SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito: 25/02/2002

Assinatura: *[Handwritten Signature]*

Métodos alternativos de reconhecimento de padrões para sistemas de detecção de intrusão

Daniel Rodrigues Ambrósio

Orientador: *Prof. Dr. Edson dos Santos Moreira*

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Fevereiro/2002

A Comissão Julgadora:

Prof. Dr. Edson dos Santos Moreira

A stylized handwritten signature consisting of a large 'M' followed by a horizontal line.

Profa. Dra. Regina Melo Silveira

A handwritten signature that appears to read 'RM Silveira'.

Prof. Dr. Adilson Gonzaga

A handwritten signature that appears to read 'Adilson Gonzaga'.

*Aos meus pais e irmão,
as pessoas que mais amo no mundo.*

AGRADECIMENTOS

Gostaria de usar esse espaço para expressar meus mais sinceros agradecimentos às pessoas que contribuíram, cada uma à sua maneira, para a realização desse trabalho:

Prof. Dr. Edson dos Santos Moreira: Muito obrigado pelo enorme voto de confiança, acreditando que eu seria capaz de terminar esse trabalho mesmo estando a mais de 9000 km de distância. Seu incentivo, apoio e palavras de motivação foram fundamentais nos momentos mais difíceis.

JuRico: Você foi fundamental para fazer com que esse tempo longe das pessoas que amamos não fosse tão difícil. Seu companheirismo, amizade e carinho 100% presentes me sustentaram o tempo todo. Devo muito a você!

JuMoraes: Pelo árduo trabalho de revisar e corrigir nos mínimos detalhes TODAS as referências, uma por uma, editora por editora, autor por autor, vírgula por vírgula. Sua paciência e capacidade também estão evidenciadas neste trabalho.

Lu ("maringirl do coração"): Você ainda não passou por sua grande provação até esse momento em que escrevo essas linhas, porém te agradeço por antecipação pela paciência e perseverança em encaminhar oito cópias impressas desse documento às devidas autoridades!

Rudinei: Pelas dicas valiosas nessa reta final, sua orientação e experiência foram para mim de muita valia. Essas poucas linhas de agradecimento não se comparam a todas as linhas que você ajudou a desenvolver nesse trabalho.

Jan: Pelo seu súbito ressurgimento das cinzas e pelo detalhado relatório de erros encontrados durante a revisão. Obrigado por investir parcialmente seu tempo de descanso em mim!! E viva as férias universitárias!!

Beth, Laura e Marília: Funcionárias como vocês são difíceis de se encontrar; pessoas como vocês infelizmente são poucas as que conhecemos em nossas vidas. Muito obrigado por toda a ajuda, paciência e informação dispensadas durante esses anos que nos conhecemos.

Amigos do Grupo de Segurança: Agradeço a vocês pelos ótimos momentos que passamos nos invernos do Laboratório Intermídia. As discussões técnicas e principalmente as NÃO-técnicas alimentaram a esperança que esse dia chegaria! *Mauro*, seu trabalho e sua experiência

serviram de alicerce para esse documento; *Stênio*, não se preocupe, voltarei para o Brasil para assistir um Palmeiras X Cruzeiro ao seu lado; *Dalton*, suas respostas praticamente em tempo real aos meus e-mails foram de muita valia; *Chico e Milagres*: Chico, avise o Milagres que o milagres.com está muito legal; *Tom e Brandão*: obrigado a vocês pelo compartilhamento de conhecimento.

$Fx F^y x$, onde $x \in ['a', 'e']$; $y = ^$ para $x = 'e'$; $y = ^$ para $x = 'a'$: Obrigado pelos megabytes diários de e-mails e pela revisão desse trabalho, mesmo que essa última tenha sido feita mediante a oferta do recebimento de cervejas alemãs como pagamento.

Funcionárias da Biblioteca do ICMC: Proponho o uso de uma placa na entrada biblioteca: “Prazer em bem servir”. É incrível a simpatia, boa vontade e alegria com que somos atendidos quando as procuramos!

CNPq: Obrigado pelo apoio financeiro, fundamental para a dedicação integral ao trabalho de mestrado.

Rolf F. Katzenberger: Thanks a lot for your understanding and support, specially during the latest moments of the development of this work. Danke Schön!

E “*last but not least*” eu gostaria de agradecer aos idealizadores e projetistas da ARPANET do final dos anos 60, pois sem a atual infra-estrutura da Internet teria sido impossível realizar esse trabalho à distância.

SUMÁRIO

1	<i>Introdução</i>	1
1.1	Histórico do Grupo	2
1.2	Estrutura da Dissertação	4
2	<i>Segurança Computacional</i>	5
2.1	Definições	5
2.2	Intrusos e Intrusões	6
2.3	Aspectos de Segurança	12
2.4	Considerações Finais	14
3	<i>Sistemas de Detecção de Intrusão</i>	17
3.1	Definições	17
3.2	Classificação dos SDI	19
3.2.1	Tratamento dos Dados	19
3.2.2	Método de Detecção	23
3.2.3	Tipo de Reação	27
3.3	Componentes de um Sistema de Detecção de Intrusão	29
3.4	Considerações Finais	29
4	<i>Métodos de Reconhecimento de Intrusões</i>	31
4.1	Sistema Imunológico Computacional para Detecção de Intrusões	31
4.1.1	Projeto do Sistema	32
4.1.2	Resultados Experimentais e Análise	36
4.2	Filtros Baseados em Reconhecimento de Padrões	40
4.2.1	Detecção de Intrusão como um problema de Reconhecimento de Padrões	40
4.2.2	Probabilidade de Reconhecimento	42
4.2.3	Resultados Experimentais	43
4.3	Aprendizado de Comportamento de Programas	46
4.3.1	Rede Neural Recorrente Elman	46
4.3.2	Transdutor de String	48

4.3.3 Testador de Estados	49
4.3.4 Performance dos Algoritmos	50
4.4 Detecção de Intrusões Distribuídas e Desconhecidas	53
4.4.1 MADAM ID	53
4.4.2 Visão Geral do CIDF	55
4.4.3 MADAM ID como um Mecanismo de Modelagem do CIDF	56
4.4.4 Experimentos	59
4.5 Minerando Dados de Execução	61
4.5.1 Construindo Autômatos Finitos a partir de Informação de Auditoria	61
4.5.2 Mineração de Dados usando MEF ensinadas	66
4.5.3 Experimentos	67
4.6 Considerações Finais	71
5 Análise e Proposta	73
5.1 Arquitetura do Sistema	74
5.2 Análise dos Trabalhos Apresentados	76
5.2.1 Sistema Imunológico Computacional para Detecção de Intrusões	76
5.2.2 Filtros Baseados em Reconhecimento de Padrões	79
5.2.3 Aprendizado de Comportamento de Programas	82
5.2.4 Detecção de Intrusões Distribuídas e Desconhecidas	83
5.2.5 Minerando Dados de Execução	84
5.2.6 Classificação dos Métodos	86
5.3 Conclusões	87
5.4 Trabalhos Futuros	88
6 Referências	91

LISTA DE FIGURAS

<i>Figura 1: Intruso passivo (a) e intruso ativo (b)</i>	7
<i>Figura 2: Evolução das ferramentas e técnicas de ataque e do conhecimento dos atacantes</i>	8
<i>Figura 3: Invasões reportadas ao CERT</i>	11
<i>Figura 4: Vulnerabilidades reportadas ao CERT</i>	12
<i>Figura 5: Posicionamento de sensores de um SDI de Rede</i>	21
<i>Figura 6: Ciclo de vida de um anticorpo CDIS</i>	34
<i>Figura 7: Resultados de testes para a primeira fase, com conjuntos de dados pequenos</i>	39
<i>Figura 8: Busca de assinaturas como um problema de reconhecimento de padrão aproximado múltiplo</i>	41
<i>Figura 9: Resultados dos testes experimentais</i>	43
<i>Figura 10: Topologia padrão feed-forward e uma Rede de Elman</i>	47
<i>Figura 11: O processo de mineração de dados para a construção de modelos</i>	54
<i>Figura 12: As interfaces CIDF dos SDI</i>	58
<i>Figura 13: Ilustração da fusão de estados</i>	63
<i>Figura 14: Escores para programas usados no treinamento para o Experimento 1</i>	68
<i>Figura 15: Escores para os programas não usados no treinamento para o Experimento 1</i>	69
<i>Figura 16: Escores para programas usados no treinamento para o Experimento 3</i>	70
<i>Figura 17: Escores para os programas não usados no treinamento para o Experimento 3</i>	71
<i>Figura 18: Modelagem em camadas para o sistema baseado em agentes.</i>	75
<i>Figura 19: Ciclo de vida de um anticorpo CIDS com suporte a agentes</i>	78
<i>Figura 20: Proposta para o uso de anticorpos</i>	79
<i>Figura 21: Agente de Filtragem na Camada de Vigilância</i>	81

LISTA DE TABELAS

<i>Tabela 1: Características do Pacote</i>	33
<i>Tabela 2: Parâmetros principais para os três padrões</i>	45
<i>Tabela 3: Lista de programas monitorados pelo autômato de detecção de intrusão</i>	51
<i>Tabela 4: Um exemplo de padrões de intrusão</i>	55
<i>Tabela 5: Classificação dos métodos apresentados</i>	86

LISTA DE ABREVIATURAS

AF	Autômato Finito
BSM	<i>Basic Security Module</i>
CDIS	<i>Computer Defense Immune System</i>
CERT	<i>Computer Emergency Response Team</i>
CIDF	<i>Common Intrusion Detection Framework</i>
CIS	<i>Computational Immune System</i>
CISL	<i>Common Intrusion Specification Language</i>
CPU	<i>Central Processing Unit</i>
CSI	<i>Computer Security Institute</i>
DARPA	<i>Defense Advanced Research Projects Agency</i>
DDoS	<i>Distribute Denial of Service</i> (também no Glossário)
DOD	<i>Department of Defense</i>
DoS	<i>Denial of Service</i> (também no Glossário)
DTD	<i>Document Type Definition</i>
GIDO	<i>Generalized Intrusion Detection Object</i>
GUI	<i>Graphical User Interface</i>
HIDS	<i>Host Intrusion Detection System</i> ou <i>Host-based Intrusion Detection System</i>
HMM	<i>Hidden Markov Model</i>
ICMC	Instituto de Ciências Matemáticas e de Computação
ICMP	<i>Internet Control Message Protocol</i>
IP	<i>Internet Protocol</i>
LL	<i>Lincoln Laboratory</i>
MEF	Máquina de Estados Finitos
MIT	<i>Massachusetts Institute of Technology</i>
NFR	<i>Network Flight Recorder</i>
NIDS	<i>Network Intrusion Detection System</i> ou <i>Network-based Intrusion Detection System</i>
NNIDS	<i>Network Node Intrusion Detection System</i>
PCN	Política de Continuidade de Negócios
RNA	Redes Neurais Artificiais
SDI	Sistema de Detecção de Intrusão
SID	<i>Semantic Identifier</i>
SNMP	<i>Simple Network Management Protocol</i>

TCP	<i>Transfer Control Protocol</i>
TFN	<i>Tribal Flood Network</i>
UDP	<i>User Datagram Protocol</i>
XML	<i>eXtensible Markup Laguage</i>

RESUMO

Esse trabalho apresenta um estudo das atuais alternativas para reconhecimento de padrões aplicados à segurança computacional, mais precisamente aos Sistemas de Detecção de Intrusão. São estudadas as abordagens propostas por cinco trabalhos publicados em simpósios internacionais e os resultados experimentais apresentados por cada um. Esses trabalhos fazem uso de variados métodos para reconhecimento de padrões, como Redes Neurais Artificiais, Algoritmos Evolucionários e Mineração de Dados entre outras. Uma avaliação crítica de cada uma das abordagens é apresentada, e são feitas propostas de implementação de alguns dos métodos avaliados, objetivando suportar um Sistema de Detecção de Intrusões baseado em agentes móveis. Esse sistema foi modelado em estudos realizados pelo Grupo de Segurança Computacional do Instituto de Ciências Matemáticas e de Computação.

ABSTRACT

In this document we present a study of recent alternatives for pattern recognition applied to computer security, more specifically to Intrusion Detection Systems. We study the approaches proposed by five papers published in international symposiums and the experimental results each of them achieved. These studies make use of diverse pattern recognition methods, such as Artificial Neural Networks, Evolutionary Algorithms and Data Mining among others. We present a critical evaluation of each of these approaches, and some implementation suggestions for the evaluated methods, aiming to support an Intrusion Detection System based on Mobile Agents. The referred system was modeled in previous researches by the members of the Computational Security Group of the Instituto de Ciências Matemáticas de São Carlos.

1

1 INTRODUÇÃO

Como o número de usuários que dependem da segurança computacional passou de apenas algumas organizações que lidam com dados secretos para todo o mundo conectado à Internet, as necessidades de segurança computacional também acompanharam essa mudança radical. No mundo atual de redes internacionais e comércio eletrônico, todo sistema computacional é um alvo potencial. Raramente se passa um mês em que não surjam notícias na mídia de uma grande empresa ou organização que tenha tido seus sistemas invadidos.

Os altos índices de informatização, conectividade, negócios pela Internet e compartilhamento de dados tornaram a informação um dos bens mais valiosos e vulneráveis das empresas. Com isso, incidentes nas redes de computadores passaram a afetar diretamente os resultados do negócio e o valor das empresas.

Apesar dos riscos, o interesse em redes de computadores e na Internet nunca foi maior. O número de computadores ligados à Internet tem dobrado a cada ano por aproximadamente uma década¹. Por todas as indicações, é provável que esse crescimento continue por alguns anos. Com esse crescente interesse na Internet, é natural que o interesse na proteção da informação também cresça.

A tecnologia de segurança de rede mais utilizada hoje em dia é o *firewall* [BERNARDES, 1999; MÓDULO, 2001]. Este sistema previne a entrada não autorizada utilizando-se de

¹ Informação retirada do documento "Digital Economy 2000" do U.S. Department of Commerce. Disponível online em <http://www.esa.doc.gov/dc2k.htm> (acesso em 29/01/2002).

mecanismos de controle de acesso externo. Porém não existe nenhum sistema que possa ser considerado o estado da arte em matéria de proteção e ainda, que forneça um elevado grau de segurança enquanto permite uma certa flexibilidade e liberdade no uso dos recursos computacionais.

Existem fatores que tornam muito difícil impedir que atacantes eventualmente tenham acesso a um sistema. A maioria dos computadores possui algum tipo de “furo de segurança” que permite a atacantes externos ou usuários legítimos do sistema, terem acesso a informações confidenciais. Mesmo um sistema supostamente seguro pode ser vulnerável a usuários internos abusando de seus privilégios ou ser comprometido por práticas impróprias. Em vista disto, uma vez que um ataque pode ser considerado inevitável, existe uma óbvia necessidade por mecanismos que possam detectar atacantes tentando penetrar no sistema ou usuários legítimos fazendo mau uso de seus privilégios.

Com o crescente aumento no número de ataques internos, a utilização de mecanismos para a detecção de intrusões deve ser ampliada. Visto que este tipo de ataque, ocasionado pelos próprios usuários do sistema dificulta a localização imediata, torna-se necessário o uso integrado de diversas tecnologias para aumentar a capacidade de defesa de um site². Entre essas tecnologias está o uso de mecanismos para reconhecimento de padrões, buscando maneiras de reduzir o custo e aumentar a precisão dos algoritmos usados na detecção e identificação de intrusões. Várias pesquisas estão sendo feitas nesse sentido, e seria de grande interesse para o Grupo de Segurança Computacional (Laboratório Intermídia, ICMC – USP) executar um estudo comparativo entre as tecnologias propostas, aplicando-as ao seu modelo de um Sistema de Detecção de Intrusão baseado em Agentes Móveis.

1.1 Histórico do Grupo

O principal foco de pesquisa do Grupo de Segurança atualmente são os Sistemas de Detecção de Intrusão. O sistema proposto e desenvolvido pelo grupo evoluiu significativamente ao longo dos últimos anos, e podemos destacar cinco fases bem definidas desse processo, a saber:

² Site é aqui citado como sendo o conjunto de *hardware* e *software* que se deseja proteger, não somente no sentido que se tornou comum com a popularização da Internet, onde um *site* seria apenas um subconjunto desse sistema.

1. Projeto MultiView – implementado através da integração de várias ferramentas existentes em UNIX (*gawk*, *ping*, *etherfind*, etc.) e estruturas de dados complexas em um ambiente gráfico bastante robusto e capaz de visualizar topologias de rede, mostrar informações de sub-redes e de equipamentos [ODA, 1994]. Nessa fase ocorreu ainda, a adição de recursos multimídia para vídeo-conferência [LIEIRA, 1995, MORAES, 1995] e o desenvolvimento de um agente SNMP³ [CICILINI, 1994].
2. Projeto NetTracker – resultado da evolução do modelo de gerenciamento incluindo administração via WWW, método de extensão dos serviços do gerente e carregamento dinâmico de módulos [MOURO, 1997, MOURO, MORISHITA & MOREIRA, 1997]. A implementação foi feita em linguagem Java e uma implementação de SNMPv2 deveria ter sido integrada, mas os órgãos de padronização e o mercado abandonaram aquele padrão e lançaram uma nova versão: o SNMPv3, que possui características de modularidade semelhantes ao NetTracker [MORISHITA, 1997].
3. A terceira fase consiste na integração de serviços de detecção de intrusos ao NetTracker. Um sistema de detecção de intrusos baseado em redes neurais [CANSIAN, 1997; BONIFÁCIO et al., 1998a] já foi desenvolvido e adaptado para que seja possível utilizá-lo através do ambiente NetTracker. Também nesta fase foi especificado e desenvolvido um protótipo de um ambiente de gerenciamento de segurança apoiado por agentes móveis [REAMI, 1998].
4. Recentemente, o grupo de segurança computacional produziu boa quantidade de material para suportar o desenvolvimento de um Sistema de Detecção de Intrusão baseado em Agentes Móveis. BERNARDES [1999] apresenta uma modelagem para as especificações apresentadas na tese de mestrado de REAMI [1998], modela alguns cenários de utilização do ambiente e ainda, implementa um cenário de verificação de anomalias. Para uma decisão sobre qual ambiente servidor para agentes móveis seria melhor utilizado na segurança computacional, PEREIRA FILHO [2001] realizou uma pesquisa para a determinação das características de tais ambientes.
5. Na fase mais atual, TAVARES [2002] propõe mecanismos de integração de dispositivos de segmentação de redes (*switches*) com as tecnologias de detecção de intrusão usadas e

³ Simple Network Management Protocol.

propostas pelo grupo até o momento. Em paralelo ao trabalho de TAVARES [2002], essa pesquisa visa comparar as técnicas de reconhecimento de padrões atualmente propostas para o uso em sistemas de detecção de intrusão.

1.2 Estrutura da Dissertação

O Capítulo 2 apresenta pesquisas e conceitos relacionados à segurança computacional. Uma análise das pesquisas aqui apresentadas é suficiente para dar uma visão global da situação atual da segurança computacional.

O Capítulo 3 é dedicado aos Sistemas de Detecção de Intrusão, e são apresentadas algumas de suas características, o seu uso, e quais os modelos existentes atualmente.

O Capítulo 4 relaciona os recentes avanços na área de detecção de intrusos. São apresentados trabalhos que oferecem alternativas para a implementação de reconhecimento de padrões.

O Capítulo 5 apresenta uma análise dos métodos apresentados no capítulo 4, uma proposta de atualização do sistema de detecção de intrusão do ICMC e também os problemas encontrados durante o desenvolvimento desse trabalho. Finalmente, são feitas sugestões para trabalhos futuros que possam tomar esse como base.

Como informação adicional, pode-se encontrar nos apêndices um glossário e também uma lista com links relacionados à segurança computacional.

2 SEGURANÇA COMPUTACIONAL

Esse capítulo apresentará uma discussão sobre os aspectos envolvidos com a segurança computacional. Dentre esses aspectos, destaco as recentes pesquisas aqui analisadas, que mostram o cenário atual no mundo e no Brasil da segurança computacional, e servem como motivação para a pesquisa nessa área.

Não é possível iniciar uma exploração da Segurança Computacional sem antes definir o que significa o próprio termo. Essa exploração deve ser feita baseando-se em algumas linhas gerais que podem ajudar a trazer uma organização à grande quantidade de conceitos e mecanismos de segurança que podem ser encontrados hoje em dia.

2.1 Definições

A tecnologia modifica-se muito rapidamente, portanto, fazer uma distinção entre ‘segurança computacional’, ‘segurança da tecnologia da informação’ ou ‘segurança da informação’ seria arriscar a diferenciar conceitos que se misturam até mesmo na literatura especializada. Sendo assim, esses termos serão tratados neste trabalho como sinônimos.

Além desses, outros termos como *segurança*, *proteção* e *privacidade* geralmente têm mais de um significado, principalmente se considerarmos os diferentes contextos onde eles podem ser usados. Mesmo os profissionais que trabalham com segurança da informação não concordam em exatamente o que todos eles significam. Somente como um exemplo, a seguir é apresentada a definição da palavra ‘segurança’ retirada do Dicionário Larousse [GRANDE, 1999]:

Segurança *s.f.* Qualidade, caráter de uma coisa ou pessoa em que se pode fiar, sobre a qual ou quem se pode contar de maneira certa.

Segurança trata também da proteção da propriedade. Isso significa que é preciso conhecer as suas propriedades e o valor que elas têm [CANSIAN, 1997]. Essa observação geral é claro, vale também para o mundo dos computadores, e a análise de riscos é uma parte importante da estratégia de segurança da informação. Isso pode ser comprovado em BONIFÁCIO Jr. [1998], que diz que geralmente um atacante irá analisar a relação custo/benefício, ou seja, o quão custoso e complicado será invadir um determinado sistema ponderado aos lucros que ele alcançará com tal invasão. Uma vez que esta proporção se torne inviável para o atacante, pode-se dizer que foi atingido um bom nível de segurança.

São apresentadas a seguir, duas definições para o termo ‘segurança computacional’ encontradas na literatura especializada, que enfocam o assunto por diferentes ângulos e são ao mesmo tempo complementares.

1. Segurança Computacional: “Um computador é seguro se você pode confiar que ele e seus programas se comportarão como esperado”.
2. Segurança Computacional: “Segurança Computacional lida com a prevenção e detecção de ações não autorizadas dos usuários de um sistema computacional”.

A primeira definição, de GARFINKEL & SPAFFORD [1996] é um pouco mais abrangente e trata a questão da segurança sem distinguir a capacitação do profissional responsável pela segurança do usuário comum. A segunda definição, de GOLLMANN [1999] considera o lado do profissional de segurança, que é o responsável por prevenir, detectar e reagir a ações não autorizadas.

2.2 Intrusos e Intrusões

Houve um tempo em que os profissionais da segurança discutiam entre si sobre o termo *hacker*. Alguns achavam que deveria referenciar pessoas com habilidades de programação e conhecimento extraordinários, enquanto outros achavam que o *hacker* era o criminoso, aquele que faz invasões e rouba informações. Essa discussão é antiga, mas hoje em dia a mídia (principalmente a não especializada) e a indústria do entretenimento resolveram a questão e têm usado a palavra para indicar uma pessoa que tem um talento excepcional com

computadores e que faz um mau uso dessa habilidade. Por outro lado, nos meios especializados em Segurança Computacional, adota-se o nome *cracker* [MAXIMUM, 1998] para designar pessoas com tais características. Sendo assim, neste trabalho usarei os termos *cracker*, invasor ou intruso como uma maneira única de me referir a essas pessoas.

Os intrusos podem ser classificados em ativos ou passivos. Um intruso passivo simplesmente monitora a rede em busca de informações como senhas, números de cartões de créditos e informações confidenciais; o intruso ativo atua modificando o conteúdo dos pacotes por ele acessados. (Figura 1)

Figura 1: Intruso passivo (a) e intruso ativo (b)

Nos anos 80, os intrusos eram normalmente pessoas com alto conhecimento tecnológico e que criavam seus próprios métodos para invadir sistemas. O uso de ferramentas automáticas e *scripts* era exceção e não a regra. Hoje em dia, absolutamente qualquer pessoa pode atacar uma rede, devido à facilidade com que se conseguem tais ferramentas e *scripts* [ALLEN et al., 2000]. Enquanto os intrusos mais experientes estão se tornando mais eficientes e espertos, o que podemos comprovar pelo crescimento da sofisticação nos tipos de ataques (Figura 2), o conhecimento requerido por parte de intrusos novatos para copiar e iniciar alguns desses métodos de ataques está decaindo.

Figura 2: Evolução das ferramentas e técnicas de ataque e do conhecimento dos atacantes
Fonte: [ALLEN, 2000]

Mesmo que durante uma intrusão nada tenha sido removido ou alterado, geralmente os administradores de sistema precisam gastar horas ou dias reconfigurando um sistema comprometido para voltar a ter confiança na sua integridade. As pessoas que penetram em sistemas simplesmente para “dar uma olhada” acabam causando danos reais, mesmo que não leiam informações confidenciais e não apaguem nenhum arquivo.

Se a segurança computacional uma vez foi alvo de diversão e jogos, com uma rápida análise dos investimentos que estão sendo feitos nesse setor, podemos afirmar que esses dias ficaram no passado. A pesquisa [MÓDULO, 2001], por exemplo, mostra que no Brasil 80% das empresas pretendem aumentar os investimentos na área de segurança para esse ano.

