados; dois conjuntos de características distintas.

O modelo foi implementado através de um conjunto de classes escritas em linguagem *Perl*. Estas classes executam todas as camadas descritas no modelo: coleta e normalização, agrupamento e classificação. Além destas, outras funcionalidades foram implementadas, como a anotação dos dados de aprendizado e o suporte para extração dos resultados e estatísticas.

Nas próximas seções detalhamos as bases de dados e os detalhes da implementação do modelo usados em nossos experimentos. Por fim, apresentamos os resultados dos experimentos realizados junto a estas bases.

### 5.1 DARPA

O desafio DARPA foi desenvolvido dentro de uma parceria entre o próprio DARPA (*Defense Advanced Research Project Agency*) e o MIT Lincoln Labs para testar sistemas de detecção de intrusão [22].

Trata-se do primeiro padrão de testes para avaliação de sistemas de intrusão. A base de dados disponibilizada pelo desafio em conjunto com as compilações dos resultados obtidos por seus contendores representam importante ferramenta de comparação (*benchmark*) para pesquisas na área.

Os patrocinadores do desafio simularam a rede (LAN) de uma base da Força Aérea Americana, com diferentes sistemas e um acesso simulado à Internet.

A simulação contém cinco máquinas que são alvos de ataques (Solaris 2.5.1, SunOS 4.1.4, Linux Red Hat 5.0 Kernel 2.0.32, Windows NT 4.0 Build 1381 SP1 e Windows 98), um *sniffer* (software que acessa todos os pacotes da rede — podendo armazená-los para análises futuras), atacantes internos e máquinas virtuais. Adicionalmente, simula-se uma rede externa composta por *sniffer*, atacantes externos e máquinas virtuais.

A Figura 5.1 apresenta uma visão topológica da rede simulada usada no desafio.

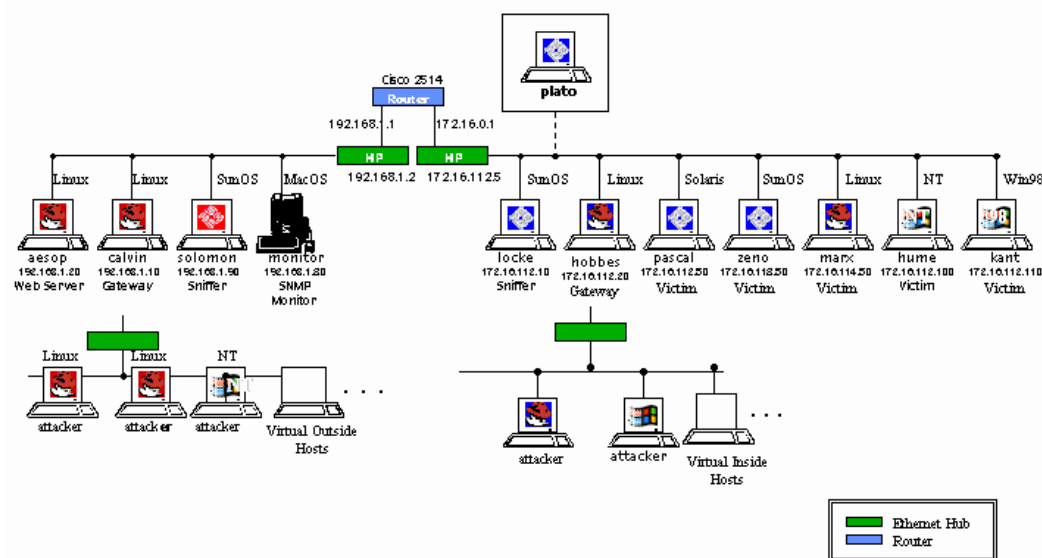


Figura 5.1: Topologia da Rede do desafio DARPA

Além do tráfego de rede e operações de sistemas usuais, ataques foram perpetrados ao longo de cinco semanas de experimento. A lista dos ataques e suas frequências podem ser encontradas no apêndice A.

Os dados disponibilizados por este desafio são: arquivos *tcpdump* (arquivos contendo o *dump* — registro completo — de todos os pacotes TCP/IP trafegados na rede), arquivos BSM (*Basic Security Module*) — sistema que registra *logs* de auditorias —, arquivos de alertas Windows NT e outros dados de auditoria.

No desafio original, eram fornecidos aos participantes os dados das cinco semanas de simulação, bem como a anotação dos ataques ocorridos nas três primeiras semanas. O objetivo dos participantes consistia em determinar quais ataques ocorriam nas duas

últimas semanas de dados fornecidos.

Os ataques eram classificados em cinco grupos: negação de serviço (DoS), acesso remoto (R2L), escalada de privilégios (U2R), exploração (*Probe*) e roubo de dados (*Data*).

Considera-se uma detecção bem sucedida aquela que corretamente aponta o alvo do ataque e a hora da ocorrência, com uma tolerância de um minuto antes do início e após o término do ataque. Preferencialmente, o sistema deveria também determinar a classificação do ataque em um dos grupos supracitados.

Quatro em cada cinco trabalhos práticos na área de detecção de intrusão utilizam o desafio DARPA como massa de testes, dentre eles: Ohta et al [46], Mahoney e Chan [33], Bowen et al [3], Mukkamala et al [41], Faraoun e Boukelif [16] [15], Tandon e Chan [55], Lee et al [30], Mukkamala e Sung [40], Valdes e Skinner [59], e Sabata e Ornes [51].

## 5.2 SotM

Apesar da grande difusão da base de dados do desafio DARPA, algumas críticas têm sido feitas à mesma: (a) o fato se referir a um ambiente simulado, e não estritamente real; e, (b) sua idade e, por conseguinte, a atualidade dos ataques envolvidos.

Para complementar nosso experimento incluímos uma segunda base de testes com características distintas; complementares.

O desafio *Scan of the Month*, promovido pelo *The Honeynet Project* (ver Seção 2.1.2), objetiva “auxiliar a comunidade a desenvolver habilidades de análise forense para decodificar ataques reais”. Seu conteúdo baseia-se em ataques reais, capturados junto a redes constituídas de potes de mel. Os *logs* e informações destes ataques são publicados em seu website [24] como exercícios que devem ser respondidos pela comunidade.

Dentre os vários desafios existentes escolhemos o de número 34, por configurar-se como o mais complexo e completo, e que melhor se adequa a nossos objetivos de testes.

As evidências disponibilizadas incluem:

- *Logs* de servidor web Apache;
- Entradas de arquivo *Syslog*;
- *Logs* de NIDS Snort;
- *Logs* de *firewall* ipTables.

Utilizamos as três melhores respostas — providas por Matt Richard and Michael Ligh, Chris Kronberg, e Andrew — e, através da interpretação destas, anotamos os dados com os ataques desferidos, de forma a serem usados posteriormente em nosso aprendizado supervisionado. A lista destes ataques está descrita no Apêndice B.

Esta base tem características distintas das do desafio DARPA. Primeiramente, trata-se de ataques reais desferidos por *crackers* anônimos da Internet. As técnicas utilizadas estão mais próximas daquelas enfrentadas no dia-a-dia das instituições conectadas à Internet. Em segundo lugar, pelo fato de se tratar de uma rede de mel, observamos que o volume de tráfego não associado aos ataques (“ruído de fundo”) é baixo. Devemos ter estas duas características em mente quando analisarmos os resultados dos experimentos.

### 5.3 Instanciação do Modelo

Para a realização dos experimentos implementamos as camadas do modelo para processar as fontes de alertas das duas bases de dados em questão.

A instanciação visa adequar a arquitetura do modelo à realidade dos dados de teste, bem como suportar diferentes tipos de classificadores para um estudo comparativo. Ela compreende o desenvolvimento de módulos que desempenhem as funções descritas pela abordagem.

De uma forma simplificada, há que se desenvolver um módulo de coleta e normalização para cada fonte de eventos usada no experimento, a saber: *Windows Security Log*, *BSM Log*, *tcpdump* e *Snort log*, *Syslog* e *Apache Access Log*, conforme mostra a Figura 5.2.

Cada um destes módulos manipula e traduz os dados de origem, gerando alertas no formato normalizado descrito na Seção 4.2.2.

Os dados (alertas) normalizados provenientes desta camada são agrupados pelo módulo de *AlertFusion*, constituindo os meta-alertas. Este módulo deve preencher os requisitos estabelecidos pelo modelo e descritos na Seção 4.3. Em especial, todo meta-alerta consiste de um registro compatível com o formato disposto na Tabela 4.2

Na camada mais alta, temos um módulo classificador para cada técnica de classificação usada no experimento, a saber: SVM, Rede *Bayesiana*, Árvore de Decisão e Coletânea (*Ensemble*).

Este módulos traduzem os registros de meta-alertas nos formatos de dados requeridos pelos pacotes de classificação. Conforme explicitaremos adiante, os módulos de classificação implementam a interface com pacotes de mineração de dados.

Nas próximas seções aprofundaremos os detalhes da implementação dos diversos módulos componentes da arquitetura.

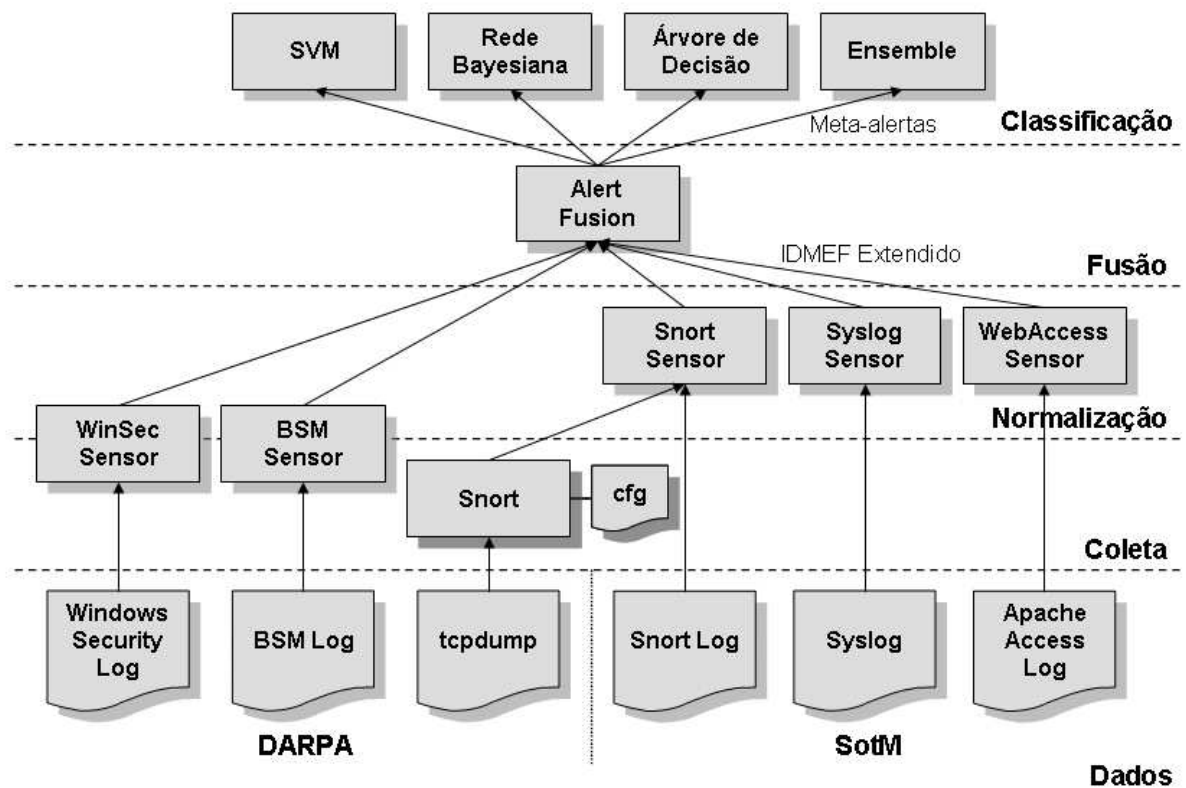


Figura 5.2: Instanciação do Modelo



### 5.3.1 Instanciação de Coleta e Normalização

Para a camada de coleta e normalização, desenvolvemos módulos de software que denominamos de sensores. Em nossa implementação, há dois tipos de sensores: passivos simples (cardinalidade 1x1 — ver Seção 4.2.1) e passivos com filtragem baseada em heurísticas.

Sensores passivos simples são utilizados para interface com fontes de alertas que já proveem eventos indicativos de incidentes de segurança. Dentre estas fontes de dados estão: *logs* de NIDSs, *logs* de HIDSs, e *logs* de acessos a sistemas.

Os sensores passivos com filtragem baseada em heurísticas foram utilizados para interface com sistemas como os *logs* BSM e o *log* de eventos de segurança do Windows. As entradas presentes em tais fontes não correspondem exclusivamente a alertas de segurança. Seu conteúdo precisa ser interpretado — tipicamente por um IDS — antes de serem normalizados e processados. Devido à ausência de sistemas de IDS públicos que interpretassem *logs* de BSM e Windows, optamos pelo desenvolvimento de tais módulos, utilizando-nos, para tal, de heurísticas simples. Estas heurísticas indicam situações conhecidas de potenciais incidentes na segurança da infraestrutura.

Independentemente de sua fonte de coleta, todos os alertas são transcritos segundo a taxonomia proposta. Os tipos de alertas processados e classificados podem ser encontrados nos Apêndices C e D.

Além do mapeamento entre um alerta e sua taxonomia, sensores também proveem a associação entre este mesmo alerta e sua classe. A definição da classe de um alerta é de especial importância para o suporte à camada de agrupamento. Tais mapeamentos são tipicamente configuráveis junto aos sensores e consistem em importante aspecto da atividade de normalização de alertas de segurança.

Nas subseções seguintes refinaremos cada módulo de coleta e normalização da implementação.

#### Snort

Há dois cenários diferentes para tratamento de eventos do Snort. No caso do desafio SotM, os *logs* de saída do Snort já são providos em formato texto. Neste caso, o próprio *SnortSensor* — implementado no escopo deste trabalho — realiza o *parsing* do arquivo e sua normalização nos campos propostos pelo modelo.

Para o caso da base DARPA, ao invés de *logs* Snort, são fornecidos o *dump* de todos os pacotes trafegados na rede. Para seu processamento, fizemos uso do Snort, o qual foi configurado com regras padrões (conforme distribuição disponível na página do projeto Snort [54]). O resultado são entradas de *log* do tipo abaixo:

```
[**] [1:553:7] POLICY FTP anonymous login attempt [**]
```

```
[Classification: Misc activity] [Priority: 3]
03/08-11:00:01.016166 172.16.114.148:1025 -> 197.218.177.69:21
TCP TTL:64 TOS:0x10 ID:242 IpLen:20 DgmLen:56 DF
***AP*** Seq: 0xA9E58D68 Ack: 0x7B8C0127 Win: 0x7D78 TcpLen: 20
```

Uma única alteração foi realizada no arquivo de configuração do Snort: todos os pré-processadores disponibilizados foram ligados. Estamos especialmente interessados na cobertura oferecida pelos processadores de desfragmentação, os quais auxiliam na detecção de ataques mais complexos, às custas de uma maior taxa de falsos positivos.

A título de ilustração, transcrevemos abaixo uma linha de saída do Snort associada ao funcionamento do pré-processador de desfragmentação.

```
[**] [113:1:1] (spp_frag2) Oversized fragment, probable DoS [**]
03/08-11:50:12.139067 206.229.221.82 -> 172.16.113.50
ICMP TTL:253 TOS:0x0 ID:176 IpLen:20 DgmLen:1500
Frag Offset: 0x1FCC Frag Size: 0x05C8
```

Independentemente do uso de pré-processadores, e conforme observado por Kayacik [28], sistemas normais de NIDSs, e em especial o Snort, apresentam altas taxas de falsos positivos, atingindo valores de 99% para a base DARPA. Tal fato pudemos observar na prática.

Para todo tipo de mensagem de saída do Snort, realizamos um mapeamento para a taxonomia proposta. Cuidado especial foi dispendido para que este mapeamento fosse consistente com o mapeamento realizado junto aos demais sensores. Esta consistência é vital na fusão de alertas e na extrapolação para detecção de novos ataques.

### *Syslog*

As mensagens de syslog são fornecidas pelo SotM como parte de seu desafio. O *SyslogSensor* tem um funcionamento bastante direto, consistindo basicamente de um *parsing* de entradas do seguinte formato.

```
Mar  6 08:54:20 combo sshd(pam_unix)[6390]: authentication failure;
logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=210.125.27.175 user=root
```

As entradas são mapeadas para a estrutura de dados proposta através do processo de normalização. Novamente há que se interpretar cada tipo de mensagem e prover um mapeamento consistente para a taxonomia proposta.

Algumas entradas não associadas a segurança são descartadas dentro deste módulo.

## Acessos Web

Os *logs* de acesso ao servidor web Apache do SotM são processados pelo *WebAccessSensor*. Seu funcionamento é análogo ao do *SyslogSensor*. O tipo de mensagem tratado é transcrito abaixo:

```
220.170.151.237 - - [16/Mar/2005:19:49:21 -0500] "GET / HTTP/1.1" 403 2898
"- " "Mozilla/4.0 (compatible; MSIE 5.5; Windows 98)"
```

Todas as entradas cujos códigos de retorno impliquem em erros são traduzidas em alertas, normalizados segundo formato proposto e classificados segundo taxonomia de tipos de eventos.

