HERMANO PEREIRA

SISTEMA DE DETECÇÃO DE INTRUSÃO PARA SERVIÇOS WEB BASEADO EM ANOMALIAS

Dissertação submetida ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Informática.

Curitiba PR
Fevereiro de 2011

HERMANO PEREIRA

SISTEMA DE DETECÇÃO DE INTRUSÃO PARA SERVIÇOS WEB BASEADO EM ANOMALIAS

Dissertação submetida ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para a obtenção do título de Mestre em Informática.

Área de concentração: *Ciência da Computação*

Orientador: Prof. Dr. Edgard Jamhour

Curitiba PR
Fevereiro de 2011

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central

P436s 2011

Pereira, Hermano
Sistema de detecção de intrusão para serviços Web baseado em anomalias / Hermano Pereira ; orientador, Edgar Jamhour. – 2011.
xxiv, 79 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2011
Bibliografia: f. 71-79

1. Servidores da Web. 2. Redes de computação - Medidas de segurança. 3. Redes de computação - Protocolos. 4. Informática. I. Jamhour, Edgar. II. Pontifícia Católica do Paraná. Programa de Pós-Graduação em Informática. III. Título.

CDD 20. ed. – 004

Pontifícia Universidade Católica do Paraná
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Informática

ATA DE DEFESA DE DISSERTAÇÃO DE MESTRADO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

DEFESA DE DISSERTAÇÃO Nº 05/2011

Aos 28 dias do mês de Fevereiro de 2011 realizou-se a sessão pública de Defesa da Dissertação **"Sistema de Detecção de Intrusão para Serviços Web Baseados em Anomalias."** apresentada pelo aluno **Hermano Pereira** como requisito parcial para a obtenção do título de Mestre em Informática, perante uma Banca Examinadora composta pelos seguintes membros:

Prof. Dr. Edgard Jamhour PUCPR (Orientador)	Edgard Jamhour *(assinatura)*	APROVADO *(aprov/reprov.)*
Prof. Dr. Manoel Camillo de Oliveira Penna Neto PUCPR		APROVADO
Prof. Dr. Cassio Ditzel Kropiwiec UTP		APROVADO

Conforme as normas regimentais do PPGIa e da PUCPR, o trabalho apresentado foi considerado APROVADO (*aprovado/reprovado*), segundo avaliação da maioria dos membros desta Banca Examinadora. Este resultado está condicionado ao cumprimento integral das solicitações da Banca Examinadora registradas no Livro de Defesas do programa.

Prof. Dr. Mauro Sérgio Pereira Fonseca
Diretor do Programa de Pós-Graduação em Informática

Programa de Pós-Graduação em Informática PPGIa PUCPR

Rua Imaculada Conceição, 1155 Prado Velho CEP 80215 901 Curitiba Paraná Brasil
Tel.: (41) 3271 1669 Fax: (41) 3271 2121 www.ppgia.pucpr.br

Dedico aos meus pais,
Ivônio de Castro Pereira
e Zuelita Schindler.

Resumo

Este trabalho apresenta uma proposta de arquitetura para um sistema de detecção de intrusão baseado em anomalias que utiliza algoritmos de agrupamento para a detecção de ataques em servidores *web*. Os algoritmos de agrupamento são utilizados para criar modelos de detecção de intrusão com base em treinamentos que são realizados sobre parte das informações que, por exemplo, passam por uma rede de computadores. Esse método de treinamento é conhecido como aprendizado não-supervisionado, pois o treinamento pode ser efetuado sem o conhecimento prévio de registro de ataques dentre as informações coletadas. Nos trabalhos relacionados, os grupos resultantes desses treinamentos são rotulados como normais ou, no caso de anomalias, como ataques. Porém, quando esses grupos são utilizados como modelos de detecção de intrusão, os grupos menores ou isolados (*outliers*) que contém informações legítimas poderão ser detectados como ataques e aumentar o índice de falsos positivos. Com o intuito de reduzir esse índice, apresenta-se neste trabalho um método heurístico para atribuir rótulos de reputação aos grupos de acordo com a quantidade e a origem das informações. Diferentemente de rotular como normal ou ataque, os rótulos dos grupos podem variar de péssima a excelente dentro de uma escala empírica. Portanto, para realizar os experimentos foram coletadas as requisições de servidores *web* onde alguns campos do protocolo HTTP (*HyperText Transfer Protocol*) foram selecionados para o treinamento e atribuição de rótulos de reputação. Assim, a cada requisição inspecionada no momento da detecção de intrusão, os rótulos atribuídos aos campos foram combinados para determinar se tratava ou não de ataque. Para testar a eficiência da arquitetura proposta, a mesma foi implementada e seus resultados foram comparados com os resultados do sistema de detecção de intrusão conhecido como Snort, utilizando o mesmo conjunto de requisições. Ao final, o modelo de detecção proposto obteve melhores índices quando o treinamento foi realizado sobre uma quantidade mínima de 15% do total de requisições.