O CSI apresentou em 2000 os resultados de uma pesquisa⁴ sobre quebras de segurança ocorridas durante o ano de 1999. Essa pesquisa foi publicada em 22/03/2000 e realizada junto a 643 parceiros de segurança computacional do governo americano, dentre eles, agências governamentais, instituições financeiras, instituições médicas e universidades:

⁴ Essa pesquisa é conduzida conjuntamente pelo CSI (Computer Security Institute) e o San Francisco Federal Bureau of Investigation's (FBI) Computer Intrusion Squad.

- 90% dos que responderam tiveram alguma quebra de segurança nos últimos 12 meses
- 74% sofreram perdas financeiras devido a quebras de segurança
- 42% foram capazes de quantificar suas perdas; esses 273 totalizaram US\$265, 589,940
- 59% apontaram a Internet como “a principal porta de entrada” da maioria das ameaças
- 71% detectaram acesso não autorizado por usuários internos
- 25% detectaram (e reportaram) invasões externas
- 27% sofreram ataques de *Denial of Service (DoS)*

Os números para essa pesquisa no ano de 2001 são diferentes e apresentam um crescimento de 42% nos prejuízos causados por *crackers* chegando a US\$ 378 milhões. A pesquisa, publicada no dia 12/03/2001 foi realizada com 538 profissionais de segurança de companhias de vários setores, e revelou que as maiores perdas estão relacionadas ao roubo de informações confidenciais, representando US\$ 151 milhões do total [COMPUTER, 2001].

- 85% registraram problemas de segurança no ano passado
- 64% delas tiveram prejuízos
- 70% apontaram a Internet como "a principal porta de entrada" da maioria das ameaças
- 36% detectaram (e reportaram) invasões externas

Um dos criminosos digitais mais perigosos é geralmente um funcionário (ou ex) da própria empresa. Em MÓDULO [2001] 53% das empresas apontam esse como sendo a principal ameaça às suas informações, e em FONTES [1999] esse número é ainda maior, chegando a 70%. Um ataque de um ex-funcionário descontente que queira se vingar, por exemplo, pode ser extremamente prejudicial, uma vez que supostamente ele conhece quais computadores são vitais, quais arquivos vão prejudicar mais a empresa se apagados ou danificados e, sobretudo quais são as defesas existentes.

Essas fraudes e tentativas de acesso indevido provocadas por usuários e ex-funcionários, relacionadas na pesquisas, talvez tenham como motivo principal o fato de a empresa não tratar de forma profissional a questão da segurança da informação. Uma abordagem amadora

(informal e não estruturada) pode ter boas intenções, mas normalmente, não é compatível com o porte da empresa e nem com os valores envolvidos no negócio. No Brasil, as empresas parecem estar atentas a esse cenário, já que 71% das que responderam à MÓDULO [2001] pretendem investir em Políticas de Segurança. Deve-se levar em conta também, que na maioria dos casos, os ataques vindos de dentro não são tentativas de sabotagem, mas sim uso errado de tecnologia. Ou a pessoa tem acesso a mais informação do que precisa e começa a acessá-la por curiosidade, ou então faz alguma coisa errada e acaba alterando-a [LOPES 2000].

Nas pesquisas internacionais, o número de empresas que reportaram problemas de segurança é muito maior do que nas pesquisas feitas com organizações nacionais. Esse número pode corresponder a uma preocupação com a segurança por parte dos indivíduos ou um total desconhecimento, que leva à falsa sensação de segurança. Para os especialistas em segurança, há muitas invasões que sequer são notadas pelas empresas, e essa afirmação é confirmada na pesquisa MÓDULO [2001], onde 31% das empresas entrevistadas responderam que não sabem sequer se foram invadidas. Mesmo que o ataque seja percebido, as companhias são relutantes em admiti-lo a menos que o fato seja intensamente noticiado. Invasões ou mesmo ataques do tipo *Distributed Denial of Service (DDoS)* direcionados a grandes empresas da Internet acabam por manchar a imagem das organizações.

Pode-se também ver pelos números da pesquisa, que a Internet é hoje o principal meio de acesso usado por atacantes externos, e é também através dela que são disponibilizados documentos que explicam técnicas de *hacking*, métodos de invasão e monitoração de casos em andamento. Usando esses documentos do tipo “receita de bolo” que explicam passo a passo algumas técnicas de intrusão, uma nova categoria de *crackers* acabou surgindo: são os *script-kiddies*, termo usado para descrever garotos (nem sempre tão garotos assim) que usam *scripts* e programas prontos, facilmente encontrados na rede, para efetuar ataques.

Existem também, por outro lado, vários sites na rede que disponibilizam informações que auxiliam os administradores. Uma ótima fonte de informações é o site da CERT *Computer Emergency Response Team*⁵, que divulga os mais novos *exploits* encontrados⁶. Um *exploit* é

⁵ Pode ser on-line em consultado em: <http://www.cert.org> (acesso em 22/10/2001).

⁶ Consulte o “Apêndice A: Links” para ver uma lista de *links* relacionados à segurança computacional.

uma técnica ou programa que explora uma falha conhecida de outros programas, inclusive sistemas operacionais. No entanto, essas falhas só são indicadas no site depois que um *patch* ou uma nova versão corrigida para o programa vulnerável tenham sido divulgados. No Brasil, ainda poucos sites conseguem se manter atualizados sobre o assunto, sendo o da empresa Módulo Consultoria⁷ uma exceção à regra, apresentando sempre novidades, artigos e incidentes reportados.

O grande problema é que os invasores parecem dispor de muito mais tempo para criar e descobrir *exploits* do que os administradores para se atualizarem. Por isso, e pela demora na liberação dos *patches*, tem-se um cenário em que uma rede pode alcançar um estado altamente vulnerável muito rapidamente. Para se ter uma idéia do grande número de *exploits* surgidos a cada ano, exponho dados de uma pesquisa⁸ realizada pelo CERT sobre o panorama mundial de invasões e ataques. Segundo a pesquisa, o número de incidentes no ano de 2002 deve dobrar com relação ao ano anterior, e a evolução desse cenário pode ser visualizada na Figura 3. Os valores representados para o ano 2001 são referentes somente aos três primeiros quartos do ano.

Figura 3: Invasões reportadas ao CERT

⁷ Pode ser on-line em consultado em <http://www.modulo.com.br> (acesso em 22/10/2001).

⁸ Os resultados da pesquisa foram disponibilizados no dia 15 de outubro de 2001 e pode ser consultado eletronicamente em http://www.cert.org/stats/cert_stats.html.

O CERT divulgou também, na mesma pesquisa, o número de vulnerabilidades relatadas, e como no gráfico anterior, nos números mostrados na Figura 4 só foram computados os três primeiros quartos do ano de 2001.

Figura 4: Vulnerabilidades reportadas ao CERT

2.3 Aspectos de Segurança

Uma primeira tentativa no sentido de entender os aspectos da segurança computacional deve examinar como a propriedade da informação pode ser comprometida. A definição freqüentemente usada propõe que se cubram três aspectos principais:

- **Confidencialidade:** garante que somente aqueles que têm direitos de acesso a um recurso ou informação possam efetivamente acessá-lo.
- **Integridade:** garante que a informação não será modificada.
- **Disponibilidade:** garante que os recursos ou a informação desejados estarão sempre disponíveis quando necessário. Os ataques do tipo *denial of service* aproveitam furos que causam falhas na disponibilidade do sistema.

Esses são os aspectos principais. Pode-se argumentar que a lista está incompleta porque não leva em consideração os aspectos de comunicação ou de aplicações do tipo de comércio eletrônico. Em REAMI [1998] é citado um outro princípio tido como básico, que considera a comunicação:

- **Autenticidade (autenticação):** garante que uma pessoa usando o sistema é mesmo quem diz ser.

Ainda para aqueles preocupados com as aplicações de comércio eletrônico podemos citar um outro princípio, que ao ser comprometido, colocaria em risco também a segurança do sistema. Contabilidade é em grande parte, relacionada ao comércio eletrônico, e apesar de esse último ser relativamente novo, já existiam referências a esse aspecto da segurança da informação na literatura clássica, em documentos como o *Orange Book*⁹.

- **Contabilidade:** faz com que o usuário seja responsável por suas ações, impedindo que a pessoa negue ter feito algo que tenha realmente feito.

Fundamentalmente, segurança computacional é um conjunto de soluções técnicas usadas para resolver problemas não-técnicos. Pode-se gastar uma grande quantidade de tempo, dinheiro e esforço em segurança computacional, mas o problema de perda de dados acidental, por exemplo, dificilmente pode ser totalmente resolvido. Dado um certo conjunto de circunstâncias – problemas de softwares, acidentes, erros, vírus, má sorte, mau tempo ou um atacante motivado e bem equipado – qualquer computador pode ser comprometido, mesmo os que não estão conectados em rede.

Uma quebra de segurança pode ocorrer onde existir uma falha. Segundo BERNARDES [1999], existem três causas principais para essas falhas: softwares, administradores e usuários.

- **Softwares:** de maneira geral, qualquer software está susceptível a apresentar falhas que podem ser exploradas por atacantes. Existe ainda o risco de que, ao adquirir um software de procedência duvidosa esse contenha *backdoors*, ou seja, um *Trojan Horse* (Cavalo de Tróia). Um *backdoor* é um tipo de ataque em que um software inserido ou modificado em uma rede pode, a partir de uma combinação especial de caracteres, ou de um evento de tempo, ter um comportamento diferente do esperado. Um *Trojan Horse*, como o próprio nome diz, é associado ao Cavalo de Tróia da história, ou seja, traz dentro de si uma “surpresa”. Aos olhos do usuário o programa roda normalmente,

⁹ Inicialmente publicado em 1983, o “Critério de Avaliação de Sistemas Computacionais Confiáveis” (*Trusted Computer System Evaluation Criteria*) do Departamento de Defesa (DOD) (DOD-5200.28-STD) é conhecido como o *Orange Book* que é um padrão para a segurança computacional até os dias de hoje.

mas pode estar executando alguma outra ação por trás sem que se perceba. Um exemplo é quando o programa de *login* é modificado, e além de fazer o que normalmente faria, ou seja, autenticar o usuário, ao receber a senha, grava-a em algum lugar, de onde será lida depois pelo criador do *Trojan*. Outro problema do software é a demora dos fabricantes em lançar *patches* para corrigir problemas de segurança que tenham sido descobertos e divulgados.

- **Administradores:** Esse costuma ser um grande problema em organizações que têm aquele profissional “multitarefa”, isto é, ele acumula funções e não pode se dedicar exclusivamente à segurança da informação. Geralmente essas pessoas não conseguem dar a devida importância à segurança da rede, quer seja por falta de informação, por falta de tempo, ou ainda, por desinteresse ou negligência dele próprio ou da gerência. Sendo assim, acabam incorrendo em erros graves como: não instalação de sistemas de proteção; não instalação de sistemas de auditoria (ou quando instalados não analisam as informações geradas); não aplicação de *patches* de problemas conhecidos; não orientar os usuários a se manterem atualizados.
- **Usuário:** Esse é o outro grande perigo para a segurança de uma rede: usuários mal informados ou mal intencionados podem causar grandes prejuízos. Como dito antes, um funcionário (ou ex) geralmente tem grande conhecimento do sistema, o que facilita ataques para concretizar vinganças contra colegas, superiores, ou mesmo a empresa. Os usuários legítimos podem expor o sistema a ataques, usando senhas fracas, sendo vítimas de engenharia social¹⁰, utilizar programas de origem duvidosa e muitas outras ações que podem ir contra a política de segurança ou que não estejam previstas, mas que podem colocar a rede em perigo.

2.4 Considerações Finais

Como já é possível deduzir de tudo que foi apresentado até aqui, é praticamente impossível conseguir construir um sistema totalmente à prova de falhas e intrusões. Sendo assim, os profissionais de segurança não se referem a um sistema computacional como sendo ‘seguro’

¹⁰ A engenharia social é um conceito antigo que explora a inocência dos usuários de sistemas. Um atacante pode, por exemplo, ligar para um usuário legítimo do sistema e fazendo-se passar pelo administrador do sistema, pedir a sua senha para fazer testes de rotina.

ou ‘inseguro’. Ao invés disso, é melhor adotar uma terminologia que especifique o ‘nível de confiança’ que se pode ter no sistema em questão. Para atingir um nível satisfatório de confiança, várias técnicas e/ou ferramentas devem ser combinadas. Dentre essas podemos citar, utilização de criptografia como nas Redes Privadas Virtuais (VPN¹¹), instalação de *firewalls* para limitar o alcance de acessos externos e a definição de uma política de segurança. A implementação de várias camadas de segurança torna mais difícil a um atacante entrar e sair sem deixar pelo menos alguma pista captada pelas ferramentas de segurança.

É sempre importante salientar que um sistema que use ferramentas de segurança mal configuradas, um *firewall*, por exemplo, pode algumas vezes ser pior do que segurança nenhuma, porque acaba passando uma falsa sensação de proteção, o que provoca um certo relaxamento com as atitudes a serem tomadas na rede “protegida” por esse *firewall*.

Ainda cabe aqui citar, que devido à insegurança intrínseca a sistemas computacionais conectados ou não em redes, uma atividade vital para a sobrevivência do negócio no caso do pior (invasões, ataques, perda de dados, etc.) vir a acontecer, é preciso ter uma Política de Continuidade do Negócio (PCN) bem definida. A definição de uma PCN deve ser feita considerando todos os aspectos, até mesmo os mais improváveis, que podem afetar o funcionamento normal da estrutura de negócios da empresa. Respostas a perguntas do tipo “se meu servidor de arquivos primário parar, como posso fazer com que meus 100 desenvolvedores continuem a trabalhar?” devem ser procuradas e formuladas procurando soluções realistas e viáveis.

O capítulo a seguir apresenta uma ferramenta poderosa no auxílio à segurança computacional. São os sistemas de detecção de intrusão. Esses sistemas agem na rede interna (geralmente protegida por *firewalls*), buscando por sinais de tentativas ou de invasões concretizadas.

¹¹ Do inglês: *Virtual Private Network*. Consulte o glossário para maiores informações.

3

3 SISTEMAS DE DETECÇÃO DE INTRUSÃO

Neste capítulo é apresentada, além das definições, a classificação mais aceita na atualidade para os sistemas de detecção de intrusão, e cada uma das classificações é discutida sobre suas vantagens e desvantagens.

Técnicas de prevenção de intrusões, como encriptação, autenticação (por exemplo, senhas ou biométricas) e programação defensiva têm sido usadas para proteger sistemas computacionais como uma primeira linha de defesa. No entanto, prevenção de intrusões por si só não é suficiente porque sempre existem fraquezas exploráveis nos sistemas, devido a erros de projeto e programação, ou várias técnicas de penetração como “engenharia social”. Por exemplo, depois de ter sido identificado há anos atrás, furos de segurança do tipo *buffer overflow*, que podem levar a acesso privilegiado não autorizado, ainda existem em programas recentes. Detecção de intrusão pode ser usada como outra ferramenta para proteger sistemas de rede, porque uma vez detectada uma intrusão, por exemplo, num estágio prematuro de um ataque *Denial of Service*, reações podem ser iniciadas para minimizar os estragos, recolher evidências para processos e até mesmo lançar contra-ataques.

3.1 Definições

Detecção de Intrusão é definida em SPAFFORD et al. [1998] como “o problema de identificar pessoas que estejam usando um sistema computacional sem autorização (por exemplo, *crackers*) e aqueles que têm acesso legítimo ao sistema mas estão abusando de seus privilégios” e intrusão seria “qualquer conjunto de ações que tentem comprometer a integridade, confidencialidade ou disponibilidade de um recurso”.

Se uma senha fraca é quebrada, medidas de controle de acesso podem fazer pouco para prevenir roubo ou corrupção de informação legalmente acessível ao usuário que teve sua

senha comprometida [KUMAR & SPAFFORD, 1994]. Isso quer dizer que mesmo depois de definida uma boa política de segurança e da configuração e instalação de programas e tecnologias para a proteção do sistema, ainda existe a possibilidade de falhas e/ou de ações maliciosas de usuários tanto externos quanto internos. Portanto, torna-se desejável a existência de sistemas capazes de realizar a detecção de tais falhas e informar o administrador da rede. Tais sistemas são conhecidos como Sistemas de Detecção de Intrusão (SDI)¹².

As funcionalidades de um sistema de detecção tornam-se de vital importância na medida em que fornecem meios de inferir sobre o conteúdo das conexões permitidas e detectar as que apresentem um comportamento suspeito ou não condizente com a política de segurança implantada.

É importante ressaltar que um sistema de detecção de intrusão não inclui prevenção de intrusão, somente detecção e a comunicação de sua ocorrência ao administrador. Alguns sistemas também iniciam um processo de reação a uma intrusão detectada e confirmada.

Em SPAFFORD et al. [1998] são apresentadas as características desejáveis de um SDI:

- Ele deve rodar continuamente com um mínimo de supervisão humana.
- Ele deve ser tolerante a falhas no sentido de ser capaz de se recuperar caso o sistema caia, tanto acidentalmente ou por ações maliciosas.
- Ele deve resistir à subversão, ou seja, o próprio sistema tem que ser capaz de identificar se foi modificado por um atacante.
- Ele deve impor um *overhead* mínimo no sistema onde está rodando para que não interfira em sua performance.
- Ele deve ser capaz de ser configurado de acordo com a política de segurança do sistema que ele está monitorando.
- Deve ser capaz de adaptar mudanças no sistema e no comportamento dos usuários.

¹² Do inglês: *Intrusion Detection Systems*.

- O SDI deve ser escalável para suportar a carga de monitorar várias estações e ainda produzir resultados confiáveis e a tempo.

3.2 Classificação dos SDI

Os SDI são classificados sob diferentes ópticas na literatura da área. As principais classificações utilizadas são em termos da forma como o sistema aborda o problema de detectar intrusão e em termos do tratamento dos dados.

3.2.1 Tratamento dos Dados

Essa é a classificação mais comum encontrada para os Sistemas de Detecção de Intrusão. De acordo com o método usado para o tratamento dos dados, os SDI são classificados em:

3.2.1.1 Sistemas Baseados em Rede (NIDS)

A maioria dos sistemas comerciais de detecção de intrusão é baseada em rede [BACE & MELL 2001; ALLEN et al., 2000; GHOSH, MICHAEL & SCHATZ, 2000]. Os SDI baseados em rede examinam os dados que trafegam pela rede através da monitoração on-line dos pacotes procurando por indicações de que um ataque possa estar ocorrendo. São freqüentemente formados por sensores espalhados pela rede que trabalham em modo promíscuo, ou seja, capturando todos os pacotes endereçados ao segmento de rede onde eles se encontram.

Os Sistemas Baseados em Rede (NIDS¹³) apresentam como principais vantagens:

- Apenas alguns sensores bem posicionados podem monitorar uma rede com muitas estações.
- A adoção de um sistema deste tipo tem pouco impacto em uma rede já existente, porque esses sistemas geralmente são dispositivos passivos que “escutam” os cabos de rede, sem interferir em sua operação normal.
- Esses sistemas podem ser feitos invisíveis aos atacantes usando métodos de “stealth” [BACE & MELL 2001].

¹³ Do inglês: *Network Intrusion Detection Systems*.

As maiores vantagens dos SDI baseados em rede não se aplicam às modernas redes baseadas em *switches*. Esses dispositivos dividem a rede em muitos pequenos segmentos (geralmente um cabo de *Ethernet* por estação) e fornecem links dedicados entre as estações servidas pelo mesmo *switch*. Além desse problema, outras desvantagens desse tipo de sistema podem ser levantadas:

- Podem ter dificuldade em processar os pacotes em uma rede grande e de tráfego intenso, podendo falhar na tarefa de reconhecer ataques originados durante os períodos de pico de uso. Alguns fabricantes estão tentando resolver esse problema implementando a solução de SDI totalmente em *hardware*, o que torna o sistema mais rápido.
- Eles não conseguem analisar tráfego encriptado na rede, e esse problema está se tornando mais sério, uma vez que várias empresas (e atacantes) estão usando VPNs.
- A maioria dos SDI baseados em rede não pode dizer se um ataque teve sucesso ou não; eles podem ao máximo dizer que um ataque está em progresso. Isso significa que depois que o SDI alerta sobre uma intrusão, o administrador precisa investigar manualmente cada estação atacada para saber se ela foi ou não realmente afetada.
- Alguns sistemas existentes hoje em dia têm problema em lidar com ataques baseados em rede que usam fragmentação de pacotes. Esses pacotes mal-formados podem fazer com que o sistema fique instável ou pare de funcionar.

Figura 5: Posicionamento de sensores de um SDI de Rede

A Figura 5 mostra a disposição de sensores de um SDI. Na figura, os sensores são representados como uma máquina normal, mas destacados com um fundo diferente. O primeiro deles (de cima para baixo) monitora todo o tráfego entre a Internet e a intranet. O segundo, monitora o tráfego entre a sub-rede 3 e a intranet / Internet. Os outros sensores funcionam de maneira análoga a esse último, monitorando os dados entre uma sub-rede específica e a intranet / Internet.

3.2.1.2 Sistemas Baseados em Nós de Redes (NNIDS)

Os Sistemas Baseados em Nós de Redes (NNIDS¹⁴) constituem um tipo “híbrido” de Sistema de Detecção de Intrusão que surgiu mais recentemente e consegue superar algumas das limitações dos SDI baseados em rede.

¹⁴ Do inglês: *Network Node Intrusion Detection System*.

Esse novo modelo trabalha de uma maneira similar aos baseados em rede porque, numa visão bastante simplista, também captura pacotes e os compara com uma base de dados de assinaturas de ataques. No entanto, esse novo “micro sensor” só se preocupa com pacotes direcionados ao nó de rede onde ele se encontra, dando assim origem ao nome Sistemas Baseados em Nós de Redes.

Na verdade, um nó de rede é uma estação, mas o termo quando usado dessa maneira, acaba causando confusões com os SDI baseados em *Host*. Os sensores NNIDS não trabalham em modo promíscuo, portanto, somente capturam os pacotes de rede endereçados à estação que ele estiver monitorando. O fato do NNIDS não precisar capturar todo pacote que passar pelo seu segmento, significa que eles podem ser mais rápidos e consumir menos recursos, e isso permite que eles sejam instalados nas estações existentes sem impor um grande *overhead*. Eles são também adequados a segmentos de tráfego intenso ou isolados por *switches*, ou seja, situações onde os NIDS normalmente encontram problemas.

3.2.1.3 Sistemas Baseados em *Host* (HIDS)

Os Sistemas Baseados em *Host* (HIDS¹⁵) operam em informações coletadas em um sistema individual. Isso permite que sistemas deste tipo analisem atividades com grande confiabilidade e precisão, determinando exatamente quais processos e usuários estão envolvidos em um ataque em particular. Além disso, diferentemente dos SDI baseados em rede, esses sistemas podem saber o resultado de uma tentativa de ataque, uma vez que eles têm acesso direto e monitoram os arquivos de dados e processos do sistema geralmente usados como alvos pelos atacantes.

Geralmente usam informações de duas fontes para executarem sua tarefa: informação de auditoria do sistema operacional¹⁶ e logs do sistema. O primeiro, geralmente é mais protegido e fornece informações mais detalhadas, os logs por sua vez, são menores e mais claros, sendo mais fáceis de serem analisados.

Os sistemas baseados em *hosts* apresentam como principais vantagens:

¹⁵ Do inglês: *Host Intrusion Detection System*.

¹⁶ Do inglês: *Operating System Audit Trail*.

- Frequentemente podem atuar em um ambiente onde o tráfego de rede é encriptado, usando os dados da estação origem antes de serem encriptados e/ou depois que são decriptados na estação destino.
- Não são afetados por redes que fazem uso de *switches*.
- Quando operam nas informações de auditoria do sistema operacional, eles podem ajudar a detectar Cavalos de Tróia ou outros ataques que envolvam falhas de integridade de software.

As desvantagens apresentadas por esses sistemas são:

- São mais difíceis de gerenciar, uma vez que as informações precisam ser configuradas e gerenciadas para cada estação monitorada.
- Como pelo menos as fontes de informação (e algumas vezes parte do sistema de análise) necessárias ao SDI residem na estação para onde o ataque é direcionado, o próprio sistema de detecção pode ser comprometido e desabilitado como parte do ataque.
- Eles não podem detectar ataques direcionados a toda a rede, já que conhecem a informação de cada estação em separado. Sendo assim, um simples *port-scan* passa despercebido pelo SDI.
- Quando usam informação de auditoria do sistema operacional, a quantidade de informação pode ser imensa, exigindo espaço adicional no sistema.

Em BACE & MELL [2001] encontramos ainda um terceiro tipo de classificação, que seriam os sistemas baseados em aplicações. Na verdade um subgrupo dos sistemas baseados em *hosts*, esses sistemas analisam os eventos ocorridos dentro de uma aplicação (*software*). A fonte de informação mais comum usada por esses SDI é os arquivos de log das aplicações.

3.2.2 Método de Detecção

De acordo com o método de detecção, os SDI têm sido historicamente estudados baseando-se em dois subtópicos: detecção por *uso indevido* e por *anomalia* [KUMAR & SPAFFORD, 1995]. Um terceiro e mais recente método seria o *baseado em especificações*, onde o

comportamento correto e permitido a um sistema é especificado, e eventos que não encaixem com essa especificação geram alertas [KO et al., 2000].