Há uma única exceção no processamento deste tipo de *log*. Como o mesmo descreve os acessos realizados por atores externos ao servidor web, ele constitui um bom ponto de verificação de ataques do tipo *buffer overflow* ou injeção de código (*code injection*). Optamos por desenvolver uma única regra adicional ao *parsing* simples das entradas do *log*. Esta regra cria um alerta sempre que a URL superar uma constante definida (em nosso caso, 100 bytes). Tal regra objetiva identificar acessos maliciosos como o descrito abaixo:

```
201.9.104.83 - - [17/Mar/2005:05:43:04 -0500] "GET /default.ida?XXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXX%u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%u7801%u9090%u6858
%ucbd3%u7801%u9090%u9090%u8190%u00c3%u0003%u8b00%u531b%u53ff%u0078
%u0000%u00=a HTTP/1.0" 404 1061 "- " "- "
```

A inserção de códigos maliciosos em chamadas a servidores HTTP constitui uma das formas mais comuns de ataques a um servidor web. Uma regra tão simples quanto a mencionada acima prova-se útil na detecção deste tipo de ataque.

## BSM

O processamento de *logs* de BSM é mais complexo. Tal sistema, quando ativado, registra todos os acessos realizados ao sistema operacional por aplicações de um servidor Solaris. Logo, suas entradas não implicam necessariamente em alertas de segurança, conforme trecho extraído de seu *log* abaixo:

```
file,Fri Mar 5 09:54:01 BRT 1999, + 747 msec,
```

```
header,36,2,system booted,na,Fri Mar  5 09:53:32 BRT 1999, + 219 msec,
text,booting kernel
header,91,2,getmsg(2),,Fri Mar  5 09:54:23 BRT 1999, + 481 msec,argument,
1,0x3,fd,argument,4,0xefffddccc,pri,subject,-2,root,root,root,root,96,0,
0 0 0.0.0.0,return,success,0,trailer,91
```

Idealmente, utilizaríamos um HIDS que interpretasse as entradas do BSM e dele extrairíamos os alertas de segurança. Em sua ausência, optamos por desenvolver algumas heurísticas que indicassem as ameaças mais óbvias a um servidor Unix. As heurísticas selecionadas, e implementadas, — em sua maior parte derivadas de ameaças descritas por Garfinkel e Stappford [20] — foram:

1. Detecção da criação de arquivos *core*.
2. Acessos suspeitos a arquivos *core*, ou edição dos mesmos com programas como: *cat*, *grep*, *vi*, *more*, *tail*, *head*.
3. Erros de *login* devido a senha incorreta.
4. Erros de *login* devido a usuário incorreto.
5. Invocação de chamada de sistema *exec* com argumentos muito longos (possível ataque de *buffer overflow*).
6. Detecção de *strings* formatadas em acessos a chamadas de sistema.
7. Acesso suspeito a diretório *home* de um usuário.
8. Operação de leitura e/ou edição suspeita sobre arquivos de configuração do sistema, tais como: *passwd*, *shadow*, *rhosts*, *cron*.
9. Operação de edição suspeita sobre arquivo de *log* do sistema, tais como: *sulog*, *lastlog*, *messages*, *acct*, *wtmp*, *wtmpx*, *pacct*, *saveacct*.
10. Operação suspeita para ganho de privilégios através de *setuid*.

Deve-se notar que este conjunto de heurísticas não é, nem objetiva ser, uma lista exaustiva de técnicas de detecção de intrusão em sistema Unix. Há sistemas comerciais de HIDS que possuem cobertura mais ampla que a lista supracitada. Na literatura há trabalhos extensos acerca da detecção de intrusões em sistema Unix a partir da análise de chamadas ao sistema, como aquele publicado por Tandon e Chan [55].

Não obstante, a lista acima apresenta subsídio importante para complementar os alarmes detectados por sistemas de NIDS e propiciar suporte à busca da consciência situacional.

### Logs de Segurança Windows

O processamento de *logs* de segurança do Windows apresenta desafio análogo ao enfrentado com o BSM. Suas entradas também não implicam diretamente em alertas de segurança, impedindo seu uso direto, conforme exemplo de trecho de seu *log* abaixo:

```
2/7/1999,01:11:29,Security,Auditoria com êxito,Acesso a objetos ,562,
S-1-5-21-742865521-1025978620-313593124-500,HUME,Identificador fechado:
```

```
Servidor de objetos: Security
Identificação do identificador: 32
Identificador do processo: 2154647584
Nome do arquivo de imagem: %4
```

```
2/7/1999,01:11:29,Security,Auditoria com êxito,Acesso a objetos ,560,
S-1-5-21-742865521-1025978620-313593124-500,HUME,"Objeto aberto:
```

```
Servidor de objetos: Security
Tipo de objeto: Section
Nome do objeto: \NLS\NlsSectionCType
Identificação do identificador: 32
Identificação da operação: {0,39287}
Identificação do processo: 2154647584
Nome do arquivo de imagem: Administrator
Nome de usuário primário: EYRIE
Domínio primário: (0x0,0x2808)
Identificação do logon primário: -
Nome de usuário cliente: -
Domínio do cliente: -
Identificação do logon do cliente: Mapear a seção para leitura
```

```
Acessos: -
Privilégios: %16
Contagem Sid restrita: %17
```

```
"
```

Além do *parsing* de seus registros, temos novamente de elencar um conjunto de heurísticas para detecção de intrusões. O processo consistiu numa transposição dos conceitos empregados em Unix para o ambiente Windows. O conjunto resultante de heurísticas é:

1. Invocação de aplicativo Dr. Watson (*drwtsn32.exe*) — análogo à geração de arquivo de *core* em Unix.

2. Acessos suspeitos a base de dados de configuração, incluindo: contas de usuários (*sam\_user*), sinônimos (*sam\_alias*), grupos de usuários (*sam\_group*), domínios (*sam\_domain*), e servidores (*sam\_server*).
3. Sucessos e falhas em *logins* no sistema através de diversas fontes: *advapi*, *ksecdd*, *iusr*.
4. Indicações de reinício (*restart*) ou desligamento (*shutdown*) — *hard* ou *soft* — do sistema operacional.
5. Acesso a aplicativos suspeitos: *posix.exe*, *psxss.exe*, *cat.exe*, *regedit.exe*, *net.exe*, *at.exe*, *cmd.exe*, *newdsn.exe* e *explore.exe*.

Deve-se ressaltar que, a exemplo do caso anterior, esta lista não pretende ser exaustiva com relação a todas as ameaças potenciais a um sistema Windows. Tão pouco pode-se afirmar que um alerta gerado pelos critérios acima necessariamente implique num incidente de segurança. Por conta deste fato, preferimos referir-nos a tal conjunto como heurísticas de segurança.

### 5.3.2 Fusão de Alertas

O módulo *AlertFusion* implementa o algoritmo descrito na Seção 4.3. Trata-se da camada mais intensa em processamento do modelo. Por isto, providências foram tomadas para otimizar seu desempenho.

A mais importante destas providências consiste em manter em memória principal os meta-alertas passíveis de fusão pelos próximos alertas. Isto é feito através de um mecanismo de *cache* de meta-alertas. As entradas deste *cache* são descartadas quando suas datas estão fora do alcance da janela de tempo usada pelo algoritmo de fusão. Deve-se notar que este algoritmo requer que os alertas sejam processados em ordem cronológica, ou muito próxima desta.

A opção pelo uso da linguagem *Perl* trouxe algumas facilidades no tratamento dos conceitos de pertinência (usados no algoritmo). Estes casos puderam ser traduzidos em operações de busca de *substrings*, as quais são extremamente otimizadas na linguagem *Perl*.

Uma implementação mais sofisticada poderia modelar os atributos de uma forma mais elaborada e otimizar ainda mais as operações de pertinência. Outra possibilidade de otimização reside na criação de árvores invertidas com índices para os diferentes atributos utilizados nas comparações no algoritmo de fusão.

Para os fins deste trabalho, o uso do modelo de *cache* e das funcionalidades da linguagem *Perl* mostraram-se suficientes para um bom desempenho nos experimentos.

### 5.3.3 Métodos de Classificação

Uma das principais contribuições deste trabalho reside em testarmos técnicas recentes de classificação dentro do contexto de detecção de intrusão. Não é de nosso conhecimento um trabalho que tenha envolvido tantas técnicas diferentes e recentes, aplicadas sobre uma massa de dados tão vasta, normalizada e agrupada.

Dentre as opções de classificadores existentes no campo de aprendizado de máquina, estas foram as escolhas:

1. SMVs: as técnicas de SVM têm chamado grande atenção da comunidade nos últimos anos. Estamos especialmente interessados na combinação entre as propriedades de *kernel* e o vetor de bits de tipos de alertas;
2. Redes *Bayesianas*: também trata-se de uma técnica de grande disseminação nos últimos tempos. Sua abordagem eminentemente estatística representa um contraponto às SVMs. A possibilidade de se contar com uma representação gráfica da rede colabora para a compreensão dos meta-alertas classificados;
3. Árvores de Decisão: sua simplicidade promove o desenvolvimento da intuição sobre os dados.
4. Coletânea (*ensemble*): combinar múltiplas técnicas ou instâncias de parâmetros diferentes na busca de um classificador de alto desempenho.

Nas próximas subseções detalharemos as implementações destes classificadores.

#### Utilizando SVMs

Para a implementação do classificador baseado em SVM utilizou-se a libSVM desenvolvida por Chang e Lin [8].

A preparação dos atributos descritos na Seção 4.3 para uso na SVM exigiu alguns cuidados e ajustes.

Os atributos *alert\_count*, *tgt\_node\_count* e *tgt\_port\_count* foram discretizados em quatro valores possíveis. Esta ação diminuiu a sensibilidade do algoritmo ao conjunto de dados de aprendizado. De uma forma intuitiva, não queremos que um ataque de negação de serviço que contém 200 alertas em seu meta-alerta deixe de ser detectado porque no conjunto do aprendizado supervisionado utilizamos um meta-alerta com 500 alertas.

A discretização ocorreu em faixas de valores. No caso de *alert\_count*, por exemplo, as faixas usadas foram 0–2, 3–5, 6–51 e mais que 51. Uma forma alternativa teria sido trabalhar com o logaritmo do atributo.

Os atributos *ext\_class* e *alert\_type\_set* foram linearizados de forma que cada valor possível do primeiro e cada bit do vetor do segundo fossem mapeados numa dimensão própria (uso de *dummy variables*).

Como nosso meta-alerta indica o tipo de ataque associado, basta para a SVM decidir entre um ataque ou um falso alarme. Desta feita, sua natureza binária de classificação adequa-se ao problema sem necessidades de ajustes.

A otimização do uso de SVM seguiu o guia proposto por Hsu et al [61], com foco para o processo de *scaling* das diferentes dimensões.

O aprendizado supervisionado foi realizado com diferentes pesos para classificações positivas e negativas (assimetria de custos; ver Seção 2.2.6). De forma geral, entendemos que falsos negativos são mais críticos que falsos positivos. Em nosso experimento utilizamos diferentes pesos para penalização dos falsos negativos, conforme mostraremos adiante. Tais resultados permitem que organizações escolham a melhor relação de assimetria de custos, segundo sua cultura e política de segurança. Em outras palavras, o número de falsos positivos que uma organização aceita — em troca da detecção de um positivos verdadeiro — é função de sua tolerância a riscos, disposição de investir em segurança, políticas internas, entre outros. O uso da técnica de assimetria de custos para desenho da curva ROC permite explicitar este compromisso e auxiliar na determinação do ponto ideal para uma dada organização.

### Utilizando Redes *Bayesianas*

A natureza eminentemente probabilística do fato de um conjunto de evidências designar um ataque de segurança torna o uso de técnicas *bayesianas* uma alternativa natural para o problema. Uma das primeiras iniciativas deste tipo ocorreu no projeto eBayes TCP de Valdes et al [59].

Em nossa implementação lançamos mão da ferramenta Weka [57], desenvolvida pela Universidade de Waikato. Trata-se de uma coleção de algoritmos de aprendizado de máquina que pode ser usada em atividades de mineração de dados. Além do suporte a Redes *Bayesianas*, também possui implementações de Árvores de Decisão.

Originalmente nossa intenção residia em utilizar a possibilidade de se definir a topologia da rede e usar o aprendizado supervisionado para a definição das tabelas de probabilidades condicionais (CPTs, ver Seção 2.2.2). A definição da topologia permite aumentar o grau de semântica da inferência. Poder-se-ia, por exemplo, reproduzir na topologia da rede conceitos como a anatomia de um ataque, conforme apresentado por Scambray et al [52].

A Figura 5.3 retrata um exemplo de fragmento de topologia de rede *bayesiana* que traduz os conceitos da anatomia de um ataque, dada a taxonomia definida neste trabalho.

Note-se a introdução dos nós PROBE, R2L, U2R e DoS. Estas são variáveis ocultas



(*hidden variables*) que objetivam ligar logicamente as evidências (tipos de alertas) ao modelo da anatomia do ataque. Tal anatomia é representada pelas arestas que ligam tais nós, numa visão simplificada daquela defendida por Scambray et al [52].

Infelizmente não logramos localizar implementações de redes *bayesianas* disponíveis que apresentassem simultaneamente as funcionalidades de importação de topologia e suporte a variáveis ocultas. No caso do Weka, por exemplo, a importação da topologia é possível, mas, neste contexto, não há suporte para as variáveis ocultas. Em função desta nova realidade, diferimos a idéia de uso de uma topologia pré-definida (com maior teor semântico) para trabalhos futuros, e voltamo-nos ao uso do Weka tanto para a determinação da topologia quanto das tabelas de probabilidades condicionais, através do classificador *weka.classifiers.bayes.BayesNet*.

Para uso pelo classificador *weka.classifiers.bayes.BayesNet* do Weka, os atributos de classificação descritos na Seção 4.3 sofreram a mesma manipulação descrita na Seção 5.3.3, incluindo discretização dos campos numéricos e uso de múltiplas dimensões para o vetor de bits e campos multivalorados.

Os pesos no aprendizado foram implementados através do uso de matrizes de custo (*cost-matrix*), mediante o metaclassificador *weka.classifiers.meta.CostSensitiveClassifier*.

O resultado da classificação reside na definição do estado da variável *booleana is\_attack*. Se verdadeira, trata-se de um ataque; caso contrário, o meta-alerta é um alarme falso.

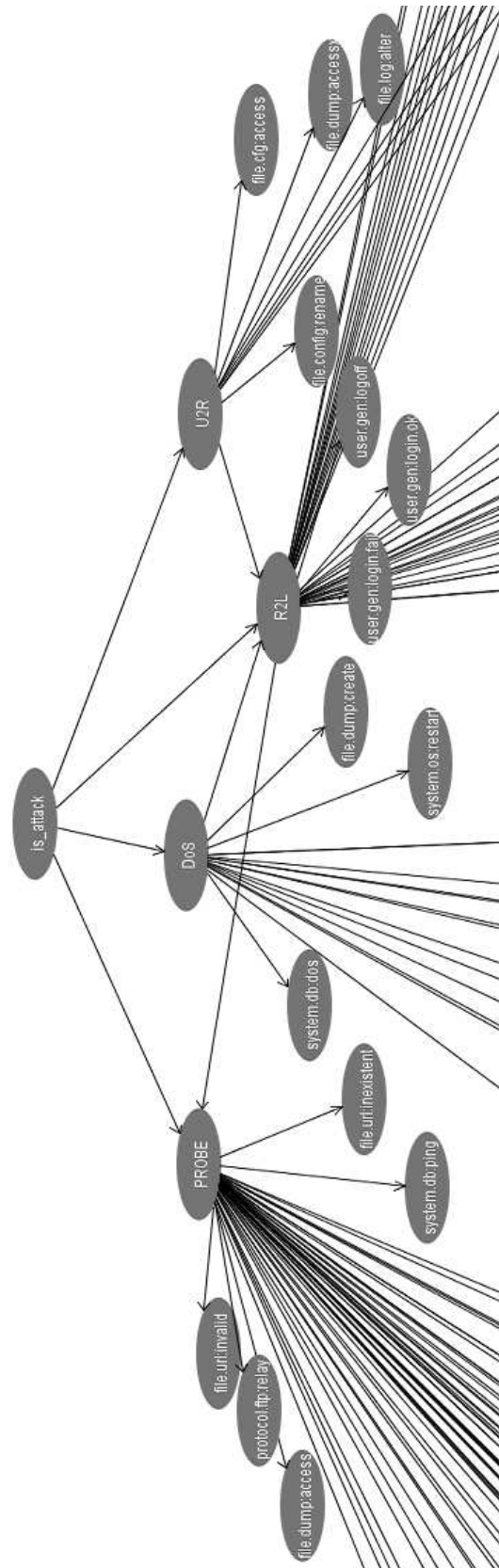


Figura 5.3: Uma Topologia de Rede *Bayesiana* Refletindo a Anatomia de um Ataque

### Utilizando Árvores de Decisão

Para implementação do classificador baseado em Árvore de Decisão, recorreremos novamente ao uso de Weka. Utilizamos o classificador *weka.classifiers.trees.J48*, o qual implementa o algoritmo C4.5.

Todo o tratamento dos atributos, matrizes de custo e metaclassificadores é idêntico ao usado na Rede *Bayesiana*.

### Utilizando *Ensembles*

Dados os bons e complementares resultados obtidos com os classificadores SVM e Redes *Bayesianas*, decidimos pela construção de uma coletânea (*ensemble*) destes dois. Como critério de eleição, optamos pela solução mais conservadora pela ótica de reduzir falsos negativos: se um dos classificadores julgar que o meta-alerta corresponde a um ataque, então o *ensemble* decide por um ataque.