Palavras-chave: sistemas de detecção de intrusão, agrupamento de dados, ataques web.

Abstract

This paper presents a proposal of architecture for an anomaly based intrusion detection system where clustering algorithms are implemented to detect webservers attacks. The clustering algorithms are used to create intrusion detection models when training are performed over partial information (e.g. transmission over a network). This training method is known as unsupervised learning, because the training can be performed without prior knownledge of the occurrence of attempts attacks on collected information. In related work, the resulting clusters from these trainings are labeled as normal or attack (in anomalous case). However, when these clusters are used as intrusion detection models, the smaller or isolated (outliers) groups containing legitimate information may be detected as attacks and increase the false positives rate. In order to reduce this rate, this paper presents a heuristic method to assign reputation labels to clusters according to the amount and source of information. Instead of labeling as normal or attack, the labels can vary from bad to excellent within an empirical scale. Therefore, to perform the experiments were collected webserver requests where some fields of the HTTP protocol (HyperText Transfer Protocol) were selected for training and assignment of reputation labels. Thus, for each request inspected during an intrusion detection, the labels assigned to the fields were combined to determine whether it was an attack or not. To test the effectiveness of the proposed architecture, it was implemented and his result were compared to the results of the intrusion detection system called Snort, using the same set of requests. Finally, the proposed model achieved better detection rates when the training was performed on a minimum of 15% of all requests.

Keywords: intrusion detection systems, data clustering, web attacks.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviações

A

AAFID	*Autonomous Agents for Intrusion Detection*
AHC	*Agglomerative Hierarchical Clustering*
AODV	*Ad hoc On-Demand Distance Vector*
ASIM	*Automated Security Measurement System*

B

BSM	*Basic Security Module*

C

CDIS	*Computer Defense Immune System*
CDX	*Cyber Defense Exercise*
CERT	*Computer Emergency Response Team*
CLAD	*Clustering for Anomaly Detection*
CMDS	*Computer Misuse Detection System*
CMS	*Content Management System*
COD	*Common Outlier Detection*
CPAN	*Comprehensive Perl Archive Network*
CR	*Carriage Return*
CTF	*Capture The Flag*

D

DARPA	*Defense Advanced Research Projects Agency*
DBMS	*Database Management System*
DDoS	*Distributed DoS*
DFA	*Deterministic Finite Automata*
DIDS	*Distributed IDS*
DNS	*Domain Name System*
DoS	*Denial of Service*

E

EM	*Expectation-Maximization*
EMERALD	*Event Monitoring Enabling Responses to Anomalous Live Disturbances*

F

FIRE	*Fuzzy Intrusion Recognition Engine*
fn	*falsos negativos*
fp	*falsos positivos*
fpMAFIA	*frequent-pattern in* pMAFIA

G

GMT	*Greenwich Mean Time*
GrIDS	*Graph-based Intrusion Detection System*

H

HIDS	*Host-based Intrusion Detection System*
HMM	*Hidden Markov Models*
HP	*Hewlett-Packard*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HTTP Secure*

I

IDA	*Intrusion Detection Agent*
IDES	*Intrusion Detection Expert System*
IDIOT	*Intrusion Detection In Our Time*
IDMEF	*Intrusion Detection Message Exchange Format*
IDP	*Intrusion Detection and Prevention*
IDS	*Intrusion Detection System*
IDWG	*Intrusion Detection Work Group*
INBOUNDS	*Integrated Network-Based Ohio University Network Detective Service*
IP	*Internet Protocol*
IPS	*Intrusion Prevention System*
ISS	*Internet Security System*
ITOC	*Information Technology Operations Center*