3.2.2.1 Detecção por Uso Indevido

Neste modelo, a detecção é feita através da procura de tentativas de explorar pontos fracos conhecidos do sistema, que podem ser descritas por um padrão específico ou sequência de eventos ou dados (a “assinatura” da intrusão). Por exemplo, uma regra de assinatura para o ataque de “adivinhação de senhas” pode ser “falharam mais que quatro tentativas de login nos últimos 2 minutos” [LEE et al., 2000]. Esse é o método usado pela maioria dos SDI comerciais [BACE & MELL, 2001; GHOSH, MICHAEL & SCHATZ, 2000].

Os sistemas que fazem detecção por uso indevido apresentam como principais vantagens:

- São efetivos em detectar ataques sem gerar uma quantidade exagerada de alarmes falsos.
- São capazes de diagnosticar rápida e confiavelmente o uso de uma ferramenta ou técnica específica de ataque. Isso pode ajudar os gerentes de segurança a priorizar as medidas de correção.

As desvantagens apresentadas por esses sistemas são:

- Só podem detectar os ataques que sejam conhecidos para o sistema e por isso precisam ser atualizados constantemente com as assinaturas dos novos ataques [KUMAR, 1995].
- Muitos detectores de uso indevido são projetados para usar assinaturas bem definidas, e isso os previne de detectarem variações de ataques comuns. Detectores de uso indevido baseados em estado (*state-based*) podem não ter essa limitação [BACE & MELL, 2001].

3.2.2.2 Detecção por Anomalia

Neste método, a detecção é feita procurando por mudanças nos padrões de utilização ou comportamento do sistema. É construído um perfil representando o comportamento normal dos usuários, das estações e/ou das conexões de rede. Esses perfis são construídos com dados históricos colhidos através de um período de operação normal ou empiricamente. A premissa

central de detecção por anomalia é que *atividade intrusiva* é um subconjunto de *atividade anômala*. Isso parece bem razoável, já que se um intruso consegue acessar uma conta de um sistema, sem ter a menor noção do comportamento padrão do usuário que teve a conta comprometida, existem grandes chances de que o seu comportamento seja anômalo [KUMAR, 1995]. Por exemplo, o perfil normal de um usuário pode conter a frequência média de alguns comandos do sistema usado em suas sessões de uso. Se em uma sessão sendo monitorada, as frequências forem significativamente maiores ou menores, então um alarme é lançado [LEE et al., 2000].

Esse modelo foi e continua sendo objeto de um grande número de pesquisas. As métricas e técnicas usadas nesses sistemas incluem:

- Detecção de limiar (*threshold*) onde alguns atributos do usuário e do sistema (número de arquivos acessados por um usuário em um determinado período de tempo, número de tentativas de efetuar login no sistema que falharam, quantidade de CPU utilizada por um processo, etc) são expressos através de cálculos, com um limite estabelecido como permissível. Este nível pode ser determinado estatística ou heurísticamente, isto é, projetado para mudar com valores reais observados durante algum tempo.
- Medidas estatísticas, tanto paramétricas (onde a distribuição dos atributos com perfis traçados é assumida como pertencendo a um padrão em particular) quanto não paramétricas (onde a distribuição dos atributos com perfis traçados é “aprendida” de um conjunto de valores históricos, observados durante algum tempo).
- Medidas baseadas em regras, que são similares às medidas estatísticas não paramétricas no sentido que os dados observados definem padrões de uso aceitáveis, mas diferenciam no sentido que esses padrões são especificados como regras, e não quantidades numéricas.
- Outras medidas, incluindo redes neurais, algoritmos genéticos e modelos de sistemas imunes.

Segundo BACE & MELL [2001] somente as duas primeiras métricas podem ser encontradas hoje em sistemas comerciais.

Podemos apontar as seguintes vantagens em sistemas que fazem detecção por anomalia:

- Detectam comportamento incomum e portanto têm a habilidade de detectar sintomas de ataques sem especificar conhecimento dos detalhes.
- Podem produzir informação que pode ser usada para definir assinaturas para detectores por uso indevido.

Algumas das desvantagens desses sistemas são:

- Produzem um grande número de alarmes falsos devido ao comportamento imprevisível de usuários e das redes [KUMAR, 1995; LEE et al., 2000].
- Geralmente requerem extensivos “conjuntos de treinamento” de registros de eventos do sistema para poderem caracterizar padrões de comportamento normais.
- Um atacante pode modificar seu comportamento lentamente, de maneira que o SDI não perceba [KURI et al., 2000].

3.2.2.3 Detecção Baseada em Especificações

Técnicas baseadas em especificações foram propostas como uma alternativa que combina as forças de detecção por anomalia e uso indevido. Nessa abordagem, são criadas especificações que caracterizam o comportamento legítimo dos programas. Como esse método é baseado em comportamentos legítimos, ele não gera alarmes falsos quando comportamentos não comuns (mas legítimos) são encontrados. Além disso, a taxa de falsos positivos pode ser comparada à dos métodos de uso indevido. Embora as promessas da abordagem baseada em especificações já estejam sendo discutidas há algum tempo, a pergunta se esses benefícios podem mesmo ser implementados na prática, ainda estava sem resposta, e o trabalho de UPPULURI & SEKAR [2000] visa dar algumas dessas respostas.

Além disso, diversos SDI empregam ao mesmo tempo técnicas para detecção de anomalias e de uso indevido. Algumas técnicas são baseadas em pré-determinação de padrões futuros, a partir do comportamento anterior, enquanto outras, utilizam estatística para determinar o comportamento anômalo. Em ambos os casos, quando o comportamento monitorado não está de acordo com o comportamento esperado, pode haver um indicativo de intrusão. Os modelos são baseados na hipótese de que a exploração das vulnerabilidades de um sistema envolve algum tipo de desvio. Deste modo, violações de segurança podem ser detectadas a partir de

algoritmos agindo nos computadores em questão, ou sobre a rede como um todo [BACE & MELL, 2001].

3.2.3 Tipo de Reação

As duas classificações acima são clássicas na literatura sobre Sistemas de Detecção de Intrusão. Em BACE & MELL [2001] um novo tipo de classificação é sugerido. Os autores classificam os SDI também de acordo com o tipo de reação. De acordo com essa classificação, os SDI são separados em sistemas com:

3.2.3.1 Reação Ativa

Reações de SDI ativos são ações automáticas tomadas quando certos tipos de intrusão são detectados. Existem três categorias de reações ativas:

A reação ativa mais inócua, porém mais produtiva, é a de **Coletar Informação Adicional**. Muitos de nós já tivemos reações desse tipo como, por exemplo, quando se acorda à noite com algum barulho estranho. A primeira coisa que se faz é tentar escutar melhor, procurando por informações adicionais que permitam decidir se é preciso ou não reagir. Para o SDI isso pode envolver aumentar o nível de sensibilidade da origem de informações (por exemplo, aumentar a sensibilidade do sensor que captura os pacotes da rede, não apenas visando uma porta ou sistema em particular). A informação adicional coletada pode ajudar a resolver a detecção do ataque, ou fornecer informações extras que possam dar suporte a uma investigação e/ou apreensão do intruso, e também fornecer provas para ações legais civis e criminais.

Mudar o Ambiente é uma outra reação ativa e pode, por exemplo, ativar o sistema para parar um ataque em progresso e depois inibir o acesso do atacante, tipicamente, bloqueando o endereço IP de onde aparentemente se origina o ataque. Bloquear um atacante habilidoso é muito difícil, mas os SDI podem frequentemente deter atacantes menos experientes usando alguma dessas técnicas:

- Injetar pacotes TCP de *reset* na conexão do atacante ao sistema-vítima terminando, assim, a conexão.
- Reconfigurar roteadores e *firewalls* para bloquear pacotes originados da aparente localização do atacante (endereço IP).

- Reconfigurar roteadores e *firewalls* para bloquear portas de rede, protocolos ou serviços sendo usados por um atacante.
- Em situações extremas, reconfigurar roteadores e *firewalls* para cessar todas as conexões que usam uma específica interface de rede.

Contra-atacar é uma forma de reação mais agressiva e pode envolver ações como buscar ativamente mais informações sobre a estação ou sistema do atacante, ou mesmo lançar ataques contra ele. Mas essa atitude pode ser perigosa legalmente falando, uma vez que o atacado pode estar incorrendo no mesmo crime que o atacante, e isso pode significar um risco muito maior do que o próprio ataque representava.

3.2.3.2 Reação Passiva

Reações passivas fornecem informações aos usuários do sistema, confiando em humanos para tomarem ações subseqüentes baseadas nessas informações.

Alarmes e notificações são gerados pelos SDI para informar usuários quando ataques são detectados. A forma mais comum de alarme é uma mensagem na tela ou uma janela *popup*, e a informação passada é bastante variável, indo de algo como “uma invasão está acontecendo” até longas mensagens incluindo endereço IP do atacante e do atacado, a ferramenta ou técnica usada e o resultado da tentativa de ataque. Notificações também são uma parte fundamental, permitindo que o SDI seja configurado para avisar o administrador através de *paggers* ou celulares ou até mesmo o mais tradicional *e-mail* (mas essa notificação é mais facilmente bloqueada pelo atacante).

Alguns sistemas de detecção de intrusão são projetados, como destacado no parágrafo anterior, para gerar alarmes e alertas. Porém, alguns deles são capazes de usar **chamadas (traps) SNMP**, e mensagens para enviar alarmes e alertas aos consoles de gerenciamento central da rede, onde podem ser analisados por pessoal especializado. Um dos grandes benefícios dessa abordagem é que a infra-estrutura da rede pode ser adaptada como reação a um ataque detectado.

Outro problema sendo abordado em pesquisas é o fato de que os administradores gastam boa parte do seu tempo analisando os relatórios e alarmes gerados pelos SDI. É fácil imaginar um sistema protegido, por exemplo, por um NIDS e um HIDS onde ambos possuem diferentes

sistemas de alerta e notificação. O trabalho de VALDES & SKINNER [2001] propõe uma correlação probabilística entre os alertas semelhantes gerados por diferentes SDI.

3.3 Componentes de um Sistema de Detecção de Intrusão

A funcionalidade de um SDI pode ser logicamente distribuída em três componentes: sensores, analisadores e uma interface de usuário.

- **Sensores** são responsáveis por coletar os dados. A entrada para um sensor pode ser qualquer parte de um sistema que possa conter evidência de uma intrusão. Exemplos de tipo de entrada para um sensor são pacotes de rede, arquivos de log, e rastreamento de chamadas de sistema. Os sensores recolhem e transferem essa informação para o analisador.
- **Analisadores** recebem dados de um ou mais sensores ou de outros analisadores. O analisador é responsável por determinar se uma intrusão ocorreu. A saída deste componente é uma indicação da ocorrência de uma intrusão e pode incluir evidências que suportem essa indicação. O analisador pode fornecer orientação sobre ações a serem tomadas quando se depara com uma intrusão.
- A **Interface** de um SDI permite que um usuário veja a saída do sistema ou controle o comportamento do mesmo.

Além desses três componentes essenciais, um SDI pode ser suportado por um “*honeypot*”, isto é, um sistema projetado e configurado para ser visível para um intruso e que demonstre ter vulnerabilidades já conhecidas. Tal sistema cria um ambiente de onde se pode obter informações adicionais que ajudem na análise de intrusões. O *honeypot* serve como um sensor para um SDI, simplesmente esperando que um intruso ataque o sistema aparentemente vulnerável, tendo ainda a habilidade de detectar intrusões em ambiente relativamente controlado.

3.4 Considerações Finais

A pesquisa relacionada à detecção de intrusão produziu várias ferramentas nos anos 90. No entanto, como muitas foram desenvolvidas por estudantes explorando e desenvolvendo conceitos, essas ferramentas não receberam mais manutenção depois de servirem a esse propósito. Mesmo assim, essas ferramentas influenciaram a direção de pesquisas

subseqüentes e também de desenvolvimento de sistemas comerciais. Porém ao contrário da literatura de produtos desenvolvidos como objeto de pesquisa, a documentação encontrada para os produtos comerciais é geralmente voltada para o marketing. Isso torna difícil identificar as funcionalidades que estes produtos realmente suportam. Por exemplo, às vezes é preciso fazer uma leitura cuidadosa para identificar pontos técnicos básicos, como por exemplo, se o sistema é baseado em rede ou em *host*.

A tecnologia de Detecção de Intrusão é ainda imatura e muito dinâmica. Por causa das rápidas mudanças na área, informações como listas, pesquisas ou revisões ficam rapidamente desatualizadas. Sendo assim, o local para se coletar informações mais atualizadas pode ser em sites especializados, que procuram usar o dinamismo da Internet para proporcionar informações atualizadas na área de interesse.

O capítulo seguinte apresenta alguns dos métodos mais recentes usados na detecção de intrusos.

4

4 MÉTODOS DE RECONHECIMENTO DE INTRUSÕES

Neste capítulo serão apresentados alguns trabalhos que representam os mais recentes avanços na área de detecção de intrusões. As pesquisas aqui apresentadas cobrem várias alternativas para reconhecimento de padrões na tentativa de suportar os SDI, dentre elas, redes neurais, mineração de dados e algoritmos genéticos.

4.1 Sistema Imunológico Computacional para Detecção de Intrusões

Esta seção apresenta um trabalho que é uma tentativa de desenvolver um SDI baseado em rede capaz de agir por antecipação, baseado no conceito de Sistema Imunológico Computacional (CIS)¹⁷. O objetivo principal é fornecer detecção de intrusão, baseada em redes e pacotes simples (não considera o *payload*) usando um CIS [WILLIAMS et al., 2001]¹⁸. Satisfazer esse objetivo implicava em dois principais desafios de projeto:

1. Definir uma estrutura de assinaturas, ou anticorpos.
2. Definir uma função de encaixe imperfeito; essa função realiza um mapeamento entre a assinatura e o espaço *non-self*. Esse mapeamento deve ser construído de maneira geral

¹⁷ Do inglês: *Computational Immune System*.

¹⁸ Todas as afirmações desta seção (4.1) são provenientes dessa fonte de referência, exceto onde explicitamente indicado.

no sentido que uma determinada assinatura deveria encaixar uma variedade de pacotes *non-self*, todos relacionados pela característica da assinatura.

4.1.1 Projeto do Sistema

Computer Defense Immune System (CDIS) é um Sistema Imunológico Computacional multi-agente, hierárquico e distribuído, modelado com base nos arquétipos biológicos do processo imunológico. Foi inicialmente criado pensando-se no problema dos vírus, mas foi estendido para tentar resolver também os problemas relacionados à detecção de intrusão. A pesquisa de WILLIAMS et al. [2001] modificou o processo de ciclo de vida do CDIS e implementou dois protótipos: Warthog e Ferret. Ferret é o mais novo dos dois protótipos e resolve alguns dos problemas de lógica e implementação do protótipo anterior, porém ainda não está terminado e para os testes descritos no artigo, o Warthog foi usado.

Ambos os protótipos trabalham com pacotes TCP, UDP e ICMP. Uma grande quantidade de pacotes que não contém ataques é usada como os dados *self*. Esses pacotes são carregados na base de dados (Warthog) ou na memória (Ferret) e usados no processo de seleção negativa descrito na seção 4.1.1.3. Os dados de testes são pacotes TCP, UDP e ICMP que podem ou não conter ataques.

4.1.1.1 Características de Pacotes

Como dito anteriormente, um dos maiores desafios encontrados para a construção do sistema seria como projetar uma estrutura e a representação das assinaturas. Uma vez que o domínio do problema é baseado em rede, e a unidade básica de tráfego de uma rede é o pacote, as assinaturas no CDIS são construídas com base nos atributos dos pacotes. Foram usadas 28 características para o TCP e 16 para UDP e ICMP. A Tabela 1 mostra as características usadas para cada protocolo¹⁹. Todos os pacotes são representados como uma string binária de 320 bits, independentemente do protocolo.

¹⁹ Cada parte de um endereço IP separada por pontos é uma característica, embora na tabela eles sejam mostrados como uma característica única.

Nome do Campo	Valores possíveis
Campos IP (comuns a todos os pacotes)	
Tipo de Protocolo	TCP, UDP, ICMP
Número de Identificação IP	0 - 65535
Time to Live (TTL) IP	0 - 255
Flags IP	0 - 65535
Comprimento total do pacote	0 - 65535
Endereço origem do pacote (A.B.C.D)	Endereço IP válido
Endereço destino do pacote (A.B.C.D)	Endereço IP válido
Campos exclusivos para TCP	
Porta Origem TCP	0 - 65535
Porta Destino TCP	0 - 65535
Número da Sequência TCP	0 - 4294967295
Número de Próxima Sequência TCP	0 - 4294967295
Número de Reconhecimento TCP	0 - 4294967295
Todas as Flags TCP	0 - 255
Flags Individuais TCP (PCWR, Echo, Urgent, Ack, Push, Reset, Syn, Fin)	Boolean
Tamanho do pacote TCP	0 - 65535
Campos exclusivos para UDP	
Porta Origem UDP	0 - 65535
Porta Destino UDP	0 - 65535
Comprimento de Dados UDP	0 - 65535
Campos exclusivos de ICMP	
Tipo ICMP	0 - 255
Código ICMP	0 - 255
Comprimento de Dados ICMP	0 - 65535

Tabela 1: Características do Pacote

4.1.1.2 Características de Anticorpos

Anticorpos são assinaturas para detectar pacotes *non-self*, ou antígenos. Um anticorpo consiste de duas ou mais das características de pacotes mostradas na Tabela 1. Para criar um anticorpo, um protocolo é randomicamente escolhido, e depois alguns valores são randomicamente atribuídos aos atributos daquele protocolo. O par atributo-valor de um anticorpo especifica uma sub-região no hiperespaço correspondente a todos os pacotes possíveis. Se todos os atributos fossem definidos para um anticorpo em particular, então esse anticorpo especificaria um ponto no espaço.

Uma vez que uma sub-região ou ponto nesse espaço pode ter um uso muito limitado em termos de capturar ataques anômalos, cada anticorpo também tem, para cada atributo não-booleano selecionado, uma área de alcance randomicamente selecionada ao redor do valor especificado. Com essas áreas de alcance, cada anticorpo representa um hipervolume de 16 ou 28 dimensões, em um hiperespaço de 16 ou 28 dimensões para o protocolo desse anticorpo. E esses hipervolumes podem conter muitos pacotes possíveis.

4.1.1.3 O Processo CDIS

O ciclo de vida de um anticorpo de intrusão de rede no CDIS é mostrado na Figura 6. A arquitetura CDIS permite mortes periódicas e criação de novos anticorpos, mas essas características, mostradas como linhas tracejadas, não foram implementadas em nenhum dos protótipos construídos. Os processos que foram implementados são descritos a seguir.

Figura 6: Ciclo de vida de um anticorpo CDIS

Criação do Anticorpo e Seleção Negativa. No início, o sistema gera randomicamente um número pré-determinado de anticorpos. Cada anticorpo passa, então, por seleção negativa, que significa que o anticorpo é checado para determinar se algum pacote *self* fica dentro de seu hipervolume. Se isso acontecer, o anticorpo é descartado e um novo é gerado.

Maturação de Afinidade. Uma vez que os anticorpos são criados randomicamente, a sua posição e área de alcance provavelmente não serão ótimas; nesse contexto, ser ótimo significa que o anticorpo é tão grande quanto possível sem incluir nenhum pacote *self*. O objetivo da maturação de afinidade é maximizar o hipervolume definido pelas áreas de alcance das

características do anticorpo. Como este é um problema de otimização e computacionalmente caro, uma busca usando algoritmos genéticos é realizada em cada um dos parâmetros de área de alcance do anticorpo, construindo uma função de aptidão que premia o crescimento até que o anticorpo colida com pacotes *self*. Quando isso acontece a função de aptidão é modificada para compensar encolhimento até que o anticorpo não se encontre mais com os pacotes *self*. Esse processo é também muito custoso e, por isso, é opcional no ciclo de vida, podendo ser substituído, por exemplo, por um número maior de anticorpos, que também cobriria uma área maior do espaço de eventos *non-self*.

Função de Encaixe Imperfeito. Como no sistema imunológico do corpo humano, a detecção de um antígeno é efetuada atando-se o anticorpo a ele via alguma técnica de encaixe imperfeito. A técnica de encaixe imperfeito do CDIS usa áreas de alcance ao invés de pontos, onde cada anticorpo define um hipervolume como foi explicado em 4.1.1.2. Portanto, a função é imperfeita porque identifica pacotes dentro do hipervolume, ao invés de identificar (ou não) especificamente um ponto único.

Existem muitas funções de encaixe de padrões ou encaixe imperfeito. Esses comparadores podem ser encontrados em duas variedades: medidas de semelhança que expressam o quão semelhantes dois objetos são, ou funções de diferença que medem o quão diferentes os objetos são. Informalmente, objetos que estão próximos no espaço de características devem ser similares, enquanto aqueles que estão distantes são menos similares. No contexto de um CIS, padrões gerados pelo sistema são comparados com anticorpos contendo tanto padrões *self* quanto *non-self*, e são rejeitados ou aceitos dependendo do resultado dessa comparação. Uma vez que todos os anticorpos CDIS passaram por seleção negativa, é garantido que eles não contenham pacotes *self* conhecidos. Dessa maneira, qualquer pacote que encaixe com um anticorpo CDIS, dentro do *threshold* definido pelos parâmetros de alcance de área no anticorpo, pode ser classificado como *non-self*.

Costimulation. Como CDIS define o *non-self* como qualquer coisa que esteja fora dos dados de treinamento do *self*, o espaço *self* tem que ser muito bem definido para que o CDIS trabalhe efetivamente. No entanto, como o *self* não é definido perfeitamente e desvia com o tempo, é provável que o sistema levante falsos positivos. O CDIS deve rodar de maneira distribuída, simultaneamente em múltiplos computadores monitorando a mesma rede, trabalhando juntos para detectarem ataques. *Costimulation* é então o processo pelo qual os anticorpos cruzam a informação dos pacotes que chegam com outros anticorpos, tentando

assim, reduzir o número de falso positivo gerado. Um pacote que chega não é classificado como um ataque a não ser que múltiplos anticorpos o detectem; dessa forma, se apenas um anticorpo detecta um pacote em particular, esse pacote é descartado e não é reportado como um ataque. Embora esse processo possa reduzir o número de falsos positivos detectados, ele também pode aumentar o número de ataques não identificados, uma vez que todos os pacotes detectados precisam ser “costimulados”. O equilíbrio entre falsos positivos e falsos negativos é um assunto muito importante.

4.1.2 Resultados Experimentais e Análise

Os objetivos do teste incluem determinar:

- O sistema é capaz de separar *self* de *non-self*?
- O sistema é capaz de detectar ataques desconhecidos, isto é, ataques para os quais não fora treinado?
- Como o número de anticorpos afeta as taxas de erro falso positivo e falso negativo?
- A técnica de maturação de afinidade baseada em Algoritmo Genético é efetiva em aumentar a generalidade dos anticorpos?
- Como o processo de *costimulation* afeta as taxas de erro de falso positivo e falso negativo?

Os testes foram realizados em duas fases separadas. A primeira fase de testes usou poucos dados para reduzir o tempo necessário para a execução dos testes, enquanto a segunda fase usa quantidades de dados mais realistas para simular o mundo real. Pesquisadores do Lincoln Laboratory (LL) no Massachusetts Institute of Technology (MIT) criaram um arquivo com dados para detecção de intrusão para a Avaliação de SDI da DARPA 1999 [LIPPMANN et al., 2000a]²⁰, e esses dados foram usados para a fase de testes.

4.1.2.1 Conjunto de dados de Treinamento ou *Self*

Para o teste menor, os dados *self* consistem dos primeiros 10K pacotes dos dados filtrados (somente pacotes TCP, UDP e ICMP) da quinta-feira. Para o teste maior, os dados *self* são todos os 1.308.081 pacotes filtrados da segunda-feira. Não foram incluídos dados de ataques,

²⁰ Mais informações sobre a avaliação podem ser obtidas on-line em <http://www.ll.mit.edu/IST/ideval> (acesso em 12/02/2002).

e os processos de seleção negativa e maturação de afinidade foram aplicados ao conjunto de dados para construir os anticorpos.

4.1.2.2 Conjunto de dados de Teste e Ataque

Os conjuntos de dados de teste do LL eram livres de ataques, onde, para ter maior controle sobre os dados sendo testados, os pesquisadores injetaram algumas assinaturas de ataques. Para o teste menor, os dados de teste são os primeiros 20K pacotes dos dados filtrados da quinta-feira, enquanto para o teste maior, foram usados todos os 1.085.895 pacotes filtrados da terça-feira. Ambos os conjuntos foram injetados com os mesmos ataques.

O conjunto de ataques é formado por um conjunto de 2.643 pacotes; parte deles, foi retirada parcialmente do conjunto de dados de ataque do LL, a outra parte foi gerada usando a ferramenta de análise de segurança Nessus²¹.

4.1.2.3 Ambiente de Teste

Os experimentos rodaram em um conjunto de seis máquinas Dell Pentium III de 933MHz, cada uma com 256MB RAM e Windows 2000 Professional Edition como sistema operacional. A base de dados orientada a objetos usada foi a Objectivity 6.0, e o protótipo Warthog foi desenvolvido em Java e usava a máquina virtual da Sun, JDK 1.3.