## 5.4 Experimentos

Dada a implementação descrita na Seção 5.3, e as fontes de dados do DARPA e SotM (Seções 5.1 e 5.2), executamos quatro grandes cenários de experimentos como forma de verificação de nossa proposta:

1. DARPA 3x2: com base nos dados DARPA, treinamos nossos classificadores com as três semanas de dados anotados, e testamos seu desempenho contra as duas semanas remanescentes. Trata-se da reedição do próprio desafio DARPA.
2. DARPA kFold-5: com base nos dados DARPA, realizamos os testes com base num *K-Fold* (ver Seção 2.2.5) de cinco.
3. SotM kFold-5: com base nos dados SotM, realizamos os testes com base num *K-Fold* de cinco.
4. Alertas x Meta-alertas: reeditar o teste DARPA kFold-5 sobre os alertas, ao invés dos meta-alertas, usando SVMs; contrastar os resultados obtidos com meta-alertas e aqueles obtidos com alertas.

As próximas seções detalham estes experimentos e os resultados observados.

### 5.4.1 Experimento 1: DARPA 3x2

Neste experimento, reproduzimos o desafio DARPA original. Treinamos nossos classificadores de forma assistida com os dados das três primeiras semanas observadas. Os ataques são anotados individualmente, com base nas informações fornecidas pelo próprio DARPA. O sistema é testado contra as duas semanas de dados remanescentes.

#### Coleta e Normalização

Os módulos de coleta e normalização são aplicados às cinco semanas de dados da base DARPA. A Tabela 5.1 sumariza o resultado do processo de coleta e normalização junto a esta base de eventos.

Tabela 5.1: Fontes de Eventos DARPA

	Semana 1		Semana 2		Semana 3		Semana 4		Semana 5	
Analizador	Registros	Alertas	Registros	Alertas	Registros	Alertas	Registros	Alertas	Registros	Alertas
Snort Interno	142.674	142.674	47.405	47.405	18.742	18.742	17.169	17.169	34.652	34.652
Snort Externo	143.098	143.098	47.826	47.826	21.687	21.687	23.032	23.032	53.612	53.612
BSM	2.063.809	846	2.151.011	728	2.147.384	10.752	1.841.269	701	2.949.363	912
Windows	581.192	2.953	3.650.045	405	3.574.791	419	2.292.926	643	2.476.508	852
<b>Total</b>	2.930.773	289.571	5.896.287	96.364	5.762.604	51.600	4.174.396	41.545	5.514.135	90.028

Pode-se notar que o módulo *SnortSensor* aplicado ao *log* dos dois Snorts dispostos na rede representou a maior parte dos alertas coletados. Para este sensor, um registro do *log* do Snort corresponde a um alerta gerado. Trata-se do caso típico de um sensor passivo, com cardinalidade 1x1. No caso dos sensores de BSM e Windows, vemos o efeito das heurísticas sobre os registros; apenas um subconjunto destes é promovido à categoria de alerta. Neste caso, os sensores atuam como o primeiro filtro das informações submetidas para fusão. De um total de mais de 24 milhões de registros analisados, a atuação do Snort e das heurísticas dos demais sensores reduz o domínio de atuação para menos de 570 mil alertas.

### Fusão e Razão de Redução

Os alertas coletados e normalizados pelos sensores são processados pelo módulo *AlertFusion*.

A fusão aplicada a estes alertas importa na geração de 19.550 meta-alertas. Isto representa mais de uma ordem de grandeza de redução.

Assumindo o uso do classificador SVM, com penalidade máxima para falsos negativos (gerando, portanto, o número máximo de falsos positivos) temos um total de 268 meta-alertas classificados como ataques.

Contrastando o número de ataques indicados com o número de alertas, temos uma taxa de redução de dados para informação (*Data to Information Ratio*) de 2.124, superando os valores obtidos por Valdes e Skinner [58], e Sabata e Ornes [51].

Tabela 5.2: Redução de Eventos

Entidade	Quantidade
Registros	24.278.195
Alertas	569.108
Meta-alertas	19.550
Ataques Indicados	268
Razão de Redução	2.124

### Classificador – SVM

Inicialmente avaliamos os resultados obtidos utilizando-se o classificador SVM.

Ao variarmos a assimetria de custo de aprendizado, representada pelo parâmetro peso (*weight*) atribuído a um falso negativo com relação a um falso positivo (assimetria de custos), obtemos os resultados da Tabela 5.3.

Tabela 5.3: Resultados para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

	svm-w1	svm-w10	svm-w20	svm-w30	svm-w40	svm-w50	svm-w60	svm-w70	svm-w80	svm-w90	svm-w100
<b>TP</b>	0	37	46	46	55	56	59	61	62	62	63
<b>FP</b>	0	45	46	51	91	113	144	195	200	203	205
<b>TN</b>	7952	7907	7906	7901	7861	7839	7808	7757	7752	7749	7747
<b>FN</b>	285	248	239	239	230	229	226	224	223	223	222
<b>TP/Dia</b>	0,0	3,7	4,6	4,6	5,5	5,6	5,9	6,1	6,2	6,2	6,3
<b>FP/Dia</b>	0,0	4,5	4,6	5,1	9,1	11,3	14,4	19,5	20,0	20,3	20,5
<b>FPR</b>	0,00	0,01	0,01	0,01	0,01	0,01	0,02	0,02	0,03	0,03	0,03
<b>TPR</b>	0,00	0,13	0,16	0,16	0,19	0,20	0,21	0,21	0,22	0,22	0,22

A coluna svm-w1 indica o cenário em que falsos positivos e falsos negativos possuem o mesmo peso. Na extremidade oposta, svm-w100 representa o cenário em que um falso negativo tem um peso 100 vezes superior a um falso positivo.

Esta mesma informação pode ser visualizada graficamente na Figura 5.4. Pode-se observar a aparente conformidade de uma curva ROC típica (ver Seção 2.2.6), estabelecida acima da linha da indiferença.

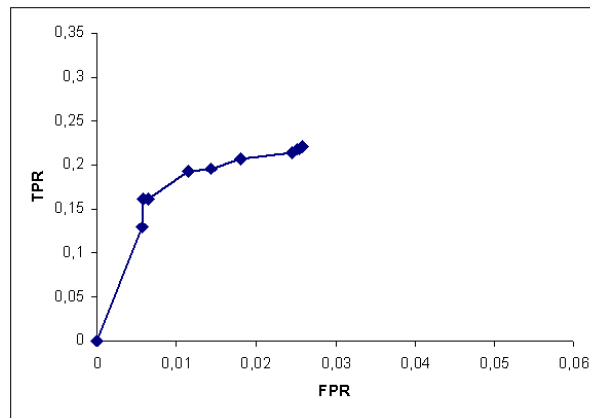


Figura 5.4: Curva ROC para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

O número de falsos negativos supera o número de positivos verdadeiros. Uma visão mais detalhada é provida pela Figura 5.5.

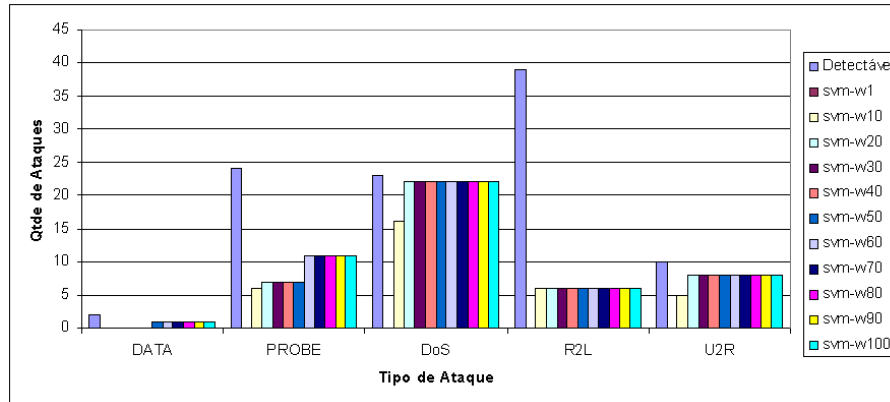


Figura 5.5: Detecção por tipo de ataque para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

Verificamos que a maior parte dos ataques são do tipo R2L; muitos deles, conforme mencionado por Lippmann et al [22] são ataques novos, inexistentes na massa de dados disponibilizada para o aprendizado assistido. Outra classe de ataques em que a SVM tem dificuldades de detecção é a de PROBE.

Em contrapartida, a taxa de detecção de ataques de DoS aproxima-se de 100%. O classificador também desempenha bem na detecção dos ataques de U2R.

De forma geral, o classificador apresenta baixa taxa de falsos positivos, porém tem dificuldades em extrapolar para novos ataques. Independentemente da variação do peso dos falsos negativos, os resultados parecem indicar que o SVM delinea uma superfície de classificação mais “rugosa”, dificultando a extrapolação para novos cenários.

### Classificador – Rede *Bayesiana*

O classificador baseado em Rede *Bayesiana* apresenta comportamento bastante diverso daquele observado com a SVM.

As taxas de detecção (TPR) são 50% mais altas neste classificador, conforme observa-se na Tabela 5.4. Todavia, este resultado ocorre às custas de um aumento de aproximadamente 100% no número de falsos positivos obtidos.

Não obstante o aumento do número de falsos positivos, estes mantêm-se muito abaixo do limite de 100 falsos positivos por dia, conforme proposto pelo desafio DARPA.

Tabela 5.4: Resultados para classificador Rede *Bayesiana* aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

	bayes-w1	bayes-w10	bayes-w20	bayes-w30	bayes-w40	bayes-w50	bayes-w60	bayes-w70	bayes-w80	bayes-w90	bayes-w100
<b>TP</b>	54	66	74	85	86	86	88	89	89	90	90
<b>FP</b>	96	261	285	377	386	386	394	394	394	394	397
<b>TN</b>	7856	7691	7667	7575	7566	7566	7558	7558	7558	7558	7555
<b>FN</b>	231	219	211	200	199	199	197	196	196	195	195
<b>TP/Dia</b>	5,4	6,6	7,4	8,5	8,6	8,6	8,8	8,9	8,9	9,0	9,0
<b>FP/Dia</b>	9,6	26,1	28,5	37,7	38,6	38,6	39,4	39,4	39,4	39,4	39,7
<b>FPR</b>	0,01	0,03	0,04	0,05	0,05	0,05	0,05	0,05	0,05	0,05	0,05
<b>TPR</b>	0,19	0,23	0,26	0,30	0,30	0,30	0,31	0,31	0,31	0,32	0,32

Com base nestas informações obtemos a curva ROC disposta na Figura 5.6. Novamente a curva estabelece-se bastante acima da linha da indiferença, atestando a efetividade do classificador.

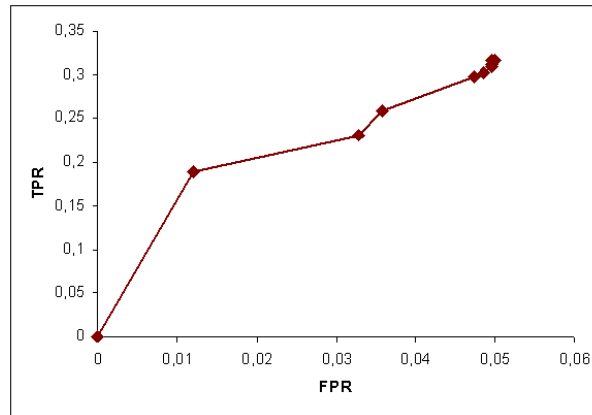


Figura 5.6: Curva ROC para classificador Rede *Bayesiana* aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

Sua conformação varia um pouco em relação às curvas tradicionais, sendo mais difícil a



identificação do gradiente de 45 graus, tipicamente usado como referência do ponto ótimo de funcionamento do classificador.

Alguma luz sobre este comportamento pode ser obtida pela análise da Figura 5.7.

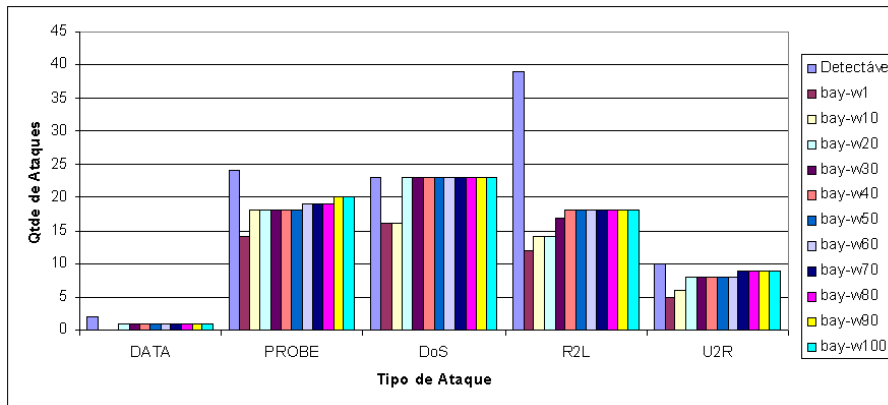


Figura 5.7: Detecção por tipo de ataque para classificador Rede *Bayesiana* aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

À medida que a assimetria de custos aproxima-se de 100 (custo de um falso negativo é 100 vezes maior que o custo de um falso positivo), novos ataques, nas categorias PROBE, R2L e U2R, são detectados.

Este classificador apresenta uma capacidade de extrapolação maior que o demonstrado pelo SVM. Alguns ataques novos — que não foram detectados por nenhum dos contendores do desafio DARPA original —, como *lsdomain*, *portsweep*, *queso*, *snmpget* e *ntfsdos*, foram corretamente identificados por este classificador.

A combinação de uma taxonomia genérica e um classificador bayesiano demonstrou a possibilidade de se detectar ataques para os quais não havia massa de aprendizado específica.

Deve-se notar que, à exceção dos ataques da classe R2L, os demais ataques detectáveis foram, em sua maioria, corretamente identificados pelo classificador baseado em Rede *Bayesiana*. Neste trabalho, um ataque é dito detectável se houver reais evidências de sua existência nos alertas fornecidos pelo Snort ou pelos demais sensores usados nos experimentos.

### Classificador – Árvore de Decisão

O classificador baseado em árvore de decisão apresentou os resultados mais fracos dentre o grupo. O número de falsos positivos permaneceu baixo, porém o número de verdadeiros positivos também, conforme indicado na Tabela 5.5.

Tabela 5.5: Resultados para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

	j48-w1	j48-w10	j48-w20	j48-w30	j48-w40	j48-w50	j48-w60	j48-w70	j48-w80	j48-w90	j48-w100
<b>TP</b>	33	50	50	50	50	51	51	51	51	51	51
<b>FP</b>	14	58	81	81	81	114	114	114	114	114	114
<b>TN</b>	7938	7894	7871	7871	7871	7838	7838	7838	7838	7838	7838
<b>FN</b>	252	235	235	235	235	234	234	234	234	234	234
<b>TP/Dia</b>	3,3	5,0	5,0	5,0	5,0	5,1	5,1	5,1	5,1	5,1	5,1
<b>FP/Dia</b>	1,4	5,8	8,1	8,1	8,1	11,4	11,4	11,4	11,4	11,4	11,4
<b>FPR</b>	0,00	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01	0,01
<b>TPR</b>	0,12	0,18	0,18	0,18	0,18	0,18	0,18	0,18	0,18	0,18	0,18

Sua curva ROC (Figura 5.8) tem conformação tradicional e mantém-se acima da reta da indiferença. Todavia, sua capacidade de detecção mantém-se aproximadamente 20% abaixo daquela do classificador SVM, e quase 50% abaixo daquela da Rede *Bayesiana*.

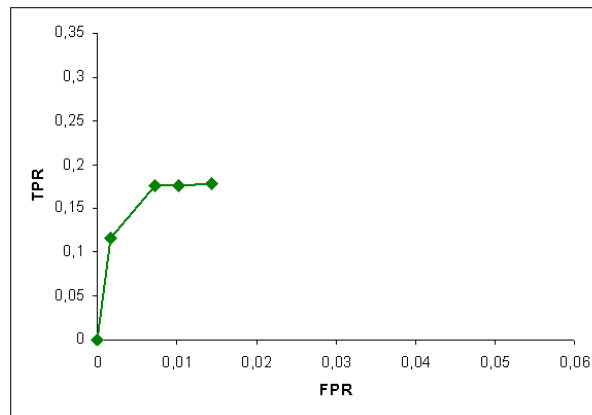


Figura 5.8: Curva ROC para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

Sua saturação acontece mais rapidamente do que nas outras técnicas. Com assimetria de custos da ordem de 10, o classificador atinge seu comportamento limítrofe. Este fenômeno é observável também no gráfico da Figura 5.9.

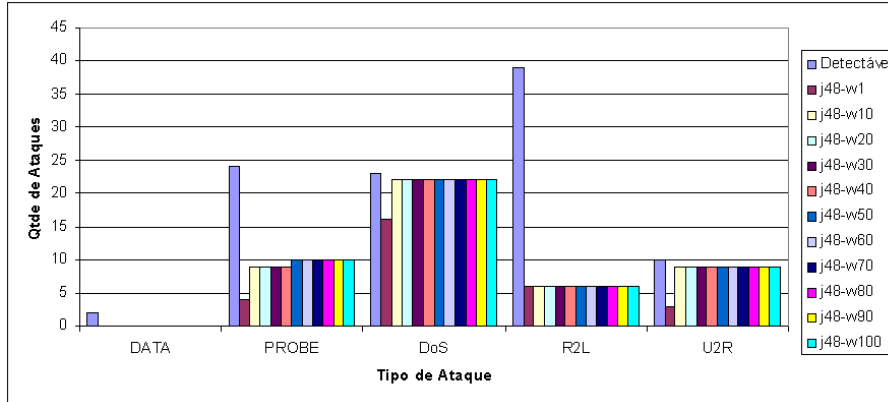


Figura 5.9: Detecção por tipo de ataque para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

É patente a dificuldade do classificador em detectar ataques das classes R2L e Probe.