J

K

KDD	*Knowledge Discovery and Data Mining*

L

LAMBDA	*A Language to Model a Database for Detection of Attacks*
LF	*Line Feed*
LIDS	*Log-based Intrusion Detection Systems*
LML	*Log Monitoring Lackey*
LOF	*Local Outlier Factor*

M

MAFIA	*Merging of Adaptative Finite Intervals*
MINDS	*Minnesota Intrusion Detection System*
MIT	*Massachusetts Institute of Technology*
MOSG	*Mixture-Of-Spherical Gaussians*

N

NCD	*Normalized Compression Distance*
NetSTAT	*Network State Transition Analysis Tool*
NFA	*Nondeterministic Finite-state Automata*
NFR	*Network Flight Recorder*
NIDS	*Network-based Intrusion Detection System*
NSM	*Network Security Monitor*
NSTAT	*Network State Transition Analysis Tool*

O

OpenVAS *Open Vulnerability Assessment System*
OSSEC *Open Source Security*
OSSIM *Open Source Security Information Management*
OWASP *Open Web Application Security Project*

P

PCAP *Packet Capture Library*
PERL *Pratical Extraction and Report Language*
PHP *PHP: Hypertext Preprocessor*
pMAFIA *parallel* MAFIA

Q

R

RAID *Recent Advances in Intrusion Detection*
RFC *Request For Comments*
RFI *Remote File Inclusion*

S

SAIC *Science Applications International Corporation*
SMTP *Simple Mail Transfer Protocol*
SOM *Self-Organizing Maps*
SQL *Structured Query Language*
SRI *Stanford Research Institute*
STAT *State Transition Analysis Technique*
SVM *Support Vector Machine*

T

TCP *Transmission Control Protocol*

U

URI *Uniform Resource Identifier*
URL *Uniform Resource Locator*
USTAT *Unix State Transition Analysis Tool*

V

VM *Virtual Machine*
VMI *Virtual Machine Introspector*
VMM *Virtual Machine Monitor*
vp *verdadeiros positivos*

W

WAF *Web Application Firewall*
WEBDAV *Web-based Distributed Authoring and Versioning*

X

XSS *Cross-site Scripting*

Y

Z

Capítulo 1

Introdução

Os Sistemas de Detecção de Intrusão (*Intrusion Detection Systems*) são sistemas que atuam junto ao sistema operacional ou a uma rede de computadores para detectar atividades maliciosas. Para identificar um ataque em potencial, os IDSs são implementados com métodos que geralmente são baseados em mau uso *(misuse-based)* ou baseados em anomalia *(anomaly-based)*. Os IDSs baseados em mau uso são comuns no mercado pois permitem representar os ataques através de padrões, tais como regras e assinaturas. A manipulação direta dessas regras permitem tanto ao operador fazer tratamento dos falsos positivos, como permitem ao fornecedor do dispositivo manter uma base de padrões de ataques. Por outro lado, os IDSs baseados em anomalias não são comuns de encontrar no mercado por causa de dois motivos principais: precisam ser treinados com o ambiente e costumam gerar uma taxa alta de alarmes falsos. Mas apresentam certas vantagens, tais como detectar ataques recentes e obter um baixo índice de falsos negativos.

Ao focar os IDSs baseados em anomalias, percebe-se que o índice de falsos positivos pode ser afetado pelo modo como o treinamento é realizado, pois os modelos de detecção de intrusão são construídos de acordo com as informações coletadas durante o treinamento. No caso dos métodos de aprendizagem de máquina que utilizam agrupamento de dados (*data clustering*), as informações semelhantes são agrupadas e utilizadas para extrair um modelo de comportamento do ambiente. Para cada grupo resultante, um rótulo de normalidade ou de ataque é atribuído de acordo com um critério pré-definido, no qual, de acordo com os trabalhos relacionados, considera-se que os grupos menores ou isolados (*outliers*[1]) são ataques. Com o objetivo de fazer uma melhor avaliação desses grupos e consequentemente reduzir o índice de falsos positivos, este trabalho apresenta um método heurístico para a atribuição de rótulos de reputação para os grupos de acordo com uma escala empírica. Esse método heurístico faz parte da arquitetura de IDS baseado em anomalia proposta neste trabalho.