4.1.2.4 Processo de Teste

A primeira fase de testes seguiu todos os passos do ciclo de vida mostrado na Figura 6. Os 10K pacotes *self* foram usados para seleção negativa e maturação de afinidade, enquanto os 20K pacotes de teste mais os 2.643 pacotes de ataque foram usados para o passo de detecção. Primeiro, Warthog aplicou seleção negativa para gerar conjuntos de anticorpos. Conjuntos de 32, 64, 128, 256, 512, 1024 e 2048 anticorpos foram criados para cada replicação de teste. Esses conjuntos de anticorpos foram usados para escanear os dados gravados por ataques, e os resultados desse teste foram gravados. Depois, cada um dos conjuntos de anticorpos passou pelo processo de maturação de afinidade para generalizar os anticorpos. Esses anticorpos generalizados foram então usados no processo de detecção para checar os pacotes de teste.

²¹ Pode ser encontrado on-line em <http://www.nessus.org/> (acesso em 21/02/2002).

Finalmente, *costimulation* foi aplicada em cada conjunto de anticorpos antes e depois do processo de maturação de afinidade, para que os efeitos dessa etapa pudessem ser analisados.

A segunda fase de testes também seguiu o ciclo de vida da Figura 6, exceto que o processo opcional de maturação de afinidade não foi usado porque era muito lento quando o número de pacotes *self* crescia acima de 1 milhão. Conjuntos com 1024 e 2048 anticorpos não foram criados, também por causa da lentidão no processo de criação. Depois, o processo de detecção checkou os 1.085.895 pacotes de teste mais os 2.643 de ataques contra cada um dos conjuntos de anticorpos. Finalmente a *costimulation* foi aplicada.

4.1.2.5 Resultados Experimentais

Os testes confirmaram que o protótipo Warthog cobriu os cinco principais objetivos como descritos na seção 4.1.2. O CDIS pode separar *self* de *non-self* e é capaz de detectar alguns ataques desconhecidos. Os testes mostraram que o número de anticorpos afeta as taxas de erro dos falsos positivos e falsos negativos; mostraram também que o processo de maturação de afinidade foi útil apenas quando o número de anticorpos era baixo. O processo de *costimulation* provou ser capaz de baixar as taxas de erro dos falsos positivos com pequeno impacto nos falsos negativos.

Analisando os resultados mostrados na Figura 7, é possível notar que a taxa de erro para os falsos negativos se aproximava de zero enquanto o número de anticorpos crescia acima de 256, e a taxa de falsos positivos, mesmo aumentando quando o número de anticorpos crescia, continuava muito baixa em todos os testes. A aplicação de *costimulation* reduziu a taxa de falsos positivos para níveis ainda mais baixos, e não afetou significativamente a taxa de falsos negativos.

Os resultados indicam que aumentar o número de anticorpos acima de 512 não melhora de forma significativa a taxa de falsos negativos, mas aumenta a de falsos positivos. Inicialmente se supôs que a maturação de afinidade seria necessária para manter baixa a taxa de falsos negativos, pelo fato de aumentar a quantidade do espaço de eventos *non-self* coberta pelos anticorpos. No entanto, os resultados mostraram que o método de criação dos anticorpos provou ser eficaz o suficiente (com os dados do LL) sem usar o pesado processo de maturação de afinidade. É possível que para grandes quantidades de dados do mundo real, esse processo seja necessário para que a detecção seja efetiva.

Figura 7: Resultados de testes para a primeira fase, com conjuntos de dados pequenos

Nos testes com maior quantidade de dados²², a taxa de erro de falsos positivos não chegou a zero, mas foi bastante baixa. Por exemplo, o experimento com pior performance usou 512 anticorpos e classificou incorretamente (como *non-self*) 72 dos 1.088.538 pacotes de teste, enquanto a maioria dos experimentos não acusou nenhum falso positivo. Embora a taxa de erro de 0,00006 possa parecer baixa, não seria um trabalho gratificante para um analista, verificar cada um dos 72 pacotes erroneamente classificados.

²² Os gráficos de performance para esses testes podem ser encontrados em WILLIAMS et al. [2001].

4.2 Filtros Baseados em Reconhecimento de Padrões

O objetivo principal do trabalho de KURI et al. [2000]²³ é fornecer um sistema de detecção de intrusão para a rápida detecção de ataques potenciais ao invés de uma detecção precisa (i.e., exaustiva) de ataques reais. Os resultados dessa detecção (i.e., filtragem dos logs de auditoria onde ataques podem estar presentes) seriam usados como entradas para um algoritmo de detecção mais preciso. Esse conceito já era usado no SDI Gassata [MÉ, 1998], que usa um algoritmo genético para uma função semelhante.

4.2.1 Detecção de Intrusão como um problema de Reconhecimento de Padrões

Em termos gerais, o problema de detecção de intrusão é detectar a existência de uma série de eventos conhecidos *a priori* dentro do rastro de atividade de um sistema sendo protegido. Esses rastros podem se diferenciar em relação à origem, forma e conteúdo, dependendo do tipo de ataque potencial que eles tentam identificar (por exemplo, tráfego de rede, log de comandos). Nesse trabalho foi considerada a detecção de ataques usando logs de comandos digitados pelos usuários de um sistema computacional distribuído. Nesse contexto, ataques parecem ser tipicamente seqüências curtas de não mais que oito comandos.

A proposta é modelar o problema de uso indevido como um problema de reconhecimento de padrões da seguinte maneira: comandos “auditáveis” do sistema podem ser vistos como letras de um alfabeto Σ e os logs de auditoria como uma longa string de letras de Σ (i.e., o texto). As seqüências de eventos representando os ataques a serem detectados são então *substrings* (i.e., padrões) a serem localizados na string principal. Uma vez que atacantes podem introduzir falsos comandos entre aqueles que representam um ataque verdadeiro, um número limitado de letras falsificadas também precisa ser permitido quando se busca pelo padrão. O interesse da pesquisa é procurar simultaneamente por um *conjunto* de padrões. Sendo assim, o problema de detecção por uso indevido pode ser considerado como um caso particular do problema de *reconhecimento de padrão aproximado múltiplo*²⁴, onde *inserção* é a única operação de

²³ Todas as afirmações desta seção (4.2) são provenientes dessa fonte de referência, exceto onde explicitamente indicado.

²⁴ Do inglês: *Multiple Approximate Pattern Matching*.

edição permitida. A Figura 8 representa o mecanismo de representação, identificação e resultado do modelo.

Figura 8: Busca de assinaturas como um problema de reconhecimento de padrão aproximado múltiplo

O problema acima é formalizado da seguinte maneira: O texto, $T_{1..n}$, é uma sequência de n caracteres de um alfabeto Σ de tamanho σ . Os padrões $P^1 \dots P^r$ são (curtas) sequências de caracteres de Σ . Consideremos um padrão $P_{1..m}$ de tamanho m . O modelo deve reportar todas as posições do texto que encaixam com P , onde no máximo k inserções entre caracteres de P são permitidas em sua ocorrência em T . A taxa de erro é chamada $\alpha = k/m$.

O problema pode ser modelado usando o conceito de *distância de inserção*. A distância de inserção de a até b , denotada $id(a,b)$, é o número de inserções necessárias para converter a em b . Diz-se que $id(a,b) = \infty$ se isso não for possível. Claramente, $id(a,b) = |b| - |a|$ se a for uma subsequência de b , e ∞ caso contrário.

A busca pelo padrão P em um texto T é feita permitindo-se inserções. A cada posição do texto $j \in 1..n$ é preciso encontrar o número mínimo de inserções necessárias para converter P em algum sufixo de $T_{1..j}$. Isso é definido como:

$$lid(P, T_{1..j}) = \min_{j' \in 1..j} id(P, T_{j'..j})$$

O problema de busca pode então ser formalizado comó: dados $P^1 \dots P^r$, T e k , identifique todas as posições j do texto tais que $lid(P^i, T_{1..j}) \leq k$ para qualquer $i \in 1..r$. Em KURI & NAVARRO [2000] são apresentados dois algoritmos diferentes para busca multipadrão especificamente confeccionados para esse problema de reconhecimento de padrões. Eles são baseados em “bit-parallelism” uma técnica para representar o estado da busca usando os bits de uma palavra de computador de w bits (tipicamente $w = 32$ ou 64). Porém, o foco do trabalho aqui discutido, é no modelo probabilístico de ocorrências de padrões em texto quando inserções são permitidas, e a sua relação com o problema de falsos positivos e falsos negativos.

4.2.2 Probabilidade de Reconhecimento

É preciso estabelecer um limite para a probabilidade de reconhecimento de um padrão randômico de tamanho m em uma posição randômica do texto, com no máximo k inserções. Considere uma posição randômica j no texto. O padrão P existe com k inserções em uma posição do texto terminando em j , se e somente se a janela de texto $T_{j-m-k+1..j}$ contém as m letras do padrão em ordem. A posição da janela que encaixe com as letras do padrão pode ser escolhida de $\binom{m+k}{m}$ maneiras. Como k pode ter qualquer valor, a probabilidade que a janela de texto contenha o padrão com no máximo k inserções é:

$$\binom{m+k}{m} \frac{\sigma^k}{\sigma^{m+k}} = \binom{m+k}{m} \frac{1}{\sigma^m} \quad (1)$$

A probabilidade de que um padrão randômico encaixe em uma posição randômica de um texto foi testada experimentalmente. A Figura 9 (esquerda) mostra a probabilidade de encontrar um texto de 3 Mb em um padrão com $m = 300$, onde ambos (padrão e texto) foram randomicamente gerados a partir de um alfabeto de tamanho $\sigma = 68$. Como pode ser visto, existe um valor k a partir de onde a probabilidade de reconhecimento começa a crescer abruptamente de quase 0 a até quase 1. Esse fenômeno é agudo o suficiente para transformar o

valor k no parâmetro mais importante governando o comportamento do algoritmo. Esse ponto é chamado de k^* , e a correspondente taxa de erro é $\alpha^* = k^* / m$.

Figura 9: Resultados dos testes experimentais

Na parte direita da Figura 9 o limite do valor α^* para diferentes comprimentos de padrões, mostrando que α^* tende a uma constante para valores altos de m , embora seja menor para valores pequenos. Ainda experimentalmente, ficou determinado que o tamanho do alfabeto σ afeta o valor de α^* , estimado como $\alpha^* = (\sigma / 1.0856) - 0.8878$.

Todos os resultados experimentais bateram com os resultados analíticos no sentido que (a) existe um claro valor de erro α^* aonde a probabilidade de reconhecimento vai de quase 0 para 1; (b) esse ponto não depende de m de maneira assintota; e (c) ele depende linearmente de σ como previsto pela análise²⁵ ($\alpha^* = (\sigma / e) - 1$) exceto que e foi substituído por aproximadamente 1.09.

4.2.3 Resultados Experimentais

Os pesquisadores também estudaram experimentalmente como o modelo probabilístico de reconhecimento de strings permitindo inserções se relaciona com o problema de falsos negativos e falsos positivos. O interesse é determinar como α^* se relaciona com a razão de

²⁵ Em KURI et al. [2000] existe uma breve explicação da análise matemática que produz esse resultado.

falsos negativos e positivos e o número total de ataques relatados, e conseqüentemente, com a eficiência de filtragem do modelo.

Os dados de entrada para o experimento consistem de um log de auditoria e uma base de ataques, ambos bem simples. O arquivo de auditoria foi coletado usando o SDI Gassata em um ambiente real. A base de dados de ataques é formada por assinaturas de ataques, onde cada assinatura é formada por regras.

Ambos os grupos de dados são transformados em uma representação voltada ao reconhecimento de padrões em três passos:

1. Uma letra diferente é associada a cada regra.
2. Assinaturas de ataque são então convertidas em padrões, mapeando suas regras para as letras correspondentes.
3. O arquivo de auditoria é escaneado e seus eventos são comparados às regras. Eventos que encaixem com mais de uma regra, recebem as letras correspondentes. Os que não encaixam, recebem uma letra arbitrária. A sequência final é construída concatenando-se a sequência de letras correspondentes aos encaixes e as letras arbitrárias.

Foi usado um arquivo de log de auditoria de 24.847 eventos e três séries de diferentes ações como padrões de ataque²⁶:

who em cadeia: representado como um padrão de quatro eventos de um comando “who”. A probabilidade da letra correspondente no arquivo de auditoria é de 0.004382 e existem quatro ataques reais desse tipo no arquivo de auditoria.

comandos sensíveis: representado como um padrão de dez eventos de qualquer comando do conjunto {“last”, “ps”, “who”, “whois”}. A probabilidade da letra correspondente no arquivo de auditoria é 0.007187 e existem dois ataques reais desse tipo no arquivo de auditoria.

²⁶ Esses padrões não são verdadeiramente ataques, mas do ponto de vista algorítmico não faz diferença.

whois em cadeia: representado como um padrão de quatro eventos de um comando “whois”. A probabilidade da letra correspondente no arquivo de auditoria é de 0.001402 e existe um ataque real desse tipo no arquivo de auditoria.

Ataque	M	Ocorr.	Prob. letra	k máx.	Process. extra
who em cadeia	4	4	0.004382	500	8,21%
comandos sensíveis	10	2	0.007187	620	14,50%
whois em cadeia	4	1	0.001402	1570	5,74%

Tabela 2: Parâmetros principais para os três padrões

O texto que o filtro não é capaz de excluir tem que ser processado por um algoritmo mais sofisticado para determinar a presença de um ataque real. O objetivo é então determinar a efetividade do algoritmo de filtragem proposto. Isto significa: quanto texto ele é capaz de filtrar para devolver qual fração dos ataques reais ocorre no arquivo de auditoria. Junto com o k máximo recomendado pelo modelo, foi computada a fração do texto que o filtro retorna (para esse k) como candidato para avaliações futuras. Isso são simplesmente os $m+k$ caracteres que precedem cada encaixe, evitando contar múltiplas vezes as áreas sobrepostas. A Tabela 2 mostra que usando o k máximo recomendado pelo modelo analítico da seção 4.2.2, apenas 6% a 15% do texto é devolvido para ser processado por um algoritmo mais pesado.

Com respeito aos falsos negativos, foi avaliado para os três padrões em particular, as frações de texto filtrados como uma função da fração de ataques detectados. Por exemplo, com um valor de k que deixe apenas 2% do texto para avaliação posterior, 50% dos ataques são reconhecidos (e portanto 50% são falsos negativos). É possível, no entanto, balancear a taxa de falsos negativos com a velocidade da detecção. É claro que em muitos casos uma taxa de zero falsos negativos é necessária, e para isso, o valor de k determinado pelo modelo é um limite superior do valor a ser usado para o padrão correspondente.

Com respeito aos falsos positivos, foi estudada a evolução do número de encaixes como uma função de k . É claro que, para alguns padrões, usar um valor de k muito grande gera muitos falsos positivos. Mas é importante notar que esses falsos positivos podem ser descartados posteriormente por um algoritmo mais preciso (e provavelmente mais custoso) que pode analisar a saída do mecanismo de reconhecimento de padrões proposto. Para limitar os falsos positivos sem permitir falsos negativos, o valor de k determinado pelo modelo parece ser próximo do ótimo.

4.3 Aprendizado de Comportamento de Programas

A motivação do trabalho de GHOSH, MICHAEL & SCHATZ [2000]²⁷ é que uma grande quantidade das intrusões envolve o uso indevido de programas, explorando-os de maneira inteligente para conseguir privilégios não autorizados que são posteriormente usados para executar ações maliciosas. Sendo assim, se o comportamento normal de um programa puder ser representado adequada e compactamente, então as características comportamentais capturadas pelos mecanismos de auditoria podem ser usadas para a detecção de intrusões.

Como a pesquisa visa analisar os arquivos de auditoria, a plataforma Solaris foi usada durante o desenvolvimento por fornecer o Módulo de Segurança Básica (BSM)²⁸, que tem um gerenciador configurável de auditoria que facilita a gravação de eventos do sistema.

Os pesquisadores procuram abordar dois desafios: detectar ataques desconhecidos e generalizar, baseando-se no comportamento observado anteriormente para reduzir a taxa de falsos positivos a níveis aceitáveis do ponto de vista administrativo.

Foram desenvolvidos três algoritmos de aprendizado de máquina para treinar autômatos que aprendem o comportamento normal de um programa. Os autômatos treinados são posteriormente usados para detectar o uso indevido de programas. Esses algoritmos são apresentados a seguir:

4.3.1 Rede Neural Recorrente Elman

O objetivo de usar Redes Neurais Artificiais (RNA) para detecção de anomalias é o de ser capaz de generalizar a partir de dados incompletos, e também classificar dados on-line como sendo normais ou intrusivos. Através desse processo de aprendizado, as RNA desenvolvem a habilidade de classificar entradas, simplesmente sendo expostas a um conjunto de *entradas de treinamento* e à aplicação de *regras de aprendizado* bem definidas, ao invés de depender de regras de classificação explícitas fornecidas por humanos. Por causa da capacidade de

²⁷ Todas as afirmações desta seção (4.3) são provenientes dessa fonte de referência, exceto onde explicitamente indicado.

²⁸ Do inglês: *Basic Security Module*.

generalizar, as RNA podem produzir classificações razoáveis para entradas desconhecidas (assumindo que a rede foi bem treinada).

Para manter a informação de estado entre entradas, é preciso usar uma RNA com topologia recorrente. Essa topologia (em oposição a uma topologia puramente *feed-forward*) é uma em que as conexões formam ciclos. Os ciclos funcionam como *loops* de atraso – forçando a informação a ser retida indefinidamente. Novas entradas interagem com os ciclos, afetando as ativações do ciclo e as que se propagam através da rede. Sendo assim, a entrada pode afetar o estado, e o estado pode afetar a classificação de qualquer entrada.

A Rede de Elman, apresentada na Figura 10, é um exemplo conhecido de topologia recorrente. A topologia de Elman é baseada na *feed-forward* – tem uma *camada de entrada*, uma *camada de saída*, e uma ou mais *camadas ocultas* (ou *intermediárias*). Adicionalmente, essa topologia tem um conjunto de *nós de contexto* (destacados na parte direita da Figura 10). Cada um dos nós de contexto recebe entrada de um único nó oculto e envia sua saída para cada nó na camada do seu correspondente nó oculto. Como os nós de contexto dependem apenas das ativações dos nós ocultos da entrada anterior, os nós de contexto guardam informação de estado entre as entradas.

Figura 10: Topologia padrão *feed-forward* e uma Rede de Elman

No contexto do trabalho, as redes Elman classificam seqüências curtas de eventos enquanto eles ocorrem em longos fluxos de eventos. Portanto, as redes são treinadas para *prever* a próxima seqüência que vai ocorrer em qualquer ponto no tempo. A n -ésima entrada, I_n , é apresentada à rede para produzir uma saída, O_n . A saída O_n é então comparada à entrada I_{n+1} . A diferença entre O_n e I_{n+1} (isto é, a soma dos valores absolutos das diferenças dos elementos correspondentes de O_n e I_{n+1}) é a medida de anomalia de cada seqüência de eventos. Ou, em outras palavras, a medida de anomalia é o erro em prever a próxima entrada na seqüência.

4.3.2 Transdutor²⁹ de String

Um transdutor de string é um algoritmo que associa uma seqüência de símbolos de entrada com uma série de símbolos de saída. São mais freqüentemente usados em biologia e lingüística computacionais, onde são geralmente implementados usando autômatos finitos cujas transições ou estados são associados com símbolos de saída.

No contexto do trabalho, a seqüência de entrada é uma string de eventos BSM, e a seqüência de saída é uma previsão para os próximos vários eventos. O uso proposto do transdutor de string como detector de intrusão é baseado no exame das probabilidades dos símbolos de saída em cada estado. Durante o treinamento, a distribuição da probabilidade dos símbolos a cada estado é estimada, e durante os testes, desvios dessa distribuição são tomados como evidências de comportamento anômalo.

A implementação proposta para essa idéia é relativamente simples. É usado um autômato finito cujos estados correspondem a n -gram dos dados do BSM, e os símbolos de saída associados a cada estado são também l -gram do BSM (para $l < n$). Mais especificamente, o símbolo de saída representa conjuntos de l eventos do BSM que podem ser encontrados quando o autômato se encontra em um determinado estado. Durante o treinamento o objetivo é recolher estatísticas sobre os l -gram sucessores; a probabilidade de cada l -gram é estimada por contagem.

Durante a detecção de intrusões, o desvio que os l -gram sucessores apresentam de seus valores esperados é usado para a pontuação de anomalia. Essa pontuação é geralmente

²⁹ Do inglês *transducer*.

diferente de zero, mas se o programa estiver se comportando normalmente, esses desvios devem atingir uma média com o passar do tempo. Em GHOSH, MICHAEL & SCHATZ [2000] existe uma explicação com exemplo de como esse processo é implementado.

4.3.3 Testador de Estados

O objetivo do terceiro algoritmo, chamado simplesmente de “testador de estados”, é criar automaticamente autômatos finitos para representar comportamento de programas. Uma vez que dados representando comportamento intrusivo não são usados durante o treinamento, o primeiro objetivo é simplesmente construir um autômato finito que aceite todas as seqüências de auditoria dos dados de treinamento, mas sem ser muito generoso a ponto de aceitar todos os dados, ou muito rígido a ponto de rejeitar toda seqüência desconhecida de auditoria depois do treinamento.

A abordagem proposta consiste em automatizar a inferência de autômatos finitos. Algo nessa linha foi proposto por WARRENDER, FORREST & PEARLMUTTER [1999] onde os dados de treinamento são usados para encontrar modelos ocultos de Markov (*Hidden Markov Model* – HMM) no comportamento normal de programas. Essa técnica provou ser efetiva na tarefa de detecção de intrusões, mas o treinamento acabou sendo custoso. Surge então a questão se algoritmos mais simples que apenas infiram o autômato finito (AF), e não as probabilidades de transição associadas com o modelo de Markov, podem também ser efetivos sem requerer muito treinamento.

Como exemplo, poderia ser criado um AF com um único estado, onde todos eventos do BSM resultam em uma transição daquele estado de volta a si mesmo. Poderia também ser criado um AF sem ciclos que aceite exatamente as seqüências do BSM presentes nos dados de treinamento. A primeira abordagem é muito fraca, porque tende a aceitar qualquer seqüência de eventos BSM e, portanto, falha na tarefa de detectar seqüências anômalas. A segunda abordagem é provavelmente muito forte, porque rejeita qualquer seqüência como sendo anormal a menos que exatamente a mesma seqüência tenha sido vista durante o treinamento. O objetivo então é criar um AF razoavelmente expressivo, mas que ainda possa generalizar.

Primeiramente é preciso definir os estados do autômato. A técnica reportada no artigo associa cada estado com um ou mais n -grams dos dados do BSM, onde n é um parâmetro do algoritmo de aprendizado. A idéia, no entanto, é ser econômico com a criação dos novos estados, e não criar simplesmente um estado no AF para todos os n -grams de eventos BSM.

Durante o treinamento, autômatos separados são criados para os diferentes programas cujos dados de auditoria estão disponíveis. Conceitualmente, o objetivo do autômato é prever o n -gram inteiro, baseado no seu estado atual e nos primeiros l eventos de auditoria do n -gram, onde $l < n$. As transições do AF correspondem a seqüências específicas de l eventos de auditoria. Diz-se então, que um AF prevê um n -gram G se existe uma transição do estado atual para o estado correspondente a G , e se essa transição estiver rotulada com os primeiros l elementos de G . Sendo assim, o autômato prevê um conjunto de estados, e esses estados são simplesmente os alcançáveis por transições rotuladas com os primeiros l elementos de G . Se esse conjunto for vazio (por exemplo, não existe transição rotulada com os primeiros l elementos de G) então se diz que o autômato não faz previsões. De outra forma, um *erro de previsão* ocorre se o conjunto previsto de estados não contém o que está associado a G .

4.3.4 Performance dos Algoritmos

Os três algoritmos descritos anteriormente foram implementados e avaliados por um laboratório independente, o Lincoln Laboratory do MIT, na Avaliação de SDI da DARPA [LIPPMANN et al., 2000a]³⁰. O laboratório estabeleceu quatro categorias de ataques: *Denial of Service* (DoS), *port-scan*, *remote-to-local* (R2L) e *user-to-root* (U2R). Os ataques de DoS e *port-scan* são baseados em rede e deixam rastros nos pacotes de rede. Ataques *remote-to-local* também envolviam ataques baseados em rede, mas também incluíam alguns ataques que tentavam fazer uso indevido de programas. Ataques *user-to-root* tentam ganhar privilégios de super usuário na máquina atacada, tanto por uso indevido de programas ou pela execução de programas maliciosos.

Como a abordagem proposta se destina a detectar uso indevido de programas, os algoritmos são capazes de detectar qualquer tipo de ataque que deixe rastros nos arquivos de auditoria, independente de em qual categoria proposta pelo Lincoln Laboratory o ataque se encaixe. De acordo com a partição proposta pelo laboratório, o sistema estaria mais apto a detectar os ataques U2R e algumas instâncias dos ataques R2L.