Este classificador apresenta a maior dificuldade — dentre os utilizados em nossos testes — em extrapolar sua detecção e identificar novos ataques. Seu comportamento aproxima-se muito de um mecanismo de correlação determinístico; baseado em regras. Poucos falsos positivos; porém poucos positivos verdadeiros, também.

### Classificador – *Ensemble*

Quando se observa a capacidade de detecção dos classificadores SVM e Rede *Bayesiana*, nota-se que ambos têm comportamentos complementares. Os ataques de Probe são melhor detectados pelo SVM, ao passo que os demais são melhor detectados pela Rede *Bayesiana*.

Deriva-se naturalmente a idéia de mesclar estas capacidades num único classificador. Esta coletânea — *ensemble* — tem como critério de decisão indicar um ataque sempre que qualquer um dos seus classificadores constituintes (SVM e Rede *Bayesiana*) indicarem este ataque.

O resultado é uma capacidade de detecção um pouco superior àquela apresentada pela Rede *Bayesiana* simples. As detecções adicionais são obtidas ao custo de mais falsos positivos, conforme observável na Tabela 5.6. Não obstante, tal quantidade de falsos positivos segue abaixo do limite proposto pelo desafio DARPA original.

Tabela 5.6: Resultados para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

	ens-w1	ens-w10	ens-w20	ens-w30	ens-w40	ens-w50	ens-w60	ens-w70	ens-w80	ens-w90	ens-w100
<b>TP</b>	54	66	74	85	87	87	92	93	93	93	94
<b>FP</b>	96	261	285	380	403	403	438	441	444	446	451
<b>TN</b>	7856	7691	7667	7572	7549	7549	7514	7511	7508	7506	7501
<b>FN</b>	231	219	211	200	198	198	193	192	192	192	191
<b>TP/Dia</b>	5,4	6,6	7,4	8,5	8,7	8,7	9,2	9,3	9,3	9,3	9,4
<b>FP/Dia</b>	9,6	26,1	28,5	38,0	40,3	40,3	43,8	44,1	44,4	44,6	45,1
<b>FPR</b>	0,01	0,03	0,04	0,05	0,05	0,05	0,06	0,06	0,06	0,06	0,06
<b>TPR</b>	0,19	0,23	0,26	0,30	0,31	0,31	0,32	0,33	0,33	0,33	0,33

A curva ROC correspondente, descrita na Figura 5.10, indica a predominância do comportamento da Rede *Bayesiana* no resultado do *ensemble*. Isto advém do mecanismo de eleição implementado, o qual privilegia o componente que mais comumente indica a presença de ataques.

No gráfico da detecção por tipo de ataque, apresentado na Figura 5.11, pode-se ver claramente a complementaridade dos componentes em ação.

A Tabela 5.7 resume o comportamento deste classificador quando configurado em seu ponto máximo de detecção. A linha *Ataque* apresenta a totalidade de ataques presentes nas duas semanas de testes. Nem todos estes ataques são passíveis de detecção, devido a limitações dos sensores usados, dentre eles, do Snort.

Para os ataques detectáveis, representados nas linhas *Detectável*, podemos notar um bom comportamento do classificador. Mais relevante é seu comportamento se considerarmos que dentre os ataques ditos detectáveis, há ataques que não constavam da massa usada no aprendizado.

Atinge-se, portanto, um grau de detecção expressivo frente ao desafio proposto.

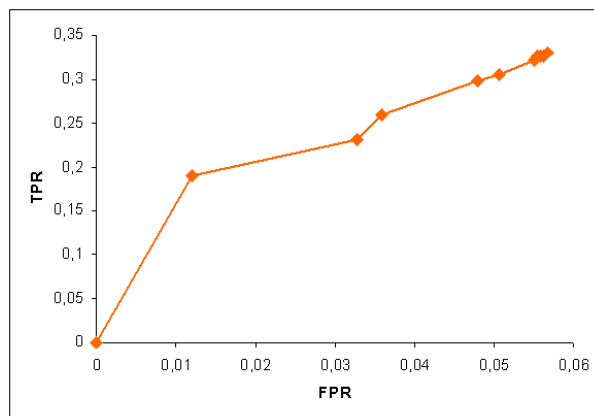


Figura 5.10: Curva ROC para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

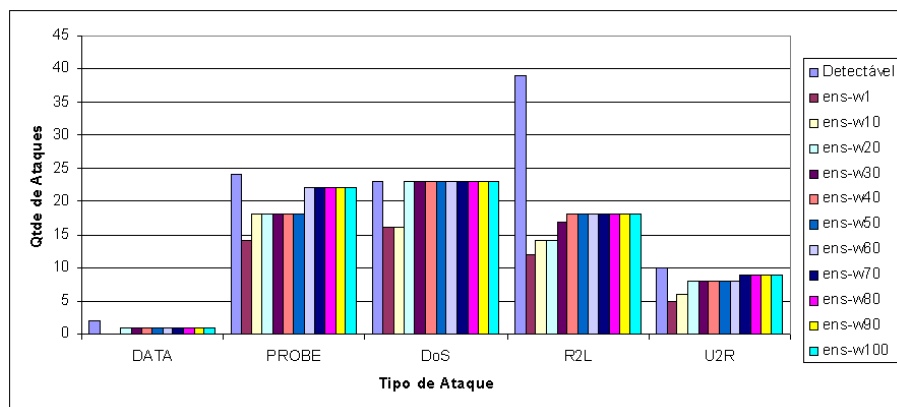


Figura 5.11: Detecção por tipo de ataque para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

Tabela 5.7: Matriz de ataques para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas semanas para testes

Sistema Operacional	Dados	Tipo de Ataque					
		DATA	DoS	PROBE	R2L	U2R	Total Geral
ALL	Ensemble-w100		0	5			5
	Detectável		0	5			5
	Ataque		2	11			13
Cisco	Ensemble-w100		1	2	4		7
	Detectável		1	2	4		7
	Ataque		1	3	4		8
Linux	Ensemble-w100	0	11	7	9	0	27
	Detectável	0	11	7	13	0	31
	Ataque	1	17	9	26	7	60
Solaris	Ensemble-w100	1	2	3	2	6	14
	Detectável	1	2	3	8	6	20
	Ataque	4	18	6	12	11	51
SunOS	Ensemble-w100		3	2	0	0	5
	Detectável		3	3	3	0	9
	Ataque		7	7	5	3	22
Windows	Ensemble-w100	0	6	3	3	3	15
	Detectável	1	6	4	11	4	26
	Ataque	1	12	8	14	10	45
<b>Ensemble-w100</b>		1	23	22	18	9	73
<b>Detectável</b>		2	23	24	39	10	98
<b>Ataque</b>		6	57	44	61	31	199

### 5.4.2 Experimento 2: DARPA kFold

Como comportamentar-se-iam estes classificadores se a massa de dados de aprendizado se aproximasse mais dos dados de teste? O que ocorreria se pudéssemos contar com uma maior gama de tipos de ataques para treinar nossos classificadores?

Para responder as perguntas acima, implementamos uma variante do experimento 1. Nesta variante, ao invés de treinarmos nossos classificadores com os meta-alertas presentes nas primeiras três semanas da massa de dados, dividimos aleatoriamente nossos meta-alertas em cinco grupos (*folds*), e realizamos a validação cruzada dentro dos grupos; treinamos o classificador com quatro grupos e testamos o mesmo contra o quinto. Este processo é repetido cinco vezes, cobrindo toda a massa de dados. Em suma, implementamos uma *K-fold cross-validation* (ver Seção 2.2.5), sendo  $K = 5$ .

Realizamos o processo acima para todos os classificadores, e os resultados observados estão dispostos nas seções subsequentes.

#### Classificador – SVM

O classificador SVM apresenta um expressivo incremento em sua taxa de detecção, quando comparado com o experimento anterior, com um aumento do número de falsos positivos por dia, conforme Tabela 5.8.

Tabela 5.8: Resultados para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

	svm-w1	svm-w10	svm-w20	svm-w30	svm-w40	svm-w50	svm-w60	svm-w70	svm-w80	svm-w90	svm-w100
<b>TP</b>	0	103	247	254	261	266	269	275	275	278	286
<b>FP</b>	0	149	911	1245	1508	1591	1876	2070	2198	2481	3127
<b>TN</b>	19209	19060	18298	17964	17701	17618	17333	17139	17011	16728	16082
<b>FN</b>	341	238	94	87	80	75	72	66	66	63	55
<b>TP/Dia</b>	0,0	4,1	9,9	10,2	10,4	10,6	10,8	11,0	11,0	11,1	11,4
<b>FP/Dia</b>	0,0	6,0	36,4	49,8	60,3	63,6	75,0	82,8	87,9	99,2	125,1
<b>FPR</b>	0,00	0,01	0,05	0,06	0,08	0,08	0,10	0,11	0,11	0,13	0,16
<b>TPR</b>	0,00	0,30	0,72	0,74	0,77	0,78	0,79	0,81	0,81	0,82	0,84

Segundo observável na Figura 5.12, a taxa de detecção do classificador atinge valores superiores a 0,70 a partir de uma assimetria de custos de 20. Deste ponto em diante, nota-se a queda expressiva do gradiente da curva; uma conformação de curva ROC das mais clássicas.

O número de falsos positivos é mais alto que o observado no experimento anterior para o mesmo classificador. Todavia, tal índice apenas supera aquele do desafio DARPA original com a assimetria de custos máxima, a saber, 100.

Na Figura 5.13, pode-se perceber a melhora na detecção dos ataques da classe R2L. Nas classes DoS, Probe e U2R, atinge-se 100% de identificação dos ataques ditos detectáveis.

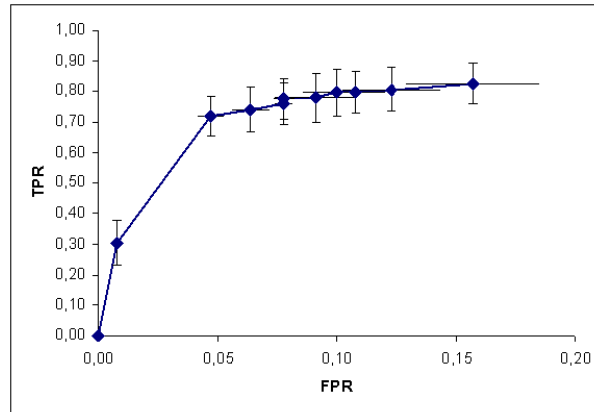


Figura 5.12: Curva ROC para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

Claramente percebe-se que o classificador SVM tem bom desempenho quando a massa de treinamento é representativa da realidade na qual operará. Neste contexto, sua taxa de detecção é alta, com níveis aceitáveis de falsos positivos.



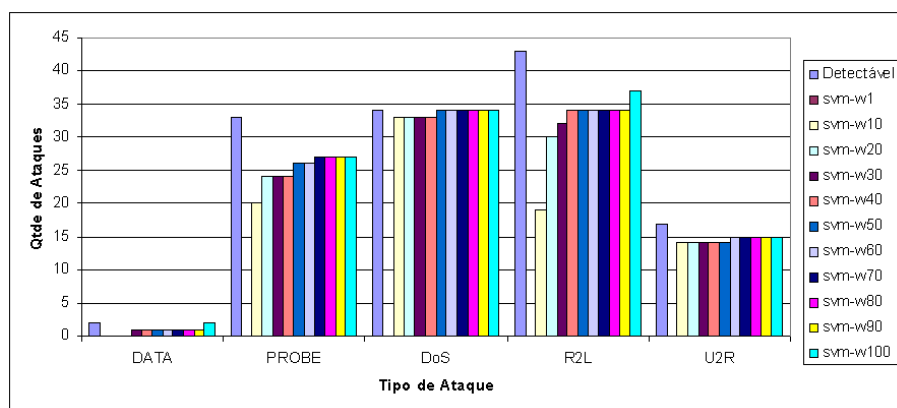


Figura 5.13: Detecção por tipo de ataque para classificador SVM aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

### Classificador – Rede *Bayesiana*

O classificador baseado em Rede *Bayesiana* comporta-se diferentemente da SVM. Embora o nível de detecção aproxime-se daquele observado na SVM, isto ocorre às custas de um alto nível de falsos positivos, conforme observável na Tabela 5.9.

Tabela 5.9: Resultados para classificador Rede *Bayesiana* aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

	bay-w1	bay-w10	bay-w20	bay-w30	bay-w40	bay-w50	bay-w60	bay-w70	bay-w80	bay-w90	bay-w100
<b>TP</b>	87	111	127	161	238	266	268	271	279	280	282
<b>FP</b>	137	508	797	1222	1794	2275	2377	2641	3474	3838	4008
<b>TN</b>	19072	18701	18412	17987	17415	16934	16832	16568	15735	15371	15201
<b>FN</b>	254	230	214	180	103	75	73	70	62	61	59
<b>TP/Dia</b>	3,5	4,4	5,1	6,4	9,5	10,6	10,7	10,8	11,2	11,2	11,3
<b>FP/Dia</b>	5,5	20,3	31,9	48,9	71,8	91,0	95,1	105,6	139,0	153,5	160,3
<b>FPR</b>	0,01	0,03	0,04	0,06	0,09	0,12	0,12	0,14	0,18	0,20	0,21
<b>TPR</b>	0,26	0,33	0,37	0,47	0,70	0,78	0,79	0,79	0,82	0,82	0,83

A partir de uma assimetria de custos de 80, o número de falsos positivos cruza a linha de 100 instância diárias. Ao observarmos a Figura 5.14, notamos que entre os pesos ( $w$ ) de 50 e 60, atingimos um gradiente próximo de 45 graus, indicando, possivelmente, o ponto ótimo de seu funcionamento.

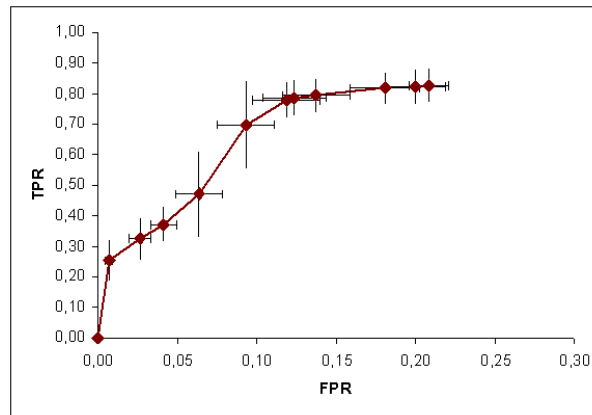


Figura 5.14: Curva ROC para classificador Rede *Bayesiana* aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

O gráfico delineado para o classificador aproxima-se das curvas ROC mais tradicionais, estabelecendo-se consideravelmente acima da reta da indiferença.

Conforme mostrado na Figura 5.15, a detecção aproxima de 100% para todas as classes de ataques detectáveis, à exceção da classe *Probe*. Aparentemente, apesar da melhoria

na detecção desta classe de ataques, ela segue sendo a maior dificuldade no desempenho deste classificador.

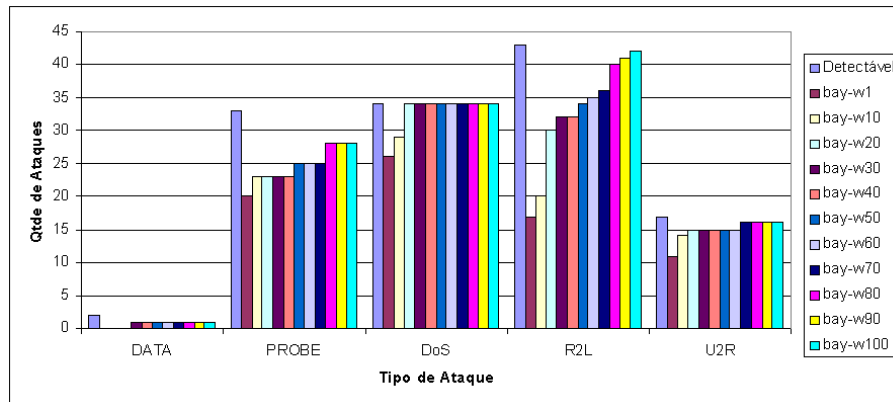


Figura 5.15: Detecção por tipo de ataque para classificador Rede *Bayesiana* aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

### Classificador – Árvore de Decisão

A exemplo do ocorrido com a SVM, o classificador baseado em árvore de decisão apresentou desempenho substancialmente melhor quando exposto a uma base de dados de treinamento mais extensa e diversa.

Conforme verificável na Tabela 5.10, a taxa de detecção aproxima-se da obtida pelos outros dois classificadores, porém com menor quantidade de falsos positivos.