Sendo assim, para testar a arquitetura proposta, foram coletadas as requisições HTTP de dois servidores *web* onde as informações de alguns campos dessas requisições foram utilizadas tanto no treinamento como no momento da detecção de intrusão. Durante os experimentos foi feito um comparativo com o Snort - um IDS baseado em regras que é conhecido tanto no mercado como na comunidade científica. Como resultado, a implementação do IDS da arquitetura proposta obteve melhores índices dado um treinamento sobre o mínimo de 15% das requisições.

[1] Definição no dicionário [Merriam-Webster, 2011]: *outlier* é uma observação estatística que é marcadamente diferente do valor das demais da amostra.

1.1 Motivação

Na comunidade científica é possível encontrar diversos trabalhos que tratam de IDSs baseados em anomalias e que buscam reduzir o número de falsos positivos. Uma técnica baseada em anomalia que está obtendo bons resultados ao ser aplicada na detecção de intrusão é o agrupamento de dados. Nesse tipo de IDS o aprendizado é realizado com o agrupamento de informações de um conjunto de dados com a intenção de extrair modelos para serem aplicados na detecção de intrusão. Após o agrupamento, cada grupo é rotulado como normal ou ataque, esse é o caso encontrado nos trabalhos de [Portnoy et al., 2001] e [Zhong et al., 2007]. Outra maneira de aprendizado é estabelecer limiares ou áreas para os grupos e considerar que os *outliers* são intrusões, como são os casos encontrados nos trabalhos de [Dokas et al., 2002], [Guan et al., 2003], [Ertöz et al., 2004], entre outros. **Porém os *outliers* continuam sendo um problema, visto que se eles representarem atividades legítimas irão influenciar na alta taxa de alarmes falsos nesses tipos de IDSs.**

Os IDSs para serviços *web* baseados em anomalia também vêm se destacando, isso ocorre porque os serviços via *web* ficaram populares na *Internet* e, consequentemente, visados por atacantes. **Fontes como os sítios** [CERT.br, 2010] **e** [Zone-H, 2010] **confirmam o aumento no número de ataques aos servidores *web* nestes últimos anos.** Sendo assim, surgiram na comunidade científica diversos trabalhos que endereçam problemas para identificar intrusões em ambientes *web*. Um dos primeiros trabalhos de IDS baseado em anomalia a tratar de HTTP foi o de [Kruegel and Vigna, 2003], este basicamente extraía o URI das requisições para treinamento e detecção de intrusão. A partir daí diversos outros trabalhos foram elaborados não só para tratar de informações no URI; mas também no conteúdo das requisições, na disposição dos recursos disponíveis no sítio *web*, e até mesmo no código-fonte do sítio fornecido pelo servidor *web*. Alguns trabalhos nessa área são [Ingham and Inoue, 2007], [Bolzoni and Etalle, 2008], [Maggi et al., 2009], [Criscione et al., 2009], [Robertson et al., 2010], entre outros.

O Snort é um IDS que está inserido tanto no mercado como na comunidade de *software livre*. Já se passaram mais de 10 anos quando Roesch disponibilizou o Snort [Roesch, 1999], e este continua um IDS popular, confiável e robusto. Hoje o Snort é referência quando o assunto é IDS baseado em regras, e é possível comprovar isso consultando o Quadrante Mágico[2] do Gartner Inc. [McAfee, 2010], onde uma versão comercial do Snort (SourceFire) pode ser encontrada entre as três primeiras das melhores soluções de IPS *(Intrusion Prevention Systems)*. Um dos últimos trabalhos que fizeram a comparação de um IDS baseado em anomalia com o Snort foi o trabalho de [Ertöz et al., 2004], onde o IDS MINDS obteve melhores resultados. **Desde então surgiram novos ataques, os quais grande parte estão representados por regras no Snort, o que torna este um IDS com potencial para ser comparado a um IDS baseado em anomalias**.