Uma vez que a implementação dos algoritmos exige o treinamento de monitores de programas, é preciso primeiro definir quais programas serão monitorados. Na prática, a

³⁰ Mais informações sobre a avaliação podem ser obtidas on-line em <http://www.ll.mit.edu/IST/ideval> (acesso em 12/02/2002).

maioria dos ataques é feita contra programas privilegiados num servidor de rede. A Tabela 3 lista os programas monitorados e também representa um super conjunto dos monitores executados no teste do LL.

admintool	cu	kerbd	nispasswd	rpld	uptime
allocate	deallocate	kernel	nmbd	rsh	userhelper
aspppd	dhcpcd	kflushd	nsd	usersd	usernetctl
at	dos	klogd	nxterm	rwhod	utmp_update
atd	eject	kswapd	pageout	sacdm	utmpd
atq	exrecover	list_devices	passwd	sadmin	uu.*
atrm	fdformat	lockd	ping	sendmail	volcheck
auditd	ff.core	login	procmail	smbd	vold
automountd	ffbconfig	lpd	ps	sperl5.00404	w
cardctl	fsflush	lpq	pt_chmod	ssh1	whodo
chage	gpasswd	lpr	pwdb_chkpwd	sshd	wu.ftpd
chfn	gpm	lprm	rcp	su	xlock
chkey	hpnpd	m64config	rdist	suidperl	xscreensaver
chsh	unt	mingetty	rdistd	syslogd	xterm
cron	in.*	mkdevalloc	rlogin	tcpd	Xwrapper
crond	inetd	mkdevmaps	routed	timed	ypbind
crontab	kcms_calibrate	mount	rpcbind	traceroute	yppasswd
ct	kcms_configure	newgrp	rpciod	umount	zgv

Tabela 3: Lista de programas monitorados pelo autômato de detecção de intrusão

4.3.4.1 Performance das Redes de Elman

Examinando primeiramente os resultados apresentados para o ataque *user-to-root*, fica claro que as redes Elman tiveram uma performance muito boa contra esse tipo de ataque. Elas atingiram 100% de detecção de ataques muito rapidamente a uma taxa de alarmes falsos próxima de 3 por dia. Um exame mais cuidadoso dos ataques mostra que a grande maioria deles envolvia o uso indevido de programas, como o *buffer overrun*.

A performance das redes Elman contra os ataques *remote-to-local* do LL não foi nem de perto tão boa quanto a performance contra U2R. A uma taxa de aproximadamente 10 alarmes falsos por dia, foram detectados aproximadamente 30% dos ataques R2L. Se o operador estiver disposto a aceitar uma taxa de alarmes falsos de até 100 por dia, a taxa de detecções corretas sobe acima de 90%. No entanto, operacionalmente falando, essa taxa de alarmes falsos não é aceitável. Parte desse resultado é explicado pelo fato de o Lincoln Laboratory praticamente não ter usado ataques R2L que envolvessem o uso indevido de programas, portanto, a maioria desses ataques cai fora do escopo proposto pela pesquisa.

Uma vez que somente programas conhecidos são monitorados, programas maliciosos não são detectados. No entanto, com a técnica proposta, é possível detectar intrusões que foram precursoras à instalação de clientes maliciosos. Em resumo, ataques que envolvam programas não monitorados ou que envolvam usos normais de programas não se encaixam no mecanismo de detecção proposto. A razão pela qual eles acabam sendo detectados (mesmo que com uma alta taxa de alarmes falsos) é que efeitos colaterais da intrusão tendem a surgir em outros programas que estão sendo monitorados.

4.3.4.2 Performance do Transdutor de String

A performance do transdutor de string é muito próxima da medida para as redes Elman. A uma taxa de aproximadamente três falsos positivos por dia, foram detectados 100% dos ataques *user-to-root*. O que é mais significativo nesse resultado, no entanto, é que uma vez que o tempo de treinamento para o transdutor de string é ordens de magnitude menor que o da rede neural Elman, pode-se atingir uma performance de detecção comparável com significativa diferença no tempo de treinamento. O treinamento das redes Elman leva um tempo da ordem de magnitude de milhares de minutos para todos os programas monitorados, enquanto o do transdutor de strings é da ordem de dezenas de minutos.

Novamente, a performance para os ataques R2L não foi muito boa pelas mesmas razões. À mesma taxa de falsos positivos foram detectados aproximadamente 15% dos ataques *remote-to-local*. Se a taxa de falsos positivos for aumentada para aproximadamente nove por dia, são detectados aproximadamente 35% dos ataques.

4.3.4.3 Performance do Testador de Estados

A uma taxa de 9 falsos positivos por dia, foram detectados 100% dos ataques *user-to-root*. Porém, o progresso em direção aos 100% de detecção é mais gradual se comparado, por exemplo, ao resultado obtido com as redes de Elman. A conclusão é que, com o testador de estados, alguém pode ajustar a performance do sistema mais facilmente para suprir os requerimentos de uma organização com relação à tolerância aos alarmes falsos. Por outro lado, a performance das redes de Elman indica, mais ou menos, detecção tudo ou nada, que não permite muitos ajustes. No entanto, é importante não generalizar mais que o necessário, já que os resultados podem variar de experimento a experimento dependendo dos ataques lançados e dos dados de treinamento.

Foram detectados 35% dos ataques *remote-to-local*, a uma taxa de alarmes falsos de nove por dia. Uma performance mais baixa que dos outros métodos, mas ainda melhor do que o padrão apresentado hoje pelos sistemas comerciais. É provável que a performance possa ser melhorada se forem usados dados mais robustos para o treinamento.

4.4 Detecção de Intrusões Distribuídas e Desconhecidas

A pesquisa de LEE et al. [2000]³¹ visa desenvolver técnicas para detectar intrusões desconhecidas e distribuídas. É descrito a seguir um sistema experimental, baseado no *Common Intrusion Detection Framework (CIDF)*, onde SDI distribuídos geograficamente podem comunicar entre si seguindo os protocolos definidos no CIDF. Por exemplo, os SDI podem trocar informações de ataques entre si que incluam a origem do ataque, método, comportamento, reação, etc. Mais ainda, ao detectar um ataque desconhecido, um SDI pode enviar os dados relevantes de auditoria para um “construtor de modelos”, que automaticamente analisa os dados e computa um novo modelo de detecção especificamente criado para esse ataque, e distribui o novo modelo para customização, tradução e instalação.

4.4.1 MADAM ID

MADAM ID (*Mining Audit Data for Automated Models for Intrusion Detection*) é um conjunto de ferramentas que podem ser aplicadas a uma variedade de dados de auditoria para gerar modelos de detecção de intrusão. O tema central da abordagem proposta é aplicar programas de mineração de dados (*data mining*) aos dados de auditoria recolhidos, para computar modelos que capturem com precisão o comportamento atual (isto é, padrões) de atividades normais e intrusivas. Os modelos resultantes podem ser mais efetivos que os criados manualmente, porque são computados e validados usando grandes quantidades de dados de auditoria.

O produto final do MADAM ID é um conjunto de regras de classificação que podem ser usadas como modelos para detecção de intrusão. Detecção de intrusão foi tratada na pesquisa como um problema de classificação, porque idealmente deseja-se classificar cada registro de

³¹ Todas as afirmações desta seção (4.4) são provenientes dessa fonte de referência, exceto onde explicitamente indicado.

auditoria em uma categoria dentre as possíveis em um conjunto discreto. Essas possíveis categorias são: *normal*, um tipo de *intrusão* em particular ou *anomalia*.

Antes de aplicarmos algoritmos de classificação, é preciso selecionar e construir o conjunto certo de características do sistema que possa conter evidências (indicadores) normais ou de intrusões. Na verdade, seleção/construção de características é o problema mais desafiador na construção de SDI, independente da abordagem de desenvolvimento usada.

Figura 11: O processo de mineração de dados para a construção de modelos

A abordagem proposta explora a natureza temporal e estatística do tráfego de rede e aplica programas de mineração de dados para computar padrões para a construção de características. Como mostrado na Figura 11, os dados crus de auditoria (por exemplo, tcpdump) do tráfego de rede, são primeiramente processados em nível de pacotes / eventos (ASCII), que depois serão sumarizados em registros de nível conexão / sessão. Cada registro é definido por um conjunto de características básicas e de propósito geral, por exemplo, hora de início, duração, *host* e porta de origem e destino, número de bytes transferidos e uma *flag* que indica o comportamento do protocolo da conexão (por exemplo, SF para SYN e FIN normais), etc. Algoritmos de mineração de dados otimizados para dados de auditoria, são aplicados então para extrair vários padrões de atividades dos dados de auditoria. Dado um extensivo conjunto de dados normais de auditoria e outro conjunto que inclua uma instância de uma intrusão em particular, os padrões de ambos os conjuntos podem ser comparados para identificar os que

são somente de intrusão. Esses padrões passam posteriormente por um *parser* encarregado de construir características apropriadas que possam prever a intrusão. Por exemplo, um padrão do ataque *SYN flood* é mostrado na Tabela 4. Posteriormente, as seguintes características são construídas para o *SYN flood*: uma contagem das conexões à mesma estação destino nos últimos dois segundos, e entre essas conexões, a porcentagem que pertence ao mesmo serviço e a porcentagem que tem a *flag* S0.

Episódio Frequente	Significado
(flag=S0, service=http, dst_host=victim), (flag=S0, service=http, dst_host=victim) → (flag=S0, service=http, dst_host=victim) [0.93, 0.03, 2]	93% do tempo, depois que duas conexões http com flag S0 (isto é, apenas um pacote SYN é enviado) são feitas à estação <i>victim</i> , uma terceira conexão similar é feita dentro de 2 segundos depois da primeira dessas duas, e esse padrão ocorre em 3% dos dados.

Tabela 4: Um exemplo de padrões de intrusão

4.4.2 Visão Geral do CIDF

A motivação do CIDF é fornecer uma infra-estrutura que permita aos componentes e sistemas de detecção, executar a análise e reação a intrusões (IDAR³²), além de compartilhar informações sobre ataques distribuídos e coordenados.

Todos os quatro tipos de componentes do CIDF (geradores de eventos, mecanismo de análise, mecanismo de reação e base de dados) trocam dados no formato GIDO³³, que é representado em um formato padrão, sendo esse padrão comum definido usando uma linguagem de especificação (CISL)³⁴. A construção principal da linguagem CISL é a expressão de propósito geral chamada expressão-S, que é formada por agrupamentos recursivos de *tags* e dados.

Dados relacionados à detecção de intrusão são representados em CISL como uma sequência de expressões-S com dois ou mais elementos. O primeiro sempre indica como interpretar os dados que se seguem, ou seja, é uma *tag* que fornece pistas semânticas sobre a interpretação

³² Do inglês: *Intrusion Detection, Analysis and Response*.

³³ Do inglês: *Generalized Intrusion Detection Object*.

³⁴ Do inglês: *Common Intrusion Specification Language*.

do resto da expressão-S. Por essa razão são chamados de Identificadores Semânticos (SID³⁵). Como um exemplo, a expressão “o usuário Pedro apagou o arquivo /etc/passwd” poderia ser expressa da seguinte maneira:

```
(Delete
  (Initiator
    (UserName 'Pedro')
  )
  (FileSource
    (FileName '/etc/passwd')
  )
)
```

4.4.3 MADAM ID como um Mecanismo de Modelagem do CIDF

O interesse da pesquisa em questão é usar o CIDF para facilitar a distribuição de novos modelos de detecção de intrusão para que novos ataques tenham um curto tempo de vida ativa. A proposta é então estudar como um construtor de modelos, por exemplo o MADAM ID, pode receber dados de ataques, e depois rápida e automaticamente produzir modelos apropriados e distribuí-los para instalação nos SDI.

4.4.3.1 Considerações de Projeto

O desafio é construir e manter um conjunto de SIDs e um dicionário de seus possíveis valores que possa ser usado para descrever precisamente os cenários de ataque. Quando existe um serviço de modelagem, ao detectar uma nova intrusão (como uma anomalia), um SDI codifica os dados de auditoria relevantes (por exemplo, tráfego de rede numa janela de tempo) em GIDOs e os transmite ao mecanismo de modelagem, isto é, ao sistema MADAM ID. Este último executa então mineração de padrões, construção de características, e aprendizado de regras usando os dados de auditoria extraídos dos GIDOs.

O mecanismo de modelagem inclui funcionalidades de bases de dados e de mecanismos de análise. O tópico mais desafiante na adição de um mecanismo de modelagem no CIDF é a codificação de características e regras nos GIDOs. Um conjunto de SIDs precisa ser definido para expressar as instruções computacionais para cada característica e regra, de maneira que um SDI, ao receber um *Modelo de Detecção de Intrusão GIDO* possa automaticamente analisar (através de um *parser*) as instruções e gerar módulos de execuções locais.

³⁵ Do inglês: *Semantic Identifier*.

A seguir são apresentados os componentes de um sistema experimental construído como protótipo de suporte às teorias apresentadas.

4.4.3.2 O Mecanismo de Modelagem

Para codificar características e regras em GIDOs, primeiramente foi definido um conjunto de características “essenciais” que todos os SDI precisam saber como computar. Essas características incluem: estação de origem, porta de origem, estação de destino, tipo de protocolo, serviço, duração, tipo, etc. Cada característica recebe então um identificador único, e são definidos vários SIDs para especificar as características e seus valores.

Os operadores de construção de características incluem *count*, *percent*, *average*, etc. Para as restrições de dados é possível usar *same*, *different*, *time window*, etc. Um exemplo de uma das características do *SYN flood* pode ser: “para as conexões com mesma estação de destino nos últimos dois segundos, a porcentagem que tem a *flag* S0 é ...”. A representação dessa característica usando expressão-S seria assim:

```
(FeatureDef
  (FeatureID 'S0_rate')
  (Constraint
    (ConstraintType 'same')
    (ConstraintValue 'destination host')
  )
  (Constraint
    (ConstraintType 'time')
    (ConstraintValue '2 seconds')
  )
  (Operation
    (Operator 'percent')
    (FeatureCond
      (FeatureID 'flag')
      (Cond
        (CondOperator 'equal')
        (CondValue 'S0')
      )
    )
  )
)
```

Uma regra de detecção de intrusão, pode então ser especificada usando a seguinte forma:

```
(DetectionRule
  (AttackType ...)
  (FeatureDef ...) [optional]
  (FeatureCond ...)
)
```

Isto é, a expressão primeiro especifica o tipo de ataque (o nome da intrusão) que pode ser detectado por essa regra. Posteriormente, descreve as definições de qualquer nova característica usadas nessa regra, seguida de uma sequência de condições de características.

4.4.3.3 Os SDI

No sistema experimental foram usados os SDI Bro [LEE apud Paxson³⁶] e NFR [LEE apud Network Flight Recorder Inc.³⁷], e como ilustrado pela Figura 12, foram implementados um cliente e um *daemon* CIDF além de modificações realizadas no Mecanismo de Filtragem de Pacotes do SDI.

Figura 12: As interfaces CIDF dos SDI

Quando o *daemon*, que está escutando em uma porta, recebe uma conexão de um CIDF cliente, ele verifica se a mensagem recebida é um GIDO válido e o decodifica em expressões-

³⁶ LEE. A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions, Springer, 2000. Apud Paxson, V. Bro: A system for detecting network intruders in real-time. In Proceedings of the 7th USENIX Security Symposium, San Antonio, TX, 1998.

³⁷ LEE. A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions, Springer, 2000. Apud Network Flight Recorder Inc. Network flight recorder. <http://www.nfr.com>, 1997.

S. Os dados decodificados são então colocados na fila compartilhada de mensagens para sinalizar ao SDI para continuar o processamento.

Ao receber a mensagem (isto é, as expressões-S), o Mecanismo de Filtragem de Pacotes enfileira um evento junto com uma expressão-S na fila de eventos que ele compartilha com o interpretador de *scripts*. Eventos relacionados ao CIDF, como por exemplo, interpretar as expressões-S para iniciar as ações apropriadas, são gerenciados pela função `cidf_process`. Um “interpretador de modelos” é implementado como um processo separado, chamado pela função `cidf_process`, para analisar (*parser*) as expressões-S que descrevem uma nova regra de detecção de intrusão. Os gerenciadores de intrusões gerados localmente podem então ser inspecionados por humanos e carregados no SDI.

O cliente CIDF fornece os meios pelos quais um SDI pode codificar e enviar GIDOs para outros componentes CIDF. Quando existir informação (dado de auditoria, alertas de intrusão, etc.) a ser enviada, a solicitação é atendida pela função `cidf_send`. Essa função primeiramente constrói expressões-S e as escreve em um arquivo local, chama o processo CIDF cliente e passa o nome desse arquivo para ele. O cliente CIDF analisa (através de um *parser*) as expressões-S no arquivo e as codifica em um GIDO, que é depois enviado a um mecanismo de modelagem ou outro SDI. Uma expressão-S em um ataque *SYN flood* é a seguinte:

```
(Attack
  (AttackSpecifics
    (AttackID 0x00000004 0x00000000)
  )
  (When
    (Time 953076974.391314)
  )
  (Message
    (Comment 'SYN flood')
    (SourceIPV4Address 181.162.15.74)
    (TCPSourcePort 1551/tcp)
    (DestinationIPV4Address 152.1.205.85)
    (TCPDestinationPort 80/tcp)
    (TCPConnectionStatus 1)
  )
)
```

4.4.4 Experimentos

O sistema experimental foi instalado na rede do campus da universidade com o MADAM ID, o Bro e o NFR em diferentes sub-redes. Foram realizados três conjuntos de testes. O primeiro

deles tinha o objetivo de testar a interoperabilidade dos componentes CIDF e teve resultados satisfatórios, que não serão apresentados nessa dissertação.

O segundo conjunto de testes, foi projetado para testar o limite do mecanismo de modelagem, ou seja, foi medido quanto tempo o MADAM ID levaria para computar um modelo de detecção depois de ter recebido os dados de auditoria. O resultado apresentou medidas de alguns segundos até poucas horas. Depois de uma análise mais detalhada descobriu-se que o processo iterativo automatizado de mineração de dados, construção de características, computação dos classificadores, e avaliação de performance requereu mais coordenação na configuração heurística dos parâmetros para os programas. Por exemplo, detectar um ataque *SYN flood* pode exigir que padrões de “mesma estação de destino e mesmo serviço” sejam computados e comparados (com padrões normais), enquanto um ataque *port-scan* exige padrões “mesma estação de destino e serviço diferente”. Uma opção a ser pesquisada é a possibilidade de o SDI incluir uma descrição “alto nível” do comportamento do ataque no GIDO, para que o mecanismo de modelagem possa ter uma chance melhor de ajustar os parâmetros corretos para os algoritmos de mineração de dados.

Outro resultado desse segundo conjunto de testes é que para intrusões que necessitem de um novo pré-processamento dos dados crus de auditoria, é virtualmente impossível expressar a complexa e delicada lógica de pré-processamento de dados usando expressões-S. Por exemplo, ataques “*teardrop*” requerem processamento especial de fragmentos IP, os quais somente podem ser expressos em linguagens de programação de alto nível como C/C++. Na verdade, especialistas humanos precisam ser envolvidos no processo de modelagem quando tal pré-processamento de dados é necessário.

O terceiro conjunto de experimentos tencionava verificar se os SDI poderiam trabalhar juntos para detectar ataques distribuídos. A ferramenta usada para gerar os ataques nesse experimento foi o DDOS *Tribal Flood Network* (TFN)³⁸. O Bro monitorava a sub-rede da estação destino e detectou o ataque, que foi iniciado por vários programas escravos, cada um rodando em uma máquina separada na sub-rede monitorada pelo NFR. Quando o NFR recebeu o relatório de instância de ataque do Bro, ele analisou (através de um *parser*) o GIDO

³⁸ O CERT divulgou uma Nota de Incidente sobre essa ferramenta de DDOS. Pode ser consultado on-line em http://www.cert.org/incident_notes/IN-99-07.html (acesso em 15/02/2002).

e reconheceu que um ataque *SYN flood* parecia ter sido iniciado da sua sub-rede (checando o `SID SourceIPV4Address`). O NFR então ativou um N-code³⁹ que detecta e bloqueia as mensagens de controle (ICMP echo reply packets) enviadas pelos atacantes mestres aos seus escravos. Ele também iniciou uma busca pelos programas escravos, usando um *daemon* especial em cada sistema de sua sub-rede para procurar e matar qualquer processo com um nome que encaixe com qualquer um na lista de programas de ataques conhecidos. Os resultados desse experimento mostraram que apenas alguns segundos após a detecção do ataque pelo Bro, o NFR foi capaz de matar todos os programas atacantes.

4.5 Minerando Dados de Execução

A pesquisa de MICHAEL & GHOSH [2000]⁴⁰ apresenta as bases de uma técnica para identificação de traços de uso indevido usando autômatos finitos ensinados automaticamente. A abordagem é baseada no processo de encontrar características que distingam entre rastros de execuções maliciosas e benignas. Em outras palavras, são discutidas técnicas de mineração de dados para os rastros de execução de programas.

Durante os experimentos, o arquivo com os rastros de execuções de programas realmente maliciosos não era diversificado o suficiente para aprendizado de máquina, então foi usada uma definição artificial de comportamento malicioso: identificar navegadores (*web browsers*) baseando-se nos rastros de execução. As máquinas de estados finitos (MEF) foram treinadas usando o Netscape como programa “malicioso”, e tentou-se identificar não apenas rastros de execuções desconhecidos do Netscape, como também rastros do Internet Explorer, que não foi usado no treinamento.

4.5.1 Construindo Autômatos Finitos a partir de Informação de Auditoria

A abordagem proposta para o aprendizado das assinaturas de intrusão tem dois passos: primeiro, a informação de auditoria é transformada em uma série de eventos simbólicos de

³⁹ Linguagem *script* de análise de rede do Sistema de Detecção de Intrusão NFR.

⁴⁰ Todas as afirmações desta seção (4.5) são provenientes dessa fonte de referência, exceto onde explicitamente indicado.

auditoria, que depois serão usados para construir um autômato finito com transições rotuladas como ruins (“ruim”), que devem ser exercitadas apenas por programas maliciosos.

4.5.1.1 Pré-processamento da informação de auditoria

Os dados de comportamento foram recolhidos dos logs de segurança do Windows NT, cujo sistema de auditoria captura várias ações executadas pelos programas, como a invocação de um programa por outro programa, a terminação de um programa e acesso aos recursos. Para cada programa cuja invocação é gravada nos logs de auditoria, um traço de execução é preparado pela gravação desses eventos básicos. Isso significa que cada evento é associado a um número único, e esse número é gravado nos traços de execução quando esse evento acontece. Por exemplo, uma entrada do log de auditoria descrevendo um acesso de leitura e escrita a um arquivo resultaria em duas entradas, cada uma delas associada a um número no rastro de execução: uma leitura e uma escrita. Isso quer dizer que um rastro de execução pode não gravar fielmente a ordem das operações executadas em um objeto em um acesso único, no entanto, isso pode não ser crítico, uma vez que os rastros de execução são usados para identificar programas e não para determinar suas semânticas de execução exatas.

4.5.1.2 Algoritmos de Fusão de Estados para MEF

Algoritmos de fusão de estados começam com uma árvore de prefixos (também chamada *acceptor tree*) descrevendo os dados de treinamento. Cada aresta da árvore de prefixos é rotulada com os eventos que podem ocorrer nos dados de treinamento, que nesse caso é o conjunto de eventos que pode ser extraído de um log de auditoria de segurança do NT. Cada sequência de eventos nos dados de treinamento corresponde aos rótulos das arestas em um caminho através da árvore de prefixos, começando na raiz e, se existir um símbolo final único, terminando em uma folha.

Figura 13: Ilustração da fusão de estados

A Figura 13 mostra uma árvore de prefixos para as seqüências baadb, baadc, babb, babc. A ilustração da esquerda (A) mostra a árvore construída a partir dessas seqüências. Note que cada caminho da raiz até uma folha tem rótulos correspondentes a cada uma das seqüências. A ilustração da direita (B) mostra o resultado da fusão dos estados 4 e 5. Esses estados foram fundidos porque as subárvores enraizadas nos estados 4 e 5 (em A) são os mesmos.

O aprendizado acontece pela fusão seletiva de estados na árvore de prefixos. Algoritmos de fusão de estados diferem em suas escolhas de quais estados serão fundidos (e isso pode incluir uma decisão de não fundir certos estados com outros). O algoritmo usado foi o *Blue-fringe* [MICHAEL & GHOSH apud Lang & Pearlmuter⁴¹], que separa conjuntos de dados em três partes: os nós vermelhos que não podem mais ser fundidos entre si, os nós azuis que só podem ser fundidos com nós vermelhos, e os nós brancos que nunca foram fundidos e ainda não são candidatos à fusão. Inicialmente o nó raiz é vermelho e seus filhos são azuis, e um algoritmo *blue-fringe* preserva as seguintes invariáveis:

⁴¹ MICHAEL & GHOSH. Using Finite Automata to Mine Execution Data for Intrusion Detection. Springer, 2000. Apud Lang, R.P.K., Pearlmuter, B. Results of the abbadingo one dfa learning competition and a new evidence driven state merging algorithm. In *Proceedings of the International Colloquium on Grammatical Inference (ICGA-98)*, volume 1433 of *Lecture Notes in Artificial Intelligence*, pages 1-12. Springer-Verlag, 1998.

- Os nós vermelhos formam um grafo arbitrário mas não podem ser fundidos uns com os outros;
- Cada filho de um nó vermelho é azul ou vermelho;
- Cada nó azul é o raiz de uma árvore.

Fundir um nó azul com um vermelho resulta em um nó vermelho, significando que os filhos do nó azul devem ser promovidos a azuis para preservar as invariáveis. O algoritmo pode também decidir promover um nó azul a vermelho sem uma fusão, significando que não existe nó vermelho adequado à fusão; aqui os nós brancos também devem ser promovidos. Finalmente, a fusão de um nó azul e de um branco pode fazer com que o autômato se torne não determinístico, e esse problema é resolvido recursivamente fundindo os filhos apropriados. Isso também pode levar a promoções.