Tabela 5.10: Resultados para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

	j48-w1	j48-w10	j48-w20	j48-w30	j48-w40	j48-w50	j48-w60	j48-w70	j48-w80	j48-w90	j48-w100
<b>TP</b>	69	238	244	246	246	249	249	256	262	273	273
<b>FP</b>	20	652	775	855	962	1150	1365	1852	2128	2562	2659
<b>TN</b>	19189	18557	18434	18354	18247	18059	17844	17357	17081	16647	16550
<b>FN</b>	272	103	97	95	95	92	92	85	79	68	68
<b>TP/Dia</b>	2,8	9,5	9,8	9,8	9,8	10,0	10,0	10,2	10,5	10,9	10,9
<b>FP/Dia</b>	0,8	26,1	31,0	34,2	38,5	46,0	54,6	74,1	85,1	102,5	106,4
<b>FPR</b>	0,00	0,03	0,04	0,04	0,05	0,06	0,07	0,10	0,11	0,13	0,14
<b>TPR</b>	0,20	0,70	0,72	0,72	0,72	0,73	0,73	0,75	0,77	0,80	0,80

A partir de uma assimetria de custos de 10, percebe-se a inflexão do comportamento do classificador, conforme indica a Figura 5.16. Por volta deste ponto, o gráfico apresenta um “cotovelo” protuberante.

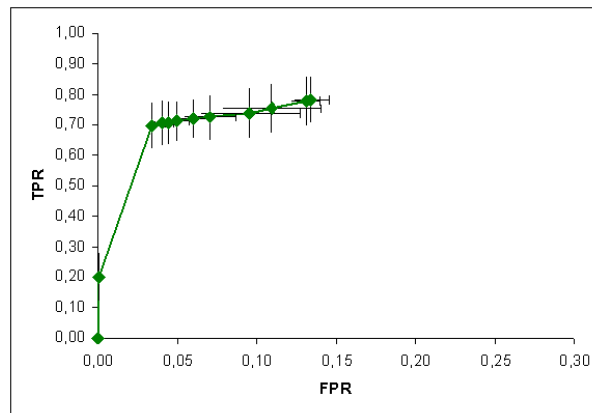


Figura 5.16: Curva ROC para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

O comportamento geral deste classificador é mais próximo da SVM também nas classes de ataques mais detectadas, conforme Figura 5.17.

Note-se que a detecção de ataques da classe *Probe* supera aquela observada pela Rede *Bayesiana*, por exemplo.

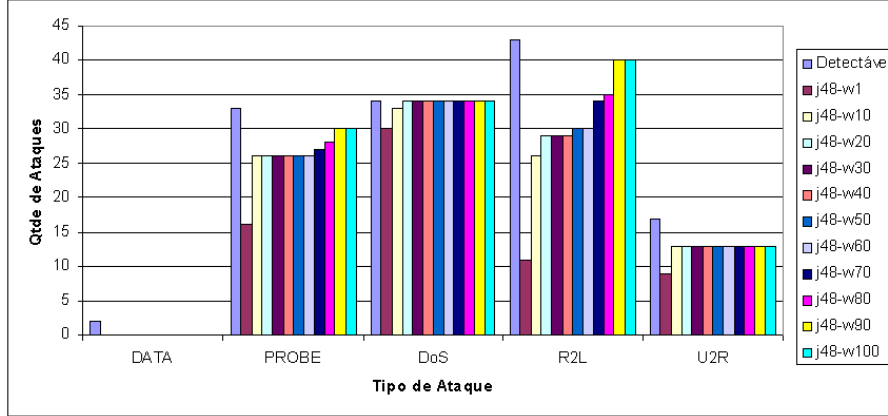


Figura 5.17: Detecção por tipo de ataque para classificador Árvore de Decisão aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

Evidencia-se, portanto, a característica de um classificador com alta “rugosidade”.

Sua performance é proporcional à maior ou menor exposição a uma base de treinamentos mais representativa e completa.

Seu ponto fraco está, evidentemente, associado à dificuldade em extrapolar sua detecção para novos tipos de ataque.

### Classificador – *Ensemble*

Construindo-se um *ensemble* com o classificador SVM e Rede *Bayesiana*, e variando-se a assimetria de custos no aprendizado dos componentes, obtemos os resultados descritos na Tabela 5.11.

Tabela 5.11: Resultados para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

	ens-w1	ens-w10	ens-w20	ens-w30	ens-w40	ens-w50	ens-w60	ens-w70	ens-w80	ens-w90	ens-w100
<b>TP</b>	87	122	259	271	274	279	280	288	296	297	303
<b>FP</b>	137	541	1492	1901	2236	2540	2717	3080	3951	4335	4714
<b>TN</b>	19072	18668	17717	17308	16973	16669	16492	16129	15258	14874	14495
<b>FN</b>	254	219	82	70	67	62	61	53	45	44	38
<b>TP/Dia</b>	3,5	4,9	10,4	10,8	11,0	11,2	11,2	11,5	11,8	11,9	12,1
<b>FP/Dia</b>	5,5	21,6	59,7	76,0	89,4	101,6	108,7	123,2	158,0	173,4	188,6
<b>FPR</b>	0,01	0,03	0,08	0,10	0,12	0,13	0,14	0,16	0,21	0,23	0,25
<b>TPR</b>	0,26	0,36	0,76	0,79	0,80	0,82	0,82	0,84	0,87	0,87	0,89

As taxas de detecção aproximam-se de 90%, porém o número de falsos positivos também é mais alto. O ponto ótimo de sua curva reside entre aqueles definidos por assimetrias de custo de 30 e 40, conforme observa-se na Figura 5.18 .

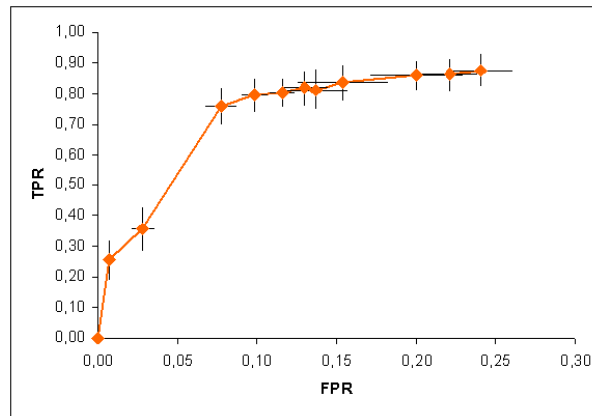


Figura 5.18: Curva ROC para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

A Figura 5.19 indica como este classificador consegue mesclar o melhor dentre os classificadores que o compõem. Temos altos índices de detecção em todas as classes de ataques.

Tal comportamento também é verificável na Tabela 5.12.

Optando-se pelo classificador definido por uma assimetria de custos de 40, temos um total de 112 detecções dentre 129 possíveis. Um índice de detecção de 86%.

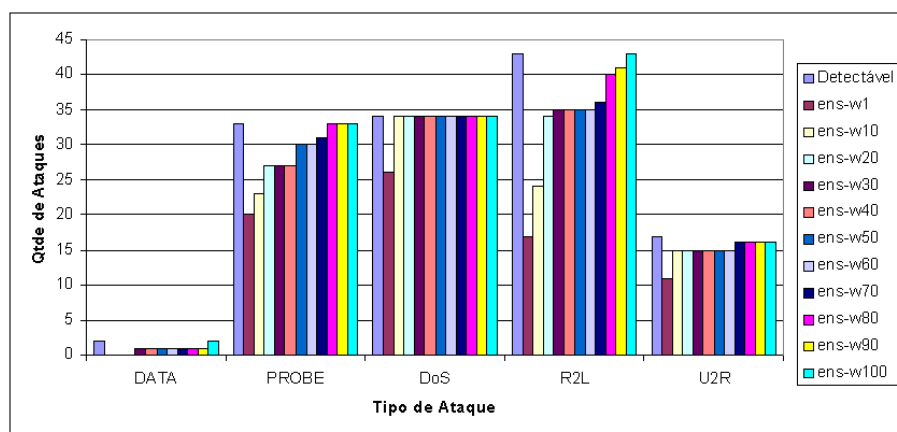


Figura 5.19: Detecção por tipo de ataque para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

Tabela 5.12: Matriz de ataques para classificador Ensemble aplicado a meta-alertas da base DARPA utilizando-se *K-fold Cross-validation*

Sistema Operacional	Dados	Tipo de Ataque					
		DATA	DoS	PROBE	R2L	U2R	Total Geral
ALL	Ensemble-w40		0	5			5
	Ensemble-w100		0	8			8
	Detectável		0	8			8
	Ataque		2	14			16
Cisco	Ensemble-w40		1	2	4		7
	Ensemble-w100		1	2	4		7
	Detectável		1	2	4		7
	Ataque		1	3	4		8
Linux	Ensemble-w40	0	15	11	15	0	41
	Ensemble-w100	0	15	11	15	1	42
	Detectável	0	15	11	15	1	42
	Alerta	3	21	14	28	10	76
Solaris	Ensemble-w40	1	4	4	10	12	31
	Ensemble-w100	1	4	4	10	12	31
	Detectável	1	4	4	10	12	31
	Ataque	5	22	7	16	17	67
SunOS	Ensemble-w40		5	2	2	0	9
	Ensemble-w100		5	3	3	0	11
	Detectável		5	3	3	0	11
	Ataque		9	7	7	3	26
Windows	Ensemble-w40	0	9	3	4	3	19
	Ensemble-w100	1	9	5	11	3	29
	Detectável	1	9	5	11	4	30
	Ataque	1	15	9	14	10	49
Total Ensemble-w40		1	34	27	35	15	112
Total Ensemble-w100		2	34	33	43	16	128
Total Detectável		2	34	33	43	17	129
Total Ataque		9	70	54	69	40	242



### 5.4.3 Experimento 3: SotM kFold

Existe alguma controvérsia quanto ao uso da base DARPA. Embora consista, até o presente momento, no padrão *de facto* para análise de sistemas de detecção de intrusão, alguns pesquisadores questionam o fato da base ser simulada, e não real. Não obstante, trabalhos continuam sendo publicados com base em seu conteúdo, conforme é percebido em [46].

Para complementar nosso estudo, utilizamos uma segunda base: o *Scan of the Month 34*.

Esta base reflete ataques reais perpetrados por *crackers* na Internet contra uma rede de mel. Como resultado, tem-se um conjunto de ataques mais homogêneos em um ambiente com baixo nível de ruído de fundo.

Nosso procedimento de teste é análogo ao usado para a base DARPA em modelo de K-fold. Ou seja, os meta-alertas, derivados de alertas anotados com base nas respostas do desafio, são divididos aleatoriamente em cinco grupos. Cada classificador é submetido a cinco sessões de treinamento (usando quatro grupos) e teste (usando o grupo restante).

Nas seções seguintes apresentamos os resultados observados.

### Coleta e Normalização

Os alertas de segurança advêm de um IDS Snort, de um *log* de acesso web, e de eventos de *Syslog*, conforme detalhado na Tabela 5.13.

Tabela 5.13: Fontes de Eventos SotM

Analizador	Registros	Alertas
Snort	69.039	69.039
Acessos Web	3.554	3.414
<i>Syslog</i>	1.158	953
<b>Total</b>	<b>73.751</b>	<b>73.406</b>

Na Tabela 5.14 verifica-se a taxa de redução de alertas em ataques indicados. A razão entre dados e informação (DIR) é de 50, consequência de um ambiente com baixíssimo tráfego de fundo (ruído).

Dadas as características de uma rede de mel, uma parte mais substancial do tráfego corresponde a ataques, se comparado com uma rede normal. Isto justifica esta queda no DIR, na comparação com os experimentos 1 e 2.

Os ataques indicados são aqueles observados com o classificador SVM em sua taxa máxima de falsos positivos.

Nas subseções seguintes abordamos o comportamento dos vários classificadores junto a esta base.

Tabela 5.14: Redução de Eventos em SotM

<b>Entidade</b>	<b>Quantidade</b>
Registros	73.751
Alertas	73.406
Meta-alertas	8.528
Ataques Indicados	1.469
Razão de Redução	50

### Classificador – SVM

Conforme indicado na Tabela 5.15, as taxas de detecção rapidamente convergem para níveis de 98%, com baixo número de falsos positivos.

Tabela 5.15: Resultados para classificador SVM aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

	svm-w1	svm-w10	svm-w20	svm-w30	svm-w40	svm-w50	svm-w60	svm-w70	svm-w80	svm-w90	svm-w100
<b>TP</b>	493	695	698	700	701	702	703	703	703	703	703
<b>FP</b>	107	733	735	735	737	737	745	747	750	759	766
<b>TN</b>	7707	7081	7079	7079	7077	7077	7069	7067	7064	7055	7048
<b>FN</b>	221	19	16	14	13	12	11	11	11	11	11
<b>TP/Dia</b>	8,1	11,4	11,4	11,5	11,5	11,5	11,5	11,5	11,5	11,5	11,5
<b>FP/Dia</b>	1,8	12,0	12,0	12,0	12,1	12,1	12,2	12,2	12,3	12,4	12,6
<b>FPR</b>	0,01	0,09	0,09	0,09	0,09	0,09	0,10	0,10	0,10	0,10	0,10
<b>TPR</b>	0,69	0,97	0,98	0,98	0,98	0,98	0,98	0,98	0,98	0,98	0,98

Estes resultados são fruto do bom funcionamento do classificador dentro de uma realidade mais homogênea de ataques. A base DARPA, por tratar-se de um ambiente construído sob medida para o teste, possui ataques muito mais heterogêneos (e complexos) do que a média do encontrado em ambientes reais. Com os dados do SotM, temos uma amostra mais próxima do comportamento típico de um *cracker*.

Para este cenário, com uma base de treino adequada, a abordagem proposta atinge altos níveis de desempenho, conforme indicado na Figura 5.20.

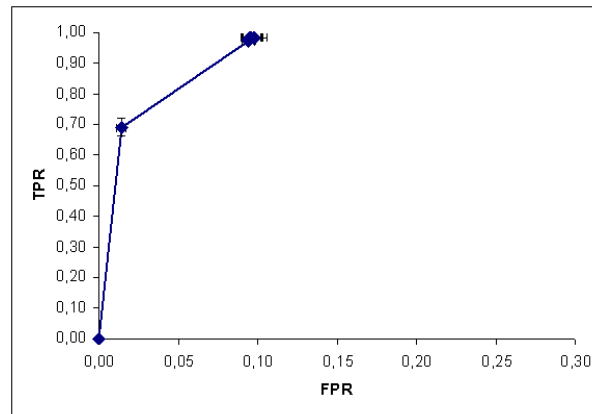


Figura 5.20: Curva ROC para classificador SVM aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

Deve-se notar que o classificador atinge seu desempenho ótimo com uma assimetria de custos da ordem de 20. Após este ponto, as taxas de detecção e de falsos positivos

permanecem basicamente inaltareadas. Temos, portanto, um processo de saturação com relação ao efeito da assimetria de custos na curva.

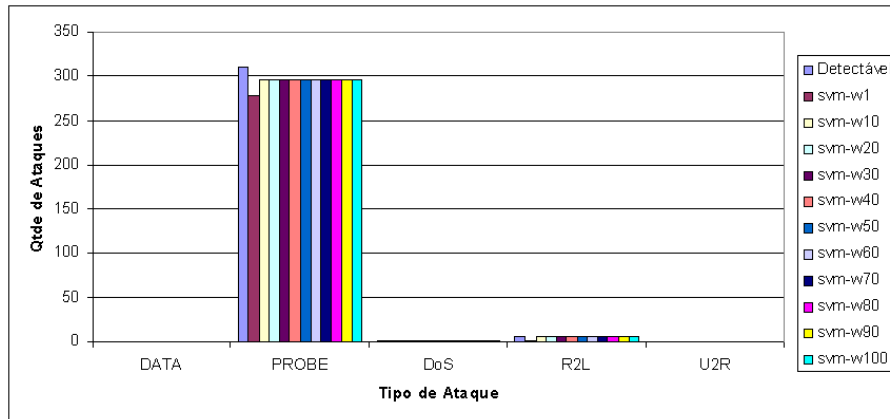


Figura 5.21: Detecção por tipo de ataque para classificador SVM aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

Conforme observável na Figura 5.21, a maior parte dos ataques da base SotM são da classe *Probe*. Sua detecção aproxima-se, porém não atinge, os 100%. Nas demais classes presentes, DoS e R2L, ela efetivamente atinge 100%.

### Classificador – Rede *Bayesiana*

O classificador baseado em Rede *Bayesiana* consegue adicionar um ponto percentual ao resultado, às custas de um aumento substancial nos falsos positivos, conforme a Tabela 5.16.

Seu ponto ótimo é atingido com uma assimetria de custos de 40, para o qual a taxa de positivos verdadeiros é de 99%.

Tabela 5.16: Resultados para classificador Rede *Bayesiana* aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

	bay-w1	bay-w10	bay-w20	bay-w30	bay-w40	bay-w50	bay-w60	bay-w70	bay-w80	bay-w90	bay-w100
<b>TP</b>	591	668	681	687	707	707	707	708	708	708	708
<b>FP</b>	236	1510	1556	1720	1792	2053	2176	2411	2429	2429	2436
<b>TN</b>	7578	6304	6258	6094	6022	5761	5638	5403	5385	5385	5378
<b>FN</b>	123	46	33	27	7	7	7	6	6	6	6
<b>TP/Dia</b>	9,7	11,0	11,2	11,3	11,6	11,6	11,6	11,6	11,6	11,6	11,6
<b>FP/Dia</b>	3,9	24,8	25,5	28,2	29,4	33,7	35,7	39,5	39,8	39,8	39,9
<b>FPR</b>	0,03	0,19	0,20	0,22	0,23	0,26	0,28	0,31	0,31	0,31	0,31
<b>TPR</b>	0,83	0,94	0,95	0,96	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Na Figura 5.22 podemos ver que o classificador atinge um platô após uma assimetria de custos de 40. A partir deste não há ganhos de detecção, porém o número de falsos positivos segue crescendo.