1.2 Desafios

Um dos maiores desafios foi a seleção de um conjunto de dados *(dataset)* que pudesse ser utilizado na comparação com o Snort. Três conjuntos de dados públicos e com

[2] O Quadrante Mágico (*Magic Quadrant*) [Gartner, 2011] foi criado pelo Gartner Inc. para publicar resultados de sua pesquisa sobre um mercado específico, dando uma visão mais ampla da posição relativa entre os concorrentes de mercado. Os fornecedores de produtos e serviços são classificados de forma gráfica em um quadrado que se divide em quatro partes: líderes, desafiadores, visionários e participantes.

ataques rotulados foram encontrados, mas infelizmente não contemplavam os ataques atuais: [KDD, 1999], [DARPA, 1998] e [DARPA, 1999]. Então dois conjuntos de requisições com ataques mais recentes foram encontrados: [iCTF, 2008] e [CDX, 2009]. Outra vez estes não puderam ser utilizados, pois foram criados a partir de uma competição que resultou em ataques não rotulados. E, além disso, a maior parte do tráfego coletado era de ataques, o que não é bom para o treinamento de um IDS baseado em anomalia. Logo, recorreu-se aos servidores da Celepar [CELEPAR, 2010] para construir um conjunto de requisições em um cenário real, atual e constituído de ataques rotulados.

Outro desafio foi a implementação da inspeção de requisições conforme a especificação que trata do protocolo HTTP no documento RFC 2616 [Fielding et al., 1999]. O IDS baseado em anomalia precisa dessa inspeção não só para a detecção de intrusão, mas até mesmo antes do treinamento para evitar que requisições incompletas ou mal formadas venham a criar modelos errados.

Para implementar um IDS baseado em anomalia também foi necessário pesquisar entre os diversos métodos baseados em estatística, conhecimento e aprendizagem de máquina. Na comunidade científica os trabalhos que implementam esses métodos são numerosos, e a seleção de um método foi o que mais dispensou tempo nesta pesquisa. Ao final, o foco foi direcionado para os algoritmos de agrupamento *(clustering)*, o que resultou em um módulo na arquitetura de treinamento do IDS proposto.

1.3 Proposta

Este trabalho apresenta uma proposta de arquitetura de IDS baseado em anomalias que utiliza técnicas de agrupamento para a detecção de ataques em requisições efetuadas para servidores *web*. A arquitetura contém duas partes: uma para treinamento (1) e outra para detecção de intrusão (2).

1) Treinamento

A contribuição principal deste trabalho está na **proposição de um método heurístico para avaliação de grupos**, e que faz parte da arquitetura de treinamento. As informações coletadas das requisições HTTP são separadas em listas nas quais se aplica um algoritmo de agrupamento. Assim o método proposto poderá ser utilizado para atribuir rótulos aos grupos resultantes de acordo com os seguintes parâmetros:

- Índice de popularidade dos grupos: para cada grupo é calculado um índice com base na quantidade de *hosts* (hospedeiros) e a frequência de requisições desses *hosts*.
- Confiabilidade dos *hosts*: onde os *hosts* que realizaram requisições em grupos populares recebem um grau de confiança.
- Índice de confiabilidade dos grupos: para cada grupo é calculado um índice de acordo com a confiabilidade dos *hosts* que fizeram requisições no grupo.
- Índice de reputação dos grupos: para cada grupo é calculado um índice que é o resultado da soma ponderada do índice de popularidade com o índice de confiabilidade.

Ao relacionar o índice de reputação com uma escala empírica, cada grupo será rotulado com uma reputação que poderá ser péssima, ruim, regular, boa, ótima ou excelente. Como resultado, um modelo de comportamento do ambiente será extraído e poderá ser utilizado na detecção de intrusão. Isso é possível pois o treinamento é realizado sem o conhecimento prévio sobre a existência de ataques nas requisições coletadas. Tal treinamento é conhecido como aprendizado não-supervisionado (*unsupervised learning*).

2) Detecção de Intrusão

Ao final, na arquitetura de detecção de intrusão, as informações de cada requisição HTTP são inspecionadas e comparadas com os grupos rotulados (ou assinaturas). Os campos são combinados e classificados de acordo com uma escala empírica para determinar se a requisição é normal, suspeita ou intrusiva.

1.4 Contribuição

A principal contribuição está no método heurístico proposto para avaliar os grupos gerados por um algoritmo de agrupamento. Os trabalhos nessa área estabelecem critérios para detecção de intrusão rotulando grupos menores ou isolados (*outliers*) como ataques durante ou após o agrupamento. Os trabalhos de [Portnoy et al., 2001] e [Zhong et al., 2007] aplicam esse critério no pós-agrupamento, assim ficam independentes do algoritmo de agrupamento utilizado. Na arquitetura proposta o tratamento para os grupos também é realizado após o agrupamento. Porém a avaliação dos grupos é diferente, pois é atribuído um rótulo de reputação para cada grupo, o que acaba reduzindo o número de falsos positivos em comparação com esses trabalhos.