A decisão de fundir ou não dois estados geralmente é baseada na estrutura de suas proles [MICHAEL & GHOSH apud Trakhtenbrot & Barzdin⁴²], mas no contexto da aplicação aqui proposta, a árvore de prefixos é muito mais esparsa do que em muitos outros problemas de aprendizado de MEF, portanto, o algoritmo usado somente checa as subárvores até uma certa profundidade. Essa idéia foi usada por FREUND et al. [1997] para melhorar a performance, mas foi usada no trabalho aqui apresentado para reduzir a quantidade de informação usada para decidir quando fundir estados, reduzindo a probabilidade de uma má fusão causada pela falta de informação. Será adotada a terminologia proposta por FREUND et al. [1997] onde, a subárvore dentro da profundidade limite de um nó é chamada de *assinatura* daquele nó.

A assinatura é, portanto, uma árvore de prefixos menor, onde cada string aceita pela árvore de assinatura do nó k consiste dos l -ésimo até o $(l + D)$ -ésimo símbolos de algum traço de execução nos dados de treinamento, onde l é a profundidade do nó k , e D é a profundidade da assinatura. Por exemplo, a assinatura de profundidade 2 do nó 2 na Figura 13 (A) aceita as strings ad , bb e bc . Essas strings (as aceitas pela árvore de assinatura do nó k) serão chamadas de *strings de assinatura* do nó k .

As strings de assinatura são usadas para implementar a seguinte heurística: são comparadas as *densidades de probabilidades* estimadas das strings de assinaturas de dois nós para

⁴² MICHAEL & GHOSH. Using Finite Automata to Mine Execution Data for Intrusion Detection. Springer, 2000. Apud Trakhtenbrot, B.A., Barzdin, Y.A. Finite Automata: Behavior and Synthesis. North-Holland, 1973.

descobrir se eles devem ser fundidos, isto é, eles só serão fundidos se as suas densidades de probabilidades parecerem similares. Para entender essa abordagem, note que alguns nós da árvore de prefixos, como o nó 2 na Figura 13 (A), são alcançados por mais de um exemplo de treinamento. Na verdade, todos os quatro exemplos de treinamento alcançam o nó 2 porque todos os quatro começam com os símbolos *ba*. Na Figura 13 (B), depois de uma fusão, todos os exemplos também alcançam o nó 4, porque todos começam com *baad* ou *bab*. A probabilidade de uma string de assinatura *s*, dado que o nó *k* é alcançado, pode então ser definida como a probabilidade de encontrarmos *s* depois de alcançado o nó *k*. Formalmente, é a probabilidade de encontrar um rastro de execução com os *l*-ésimo até *(l + D)*-ésimo símbolos da string *s*, dado que os símbolos do rastro iniciando no primeiro até o *(l - 1)*-ésimo, correspondem aos rótulos das conexões do caminho começando na raiz e terminando no nó *k*. A profundidade do nó *k* é indicada por *l*. Note que a heurística usada é implicitamente baseada na premissa que os nós agem como estados de um processo de Markov, que pode não ser verdade em todos os programas.

Sendo $v_k(s)$ o número de vezes que a string de assinatura *s* é encontrada depois de alcançado o estado *k*, e n_k o número de exemplos de treinamento que alcançam o estado *k*, então a probabilidade estimada para a string *s* dado que estamos no nó *k*, é apenas v_k / n_k . Será usado $\hat{p}(s | k)$ para denotar essa estimativa.

Para definir o escore da qualidade de uma fusão entre dois estados *j* e *k*, é usada a métrica L_d , que é definida como:

$$L_d(j, k) = \left(\sum_{i=1}^m \left| \hat{p}(s_i | j) - \hat{p}(s_i | k) \right| \right)^{1/d}$$

onde *m* é o número total de possíveis strings de assinaturas, e $\hat{p}(s_i | k)$ é zero se a string *s_i* nunca alcançar o estado *k*. Aqui, *d* pode ser qualquer número positivo e seu valor atual é um parâmetro do algoritmo de treinamento.

Em resumo, quando se considera fundir dois nós *j* e *k*, a qualidade da fusão é julgada usando-se o $L_d(j, k)$ para um *d* qualquer; um valor menor indica uma fusão melhor porque isso indica que as probabilidades (estimadas) de todas as strings são próximas entre si. Se

existirem vários pares de nós que se deseja fundir, só será fundido o par que tiver o escore L_d mais baixo.

Pode ser que todas as fusões sendo consideradas sejam tão pobres que nenhuma fusão deva ser feita. Isso acontece quando cada fusão tem uma medida L_d maior do que um *threshold* ajustado pelo usuário. Quando isso acontece, é feita uma promoção ao invés de uma fusão.

4.5.2 Mineração de Dados usando MEF ensinadas

Com o objetivo de descobrir características que podem ser usadas para distinguir executáveis maliciosos dos não-maliciosos, a abordagem proposta usa dados de treinamento rotulados; uma dada sequência de eventos de auditoria pode ser rotulada como boa “bom”, significando que o executável não é malicioso, ou “ruim”, significando que o executável é malicioso. Essa abordagem funciona da seguinte maneira:

1. Construir uma MEF a partir dos rastros de execução “bom” e “ruim”.
2. Se qualquer transição da MEF só for exercitada por rastros do tipo “ruim”, então ela é rotulada como “ruim”. Todas as outras transições são rotuladas como boas (“bom”).

Durante os testes, são percorridos os traços de um novo programa através da MEF para determinar se o executável deve ou não ser considerado malicioso. É possível então dar um alarme assim que uma transição “ruim” for exercitada, ou manter o escore do número de transições ruins que foram exercitadas pelo rastro.

Note que, de acordo com o item número 2 acima, as transições que são exercitadas somente por executáveis maliciosos são isoladas; procurando comportamento que é exclusivo de executáveis maliciosos, tenta-se encontrar as características que podem ser usadas para identificá-los. A vantagem dessa abordagem é que ela ainda vai funcionar se alguns rastros não-maliciosos forem acidentalmente incluídos com os maliciosos. Isso significa simplificar a coleta de dados de treinamento; se por exemplo, sabe-se que uma intrusão aconteceu por volta de um determinado momento, mas não se sabe ao certo quais executáveis estavam envolvidos no ataque, pode-se simplesmente jogar todos os rastros de executáveis ocorrendo perto daquele momento na categoria “ruim”.

Alguns detalhes específicos: foram usadas assinaturas de profundidade 2 quando da comparação de nós para identificar se eles poderiam ser fundidos, e usou-se a distância L_1 para comparação. O par vermelho-azul com o escore L_1 mais baixo era fundido, a menos que

o escore mais baixo fosse maior do que 1, caso em que o nó azul mais superficial era promovido a vermelho.

4.5.3 Experimentos

Nesta seção são apresentados, primeiramente o ambiente experimental, e depois algumas execuções experimentais.

4.5.3.1 Obtendo dados de comportamento para executáveis do NT

Como foi explicado anteriormente, os navegadores Netscape e Internet Explorer foram considerados executáveis “maliciosos”. O objetivo nestes testes é treinar o sistema com vários programas e com um dos dois navegadores rotulado como malicioso. A esperança é que, devido à similaridade funcional entre os dois navegadores, o sistema possa ser treinado para reconhecer qualquer um deles como “ruim”, enquanto reconhece outros programas como bons (“bom”).

Em particular, se for possível identificar traços de execução de um navegador depois de treinar o sistema somente com traços de execução do outro navegador, estará evidenciado que essa técnica pode, de fato, isolar características úteis para generalização em traços de execução.

Os logs de segurança usados nos experimentos vieram de duas fontes distintas: da avaliação de 1999 dos sistemas de detecção de intrusão feita pelo Lincoln Laboratory, e de um dos sistemas da própria universidade, usado para capturar traços adicionais do Internet Explorer, já que os dados do LL tinham um único rastro desse tipo.

4.5.3.2 Experimento 1

No primeiro experimento, o programa “ruim” foi o Netscape e os marcados como “bom” foram findfast, ls, spoolss e advanced e usaram-se os dados da segunda à quinta semanas do LL para o treinamento. Os testes foram executados usando os dados da primeira semana do LL. O classificador foi, a princípio, testado com os programas usados para treinamento, mas os traços de execução foram tirados da primeira semana. Além disso, o classificador foi testado com os seguintes programas que não foram usados durante o treinamento: dotlnetd, explorer, iexplore, nsbind, perl, posix e rpcss. Para esse experimento, um alerta era feito quanto uma transição “ruim” era alcançada.

Figura 14: Escores para programas usados no treinamento para o Experimento 1
(A) *findfast*; (B) *ls*; (C) *spoolss*; (D) *advanced*; (E) *netscape*

Os resultados para os programas de treinamento são mostrados na Figura 14. Cada programa foi executado várias vezes, e os diferentes rastros de execução são mostrados ao longo do eixo horizontal. O eixo vertical mostra o escore para cada rastro, e que pode ser um dos três valores:

- 0 (zero) - o rastro de execução foi aceito pelo autômato e não foi rotulado “ruim”;
- 0,1 - o rastro foi rejeitado antes de passar por uma conexão que o rotularia “ruim”;
- 1 - a execução foi rotulada como “ruim”.

Para os programas usados para treinar o autômato, os resultados foram mais ou menos como esperado. Os quatro programas, os quais o classificador foi instruído a rotular como bons tiveram baixos escores enquanto o Netscape teve altos escores. No entanto, alguns traços do *ls* tiveram altos escores, ou seja, estes traços levantaram alarmes falsos.

A Figura 15 ilustra os resultados obtidos para os programas que não foram usados para o treinamento. O objetivo é identificar o comportamento dos navegadores, e isso significa conseguir escores altos para o Internet Explorer (*iexplore*) e escores baixos para os outros programas. Como visto na Figura 15, três dos programas tiveram escores altos embora esse não fosse o resultado esperado; novamente isso pode ser visto como a geração de alarmes falsos. Dez dos onze rastros de execução do Internet Explorer foram identificados, mas um foi rejeitado antes de ser rotulado como “ruim”.

Figura 15: Escores para os programas não usados no treinamento para o Experimento 1
(A) perl; (B) posix; (C) iexplore; (D) inetinfo; (E) nsbind; (F) rpcss

4.5.3.3 Experimento 2

Para aperfeiçoar essa situação, isto é, eliminar os falsos positivos causados pelo nsbind, inetinfo e rpcss, e prevenir o iexplore de cair fora da MEF, é preciso usar mais dados de treinamento. Para fornecer caminhos para o rpcss, nsbind e inetinfo que não envolvam transições ruins, pode ser suficiente adicionar mais programas bons, e já que o interesse é fazer com que a MEF aceite o iexplore, mais dados de treinamento bons devem ser suficientes para isso também. Note que a intenção com essa solução é observar o comportamento do algoritmo de aprendizado, e não reduzir os falsos positivos tão rápido quanto possível.

Os resultados do segundo experimento são encorajadores. Todas as execuções do iexplore agora lançam um alarme, enquanto o nsbind baixou seus escores. Infelizmente o inetinfo e o explorer continuam gerando altos escores, e o rpcss e o ls geraram muitos falsos positivos, apesar do ls ter sido usado no treinamento.

4.5.3.4 Experimento 3

Uma maneira de fazer com que o inetinfo pare de gerar falsos positivos, seria adicionar mais dados de treinamento, continuando o processo iniciado anteriormente. No entanto, a adição de perl ou posix aos dados de treinamento como programas bons não teve o efeito desejado.

É importante notar que a adição de mais dados de treinamento não só adiciona estado e transições ao autômato, como também muda a sua estrutura, porque existem mais estados disponíveis para fusão. Dessa maneira, cada programa que fosse adicionado aos dados de treinamento como “bom” levaria potencialmente a uma MEF diferente, e é provável que cedo ou tarde o resultado seria um autômato que atende às necessidades desejadas apenas por acidente. No entanto, demonstrar isso empiricamente seria um processo muito custoso.

Figura 16: Escores para programas usados no treinamento para o Experimento 3
(A) *findfast*; (B) *ls*; (C) *spoolss*; (D) *advanced*; (E) *dotlnetd*; (F) *inetinfo*; (G) *netscape*

Portanto, para esse experimento foram adicionados traços de execução do *inetinfo* aos dados de treinamento, rotulados como “bom”. Isso produziu o efeito desejado, como ilustrado pela Figura 16, que mostra os resultados para os programas bons durante o treinamento.

Os programas usados para o treinamento agora se comportam como deveriam, ou seja, somente o Netscape gerou altos escores e o `ls` não gerou mais falsos alarmes.

Figura 17: Escores para os programas não usados no treinamento para o Experimento 3
(A) `perl`; (B) `posix`; (C) `iexplore`; (D) `nsbind`; (E) `rpcss`; (F) `explorer`

Entre os programas não usados durante o treinamento, os resultados também foram satisfatórios; os escores para os programas que não são navegadores foram baixos (exceto por um traço do `rpcss` que gerou um alarme falso), e os escores do `iexplore` foram todos altos; isso é mostrado na Figura 17.

4.6 Considerações Finais

A pesquisa acadêmica na área de sistemas de detecção de intrusões é muito dinâmica e tem produzido material em diversificadas áreas. Por exemplo, definições de padrões formais para criação de logs de auditoria [FLACK & ATALLAH, 2000], bases de dados de vulnerabilidades em um formato que possa ser interpretado por máquinas, e que atualizam automaticamente os sistemas subscritos quando uma nova vulnerabilidade é identificada [RIORDAN & ALESSANDRI, 2000], correlação estatística entre alarmes de diferentes SDI [VALDES & SKINNER, 2001], etc. Porém, o interesse desse trabalho é identificar as tecnologias de reconhecimento de padrões sendo aplicadas à detecção de intrusão. Esse capítulo apresentou trabalhos que propõem o uso de mineração de dados, máquinas de estados finitos, algoritmos genéticos, sistemas imunológicos computacionais e redes neurais como mecanismos para o reconhecimento de padrões em sistemas de detecção de intrusão.

A razão pela qual nenhum sistema comercial foi estudado, é que a informação disponível sobre eles se constitui geralmente de artigos de marketing, e encontrar especificações técnicas, mesmo as mais básicas, provou ser um trabalho exaustivo e infrutífero.

O capítulo seguinte faz uma análise de cada um dos artigos aqui apresentados, visando a sua utilização no SDI do ICMC. Posteriormente levanta as dificuldades encontradas no decorrer da pesquisa e também faz sugestões para trabalhos futuros.

5

5 ANÁLISE E PROPOSTA

O interesse em detecção de intrusões nunca foi tão grande. Por ser ainda uma área recente e que oferece vários desafios, o número de pesquisas e trabalhos que propõem soluções aos problemas relacionados a esse tema é também enorme.

Por causa disso, uma das dificuldades encontradas durante o trabalho de avaliação das metodologias de reconhecimento de padrões aplicadas à segurança computacional surgiu logo no começo: filtrar e identificar dentre essa enorme quantidade de material, os trabalhos que seriam úteis. O dinamismo dessa área de pesquisa é muito grande, tornando difícil separar técnicas que se tornaram obsoletas das que se tornaram clássicas; as que são inovadoras das que estão amadurecidas. Junte a isso tudo, o problema de fazer uma pesquisa “entre áreas”, ou seja, relacionada à segurança computacional e à inteligência artificial, duas das mais produtivas na atualidade.

Uma vez definidos os artigos a serem estudados, surgiu a próxima dificuldade: como compará-los? Não existe ainda nenhuma maneira padronizada de se comparar a performance de sistemas de detecção de intrusão, ou seja, não existem medidas de *benchmark* para tais sistemas. Alguns dos trabalhos estudados foram testados durante a avaliação de SDI da DARPA, mas mesmo assim apresentaram seus resultados utilizando medidas diferentes. Esse problema torna difícil fazer uma comparação quantitativa entre qualquer dos métodos e algoritmos aqui estudados, a não ser que cada um deles fosse implementado e testado nas mesmas circunstâncias.

Grande parte da performance dos Sistemas de Detecção de Intrusão depende da qualidade dos dados usados para treinamento e testes. Dessa maneira, uma padronização dos dados a serem

usados para testes de performance de SDI é fundamental para a definição de um bom processo de *benchmark*. Parte desse problema está sendo resolvido, porque a DARPA divulgou a base de dados de teste usada na sua Avaliação de SDI, e alguns pesquisadores estão usando essa base de dados para executar os testes de seus sistemas. Usando a mesma base de dados, eles podem comparar os resultados apresentados por seu algoritmo com o de outras abordagens. Somente uma das abordagens estudadas neste trabalho apresenta seus resultados no formato divulgado pela avaliação da DARPA: Probabilidade de Detecção / Alarmes Falsos.

Como fazer então a comparação entre propostas técnicas sem ter medidas comparáveis para cada uma delas? A solução encontrada foi a de fazer uma análise qualitativa de cada uma das abordagens estudadas, tentando levantar atributos que pudessem indicar a possibilidade ou não de sua utilização no ambiente de agentes móveis modelado por BERNARDES [1999]. A arquitetura proposta para esse sistema é apresentada a seguir.

5.1 Arquitetura do Sistema

Em sua dissertação, BERNARDES [1999] apresenta uma arquitetura baseada em agentes como especificação para um sistema de detecção de intrusão. Ele afirma que o conceito principal que envolve o SDI baseado em agentes autônomos e móveis é a simplicidade. Cada agente é uma entidade simples que irá desempenhar uma atividade específica e cooperar com outros agentes da forma mais eficiente possível. Quando uma atividade for considerada suspeita por um agente, ele irá comunicar aos demais agentes do sistema, a sua suspeita de uma possível intrusão. Neste momento, será acionado um agente (ou um conjunto deles) com um grau maior de especialização naquele tipo de suspeita.

Nenhum agente possui a autoridade de, sozinho, identificar uma intrusão. Essa decisão será tomada com base no consenso de vários agentes no sistema. Se somente um agente suspeita de uma intrusão, ele poderá ser ignorado após uma votação dos agentes envolvidos naquela suspeita. Entretanto, se mais de um agente suspeita de um comportamento anômalo, então há uma maior probabilidade de ser uma intrusão potencial e neste caso, poderá ser tomada a decisão de comunicar, um operador humano ou acionar agentes especializados em contra ataque.

A Figura 18 apresenta uma arquitetura (modelo em camadas) para o SDI proposto. As camadas são numeradas a partir da camada de Vigilância (camada 1), e cada uma delas representa um grupo de tarefas específicas desempenhadas por agentes especializados nas

funções desta camada. Através do mecanismo de troca de mensagens, um agente em uma camada aciona um ou mais agentes em uma camada superior. Em outras palavras, a camada N utiliza os serviços da camada $N-1$, desempenha suas funções e fornece serviços para a camada $N+1$.

Figura 18: Modelagem em camadas para o sistema baseado em agentes.
Fonte: [BERNARDES, 1999]

Com base em informações coletadas pelos *Agentes de Vigilância*, *Agentes de Tomada de Decisão* entrarão em ação, analisando e identificando possíveis intrusões. Caso uma ação seja considerada suspeita por estes agentes, *Agentes de Notificação* serão acionados e cuidarão de notificar o administrador da rede (via *e-mail*, *pager*, chamada telefônica, alarme, etc.) ou acionar os agentes de nível superior. Em último nível, encontram-se os *Agentes de Reação*. Estes agentes, cuidarão de contra-atacar automaticamente as possíveis intrusões, com base nas informações dos agentes de notificação ou ainda, serem acionados através de uma intervenção do administrador da rede.

A ampliação do SDI para atender uma nova configuração de ataque poderá envolver o desenvolvimento e conseqüente adição de novos agentes em uma única camada ou ainda, a criação de um novo cenário que envolverá a adição de agentes em todas as camadas.

Esse modelo foi posteriormente revisado por PEREIRA FILHO [2001]. Em sua dissertação, são levantados alguns dos possíveis problemas relacionados à segurança e integridade do próprio sistema proposto por Bernardes.

5.2 Análise dos Trabalhos Apresentados

O domínio de problemas relacionados à detecção de intrusão é enorme, variando de sistemas baseados em rede capturando pacotes de toda uma organização, até sensores baseados em estações que monitoram as atividades de um processo único em uma máquina específica. A grande maioria das propostas encontradas durante essa pesquisa aborda uma pequena parcela dos problemas relacionados a cada um dos modelos de sistemas de detecção de intrusão.

De uma maneira geral, e isso é discutido nos próprios artigos, quase todos os trabalhos estudados realizaram experimentos que são muito limitados, ora pela quantidade de dados de treinamento e/ou teste, ora pela qualidade desses mesmos dados.

Um dado importante é que os experimentos de todos os trabalhos apresentados foram realizados off-line, ou seja, em um ambiente controlado sem interferências externas. Até mesmo a aclamada Avaliação de SDI da DARPA realizou, em suas duas versões (1998 e 1999), os seus experimentos em um ambiente desse tipo.

5.2.1 Sistema Imunológico Computacional para Detecção de Intrusões

O objetivo do sistema proposto por WILLIAMS et al. [2001], apresentado na seção 4.1, é aperfeiçoar os sistemas de detecção de intrusão baseados em rede, tornando-os capazes de agirem por antecipação, ou seja, reconhecer ataques desconhecidos.

O fato de o sistema usar somente o cabeçalho, ou seja, ignorar o *payload* dos pacotes TCP têm sérias implicações práticas. A principal desvantagem dessa abordagem é que uma grande variedade de ataques não pode ser detectada quando não se inspeciona o *payload* dos pacotes, como por exemplo, em ataques de *buffer overflow*. Para investigação, fiz o download da mais recente base de regras de detecção do snort⁴³. Essa base de regras é automaticamente gerada a

⁴³ Snort é, de acordo com o site oficial, “um sistema de detecção de intrusão peso-leve, capaz de executar análise de tráfego e log de pacotes em tempo real em redes IP”. Pode ser encontrado on-line em: <http://www.snort.org> (acesso em 21/02/2002).

cada 30 minutos diretamente do programa de controle de versões e disponibilizada para download. De todas as vulnerabilidades cobertas por esse arquivo de regras (1284), um total de 93,9% (1206 regras) dependem da checagem do conteúdo do pacote (*payload*), ou seja, somente em aproximadamente 6% dessas assinaturas o conteúdo do pacote não é importante.

Os testes executados pelos autores mostram que a quantidade de anticorpos afeta as taxas de falsos positivos e de falsos negativos, e esses resultados preliminares indicam que 512 anticorpos fornecem boa detecção com baixa taxa de falsos negativos. Porém, a abordagem proposta usa busca não determinística, que é normalmente um processo computacionalmente caro. Então, seria sensato estimar e comparar o tamanho do espaço de busca representado pelos 512 anticorpos com espaço de busca representado pelos grupos de *self*. Em nenhum momento é apresentada uma informação nesse sentido, que poderia ser útil na busca de métodos alternativos à função de encaixe imperfeito usada no protótipo Warthog.

Em seu trabalho, REAMI [998] apresentou algumas metáforas interessantes para o sistema de gerenciamento de segurança apoiado por agentes móveis, inspiradas na tradição medieval japonesa, nas sociedades de insetos, etc. Porém em uma dessas metáforas, ele compara o sistema de agentes móveis ao sistema de defesa do organismo, no qual células especializadas realizam o trabalho de destruição de elementos considerados estranhos. Essas células especializadas são os anticorpos, e nesse contexto, a associação do trabalho de WILLIAMS et al. [2001] com o de BERNARDES [1999] é praticamente imediata.

Algumas considerações sobre o que poderia ser feito para viabilizar a implementação da modelagem de um sistema de agentes representando os anticorpos descritos por Williams:

- Nos resultados experimentais apresentados, o número ideal de anticorpos era 512. Porém não existem parâmetros para afirmar qual seria esse número na prática. É importante lembrar que nesses experimentos não foram aplicados os algoritmos de maturação de afinidade, então não se sabe o efeito que eles teriam na quantidade de anticorpos criados. O ponto aqui é que se esse número de anticorpos for alto, uma modelagem um pra um (de um anticorpo em um agente) tornaria o modelo do SDI baseado em agentes muito complexo, contrariando a sua atual proposta.
- Uma alternativa seria criar agentes que fossem formados por grupos de anticorpos. Para fazer essa integração ao sistema proposto por Bernardes, o ciclo de vida do anticorpo CIDS teria que ser modificado como indicado pela Figura 19. O processo

adicionado está destacado e se chama Agrupamento. Ao ser criado, um anticorpo deve ser agrupado com outros, definindo uma estrutura que contém vários hipervolumes. Os *agentes de agrupamento* (AA) criados seriam então representações dessa estrutura, capazes de identificar se um determinado pacote de rede está presente nos hipervolumes por ele mapeados.

Figura 19: Ciclo de vida de um anticorpo CIDS com suporte a agentes

- Assim como o processo de Maturação de Afinidade, o de Agrupamento pode ser computacionalmente caro dependendo do método de agrupamento escolhido. Uma alternativa para essa situação, seria fazer a criação dos agentes uma única vez em um servidor, e depois distribuí-los a todas as máquinas onde se faz a captura de pacotes.
- Os *agentes de agrupamento* devem ser suportados por um *agente de controle de agrupamentos* (ACA). Esse último deve ser capaz de: identificar quais os agrupamentos definidos atualmente, consulta-os para definir se um pacote capturado da rede é *self* ou *non-self* e retornar o resultado dessa consulta.