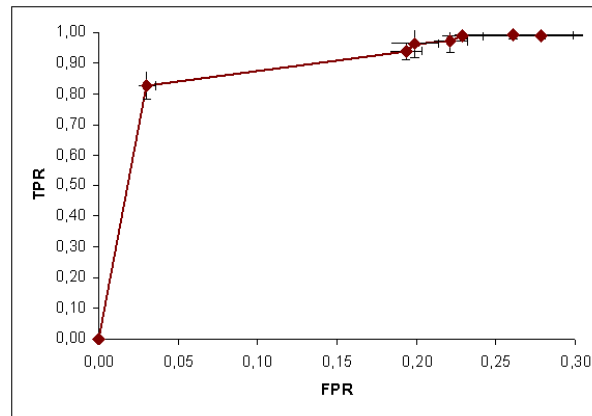


Figura 5.22: Curva ROC para classificador Rede *Bayesiana* aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

A detecção por classe de ataques aproxima-se daquela do classificador SVM, conforme verifica-se na Figura 5.23. Apenas a classe de ataques *Probe* não é completamente detectada.

Este é um classificador de excelente desempenho, desde que o volume de falsos positivos seja tolerável para o contexto no qual está inserido. Em nosso cenário, temos um positivo

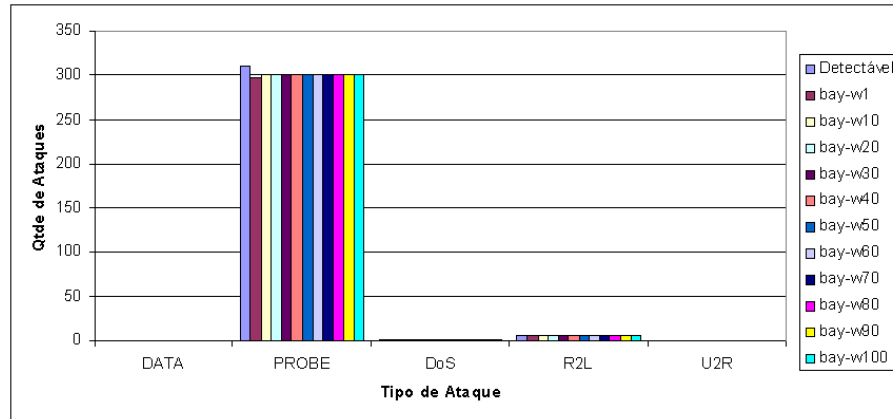


Figura 5.23: Detecção por tipo de ataque para classificador Rede *Bayesiana* aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

verdadeiro para cada dois a três falsos positivos. Não é um *trade-off* dos piores no domínio de gestão de segurança.

### Classificador – Árvore de Decisão

Curiosamente, a árvore de decisão apresenta os melhores resultados de todos os classificadores testados. Ela consegue atingir 99% de taxa de positivos verdadeiros com um número reduzido de falsos positivos, conforme Tabela 5.17.

Tabela 5.17: Resultados para classificador Árvore de Decisão aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

	j48-w1	j48-w10	j48-w20	j48-w30	j48-w40	j48-w50	j48-w60	j48-w70	j48-w80	j48-w90	j48-w100
<b>TP</b>	516	683	706	706	706	706	706	706	706	706	706
<b>FP</b>	44	461	703	706	713	774	781	798	806	840	849
<b>TN</b>	7770	7353	7111	7108	7101	7040	7033	7016	7008	6974	6965
<b>FN</b>	198	31	8	8	8	8	8	8	8	8	8
<b>TP/Dia</b>	8,5	11,2	11,6	11,6	11,6	11,6	11,6	11,6	11,6	11,6	11,6
<b>FP/Dia</b>	0,7	7,6	11,5	11,6	11,7	12,7	12,8	13,1	13,2	13,8	13,9
<b>FPR</b>	0,01	0,06	0,09	0,09	0,09	0,10	0,10	0,10	0,10	0,11	0,11
<b>TPR</b>	0,72	0,96	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Conforme verificável na Figura 5.24, o ponto ótimo de funcionamento é atingido com uma assimetria de custos de 20. Após o mesmo, a curva converge para sua saturação, onde incrementos do peso de treinamento não alteram o número de positivos verdadeiros, e há pouca mudança no de falsos positivos.

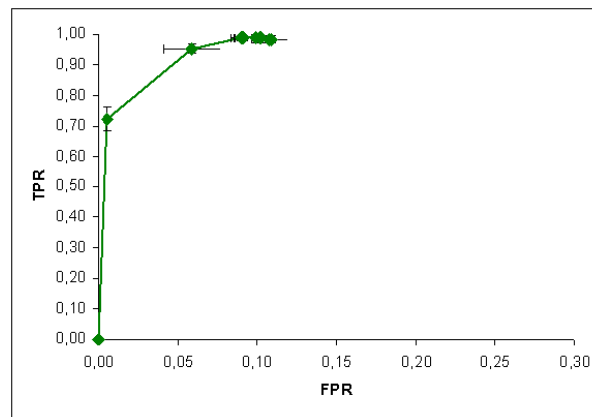


Figura 5.24: Curva ROC para classificador Árvore de Decisão aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

A detecção das classes de ataques, conforme a Figura 5.25, reflete o comportamento médio dos demais classificadores; ou seja, apenas a classe *Probe* não é totalmente detectada.

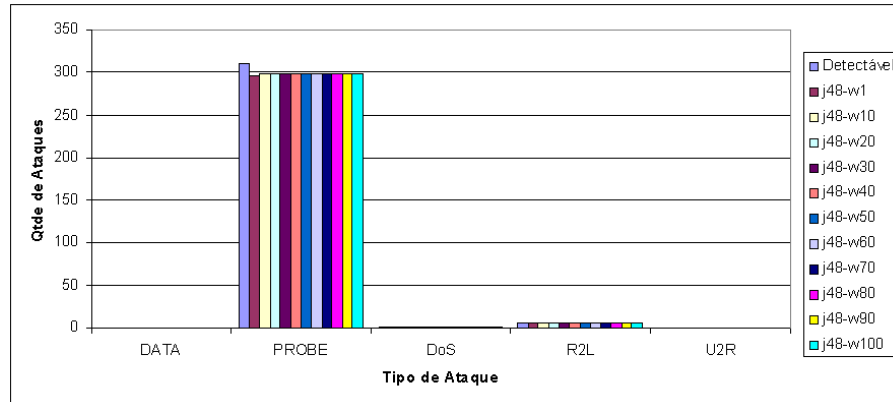


Figura 5.25: Detecção por tipo de ataque para classificador Árvore de Decisão aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

Trata-se, portanto, de um excelente classificador quando a base de treinamento reflete bem a realidade. Novamente é a “rugosidade” operando em prol deste classificador quando dados de aprendizado e dados de operação são compatíveis, homogêneos.



### Classificador – *Ensemble*

Aplicamos o mesmo *ensemble* composto de SVM e Rede *Bayesiana* à massa de dados do SotM.

Segundo a Tabela 5.18, este é o único classificador que atinge virtualmente 100% de detecção dos ataques.

Tabela 5.18: Resultados para classificador Ensemble aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

	ens-w1	ens-w10	ens-w20	ens-w30	ens-w40	ens-w50	ens-w60	ens-w70	ens-w80	ens-w90	ens-w100
<b>TP</b>	591	707	710	712	713	713	713	714	714	714	714
<b>FP</b>	237	1620	1631	1776	1818	2079	2202	2438	2459	2460	2467
<b>TN</b>	7577	6194	6183	6038	5996	5735	5612	5376	5355	5354	5347
<b>FN</b>	123	7	4	2	1	1	1	0	0	0	0
<b>TP/Dia</b>	9,7	11,6	11,6	11,7	11,7	11,7	11,7	11,7	11,7	11,7	11,7
<b>FP/Dia</b>	3,9	26,6	26,7	29,1	29,8	34,1	36,1	40,0	40,3	40,3	40,4
<b>FPR</b>	0,03	0,21	0,21	0,23	0,23	0,27	0,28	0,31	0,31	0,31	0,32
<b>TPR</b>	0,83	0,99	0,99	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Tal performance é observável na Figura 5.26.

Quando a assimetria de custos atinge o valor de 30, a taxa de detecção aproxima-se da unidade, configurando o ponto ótimo de funcionamento deste classificador.

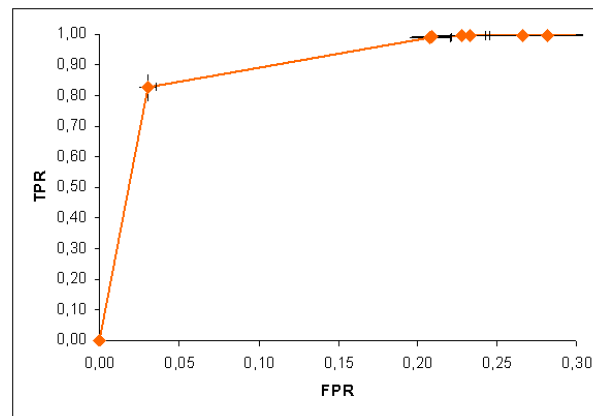


Figura 5.26: Curva ROC para classificador Ensemble aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation* e indicação de intervalo de confiança

A Figura 5.27 corrobora quão próximo da detecção da totalidade de ataques chega este classificador.

Trata-se de um classificador com uma excelente performance para a base em questão. A exemplo do que observamos nos experimentos anteriores, as características complementares da SVM e Rede *Bayesiana* colaboram para a criação de um classificador mais

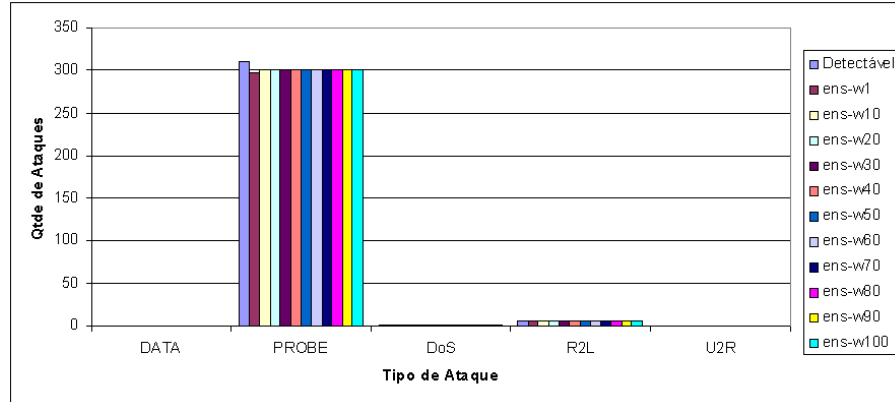


Figura 5.27: Detecção por tipo de ataque para classificador Ensemble aplicado a meta-alertas da base SotM utilizando-se *K-fold Cross-validation*

completo, sob diferentes circunstâncias e contextos (vide os resultados obtidos nos outros experimentos).

#### 5.4.4 Experimento 4: Alertas x Meta-Alertas

O modelo preconiza o uso de uma camada de fusão como entrada para a camada de classificação. A camada de fusão desempenha papel de provedor da consciência situacional (*situational awareness*) ao agrupar alertas em um mesmo meta-alerta; tal meta-alerta deveria carregar mais informações úteis para a classificação.

Todavia, cabem as perguntas: estaria um meta-alerta sendo um dado de entrada mais relevante do que um alerta para o processo de classificação? Poderia uma camada de classificação atuar sobre alertas e alcançar desempenho igual ou superior?

Para responder estas perguntas, reeditamos o experimento 1, dados da base DARPA com três semanas de treinamento e duas de testes, sobre alertas (ao invés de meta-alertas).

Esta reprodução sucedeu apenas com o classificador SVM. Os demais classificadores, dentro da implementação baseada em Weka, exigiam mais poder computacional — neste caso, memória RAM — do que o disponibilizado para o experimento.

Esta é a primeira conclusão deste novo experimento. Classificar alertas é computacionalmente mais caro do que operar sobre meta-alertas.

Os resultados obtidos com o classificador SVM são analisados sob duas óticas:

1. Granularidade de alertas: resultados obtidos são analisados dentro do domínio dos alertas. Cada alerta representa um possível ataque. Se um alerta é classificado corretamente, temos um positivo verdadeiro ou um negativo verdadeiro; caso contrário, temos um falso positivo ou um falso negativo.
2. Granularidade de meta-alertas: resultados obtidos pela classificação sobre alertas são agrupados nos seus respectivos meta-alertas (usando as mesmas técnicas de fusão descritas anteriormente). Em outras palavras, a classificação segue a granularidade dos alertas, mas a observação dos resultados vê o domínio dos meta-alertas. Dentro de uma filosofia conservadora, se um meta-alerta contém um alerta classificado como um ataque, todo o meta-alerta é considerado a identificação de um ataque. Esta granularidade simplifica a comparação de resultados com aqueles obtidos nos experimentos anteriores.

As próximas seções discorrem sobre tais experimentos.

#### Classificador - SVM - Granularidade Alerta

Ao reproduzir a classificação com a SVM sobre a massa de dados de alertas correspondentes às duas últimas semanas do desafio DARPA, obtemos os resultados dispostos na Tabela 5.19.

Tabela 5.19: Resultados para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em alertas

	svm-w1	svm-w10	svm-w20	svm-w30	svm-w40	svm-w50	svm-w60	svm-w70	svm-w80	svm-w90	svm-w100
<b>TP</b>	7809	8082	8082	8097	8097	8097	8111	8111	8551	8551	8551
<b>FP</b>	11197	12573	12573	13490	13490	13490	14129	14129	15835	15835	15835
<b>TN</b>	158417	157041	157041	156124	156124	156124	155485	155485	153779	153779	153779
<b>FN</b>	5883	5610	5610	5595	5595	5595	5581	5581	5141	5141	5141
<b>TP/Dia</b>	780,9	808,2	808,2	809,7	809,7	809,7	811,1	811,1	855,1	855,1	855,1
<b>FP/Dia</b>	1119,7	1257,3	1257,3	1349,0	1349,0	1349,0	1412,9	1412,9	1583,5	1583,5	1583,5
<b>FPR</b>	0,07	0,07	0,07	0,08	0,08	0,08	0,08	0,08	0,09	0,09	0,09
<b>TPR</b>	0,57	0,59	0,59	0,59	0,59	0,59	0,59	0,59	0,62	0,62	0,62

A taxa de detecção de ataques é superior, porém ela ocorre ao custo de um número inviavelmente elevado de falsos positivos. Mesmo sem a assimetria de custos de treinamento ( $w = 1$ ), temos um total de 11.197 falsos positivos, importando em 1.119 falsos positivos por dia. Tal número não apenas supera os limites estabelecidos pelo desafio DARPA, como inviabiliza seu uso prático.

Este resultado pode ser observado na Figura 5.28, onde verifica-se que a FPR atinge valores superiores a todos aqueles apresentados pelos experimentos anteriores. Esta taxa elevada aplicada sobre uma massa de dados mais ampla (há muito mais alertas do que meta-alertas, conforme indica a DIR), implica no elevado número de falsos positivos observados.

O número de positivos verdadeiros é tão elevado, acima de 7.800 alertas, que sua manipulação por uma equipe de segurança não é operacionalmente viável.

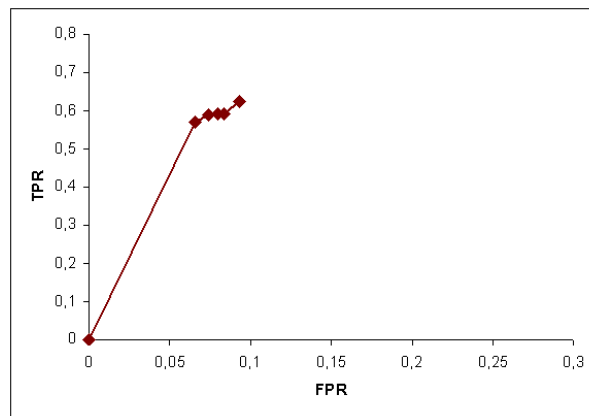


Figura 5.28: Curva ROC para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em alertas

### Classificador - SVM - Granularidade Meta-Alerta

Para emparelhar este experimento com os anteriores, tomamos os alertas classificados e aplicamos o algoritmo de fusão sobre eles. Os meta-alertas gerados recebem a classificação de incidente se um ou mais de seus alertas constituintes tiver sido classificado como um incidente.

Desta forma, temos um método de classificação orientado a alertas, com um mecanismo de observação orientado a meta-alertas.

O resultado deste experimento pode ser verificado na Tabela 5.20.

Tabela 5.20: Resultados para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em meta-alertas

	svm-w1	svm-w10	svm-w20	svm-w30	svm-w40	svm-w50	svm-w60	svm-w70	svm-w80	svm-w90	svm-w100
<b>TP</b>	41	73	74	74	74	74	74	74	74	74	74
<b>FP</b>	629	720	750	757	757	757	757	757	757	757	757
<b>TN</b>	7409	7318	7288	7281	7281	7281	7281	7281	7281	7281	7281
<b>FN</b>	244	212	211	211	211	211	211	211	211	211	211
<b>TP/Dia</b>	4,1	7,3	7,4	7,4	7,4	7,4	7,4	7,4	7,4	7,4	7,4
<b>FP/Dia</b>	62,9	72,0	75,0	75,7	75,7	75,7	75,7	75,7	75,7	75,7	75,7
<b>FPR</b>	0,08	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09	0,09
<b>TPR</b>	0,14	0,26	0,26	0,26	0,26	0,26	0,26	0,26	0,26	0,26	0,26

O número de positivos verdadeiros é inferior àquele observado com a SVM aplicada a meta-alertas. Ademais, o número de falsos positivos é bastante superior. Em suma, sua performance é inferior a que observamos com os outros classificadores aplicados aos meta-alertas. Isto também pode ser verificado na Figura 5.29.