- O resultado retornado pelo *agente de controle de agrupamentos* deve ser analisado por um agente mais especializado. Uma alternativa possível seria adaptar o *agente de análise de pacotes* (AAP) para que ele pudesse interpretar também o resultado enviado pelo ACA. O AAP é definido em BERNARDES [1999] da seguinte maneira: “agente que recebe os pacotes do agente de captura (Camada de Vigilância) e procede com sua análise. As atividades desempenhadas por este agente representam um nível de complexidade elevado, uma vez que será dotado de algum mecanismo de tomada de decisão.”

Figura 20: Proposta para o uso de anticorpos

O cenário que acaba de ser proposto é representado na Figura 20. Obviamente as considerações feitas acima são apenas sugestões iniciais, que visam indicar a possibilidade do uso das técnicas propostas por WILLIAMS et al. [2001] no sistema de detecção de intrusão baseado em agentes.

5.2.2 Filtros Baseados em Reconhecimento de Padrões

O trabalho de KURI et al.[2000], apresentado na seção 4.2, aborda o problema de performance relacionada à detecção de intrusão. O referido problema é a lentidão dos

algoritmos que detectam precisamente ataques presentes em trilhas de auditoria. Essa lentidão, causada pela complexidade do algoritmo, acaba fazendo com que os sistemas fiquem impossibilitados de tratar enormes quantidades de dados gerados durante o monitoramento do sistema. A idéia apresentada se baseia no fato que mesmo que seja difícil e lento determinar se um ataque ocorreu ou não, é possível determinar rapidamente que *não* existem ataques em grandes porções da trilha de auditoria.

Apesar de a abordagem proposta ter sido usada para detecção por uso indevido, também poderia ser usada para detecção por anomalias, por exemplo, verificando se um processo repete o comportamento apresentado durante o treinamento. Esse algoritmo também poderia usar, ao invés de logs de auditoria, pacotes de rede como dados para o seu treinamento. Obviamente, cada uma dessas outras aplicações teria que ser modelada de uma maneira semelhante, ou seja, como um problema de *reconhecimento de padrão aproximado múltiplo*⁴⁴, onde *inserção* é a única operação de edição permitida.

Os algoritmos para busca múltipadrão apresentados em KURI & NAVARRO [2000] podem alcançar velocidades de escaneamento impressionantes. Por exemplo, no caso de padrões com 4 letras e permitindo 4 inserções, os algoritmos permitem procurar por 100 padrões a uma taxa de 4 Megabytes por segundo em um servidor Sun Enterprise 450.

Vale ainda atentar que o fato de o valor de k (número de inserções) poder ser configurado, permite que se faça ajustes no comportamento do algoritmo, priorizando a velocidade ou a precisão durante a filtragem (recorde-se da Figura 9).

Os autores levantam a necessidade de realizar experimentos com conjuntos de dados maiores e mais realistas para fornecer estimativas mais precisas sobre a eficiência da filtragem. Porém, a flexibilidade de uso e a performance apresentada, tornam esse algoritmo um parceiro muito promissor para os SDI.

A proposta de KURI et al.[2000] apresenta um perfil característico da Camada de Agentes de Vigilância proposta por BERNARDES [1999] (ver Figura 18). Essa camada é formada por um conjunto de agentes responsáveis pela monitoria, coleta de informações, testes de ambiente e ajustes de configuração. Os filtros propostos em KURI et al.[2000] seriam então

⁴⁴ Do inglês: *Multiple Approximate Pattern Matching*

nesse contexto, implementados como agentes responsáveis por intermediar a transmissão do resultado produzido por alguns agentes da Camada de Agentes de Vigilância (Camada 1) para os agentes da Camada de Agentes de Tomada de Decisão (Camada 2).

Na Figura 21, são representados um Agente de Captura de Pacotes de Rede e um Agente de Captura de Eventos de Logs, cada um deles reportando seus resultados a um Agente de Filtragem. Esse agente de filtragem é que seria então responsável por fazer a transmissão dos dados filtrados e de uma possível identificação de intrusão para a camada superior. Claro que o número e o tipo dos Agentes de Filtragem pode variar de acordo com a necessidade. Uma outra opção seria, por exemplo, ter agentes diferentes e especializados para filtrar dados de diferentes origens (rede e log). Nesse modelo, os agentes que não dependessem da aplicação de filtros, continuariam se comunicando normalmente com a Camada de Tomada de Decisões.

Figura 21: Agente de Filtragem na Camada de Vigilância

Note que na Figura 21, o Agente de Filtragem tem uma seta bidirecional de comunicação com a Camada de Tomada de Decisão, a idéia dessa representação é mostrar a possibilidade que a camada superior tem de fazer o ajuste do parâmetro k do algoritmo de filtragem. O agente da Camada 2 responsável por fazer esse ajuste poderia, por exemplo, decidir incentivar a velocidade ao invés da precisão em momentos de uso crítico do sistema.

A integração de algoritmos de filtragem de alta performance aos Sistemas de Detecção de Intrusão poderia trazer inúmeros benefícios aos pesados algoritmos de detecção.

5.2.3 Aprendizado de Comportamento de Programas

A motivação do trabalho de GHOSH, MICHAEL & SCHATZ [2000], apresentado na seção 4.3, é detectar anomalias nos perfis de comportamento de programas. Essa abordagem surgiu recentemente como uma alternativa viável aos métodos de detecção de intrusão, porque na prática, grande parte das intrusões se inicia com o uso indevido de programas para a obtenção de privilégios.

Uma vantagem desse método é que, não importando o tipo de ataque sendo perpetrado, se ele envolver uso indevido de programas, as chances de detecção são enormes (de acordo com os experimentos apresentados). Por outro lado, qualquer ataque que envolva somente pacotes de rede (por exemplo, DoS) passará despercebido pelo detector. Portanto, o escopo desse método tem mais a ver com a maneira com que um ataque afeta o comportamento de um programa do que com qualquer outro atributo.

De uma maneira geral, o desempenho dos sistemas propostos com os dados do LL foi muito bom, porque atingiram aproximadamente 100% de detecção a uma taxa de menos de 10 falsos positivos por dia. Dois dos sistemas, a rede neural de Elman e o transdutor de string, detectaram todos os ataques com menos de 4 alarmes falsos por dia. Esses números se tornam mais impressionantes ao recordarmos que a proposta especifica um método de detecção por anomalia, e uma das características de sistemas desse tipo é a grande quantidade de alarmes falsos.

Concluindo, apesar de estarem limitados a detectar ataques que façam uso indevido de programas, um sistema com capacidade de generalizar e que tenha alto desempenho na detecção e produza baixas taxas de alarmes falsos, vale a pena ser investigado.

O sistema proposto por BERNARDES [1999] não prevê agentes que suportem o método de detecção baseado em programas aqui proposto. Sendo assim, a integração de, no mínimo mais dois agentes ao sistema teria que ser implementada:

- Na Camada de Vigilância, um *agente de captura de perfis de execução* precisaria ser criado. Esse agente seria responsável por capturar os eventos de execução (*system calls*) gerados pelos programas. Experimentos precisam ser feitos no sentido de determinar a necessidade da existência de um agente para cada programa sendo

monitorado, ou se um único agente seria capaz de monitorar vários programas sem perder performance.

- Na Camada de Tomadas de Decisões, um *agente de identificação de desvio comportamental* precisaria ser criado. Esse agente seria responsável por analisar se o comportamento transmitido pelo(s) agente(s) de captura de perfis de execução é anômalo ou não.

A seção *trabalhos relacionados* do artigo de GHOSH, MICHAEL & SCHATZ [2000] é uma boa referência para encontrar mais informações sobre esse método de detecção.

5.2.4 Detecção de Intrusões Distribuídas e Desconhecidas

A pesquisa de LEE et al. [2000] , apresentada na seção 4.4, discute a necessidade de novas técnicas para a detecção de intrusões desconhecidas e também de ataques distribuídos. Para o compartilhamento de informações, os autores usam o CIDE, uma infra-estrutura que fornece capacidade de análise e reação a intrusões e um mecanismo de comunicação baseado no formato GIDO. Essa infra-estrutura é equivalente a uma grande parte do sistema proposto pelo Grupo de Segurança do ICMC [REAMI, 1998]; [BERNARDES, 1999]; [PEREIRA FILHO, 2001], com a diferença de não suportar agentes, que constituem o grande valor da arquitetura proposta. A integração da CIDE à nossa arquitetura implicaria na remoção de grupos de agentes das diversas camadas, e a manutenção e suporte (por exemplo, no formato de mensagens trocadas) de duas plataformas: a CIDE e um servidor de agentes.

Uma coisa que também vale notar é que as expressões-S são definidas em um formato bastante lógico, porém proprietário. Esse formato se assemelha muito ao especificado pela linguagem XML, que é um padrão aberto para o qual pode-se encontrar várias bibliotecas de suporte. O uso dessas bibliotecas tornaria o código mais leve, pois existem *parsers* que verificam e validam um arquivo no formato XML com o seu documento de especificação (DTD). O uso de atributos de *tags*, característica das linguagens de marcação, se usado da forma correta, poderia simplificar a representação das assinaturas representadas em expressões-S. Por exemplo, convertendo para o formato XML a representação da característica “para as conexões com mesma estação de destino nos últimos 2 segundos, a porcentagem que tem a *flag* S0 é...” apresentada no item 4.4.3.2, teríamos algo do tipo:

```
<featureDef ID = "S0_rate">  
  <constraint type = "same">
```



```

        destination host
    </constraint>
    <constraint type = "time">
        2
    </constraint>
    <operation operator="percent">
        <featureCond ID="flag">
            <cond operator="equal">
                S0
            </cond>
        </featureCond>
    </operation>
</featureDef>

```

Neste exemplo simples, houve uma redução de 11,4% na quantidade de caracteres (de 263 para 233), uma característica interessante quando se deseja transmitir dados pela rede.

O problema da detecção de ataques de DDoS abordado no trabalho de LEE et al. [2000], pode também ser tratado em um ambiente de agentes, desde que os sistemas vítima e atacante estejam sob supervisão de plataformas que troquem dados em formatos compatíveis ou padrões. E por maneira semelhante, quero dizer por exemplo, que ao detectar traços de um ataque distribuído, o sistema vítima pode enviar agentes que tenham o conhecimento dos dados do ataque a outras plataformas (esse agente estaria, por exemplo na Camada de Agentes de Notificação). Cada uma das plataformas que receber esse agente verificaria então se as estações de sua rede estão originando pacotes de ataque, e em caso afirmativo, poderia contatar um agente da Camada de Reação responsável por eliminar esses processos maliciosos. Esse cenário exigiria a especificação de novos agentes para as Camadas 3 e 4 da proposta de BERNARDES [1999], visto que a detecção de ataques distribuídos não foi considerada em seu trabalho.

Os resultados da Avaliação de SDI feita pela DARPA em 1998 [LIPPMANN et al., 2000b] mostraram que os modelos de detecção produzidos pelo MADAM ID tiveram uma das melhores performances (altas taxas de verdadeiros positivos e alarmes falsos dentro dos limites toleráveis [LEE et al., 2000]) entre os sistemas participantes. O artigo aqui estudado porém, não se aprofunda nos algoritmos de criação dos modelos (exceto pela apresentação de um conjunto de experimentos), e sim na integração do MADAM ID ao CIDE.

5.2.5 Minerando Dados de Execução

A pesquisa de MICHAEL & GHOSH [2000], apresentada na seção 4.5, investiga a possibilidade de se usar sistemas de detecção de assinaturas para identificar características de

ações maliciosas nos rastros de execução de programas. Apesar de não ter sido alcançado ainda o objetivo maior, que é detectar alguns tipos de comportamento malicioso em programas, os resultados com detecção de navegadores são promissores.

Ao analisar os resultados apresentados na pesquisa, podemos identificar que alguns dos programas que inicialmente foram difíceis de distinguir dos navegadores, apresentam em alguns sentidos, comportamento semelhante a esses últimos. O `explorer`, por exemplo, tem um comportamento semelhante a um navegador, porque de um modo geral, busca e apresenta recursos baseando-se em um endereço. Semelhanças desse tipo podem ter causado as dificuldades de isolamento desses programas.

Por outro lado, os próprios autores levantam um aspecto interessante dos experimentos apresentados. Os dados de treinamento passaram por “micro ajustes” até que a performance desejada fosse alcançada. As ações tomadas para esse ajuste foram relativamente intuitivas e levaram ao resultado desejado muito rapidamente [MICHAEL & GHOSH, 2000]. Na verdade, o fator intrigante é que não foi necessário fazer diferentes tentativas com diferentes parâmetros e combinações para se atingir o resultado desejado. Algumas variações poderiam ser, por exemplo, testar diferentes parâmetros para o algoritmo de treinamento, como a profundidade das assinaturas usadas na comparação entre dois nós para determinar o score de uma possível fusão; usar combinações mais variadas dos dados de treinamento, ou até mesmo uma estratégia de fusão totalmente diferente.

Fica a questão se os resultados apresentados representam a validade da técnica proposta ou se simplesmente refletem a capacidade de intuição dos autores em definir uma configuração de treinamento. Portanto, os experimentos estão ainda em um estágio inicial, e algumas dessas questões levantadas acima são endereçadas na lista de trabalhos futuros por eles propostos.

Sendo assim, a integração dessa técnica ao sistema proposto por BERNARDES [1999] implicaria em muito trabalho de investigação para validar essa metodologia de detecção de intrusos. Fica aqui então a recomendação de acompanhar os resultados a serem apresentados no futuro por essa abordagem, mas não investir tempo nessa investigação de validade para uma possível integração.

5.2.6 Classificação dos Métodos

A seguir é apresentada uma tabela comparativa, listando cada um dos métodos e a classificação aplicada a eles de acordo com o que foi apresentado na seção 3.2, ou seja, categoriza os SDI de acordo com o método de detecção e tratamento dos dados. A Tabela 5 apresenta o resultado dessa classificação, a tecnologia de reconhecimento empregada por cada um dos sistemas, e indica se a abordagem avaliada apresenta tentativas de solver o problema de detecção de ataques desconhecidos. Note que a tabela apresenta somente as características dos métodos que foram apresentadas nos trabalhos consultados, e não as capacidades que alguns deles possam apresentar de se adaptar a outros modelos, como é o caso do trabalho número 2.

1. Sistema Imunológico Computacional para Detecção de Intrusões [WILLIAMS et al., 2001]	
Método de Detecção	Uso indevido
Tratamento dos Dados	Baseado em Rede
Detecção de Ataques Desconhecidos	Sim
Tecnologia Empregada	Sistema Imunológico Computacional
2. Filtros Baseados em Reconhecimento de Padrões [KURI et al., 2000]	
Método de Detecção	Uso indevido
Tratamento dos Dados	Baseado em Host
Detecção de Ataques Desconhecidos	Não
Tecnologia Empregada	Reconhecimento de Padrão Aproximado Múltiplo
3. Aprendizado de Comportamento de Programas [GHOSH, MICHAEL & SCHATZ 2000]	
Método de Detecção	Anomalia
Tratamento dos Dados	Baseado em Host
Detecção de Ataques Desconhecidos	Sim
Tecnologia Empregada	Rede Neural Recorrente Elman, Transdutor de String, Inferência Automática de Autômatos Finitos
4. Detecção de Intrusões Distribuídas e Desconhecidas [LEE et al., 2000]	
Método de Detecção	Uso indevido
Tratamento dos Dados	Baseado em Rede
Detecção de Ataques Desconhecidos	Sim
Tecnologia Empregada	Mineração de Dados
5. Detecção de Intrusões Distribuídas e Desconhecidas [MICHAEL & GHOSH 2000]	
Método de Detecção	Uso indevido
Tratamento dos Dados	Baseado em Host
Detecção de Ataques Desconhecidos	Sim
Tecnologia Empregada	Máquina de Estados Finitos

Tabela 5: Classificação dos métodos apresentados

Como podemos notar pela Tabela 5, apenas um dos artigos apresentados usa método de detecção por anomalias. Isso pode ser um reflexo do fato citado anteriormente que a maioria dos SDI comerciais se baseiam no método de detecção por uso indevido [BACE & MELL, 2001; GHOSH, MICHAEL & SCHATZ, 2000]. Por outro lado, também pode representar o caminho oposto, ou seja, que as pesquisas acadêmicas têm ditado o caminho a ser tomado

pelos SDI comerciais. Essa última alternativa pode ser reforçada quando olhamos a última coluna da tabela; ela mostra que das quatro abordagens que usam método de detecção por uso indevido, três delas estão tentando resolver o problema da detecção de ataques desconhecidos, um dos maiores problemas desse tipo de SDI.

5.3 Conclusões

A proposta de levantar as metodologias de reconhecimento de padrões atualmente sendo investigadas para uso em detecção de intrusões, acaba por trazer também algumas impressões gerais sobre a direção sendo tomada nessa área de pesquisas.

Como pode ser comprovado pelas tecnologias apresentadas e analisadas neste trabalho, as pesquisas estão procurando soluções para problemas específicos relacionados à detecção de intrusão, e não uma solução geral que tente resolver todos os problemas de uma só vez. A tecnologia de agentes proposta pelo Grupo de Segurança do ICMC se encaixa perfeitamente nessa futura modalidade de Sistemas de Detecção de Intrusão, ou seja, a criação e uso de ferramentas capazes de integrar várias tecnologias com a habilidade de produzirem pequenas soluções, trabalhando em conjunto para a resolução do grande problema.

Este trabalho apresentou alternativas de reconhecimentos de padrões propostas para o uso em detecção de intrusão. Procurou-se selecionar um conjunto de métodos que cobrisse várias alternativas, e esse objetivo foi atingido, porque foram apresentadas soluções usando os seguintes métodos: mineração de dados, algoritmos evolucionários, redes neurais, transdutores de strings, máquinas de estados finitos, inferência automática de autômatos finitos e uma abordagem ao problema de reconhecimento de padrão aproximado múltiplo.

Cada um desses métodos foi apresentado, destacando a sua aplicabilidade e os resultados obtidos com o seu uso na área de detecção de intrusão. Uma análise crítica e qualitativa de cada um dos métodos foi exposta, identificando as vantagens e desvantagens oferecidas por cada um deles tendo em vista o seu uso em um ambiente de agentes móveis e autônomos.

Baseando-se nessas afirmações, esse trabalho apresenta as seguintes contribuições:

- A integração de tecnologias baseadas em reconhecimentos de padrões a um sistema de detecção de intrusão baseado em agentes móveis e autônomos. Foram selecionadas as tecnologias capazes de solucionar pequenos problemas relacionados à detecção de intrusão, de acordo com a exposição feita acima. Essa necessidade de especificar

agentes dotados de mecanismos de inteligência foi destacada no trabalho de BERNARDES [1999], e tem o objetivo de suportar cenários avançados de detecção de intrusão.

- Soluções para a integração ao sistema proposto por Bernardes de propostas inovadoras, como a detecção de intrusões a partir da análise do comportamento de programas.
- Prover os resultados de uma pesquisa que analisa propostas de implementação de reconhecimento de padrões usando diferentes métodos de detecção e de tratamento dos dados.
- Estabelecimento de bases para a discussão de como aperfeiçoar o sistema proposto por Bernardes para suportar ataques distribuídos.

A evolução da área de sistemas de detecção de intrusões é muito rápida. Novos métodos e técnicas surgem com uma velocidade espantosa, tornando difícil a tarefa de manter-se atualizado. Um trabalho como este tem um grande valor nesse sentido, pois executa a coleta e a concentração de informações, transmitindo-as de uma maneira mais resumida e agregada de uma análise crítica.

5.4 Trabalhos Futuros

Como trabalhos futuros para a presente pesquisa, é proposto:

- Fazer uma investigação mais profunda do algoritmo Transdutor de String, porque a informação sobre ele é ainda escassa⁴⁵, porém ele apresentou ótimos resultados no campo de detecção de intrusões. Sua performance foi equivalente à de uma rede neural recorrente, porém com um tempo de treinamento muito menor.
- No capítulo 4, foram explicadas as limitações da proposta que usa Algoritmos Evolucionários. Porém, os resultados dessa abordagem foram promissores, e uma

⁴⁵ Por exemplo, uma pesquisa no site de buscas Google pela palavra “string transducer” retornou apenas 64 resultados (na verdade, depois de removidos os sites relacionados a música, restaram apenas 50), enquanto uma busca por “artificial neural network” retornou cerca de 216000 entradas possíveis

investigação futura poderia ser feita para modelar o uso dessa mesma técnica levando em consideração também o *payload* dos pacotes de dados. Também é preciso determinar se o esforço de implementação desse cenário seria recompensado pelo alcance de detecção oferecido pelo sistema.

- A proposta de agrupar anticorpos em um agente pode trazer bons resultados, desde que os métodos de agrupamento sejam bem definidos e eficientes. Na abordagem estudada, as características dos anticorpos são escolhidas randomicamente. Um estudo do comportamento dos anticorpos poderia revelar que algumas características revelam melhor performance na detecção de determinados grupos de ataques. Esse poderia ser um dos critérios de agrupamento.
- Em seu trabalho, BERNARDES [1999] especifica que nenhum agente deve ser capaz de decidir sozinho se uma intrusão ocorreu ou não. Essa decisão deve ser tomada em consenso. Esse processo é similar à *Costimulation* usada pelos anticorpos na abordagem de WILLIAMS et al. [2001]. Seria interessante investigar a possibilidade de aplicar a *Costimulation* aos agentes.
- Foram propostos dois agentes para suportar a abordagem de GHOSH, MICHAEL & SCHATZ [2000] no sistema proposto por Bernardes. Experimentos precisam ser feitos no sentido de determinar a necessidade da existência de um agente para cada programa sendo monitorado, ou se um único agente seria capaz de monitorar vários programas sem sofrer degradação de performance

6

6 REFERÊNCIAS

- [ALESSANDRI, 2000] ALESSANDRI, D. Using rule-based activity descriptions to evaluate intrusion-detection systems. In: RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 183-196.
- [ALLEN et al., 2000] ALLEN, J.; CHRISTIE, A.; FITTHEN, W.; McHUGH, J.; PICKEL, J.; STONER, E. **State of the practice of intrusion detection technologies**. 2000. (Technical Report CMU/SEI-99-TR-028, ESC-99-028).
- [BACE & MELL, 2001] BACE, R.; MELL, P. **Intrusion detection systems – NIST special publication on intrusion detection systems**. Disponível em: <<http://csrc.nist.gov/publications/nistpubs/800-31/sp800-31.pdf>>. Acesso em: 29/01/2002.
- [BERNARDES, 1999] BERNARDES, M.C. **Avaliação do uso de agentes móveis em segurança computacional**. São Carlos, 1999. 105 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [BONIFÁCIO Jr. et al., 1998a] BONIFÁCIO Jr., J.M.; CANSIAN, A.M.; CARVALHO, A.C.P.L.; MOREIRA, E.S. Neural networks applied in intrusion detection systems. In: IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS - IJCNN'98. **Proceedings...** Anchorage, Alaska: IEEE Computer Society, 1998. p. 205-210.
- [BONIFÁCIO Jr. et al., 1998b] BONIFÁCIO Jr., J.M.; CANSIAN, A.M.; CARVALHO, A.C.P.L.; MOREIRA, E.S. Um ambiente de segurança distribuído para a integração de firewalls com sistemas de detecção de intrusão. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES – SBRC'98, 16., Rio de Janeiro. **Anais...** Rio de Janeiro: UFF, 1998. p. 561-576.
- [BONIFÁCIO Jr., 1998] BONIFÁCIO Jr., J.M. **Sistemas de segurança distribuído: integração de firewalls com sistemas de detecção de intrusão**. São Carlos, 1998. 79 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [BRAGA, CARVALHO & LUDERMIR, 2000] BRAGA, A.P.; CARVALHO, A.C.P.L.; LUDERMIR, T.B. **Redes neurais artificiais: teoria e aplicações**. Rio de Janeiro: LTC, 2000. 262 p.
- [CANSIAN et al., 1997a] CANSIAN, A.M.; MOREIRA, E.S.; CARVALHO, A.C.P.L.; BONIFÁCIO Jr., J.M. Network intrusion detection using neural networks. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL

- INTELLIGENCE AND MULTIMEDIA APPLICATIONS - ICCIMA'97. **Proceedings...** Gold Coast, Australia, 1997. p. 276-280.
- [CANSIAN et al., 1997b] CANSIAN, A.M.; MOREIRA, E.S.; MOURO, R.B.; MORISHITA, F.T.; CARVALHO, A.C.P.L. An adaptative system for detecting intrusion in networks. In: INTERNATIONAL CONGRESS ON INFORMATION ENGINEERING, 3., 1997. **Proceedings...** Buenos Aires, Argentina, 1997. p. 96-105.
- [CANSIAN et al., 1997c] CANSIAN, A.M.; MOREIRA, E.S.; CARVALHO, A.C.P.L.; BONIFÁCIO Jr., J.M. Um modelo adaptativo para detecção de comportamento suspeito em redes de computadores. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES - SBRC'97, 15., São Carlos. **Anais...** São Carlos: UFSCar ; USP, 1997. p. 51-60.
- [CANSIAN, 1997] CANSIAN, A.M. **Desenvolvimento de um sistema adaptativo de detecção de intrusos em redes de computadores.** São Carlos, 1997. 153 p. (Tese de Doutorado). Instituto de Física de São Carlos, Universidade de São Paulo.
- [CICILINI, 1994] CICILINI, R. **Desenvolvimento de um agente SNMP para plataformas rodando DOS.** São Carlos, 1994. 107 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [COMPUTER, 2001] COMPUTER Institute Security Press Release. Disponível em: <http://www.gocsi.com/prelea/000321.html> Acesso em: 29/01/2002.
- [CROSBIE & SPAFFORD, 1995a] CROSBIE, M.; SPAFFORD, E.H. **Defending a computer system using autonomous agents.** Disponível em: <http://www.cs.purdue.edu/homes/spaf/tech-reps/9522.ps> Acesso em: 15/02/2000.
- [CROSBIE & SPAFFORD, 1995b] CROSBIE, M.; SPAFFORD, E.H. **Active defense of a computer system using autonomous agents.** Disponível em: <http://www.cs.purdue.edu/homes/spaf/tech-reps/9508.ps> Acesso em: 15/02/2000.
- [CUNNINGHAM & STEVENSON, 2001] CUNNINGHAM, R.K.; STEVENSON, C.S. Accurately detecting source code of attacks that increase privilege. In: RECENT ADVANCES IN INTRUSION DETECTION – INTERNATIONAL SYMPOSIUM - RAID 2001, 4., Davis, CA, USA. **Proceedings...** Springer-Verlag, 2001. p. 104-116.
- [FLACK & ATALLAH, 2000] FLACK, C.; ATALLAH, M.J. Better Logging through Formality. Applying formal specification techniques to improve audit logs and log consumers. In: Recent Advances in Intrusion Detection – International Workshop, RAID 2000, 3., Toulouse, França, Outubro 2000. **Proceedings...** Springer-Verlag, 2000. p. 1-16.
- [FONTES, 1999] FONTES, E. O inimigo está ao lado. **Network Computing**, v.1, n.8, p. 14, Out. 1999.
- [FREUND et al., 1997] FREUND, Y.; KEARNS, M.; RON, D.; RUBINFELD, R.; SCHAPIRE, E.; SELLIE, L. Efficient learning of typical finite automata from random walks. **Information and Computation**, v.138, n.1, p. 23-48, Out. 1997.
- [GARFINKEL & SPAFFORD, 1996] GARFINKEL, S.; SPAFFORD, G. **Practical UNIX and internet security.** 2.ed. O'Reilly & Associates, Inc., 1996. 971 p.
- [GHOSH & MICHAEL, 2000] GHOSH, A.K.; MICHAEL, C. Two state-based approaches to program-based anomaly detection. In: ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE - ACSAC 2000, 16. **Proceedings...** Disponível em: <http://www.acsac.org/2000/papers/96.pdf> Acesso em: 19/02/2002.
- [GHOSH, MICHAEL & SCHATZ, 2000] GHOSH, A.K.; MICHAEL, C.; SCHATZ, M. A real-time intrusion detection system based on learning program behavior. In: RECENT ADVANCES IN INTRUSION DETECTION – INTERNATIONAL WORKSHOP - RAID

- 2000, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 93-109.
- [GOLLMANN, 1999] GOLLMANN, D. **Computer security**. John Wiley & Sons, Inc., 1999. 320 p.
- [GRANDE, 1999] GRANDE Dicionário Larousse Cultural da Língua Portuguesa. Editora Nova Cultural Ltda., 1999. 928 p.
- [ILGUN, 1993] ILGUN, K. USTAT: a real-time intrusion detection system for UNIX. In: **COMPUTER SOCIETY SYMPOSIUM ON RESEARCH IN SECURITY AND PRIVACY**, 1993, Oakland, California. **Proceedings...** Los Alamitos, CA: IEEE Computer Society Press, 1993. p. 16-28.
- [KO et al., 2000] KO, C.; BRUTCH, P.; ROWE, J.; TSAFNAT, G.; LEVITT, K. System health and intrusion monitoring using a hierarchy of constraints. In: **RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000**, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 190-203.
- [KUMAR & SPAFFORD, 1994] KUMAR, S.; SPAFFORD, E. **An application of pattern matching in intrusion detection**. 1994. (Technical Report CSD-TR-94-013).
- [KUMAR & SPAFFORD, 1995] KUMAR, S.; SPAFFORD, E. **A software architecture to support misuse intrusion detection**. 1995. (Technical Report CSD-TR-95-009 - The COAST Project, Department of Computer Sciences, Purdue University).
- [KUMAR, 1995] KUMAR, S. **Classification and detection of computer intrusion**. 1995. (Tese de Doutorado). Universidade de Purdue.
- [KURI & NAVARRO, 2000] KURI, J.; NAVARRO, G. **Fast multipattern search algorithms for intrusion detection**. 2000. (Technical Report TR/DCC-99-11). Disponível em: <<http://citeseer.nj.nec.com/455922.html>> Acesso em: 29/01/2002.
- [KURI et al., 2000] KURI, J.; NAVARRO, G.; MÉ, L.; HEYE L. A pattern matching based filter for audit reduction and fast detection of potential intrusions. In: **RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000**, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 17-27.
- [LEE & STOLFO, 1997] LEE, W.; STOLFO, S. **Data mining approaches for intrusion detection**. In: **USENIX SECURITY SYMPOSIUM**, 7., 1998. Disponível em: <<http://www.cs.columbia.edu/~sal/hpapers/USENIX/usenix.html>> Acesso em: 13/12/1999.
- [LEE et al., 2000] LEE, W.; NIMBALKAR, R.A.; YEE, K.K.; PATIL, S.B.; DESAI, P.H.; TRAN, T.T.; STOLFO, S.J. A data mining and CDF based approach for detecting novel and distributed intrusions. In: **RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000**, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 49-65.
- [LIEIRA, 1995] LIEIRA, J.F. **Utilização de áudio e vídeo em sistemas gerenciadores de redes de computadores**. São Carlos, 1995. 111 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [LIPPMANN et al., 2000a] LIPPMANN, R.; HAINES, J.W.; FRIED, D.J.; KORBA, J.J.; DAS, K. Analysis and results of the 1999 DARPA off-line intrusion detection evaluation. In: **RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000**, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 162-182.
- [LIPPMANN et al., 2000b] LIPPMANN, R.; FRIED, D.J.; GRAF, I.; HAINES, J.W.; KENDALL, K.R.; MCCLUNG, D.; WEBER, D.; WEBSTER, S.E.; WYSCHOGROD, D.; CUNNINGHAM, R.K.; ZISSMAN, M.A. Evaluating intrusion detection systems: the 1998 DARPA off-line intrusion detection evaluation. In: **INFORMATION SURVIVABILITY CONFERENCE AND EXPOSITION**, Los Alamitos, EUA. **Proceedings...** IEEE Computer Society, 2000. p. 12-26.

- [LOPES, 1999] LOPES, F. Os super-heróis da segurança. **Network Computing**, ano 1, n. 6, p. 32, Ago. 1999.
- [LOPES, 2000] LOPES, F. A fortaleza dos negócios. **Network Computing**, v.1, n.12, p. 30-36, Fev. 2000.
- [MAXIMUM, 1998] MAXIMUM security – a hacker’s guide to protecting your internet site and network. 2.ed. SAMS Publishing, 1998. 829 p.
- [MÉ, 1998] MÉ, L. GASSATA, a genetic algorithm as an alternative tool for security audit trails analysis. In: RECENT ADVANCES IN INTRUSION DETECTION – INTERNATIONAL WORKSHOP - RAID 1998, 1., Louvain-la-Neuve, Belgium. **Proceedings...** Disponível em: <http://www.zurich.ibm.com/pub/Other/RAID/Prog_RAID98/Table_of_content.html> Acesso em: 20/02/2002.
- [MICHAEL & GHOSH, 2000] MICHAEL, C.; GHOSH, A. Using finite automata to mine execution data for intrusion detection: a preliminary report. In: RECENT ADVANCES IN INTRUSION DETECTION – INTERNATIONAL WORKSHOP - RAID 2000, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 66-79.
- [MÓDULO, 2001] MÓDULO Security Solutions S.A.. Sétima pesquisa nacional sobre segurança da informação. Disponível em: <http://www.modulo.com.br/empresa/noticias/releases/r_pesq2001.htm>. Acesso em: 29/01/2002.
- [MORAES, 1995] MORAES, S. **Voz em sistemas computacionais: projeto e implementação de módulos de processamento de voz em gerenciamento de redes**. São Carlos, 1995. 103 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [MORISHITA, 1997] MORISHITA, F.T. **Uma avaliação evolutiva dos protocolos de gerenciamento da Internet: SNMPv1, SNMPv2 e SNMPv3**. São Carlos, 1997. 68 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [MOURO, 1997] MOURO, R.B. **Uma arquitetura operacional extensível para ferramentas de gerenciamento de redes**. São Carlos, 1997. 60 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [MOURO, MORISHITA & MOREIRA 1997] MOURO, R.B.; MORISHITA, F.T.; MOREIRA, E.S. NetTracker: uma arquitetura operacional extensível para ferramentas de gerenciamento de redes. In: SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES - SBRC'97, 15., São Carlos. **Anais...** São Carlos: UFSCar ; USP, 1997. p. 64-174.
- [ODA, 1994] ODA, C.S. **Desenvolvimento de um sistema monitor gráfico baseado em protocolo de gerenciamento SNMP**. São Carlos, 1994. 111 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [PEREIRA FILHO, 2001] PEREIRA FILHO, S.F. **Avaliação para ambientes servidores para agentes móveis**. São Carlos, 2001. 100 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [REAMI, 1998] REAMI, E.R. **Especificação e prototipagem de um ambiente de gerenciamento de segurança apoiado por agentes móveis**. São Carlos, 1998. 82 p. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [RIORDAN & ALESSANDRI, 2000] RIORDAN, J.; ALESSANDRI, D. Target naming and service apoptosis. In: Recent Advances in Intrusion Detection – International Workshop, RAID 2000, 3., Toulouse, França, Outubro 2000. **Proceedings...** Springer-Verlag, 2000. p. 215–225.
- [SOBIREY, 2000] SOBIREY, M. **Michael Sobirey's intrusion detection systems page**.

- Disponível em: <<http://www-rnks.informatik.tu-cottbus.de/%7esobirey/ids.html>> Acesso em: 12/04/2000.
- [SPAFFORD et al., 1998] SPAFFORD, E.; BALASUBRAMANIAN, J.S.; FERNANDEZ, J.O.G.; ISACOFF, D.; ZAMBONI, D. **An architecture for intrusion detection using autonomous agents**. 1998. (COAST Technical Report 98/05).
- [STEIN, 1998] STEIN, L.D. **Web security: a step-by-step reference guide**. Addison Wesley & Longman, Inc., 1998. 436 p.
- [TAVARES, 2002] TAVARES, D.M. **Avaliação de técnicas de captura para sistemas detectores de intrusão**. São Carlos, 2002. (Dissertação de Mestrado). Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- [UPPULURI & SEKAR, 2000] UPPULURI, P.; SEKAR, R. Experiences with specification-based intrusion detection. In: RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 172-189.
- [VALDES & SKINNER, 2000] VALDES, A.; SKINNER, K. Adaptive, model-based monitoring for cyber attack detection. In: RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL WORKSHOP - RAID 2000, 3., Toulouse, França. **Proceedings...** Springer-Verlag, 2000. p. 80-92.
- [VALDES & SKINNER, 2001] VALDES, A.; SKINNER, K. Probabilistic alert correlation. In: Recent Advances in Intrusion Detection - International Symposium, RAID 2001, 4., Davis, CA, USA 2001. **Proceedings...** Springer-Verlag, 2001. p. 54-68.
- [VIGNA & KEMMERER, 1999] VIGNA, G.; KEMMERER, R.A. NetSTAT: A network-based intrusion detection system. **Journal of Computer Security**, v.7, n.1, p. 37-71, 1999.
- [WARRENDER, FORREST & PEARLMUTTER, 1999] WARRENDER, C.; FORREST, S.; PEARLMUTTER, B. Detecting intrusions using system calls: alternative data models. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1999. **Proceedings...** IEEE Computer Society Press, 1999. p. 133-145.
- [WILLIAMS et al., 2001] WILLIAMS, P.D.; ANCHOR, K.P.; BEBO, J.L.; GUNSCH, G.G.; LAMONT, G.D. CDIS: towards a computer immune system for detecting network intrusions. In: RECENT ADVANCES IN INTRUSION DETECTION - INTERNATIONAL SYMPOSIUM - RAID 2001, 4., Davis, CA, USA. **Proceedings...** Springer-Verlag, 2001. p. 117-133.

APÊNDICE A: LINKS

São apresentados aqui alguns links de sites para os interessados em segurança da informação com uma breve descrição do conteúdo que será encontrado nas páginas de cada um deles, assim como a data da última visita. A lista é apresentada em ordem alfabética.

http://icat.nist.gov/icat.cfm	Índice de informações sobre vulnerabilidades computacionais. Fornece busca a partir de vários parâmetros e fornece links para informações sobre vulnerabilidades e <i>patches</i> . Acesso em 06/02/2002, apresenta 3526 vulnerabilidades.
http://lists.insecure.org/	Fornece um arquivo atualizado em tempo real de alguns grupos de discussão (<i>nesgroups</i>) sobre segurança. Acesso em 06/02/2002.
http://www.cerias.purdue.edu/coast/http://www.cerias.purdue.edu/coast/intrusion-detection/	O projeto COAST (<i>Computer Operations, Audit, and Security Technology</i>) é sediado nos laboratórios da <i>Purdue University</i> e o site fornece links e um extensivo arquivo de textos, artigos e ferramentas relacionadas à segurança computacional. Acesso em 06/02/2002.
http://www.cert.org/	O site do CERT (<i>Computer Emergency Response Team</i>) fornece notícias, alertas de segurança, vulnerabilidades com os correspondentes <i>patches</i> e links relacionados à segurança da informação. Acesso em 26/02/2002.
http://www.gocsi.com/	Organização especializada em servir e treinar profissionais de segurança da informação. Fornece seminários, conferências e material para consulta, como pesquisas e links. Acesso em 06/02/2002.
http://www.modulo.com.br	Empresa brasileira que presta consultoria na área de segurança computacional. Acesso em 06/02/2002
http://www.nessus.org/	“O projeto Nessus visa prover a comunidade da Internet com um <i>port-scanner</i> grátis, poderoso e atualizado”. Acesso em 06/02/2002.
http://www.packetstormsecurity.org	Esse site tem constantes problemas com a lei, porque apresenta um extensivo conjunto de ferramentas que podem ser usadas pra fins maliciosos. Em vista disso, já foi obrigado a mudar o seu endereço, porém o que está indicado ao lado é o ativo no dia 06/02/2002.
http://www.sans.org/	O Instituto SANS (<i>System Administration, Networking, and Security</i>) é uma organização cooperativa com 156000 membros. Fornece notícias, pesquisas, artigos e alertas de segurança. Acesso em 06/02/2002.
http://www.securenet.com.br	Site em português com informações relacionadas à segurança computacional. Apresenta também uma lista de discussões sobre o assunto, chamada BOS-BR (<i>Best of Security – Brasil</i>). Acesso em 06/02/2002. Infelizmente parece estar descontinuando as atividades, pois a última notícia data de novembro de 2001.
http://www.securityfocus.com	Empresa que fornece produtos relacionados à segurança computacional, alertas de segurança, guarda uma das mais completas listas de discussão sobre vulnerabilidades (BugTraq) e uma base de dados de vulnerabilidades. Acesso em 06/02/2002.

GLOSSÁRIO

<u>Assíntota</u>	[Do gr. <i>asymptotas</i> , 'que não pode coincidir'; var. de <i>assíptota</i> .] S. f. Geom. Anal. 1. Reta que é tangente a uma curva no infinito; reta limite da família de tangentes a uma curva quando o ponto de tangência tende para o infinito (Dicionário Aurélio On-line; http://www.uol.com.br/aurelio/ ; acesso em 31/01/2002).
<u>Backdoor</u>	É conhecida como uma porta de entrada para um sistema que não deveria existir. Um <i>trojan</i> , por exemplo, quando executado em uma máquina, pode habilitar uma porta para conexões, sem que o usuário saiba.
<u>Backpropagation</u>	Algoritmo para aprendizado supervisionado de Redes Neurais Artificiais. É uma generalização da regra delta para redes de múltiplas camadas.
<u>Buffer Overrun ou Buffer Overflow</u>	Um tipo de ataque onde um usuário malicioso explora um <i>buffer</i> não checado de um programa, e sobrescreve o código do programa com seus próprios dados. Se o código do programa for sobrescrito com código executável, o efeito é mudar a operação do programa da maneira ditada pelo atacante. Se escrito com outros dados, a consequência provável é que o programa pare de executar (<i>crash</i>).
<u>Cross Validation</u>	Um método para avaliar um modelo estatístico ou algoritmo que tenha parâmetros livres. Os dados de treinamento são divididos em várias partes, e uma das partes é usada para testar se o procedimento serviu para as partes restantes. Pode ser usado para seleção de modelos ou para estimativa de parâmetros quando existem muitos parâmetros.
<u>Denial of Service (DoS)</u>	Um ataque de <i>Denial of Service</i> é um feito contra um servidor fazendo com que a um cliente seja negado o nível de serviço esperado. Em um caso leve, o impacto pode ser uma inesperada perda de performance, e no pior caso o sistema pode se tornar tão carregado que causará a sua indisponibilidade. Geralmente esses ataques não envolvem roubo ou perda de dados como motivo principal. Existem algumas variações deste tipo de ataque: <i>SYN flood</i> , <i>teardrop</i> , <i>Smurf</i> e <i>Ping of Death</i> .
<u>Distributed Denial of Service (DdoS)</u>	Versão distribuída do ataque <i>Denial of Service</i> . Neste tipo de ataque, o atacante instala algumas ferramentas especiais em muitos sites na Internet e, de sua máquina, dispara o ataque simultaneamente a todas elas. A vítima irá receber, de repente, uma quantidade monstruosa de conexões originárias dos mais diversos lugares. Isso torna difícil distinguir o ataque de um pico de acessos válidos ao site.
<u>Exploit</u>	É uma técnica ou programa que explora uma falha conhecida de um programa ou sistema operacional.
<u>Hacker</u>	(pronome) 1. Pessoa que se diverte aprendendo os detalhes de sistemas computacionais e como estender suas capacidades, em oposição à maioria dos usuários de computadores que preferem aprender somente o mínimo necessário. 2. Alguém que programa entusiasmadamente ou que goste de programar ao invés de apenas teorizar sobre a programação. ⁴⁶
<u>n-gram</u>	n-grams são usadas para descrever objetos como vetores. Isso torna

⁴⁶ Esta definição está em GARFINKEL & SPAFFORD [1996] como sendo de: Guy L. Steele, *et al.*, *The Hacker's Dictionary* e foi traduzida para a Língua Portuguesa pelo autor deste texto.

possível a aplicação de técnicas geométricas e estatísticas que são bem definidas para vetores mas não para objetos em geral. Por exemplo, um dos usos mais comuns é definir a medida de similaridade entre documentos texto baseados na aplicação de uma função matemática ao vetor que representa os documentos.

Patch

É uma espécie de “conserto” para os programas que têm alguma falha que pode ser explorada tornando-se um *exploit* ou criando um *backdoor*.

Perceptron

Modelo de Rede Neural Artificial cujo algoritmo de treinamento sempre encontra uma solução em um número finito de iterações caso as classes em questão sejam linearmente separáveis [BRAGA, CARVALHO & LUDERMIR, 2000].

Plaintext

Esse termo é muitas vezes usado genericamente como indicando um dado que não passou por um processo criptográfico. Sendo assim, não representa somente um texto simples como o nome sugere, mas todo e qualquer tipo de mensagem que possa ser criptografada.

Port-Scan

Um port-scan é um método de varredura que consiste de uma série de mensagens enviadas a um computador para identificar quais serviços de rede (cada um associado a um número de porta) o mesmo fornece. Esse método dá a um atacante uma idéia das fraquezas do sistema. Essencialmente, um port-scan é o envio de mensagens a cada porta, uma por vez. O tipo de resposta obtida indica se a porta está sendo usada, e portanto, pode ser sondada por fraquezas.

Tipo de port-scan⁴⁷:

- *Vanilla* – Uma tentativa de conectar a cada uma das 65536 portas disponíveis.
- *Strobe* – Uma tentativa de conectar a algumas portas selecionadas (tipicamente abaixo de 20).
- *Stealth scan* – Várias técnicas que previnem que a requisição de conexão seja logada (ver Stealth, IP Stealth).
- *FTP Bounce Scan* – As tentativas são direcionadas através de um servidor FTP (File Transfer Protocol) para disfarçar a localização do atacante.
- *Fragmented Packets* – A varredura é feita enviando pacotes fragmentados que podem passar através dos filtros em um *firewall*.
- *UDP* – Procura por portas UDP (User Datagram Protocol) abertas.
- *Sweep* – Varre a mesma porta em vários computadores.

Race Condition

Uma condição causada pela temporização de eventos em um pedaço de *software*. *Race Conditions* são tipicamente associadas a erros de sincronização que fornecem uma janela de oportunidade durante o qual um processo pode interferir com outro, possivelmente introduzindo uma vulnerabilidade de segurança. Por exemplo, um programa que altera um arquivo importante usa uma cópia temporária durante essa alteração. Se as permissões do arquivo temporário permitirem que ele seja editado, pode ser possível alterá-lo antes que o programa termine seu processo de edição.

Remote-to-local

Nestes ataques, um atacante que não tem uma conta na máquina vítima consegue acesso local a mesma (por exemplo, *guest*, *dict*) e altera arquivos, modifica dados em trânsito para a máquina atacada ou instala *trojans* [LIPPMANN et al., 2000a].

⁴⁷ Esses tipos são reportados pela AT&T no endereço: <http://help.broadband.att.com/legal/glossary.jsp>. Citando em 12/02/2002.

Script-kid

Termo usado para descrever atacantes que usam *scripts* e programas prontos encontrados na rede para efetuar invasões. Geralmente são garotos adolescentes que não têm conhecimento suficiente para criar suas próprias ferramentas.

SDI Model-Based

Um dos tipos de SDI baseados em regras. Nessa abordagem as tentativas de intrusão são modeladas como seqüências de comportamento do usuário. Esses comportamentos podem então ser modelados como, por exemplo, eventos dos arquivos de auditoria. É importante perceber que, o sistema de detecção de intrusão nesse caso é responsável por determinar como o comportamento de um usuário identificado pode se manifestar na trilha de auditoria. Essa abordagem apresenta algumas vantagens:

- Mais dados podem ser processados, porque a tecnologia permite reduzir o foco de dados seletivamente.
- É possível ter explicações mais intuitivas das tentativas de intrusão.
- O sistema pode prever a próxima ação do usuário.

[ILGUN, 1993] (veja também SDI State-Based).

SDI State-Based

Um dos tipos de SDI baseados em regras. Nessa abordagem a regra é codificada usando uma terminologia usada em arquivos de auditoria. Tentativas de intrusão são definidas como seqüências de estados do sistema – como definidas nos arquivos de auditoria – levando de um estado inicial de acesso limitado a, por exemplo, um estado final comprometido [ILGUN, 1993] (veja também SDI Model-Based).

Sniffing

É a técnica de captura de pacotes de comunicação diretamente na rede. Essa técnica exige que o programa que faz a captura e a placa de rede executem em “modo promíscuo”, ou seja, todos os pacotes que passarem pelo cabo onde a placa está ligada serão pegos pelo programa.

Stealth, IP Stealth

O Stealth de IP consiste em qualquer técnica empregada para esconder o IP de uma máquina quando essa se comunica através da rede. Alguns programas que fazem *port-scan*, por exemplo, escondem o seu IP dos logs da máquina destino iniciando conexões com a flag FIN ligada.

SYN Flood

Um ataque *SYN Flood* é um do tipo *denial of service* que tira proveito do funcionamento de uma conexão TCP/IP. Em um ataque deste tipo, o atacante envia ao servidor um pacote SYN (pedido de sincronização do cliente para o servidor) com um endereço IP falso (*spoofed*). O servidor então retorna um SYN-ACK e espera pelo pacote final SYN-ACK-ACK, que nunca virá porque o endereço IP não existe. O servidor então espera repetidamente, depois retransmite o pacote SYN-ACK para o falso endereço quando nenhuma resposta é obtida.

Teardrop

O ataque *teardrop* é uma variação de ataques de *Denial of Service*. Explorando uma característica do protocolo TCP/IP denominada fragmentação, segundo a qual, pacotes grandes de dados são divididos em blocos onde cada fragmento é identificado ao próximo por um marcador de *offset*. Posteriormente, os fragmentos devem ser remontados pelo sistema receptor. No ataque *teardrop*, o atacante entra um valor de *offset* confuso no segundo (ou último) fragmento, o que pode causar indisponibilidade (*crash*) do sistema visado.

Transdutor

[Do ingl. *transductor*.] S. m. Fis. 1. Qualquer dispositivo capaz de transformar um tipo de sinal em outro tipo, com o objetivo de transformar uma forma de energia em outra, possibilitar o controle de um processo ou fenômeno, realizar uma medição, etc. (retirado do Dicionário Aurélio Online. Citado em 31/01/2002. <http://www.uol.com.br/aurelio/>)

User-to-root

Também identificado por ataque U2R. Em ataques deste tipo, os intrusos tentam ganhar privilégios de super usuário na máquina atacada, tanto por

uso indevido de programas ou pela execução de programas maliciosos.

VPN

(Virtual Private Network) É uma Rede Privada Virtual que usa criptografia para criar um túnel de comunicação seguro entre duas redes e vem sendo largamente utilizada para executar comunicações seguras em ambientes não-seguros como, por exemplo, a Internet.