A partir de uma assimetria de custos da ordem de 10, o classificador atinge seu máximo de detecção. Incrementos nesta variável apenas aumentam o número de falsos positivos.

Pela ótica de detecção de ataques, vemos que o classificador aplicado a alertas apresenta dificuldade em detectar ataques do tipo R2L e DATA, conforme mostra a Figura 5.30

De uma maneira geral nota-se que aplicar classificadores a alertas, ao invés de meta-alertas, apresenta quatro desvantagens:

1. Aumento do esforço computacional no processo de classificação.
2. Aumento do tempo de processamento da fase de classificação. Em verdade, a classificação de alertas em detrimento de metaalertas apresenta um incremento de duas ordens de grandeza no tempo de processamento. Tais resultados prejudicam o uso de classificação de alertas em temporeal.
3. Aumento do número de falsos positivos.
4. Redução do número de positivos verdadeiros.

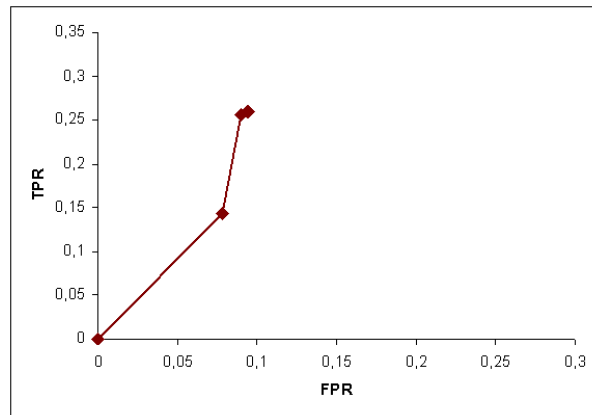


Figura 5.29: Curva ROC para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em meta-alertas

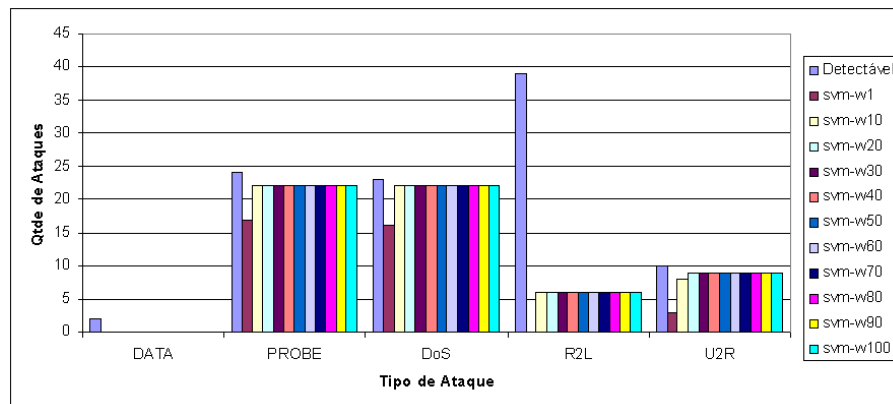


Figura 5.30: Detecção por tipo de ataque para classificador SVM aplicado a alertas da base DARPA utilizando-se três semanas de dados para treinamento e duas para testes, com observação em meta-alertas

### 5.4.5 Uma Análise Comparativa

Os experimentos realizados perfazem uma das maiores coberturas de testes em detecção de intrusão disponíveis. Tal cobertura engloba o uso de duas bases de dados, várias fontes de eventos, três tipos de classificadores independentes e um *ensemble*, além de um grande volume de alertas processados.

A maioria dos trabalhos publicados trabalham com subconjuntos das fontes de eventos disponíveis no desafio DARPA [22]. Outros utilizam apenas uma fração do volume de dados disponíveis: Faraoun e Boukelif utilizam 10% da base DARPA [16] [15]; Mukkamala et al utilizam 20% daqueles mesmos dados [41]).

Ademais, nossos experimentos apresentam melhor detecção dos ataques novos ou furtivos (*stealthy*) — *lsdomain*, *portsweep*, *queso*, *snmpget* e *ntfsdos* — em comparação com aqueles dos contendores do desafio DARPA, segundo levantamento de Lippmann et al [22]. Tais detecções ocorrem através da combinação do uso de classificadores adequados (especialmente a Rede *Bayesiana*) e de uma taxonomia especial que privilegia a extrapolação para detecção de novos ataques.

A taxa de detecção de ataques contra equipamentos Cisco atingiu performances de 100%, bem como obteve-se melhores resultados na identificação de ataques de *Probe* contra os equipamentos Linux, em comparação com os sistemas que responderam ao desafio DARPA [22].

A exemplo do que foi reportado por Kayacik e Zincir-Heywood [28], o número de falsos positivos gerados por um sistema de IDS como o Snort, contando com as mais novas regras disponíveis, é bastante elevado. Em todos os experimentos realizados, este componente colaborou com a maior parte dos alertas a serem tratados. Sem as técnicas de fusão e classificação, seria inviável a uma equipe de operações de segurança prover tratamento aos volumes de eventos gerados por um sistema Snort numa rede com grande volume de tráfego.

Dentre os ataques para os quais havia evidências (ditos detectáveis), a abordagem proposta apresenta taxas de detecção superiores àqueles apresentados por Lippmann et al [22], mesmo quando os ataques de teste não correspondem aos ataques usados para treinamento.

Outrossim, quando os classificadores utilizados são treinados com massas de dados mais representativas dos dados de teste/operação (como nos experimentos 2 e 3), as taxas de detecção superam os 87%, com baixo volume de falsos positivos.

Por fim, quando compara-se o resultado da classificação aplicada a alertas com aquela aplicada a meta-alertas, tem-se que a segunda apresenta resultados superiores (perfor-

mance e detecção), corroborando o modelo proposto.



# Capítulo 6

## Conclusão

A segurança da informação impõe-se como um desafio crescente às organizações. Novas ameaças, ferramentas automatizadas de ataques, tecnologias emergentes e vulnerabilidades crescentes tornam mais complexa a atividade de defender uma infraestrutura de tecnologia da informação.

Durante muitos anos o foco da segurança da informação residiu sobre a defesa de perímetros. Um novo contexto demarcado pela agilidade dos atacantes e a velocidade da proliferação de novas ameaças (padrões de ataques, vírus, etc) torna esta abordagem ineficaz.

Enfrentar estes desafios exige uma arquitetura de segurança mais flexível e dinâmica, pautada pela gestão e resposta rápida aos incidentes de segurança.

Todavia, responder rapidamente a incidentes exige o tratamento em tempo real dos diversos alertas de segurança. Tal tratamento envolve o processamento de centenas de milhares ou milhões de eventos por dia. Muitos destes eventos consistem em falsos positivos, drenando as energias de uma equipe de operações de segurança e tornando os custos envolvidos proibitivos.

Frente a estes desafios, o presente trabalho contribui das seguintes formas:

1. A definição de uma abordagem para o gerenciamento de alertas de segurança, representada por camadas de processamento hierarquicamente justapostas, as quais permitem a integração de novos sensores e a condução do problema de correlação de alertas de segurança com base na integração de componentes ou blocos (*building blocks*).
2. Proposição de uma taxonomia para tipos de alertas, visando aumentar a semântica de sua representação e suportar atividades de fusão e classificação.
3. Um estudo comparativo de quatro métodos (três básicos e um composto) de aprendizado de máquina para a classificação de meta-alertas de segurança entre incidentes

(ataques) ou alarmes falsos;

4. Utilizar a base de dados do DARPA [22] e SotM [47] para testar o modelo proposto e os métodos de aprendizado de máquina, permitindo que as conclusões estejam alicerçadas em ampla base de dados, abrangente coleta de eventos e múltiplas técnicas de classificação.

Esta abordagem ao problema permitiu uma solução para detecção de intrusão, contemplando a questão da redução de falsos positivos, apoiada nos seguintes preceitos:

1. Consciência situacional: visão integrada de alertas de segurança oriundos de diferentes fontes, cuja contraposição provê maior visibilidade da realidade presente sob tais alertas;
2. Fusão de alertas: redução da dimensionalidade do problema, mediante o emprego de conceitos de segurança (como taxonomias em segurança), com vistas a transformar conjuntos de alertas em meta-alertas;
3. Aprendizado de máquina sobre meta-alertas: uso de técnicas de aprendizado de máquina operando sobre alertas e meta-alertas, e não sobre o tráfego da rede ou *logs* de sistemas operacionais que deram origem aos mesmos. Tal abordagem difere da maior parte dos trabalhos correlatos presentes, onde as técnicas de aprendizado de máquina operavam diretamente sobre os dados mais primitivos da rede [59] ou dos sistemas operacionais [55].

O modelo proposto foi implementado e testado. Os resultados obtidos permitiram corroborar sua eficácia. Dentre as principais observações derivadas, destacam-se:

- Combinação da taxonomia de tipo de alertas, métodos de fusão e classificadores baseados em Redes *Bayesianas* propiciaram a extrapolação do comportamento do sistema, permitindo a detecção de ataques para os quais não havia indícios na base de aprendizado.
- Coletâneas de classificadores distintos, em especial SVM e Rede *Bayesiana*, apresentaram performances notáveis nos vários cenários de testes.
- Meta-alertas proveem melhor suporte a técnicas de classificação do que apenas alertas, corroborando o conceito da consciência situacional.

De uma forma resumida, percebe-se que o uso de técnicas de fusão desempenha papel fundamental no processamento de alertas de segurança. Elas permitem reduzir a quantidade de informações a serem analisadas por equipes de segurança. Tal redução pode ser

verificada nos altos índices de DIR (*Data to Information Reduction*) atingidos nos testes executados. Ademais, os meta-alertas resultantes são constituídos por um subconjunto dos atributos pertencentes aos alertas associados. Tal subconjunto preserva as principais informações usadas por um profissional de segurança, consistindo num mecanismo de redução de dimensionalidade que preserva o teor semântico dos dados. Os meta-alertas gerados permitem uma compreensão mais ampla do possível ataque do que a visão provida por seus respectivos alertas isoladamente. A vantagem do uso de meta-alertas sobre alertas na detecção de intrusões pode ser observado no experimento 4, no qual os resultados da classificação sobre meta-alertas mostraram-se superiores àqueles obtidos junto ao alertas, bem como mais eficientes em processamento (tornando-o mais adequado para uso em tempo-real).

Além de reduzir o volume de informações a serem analisados por uma equipe de operações de segurança, a abordagem é completada por uma camada de classificação de meta-alertas. Tal camada opera como uma forma de priorização dos meta-alertas a serem analisados. Meta-alertas classificados como possíveis ataques têm precedência de análise sobre aqueles que não o foram.

O uso de técnicas de aprendizado de máquina, aliado à taxonomia proposta e aos mecanismos de fusão, propiciou importantes resultados juntos às bases de dados de teste. Tais resultados não apenas mostraram a capacidade da abordagem em detectar ataques conhecidos, como também sua habilidade em indicar ataques que não faziam parte de sua massa de dados de aprendizado.

Em especial, a combinação de técnicas de aprendizado como SVM e Rede *Bayesiana*, por meio de um *ensemble*, parece prover um classificador de comportamento ambivalente, capaz de indicar ataques conhecidos e alguns novos ataques. Desta feita, o uso de técnicas de *ensemble* apresenta-se como uma importante alternativa — e aquela recomendada pelo autor — para a resolução de problemas de separação entre ataques reais e falsos alarmes.

Os conceitos aplicados neste modelo podem ser usados em campos adjacentes, como o processamento de eventos em sistemas de gerência de falhas (em redes de telecomunicações, por exemplo), e a detecção de ameaças à segurança física de ATMs e agências bancárias, entre outros.

O presente trabalho pode ainda ser estendido em algumas dimensões; mais notadamente:

- Estender o modelo para permitir a fusão de meta-alertas em meta-alertas de hierarquias superiores, permitindo mesclar classes de ataques distintas e ataques multi-estágios.
- Estudar a aplicação de agrupadores incrementais (*incremental clustering*) como alternativa ao algoritmo de fusão de alertas;

- Analisar novos critérios para agrupamento de eventos, como o uso de sessões, processos e usuários, por exemplo;
- Implementar um classificador baseado em Rede *Bayesiana* com topologia pré-definida e variáveis ocultas, como forma de obter respostas de maior inteligibilidade a especialistas humanos;
- Completar a Taxonomia proposta na Seção 4.2.3 para suporte a cenários mais amplos da detecção de intrusão;
- Estender a abordagem para sua integração com práticas de gerência de nível de serviço, permitindo identificar, a partir das relações entre itens de configuração de um CMDB (*Configuration Management Data Base*), os impactos de um incidente de segurança junto aos serviços prestados.

# Referências Bibliográficas

- [1] J. Allen, A. Christie, W. Fithen, J. McHugh, J. Pickel e E. Stoner. State of the practice of intrusion detection technologies. Relatório Técnico CMU/SEI-99TR-028, Carnegie Mellon University, 1999. Disponível em <http://www.sei.cmu.edu/pub/documents/99.reports/pdf/99tr028.pdf>; acessado em 06/04/2007.
- [2] F. R. Bach, D. Heckerman e E. Horvitz. On the path to an ideal roc curve: Considering cost asymmetry in learning classifiers. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, p. 9–16, Barbados, 2005.
- [3] T. Bowen, D. Chee, M. Segal, R. Sekar, T. Shanbhag e P. Uppuluri. Building survivable systems: An integrated approach based on intrusion detection and damage containment. In *DARPA Information Survivability Conference (DISCEX)*, 2000.
- [4] S. Boyer, O. Dain e R. Cunningham. Stellar: A fusion system for scenario construction and security risk assessment. In *Proceedings of the Third IEEE International Workshop on Information Assurance*, p. 105–116. IEEE Computer Society, 2005.
- [5] A. Buecker, W. Filip, S. Henley e S. Mohanty. Deployment guide series: Ibm tivoli security operations manager 4.1. Relatório técnico, IBM Tivoli Software, 2008.
- [6] D. J. Burroughs, L. F. Wilson e G. V. Cybenko. Analysis of distributed intrusion detection systems using bayesian methods. In *Proceedings of IEEE International Performance Computing and Communication Conference*, p. 329–334, Phoenix, AZ, EUA, 2002.
- [7] CERT.br. *Estatísticas dos Incidentes Reportados ao CERT.br*, 2009. Disponível em <http://www.cert.br/stats/incidentes>; acessado em 23/02/2009.
- [8] C.-C. Chang e C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Disponível em <http://www.csie.ntu.edu.tw/~cjlin/libsvm>; acessado em 03/02/2009.

- [9] E. Charniak. Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated. *AI Magazine*, 1991.
- [10] T. Chyssler, S. Burschka, M. Semling, T. Lingvall e K. Burbeck. Alarm reduction and correlation in intrusion detection systems. In *Detection of Intrusions and Malware & Vulnerability Assessment workshop (DIMVA)*, p. 9–24, Dortmund, Alemanha, 2004.
- [11] K. Cox e C. Gerg. *Managing Security with Snort and IDS Tools*. O'Reilly, 2004.
- [12] H. Debar, D. Curry e B. Feinstein. The intrusion detection message exchange format (idmef). Internet experimental RFC 4765, 2007. Disponível em <http://tools.ietf.org/html/rfc4765>; acessado em 01/07/2008.
- [13] H. Debar, M. Dacier e A. Wespi. Towards a taxonomy of intrusion-detection systems. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, 31(9):805–822, 1999.
- [14] A. Faour, P. Leray e C. Foll. Reseaux bayesiens pour le filtrage d alarmes dans les systemes de detection d intrusion. In *Atelier Modeles Graphiques Probabilistes, 5emes journees d Extraction et Gestion des Connaissances (EGC 2005)*, 2005.
- [15] K. M. Faraoun e A. Boukelif. Neural networks learning improvement using the k-means clustering algorithm to detect network intrusions. *International Journal of Computational Intelligence*, 2006.
- [16] K. M. Faraoun e A. Boukelif. Securing network traffic using genetically evolved transformations. *Malaysian Journal of Computer Science (ISSN 0127-9084)*, 19, 2006.
- [17] Folha Online. *Ocidente deve se preparar melhor para ataques virtuais, dizem especialistas*, 2008. Disponível em <http://www1.folha.uol.com.br/folha/informatica/ult124u441496.shtml>; acessado em 17/03/2009.
- [18] Folha Online. *Invasão de hacker faz Citibank recolher cartões no Brasil*, 2009. Disponível em <http://www1.folha.uol.com.br/folha/dinheiro/ult91u500646.shtml>; acessado em 23/02/2009.
- [19] Folha Online. *Presidência uruguaia é alvo recorrente de piratas virtuais*, 2009. Disponível em <http://www1.folha.uol.com.br/folha/informatica/ult124u497295.shtml>; acessado em 02/02/2009.

- [20] S. Garfinkel e G. Spafford. *Practical Unix and Internet Security*. O'Reilly, 1996.
- [21] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, 34:579–595, November/December 1995.
- [22] J. W. Haines, R. P. Lippmann, D. J. Fried, E. Tran, S. Boswell e M. A. Zissman. The 1999 darpa off-line intrusion detection evaluation. *Computer Networks. The International Journal of Computer and Telecommunications Networking*, 34:579–595, 2000.
- [23] J. Hale e P. Brusil. Secur(e/ity) management: A continuing uphill climb. *Journal of Network and Systems Management*, 15(4):525–553, 2007.
- [24] The Honeynet Project. *The Honeynet Project Challenges*. <http://www.honeynet.org/challenges>; acessado em 03/01/2009.
- [25] ABNT NBR ISO/IEC 17799:2005. *Tecnologia da informação — Técnicas de segurança — Código de prática para a gestão da segurança da informação*, 2005.
- [26] itSMF. *itSMF International - The IT Service Management Forum*. Disponível em <http://www.itsmf.org>; acessado em 03/05/2009.
- [27] K. Julisch. Clustering intrusion detection alarms to support root cause analysis. *ACM Transactions on Information and System Security*, 6:443–471, 2003.
- [28] H. G. Kayacik e A. N. Zincir-Heywood. Using intrusion detection systems with a firewall: Evaluation on darpa 99 dataset. Relatório técnico, NIMS Technical Report 062003, 2003.
- [29] W. Lee e S. J. Stolfo. Data mining approaches for intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, San Antonio, TX, EUA, 1998.
- [30] W. Lee, S. J. Stolfo, P. K. Chan, E. Eskin, W. Fan, M. Miller, S. Hershkop e J. Zhang. Real time data mining-based intrusion detection. In *Proc. Second DARPA Information Survivability Conference and Exposition*, p. 85–100, Anaheim, EUA, 2001.
- [31] P. Liu, W. Zang e M. Yu. Incentive-based modeling and inference of attacker intent, objectives, and strategies. *ACM Transactions on Information and System Security (TISSEC)*, 8:78–118, 2005.
- [32] M. E. Locasto, J. J. Parekh, S. Stolfo, A. D. Keromytis, T. Malkin e V. Misra. Collaborative distributed intrusion detection. Relatório técnico, CU Tech Report CUCS-012-04, 2004.

- [33] M. V. Mahoney e P. K. Chan. Learning nonstationary models of normal network traffic for detecting novel attacks. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 376–385. ACM, 2002.
- [34] L. A. F. Martimiano. *Sobre a estruturação de informações em sistemas de segurança: o uso de ontologia*. Doutorado em ciências da computação e matemática computacional, Universidade de São Paulo, 2006.
- [35] L. A. F. Martimiano e E. dos Santos Moreira. The evaluation process of a computer security incident ontology. In *2nd Workshop on Ontologies and their Applications (WONTO'2006), International Joint Conference.*, 2006.
- [36] J. P. Martin-Flatin, G. Jakobson e L. Lewis. Event correlation in integrated management: Lessons learned and outlook. *Journal of Network and Systems Management*, 15(4):481–502, December 2007.
- [37] S. Mathew, C. Shah e S. Upadhyaya. An alert fusion framework for situation awareness of coordinated multistage attacks. In *Proceedings of the Third IEEE International Workshop on Information Assurance*, p. 95–104, University of Maryland, EUA, 2005.
- [38] McAfee. *Cybercrime custa US\$ 1 trilhão às empresas*, 2009. Disponível em <http://www.itweb.com.br/noticias/index.asp?cod=54552>; acessado em 02/02/2009.
- [39] Y. ming Ma, Z. tang Li, J. Lei, L. Wang e D. Li. *An Intelligent Agent-Oriented System for Integrating Network Security Devices and Handling Large Amount of Security Events*, volume 4430 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2007.
- [40] S. Mukkamala e A. H. Sung. Feature ranking and selection for intrusion detection systems using support vector machines. In *Proceedings of the Second Digital Forensic Research Workshop*, 2002.
- [41] S. Mukkamala, A. H. Sung e A. Abraham. Intrusion detection using ensemble of soft computing. In *Paradigms, Advances in Soft Computing*, p. 239–248. Springer Verlag, 2003.
- [42] J. Nash. Equilibrium points in n-person games. In *Proceedings of the National Academy of Sciences*, p. 48–49, 1950.
- [43] R. E. Neapolitan. *Learning Bayesian Networks*. Prentice Hall, 2004.



- [44] P. G. Neumann e P. A. Porras. Experience with EMERALD to date. In *Proceedings 1st USENIX Workshop on Intrusion Detection and Network Monitoring*, p. 73–80, Santa Clara, CA, EUA, 2005.
- [45] P. Ning, Y. Cui, D. S. Reeves e D. Xu. Techniques and tools for analyzing intrusion alerts. *ACM Transactions on Information and System Security (TISSEC)*, 7:274–318, 2004.
- [46] S. Ohta, R. Kurebayashi e K. Kobayashi. Minimizing false positives of a decision tree classifier for intrusion detection on the internet. *Journal of Network and Systems Management*, 16(4):399–419, December 2008.
- [47] T. H. Project. *Scan of the Month 34*. Disponível em <http://old.honeynet.org/scans/scan34>; acessado em 21/07/2008.
- [48] T. H. Project. *Know Your Enemy : Learning about Security Threats (2nd Edition)*. Addison-Wesley Professional, 2004.
- [49] S. J. Russell e P. Norvig. *Artificial Intelligence — A Modern Approach*. Prentice Hall, 1995.
- [50] B. Sabata. Evidence aggregation in hierarchical evidential reasoning. In *UAI Applications Workshop, Uncertainty in AI 2005*.
- [51] B. Sabata e C. Ornes. Multisource evidence fusion for cyber-situation assessment. In *Proc. SPIE Vol. 6242, 624201 (Apr. 18, 2006)*, Orlando, FL, EUA, 2006.
- [52] J. Scambray, S. McClure e G. Kurtz. *Hackers Exposed*. Makron Books, 2001.
- [53] B. Scholkopf e A. J. Smola. *Learning with Kernels*. The MIT Press, 2002.
- [54] Snort. *Snort the de facto standard for intrusion detection/prevention*. Disponível em <http://www.snort.org>; acessado em 03/05/2009.
- [55] G. Tandon e P. Chan. Learning rules from system call arguments and sequences for anomaly detection. In *ICDM Workshop on Data Mining for Computer Security (DMSEC)*, p. 20–29, Melbourne, FL, EUA, 2003.
- [56] M. Tian, S.-C. Chen, Y. Zhuang e J. Liu. Using statistical analysis and support vector machine classification to detect complicated attacks. In *International Conference on Machine Learning and Cybernetics*, volume 5, p. 2747–2752. IEEE, 2004.
- [57] The University of Waikato. *WEKA*. <http://www.cs.waikato.ac.nz/ml/weka/>; acessado em 01/02/2009.

- [58] A. Valdes e K. Skinner. Probabilistic alert correlation. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, p. 54–68, Davis, CA, EUA, 2001.
- [59] A. Valdes, K. Skinner e S. International. Adaptive, model-based monitoring for cyber attack detection. In *Recent Advances in Intrusion Detection (RAID 2000)*, p. 80–92. Springer-Verlag, 2000.
- [60] G. Vigna, S. T. Eckmann e R. A. Kemmerer. The stat tool suite. In *Proceedings of DISCEX 2000*, Hilton Head, SC, USA, 2000. IEEE Computer Society Press.
- [61] C. wei Hsu, C. chung Chang e C. jen Lin. A practical guide to support vector classification. Relatório técnico, Department of Computer Science, National Taiwan University, 2007. Disponível em <http://www.csie.ntu.edu.tw/~cjlin/>; acessado em 10/03/2009.
- [62] I. H. Witten e E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques (Second Edition)*. Morgan Kaufmann, 2000.
- [63] E. D. Zwicky, S. Cooper e D. B. Chapman. *Building Internet Firewalls*. O'Reilly, 2000.

# Apêndice A

## Ataques no Desafio DARPA

Tabela A.1: Ataques no Desafio DARPA

Classe de Ataque	Tipo de Ataque	Frequência
DATA	framespoofer	1
DATA	secret	8
DoS	apache2	3
DoS	back	6
DoS	crashiis	10
DoS	dosnuke	4
DoS	land	4
DoS	mailbomb	6
DoS	neptune	6
DoS	pod	6
DoS	processtable	3
DoS	selfping	3
DoS	smurf	5
DoS	sshprocesstable	1
DoS	syslogd	4
Continua na próxima página		

TabelaA.1 – continuação da página anterior

<b>Classe de Ataque</b>	<b>Tipo de Ataque</b>	<b>Frequência</b>
DoS	teardrop	3
DoS	udpstorm	2
DoS	warezclient	3
DoS	warezmaster	1
PROBE	arppoisson	4
PROBE	illegalsniffer	2
PROBE	ipsweep	10
PROBE	ls	2
PROBE	mscan	1
PROBE	NTInfoscan	4
PROBE	perl	1
PROBE	portsweep	18
PROBE	queso	4
PROBE	resetscan	1
PROBE	satan	4
PROBE	tcpreset	3
R2L	anypw	1
R2L	dict	1
R2L	ftpwrite	4
R2L	guessftp	2
R2L	guesstelnet	4
R2L	guest	3
R2L	httptunnel	5
R2L	imap	2
R2L	loadmodule	2
R2L	named	3
R2L	ncftp	5
R2L	netbus	3
Continua na próxima página		

TabelaA.1 – final

<b>Classe de Ataque</b>	<b>Tipo de Ataque</b>	<b>Frequência</b>
R2L	netcat	4
R2L	ntfsdos	3
R2L	phf	6
R2L	ppmacro	3
R2L	sendmail	2
R2L	snmpget	4
R2L	sqlattack	3
R2L	sshtrojan	2
R2L	sshtrojanInstall	1
R2L	xlock	3
R2L	xsnoop	3
U2R	casesen	3
U2R	eject	5
U2R	fdformat	3
U2R	ffbconfig	2
U2R	loadmodule	3
U2R	perl	7
U2R	ps	6
U2R	ps attack	1
U2R	sechole	3
U2R	xterm	2
U2R	xterm1	1
U2R	yaga	4

# Apêndice B

## Ataques no SotM

Tabela B.1: Ataques no SotM

Classe de Ataque	Tipo de Ataque	Frequência
DOS	invalid-web-request	1
PROBE	awstats.pl-exploit	9
PROBE	CodeRed	40
PROBE	connect-scan	15
PROBE	fp30reg-scan	48
PROBE	MySQL-bot	26
PROBE	nimda	89
PROBE	nsiislog-scan	36
PROBE	NULL.printer-scan	15
PROBE	PHP-scan	1
PROBE	proxy-req	19
PROBE	scan-multiple	1
PROBE	sumthin	12
R2L	awstats.pl-exploit	1
R2L	gethostbyname-BoF	4

## Apêndice C

### Tipos de Alertas no Desafio DARPA

Tabela C.1: Tipos de Alertas no Desafio DARPA

<b>Tipo de Alerta</b>	<b>Frequência</b>
file.config:access	8
file.config:access.suspicious	6
file.config:rename	27
file.dump:access	4
file.dump:create	28
file.log:alter	20
file.multimedia:download	32
file.priv:susp_access	13455
file.url:invalid	659
protocol.bgp:traffic.suspicious	6
protocol.dns:exploit.bof	6
protocol.dns:probe.scan	12
protocol.ftp:exploit.bof	12
protocol.ftp:login.fail.user	210
protocol.ftp:privacy.violation	5
Continua na próxima página	

TabelaC.1 – continuação da página anterior

<b>Tipo de Alerta</b>	<b>Frequência</b>
protocol.ftp:probe.scan	65
protocol.ftp:relay	28
protocol.ftp:traffic.suspicious	42
protocol.http:probe.scan	63
protocol.http:traffic.suspicious	134
protocol.icmp.port:unreachable	43319
protocol.icmp:ping	31224
protocol.icmp:probe.scan	14000
protocol.icmp:redirect	281
protocol.icmp:reply	32488
protocol.icmp:traffic.suspicious	205
protocol.icmp:unreachable	78
protocol.ip:malware.trojan	12
protocol.ip:traffic.bad	1787
protocol.ip:traffic.suspicious	25
protocol.irc:traffic.suspicious	21931
protocol.netbios.admin:access.suspicious	10
protocol.netbios.drive:access.suspicious	20
protocol.netbios.session:access.suspicious	16
protocol.rpc:probe.scan	400
protocol.snmp:probe.scan	288905
protocol.tcp.port:probe.scan	1421
protocol.tcp.port:probe.sweep	147
protocol.tcp:probe.scan	2363
protocol.telnet:login.fail	5
protocol.telnet:login.fail.user	3204
protocol.telnet:reachable	18840
protocol.telnet:traffic.suspicious	2
Continua na próxima página	



TabelaC.1 – continuação da página anterior

<b>Tipo de Alerta</b>	<b>Frequência</b>
protocol.udp.port:probe.scan	10
protocol.udp.port:probe.sweep	757
protocol.x11:traffic.suspicious	51
security.db.alias:scan	37
security.db.domain:scan	117
security.db.group:scan	19
security.db.user:scan	3336
system.db:exploit.bof	1
system.db:ping	1
system.os:exploit.bof	20
system.os:exploit.code_injection	16
system.os:exploit.fstr	39
system.os:malware.backdoor.attempt	32
system.os:malware.worm	6
system.os:probe.scan	63
system.os:restart	56
system.service:exploit.bof	3
system.tool:access.suspicious	615
system.web.api:access.suspicious	12
system.web.bin.cat:access.suspicious	15
system.web.bin.newdsn:access.suspicious	14
system.web.bin:access.suspicious	1590
system.web.cfg.passwd:access.suspicious	10
system.web.cgi.perl:access.suspicious	56
system.web.cgi.phf:access.suspicious	708
system.web.cgi.phf:exploit.rexec	15
system.web.cgi:access.suspicious	4534
system.web.dir.bin:access.suspicious	478
Continua na próxima página	

TabelaC.1 – final

<b>Tipo de Alerta</b>	<b>Frequência</b>
system.web.dir.cgi:access.suspicious	14
system.web.dir:access.suspicious	2422
system.web.dir:disclosure	6410
system.web.dll:access.suspicious	294
system.web.script.idc:access.suspicious	14
system.web.script:access.suspicious	10
system.web.script:download	16
system.web.sp:access.suspicious	405
system.web:exploit.bof	23
system.web:probe.scan	1047
system.web:request.forbidden	2244
user.admin:login.attempt	1
user.anonymous:login.attempt	5115
user.gen:login.fail	119
user.gen:login.fail.pwd	252
user.gen:login.fail.user	114
user.gen:login.ok	939
user.remote:login.fail	2
user.web:privacy.tracking	61551

## Apêndice D

### Tipos de Alertas no SotM

Tabela D.1: Tipos de Alertas no SotM

<b>Tipo de Alerta</b>	<b>Frequência</b>
file.url:inexistent	2430
protocol.ftp:probe.scan	4
protocol.ftp:relay	2
protocol.ftp:traffic.suspicious	1
protocol.http.method:unallowed	59
protocol.http:probe.scan	343
protocol.http:traffic.suspicious	2109
protocol.icmp.port:unreachable	8043
protocol.icmp:ping	4518
protocol.icmp:probe.scan	464
protocol.icmp:reply	3333
protocol.icmp:traceroute	27
protocol.icmp:traffic.suspicious	221
protocol.icmp:unreachable	41
protocol.ip.port:suspicious	17282
Continua na próxima página	

TabelaD.1 – continuação da página anterior

<b>Tipo de Alerta</b>	<b>Frequência</b>
protocol.ip:malware.trojan	28
protocol.ip:privacy.violation	4
protocol.ip:probe.scan	19
protocol.ip:probe.scan.stealth	15
protocol.ip:traffic.suspicious	3505
protocol.irc:access.suspicious	7
protocol.irc:malware.trojan	1572
protocol.irc:malware.worm	148
protocol.irc:traffic.suspicious	5786
protocol.rpc:exploit.fstr	13
protocol.rpc:probe.scan	91
protocol.smtp:probe.scan	49
protocol.smtp:relay.fail	65
protocol.snmp:probe.scan	30
protocol.ssh:probe.scan	418
protocol.tcp:traffic.suspicious	8
protocol.telnet:reachable	1
protocol.tftp:traffic.suspicious	1
system.db:dos	22
system.db:exploit.bof	4438
system.db:malware.worm	8876
system.db:ping	1
system.db:probe.scan	3514
system.os:exploit.bof	12
system.os:exploit.code_injection	1710
system.os:restart	1
system.service.logger:restart	8
system.service:probe.recon	18
Continua na próxima página	

TabelaD.1 – final

<b>Tipo de Alerta</b>	<b>Frequência</b>
system.service:shutdown	11
system.service:startup	40
system.web.api:access.suspicious	37
system.web.bin.cat:access.suspicious	1
system.web.bin.newdsn:access.suspicious	1
system.web.bin:access.suspicious	974
system.web.cgi.perl:access.suspicious	4
system.web.dir.bin:access.suspicious	428
system.web.dir.cgi:access.suspicious	1
system.web.dll:access.suspicious	441
system.web.script.idc:access.suspicious	1
system.web:exploit.bof	43
system.web:log.error	54
system.web:privacy.violation	718
system.web:probe.scan	74
system.web:request.fail	110
system.web:request.forbidden	403
user.admin:login.ok	2
user.anonymous:login.attempt	3
user.gen:login.fail	702
user.gen:login.ok	89
user.gen:logoff	88
user.web:privacy.tracking	8