Outra contribuição deste trabalho está na arquitetura proposta de IDS baseado em anomalia, a qual foi capaz de obter boas taxas de detecção de ataques e baixos índices de falsos negativos em comparação com o IDS Snort. Isso utilizando conjuntos atuais de requisições HTTP com ataques em serviços *web*.

1.5 Organização

Além deste capítulo, esta dissertação está organizada com mais seis capítulos. No Capítulo 2 é apresentado o **embasamento teórico** que faz uma abordagem geral sobre os sistemas de detecção de intrusão. No Capítulo 3 são apresentados os **trabalhos** que estão **relacionados** com esta dissertação. No Capítulo 4 a **arquitetura proposta** é detalhada para dar ênfase aos pontos da contribuição. Na sequência, no Capítulo 5, o **cenário de validação** é apresentado com a descrição dos conjuntos de requisições e dos algoritmos utilizados. No Capítulo 6 são apresentados os **resultados dos experimentos** para a validação da proposta. Ao final, no Capítulo 7, são feitas as **conclusões** e sugestões para **trabalhos futuros**.

Capítulo 2

Embasamento Teórico

Neste capítulo o objetivo é apresentar o embasamento teórico sobre os diversos tipos de IDSs com o propósito de situar o contexto em que se encontra a arquitetura proposta. Uma definição sobre IDS e um breve histórico são apresentados respectivamente nas seções 2.1 e 2.2. Na seção 2.3 são apresentados alguns modelos de classificação de IDSs e, com mais detalhes, o modelo encontrado no documento RFC 4766[1] que foi selecionado por permitir uma boa ilustração dos tipos de IDSs. Nas quatro seções seguintes os tipos de IDS são classificados, alguns trabalhos científicos são citados e são levantadas as vantagens e desvantagens de cada tipo de IDS. Sendo assim, as classificações são: por método de coleta (na seção 2.4), por estratégia de análise (na seção 2.5), por técnica de análise (na seção 2.6) e pela arquitetura (na seção 2.7). Ao final, na seção 2.8, são feitas algumas considerações sobre este capítulo.

2.1 O que é um IDS?

Os sistemas de detecção de intrusão são normalmente citados na literatura com a sigla IDS, em inglês: *Intrusion Detection Systems*.

Uma analogia aos sistemas de detecção de intrusão é como um alarme em uma casa. Antes de viajar, o dono tranca os objetos de valor dentro da casa usando correntes e cadeados. De nada adiantaria deixar a casa sozinha, se um ladrão astuto tem a liberdade para testar chaves em cadeados e serrar as correntes. Portanto, para melhorar a segurança, o dono coloca alarmes e câmeras em pontos estratégicos da casa. Se o ladrão investir contra o patrimônio, os vigias serão alertados automaticamente pelo sistema de segurança.

Não obstante, um IDS monitora um ambiente computacional assim como um alarme, procurando identificar intrusos. Se houver uma tentativa de ataque, o sistema é capaz de gerar um alerta informando o ocorrido aos responsáveis pela segurança.

Na definição de [Bace and Mell, 2001]: detecção de intrusão é o processo de monitoração de eventos que ocorrem em uma rede ou sistema de computadores e a análise destes para detectar sinais de intrusão, definidos como tentativas de comprometer a confidencialidade, integridade, disponibilidade, ou burlar mecanismos de segurança de uma rede ou computador.

[1] O documento RFC 4766 [Wood and Erlinger, 2007] trata dos requisitos na troca de mensagens de detecção de intrusão conhecidas como IDMEF (*Intrusion Detection Message Exchange Format*).

2.2 Breve Histórico dos IDSs

O primeiro método de detecção de intrusão era apenas a maneira como os administradores monitoravam as atividades dos usuários do sistema. Por exemplo, conforme [Kemmerer and Vigna, 2002], seria caracterizada uma intrusão onde um usuário que deveria estar de férias estaria acessando um terminal da empresa; ou ainda, uma impressora que é usada raramente estaria com muita atividade. Ao final da década de 1970 os *logs* de auditoria passaram a ser analisados para se detectar intrusões. Na década de 1980 surgem os primeiros projetos de sistemas de detecção de intrusão, e em seguida na década de 1990 são apresentadas muitas pesquisas e os primeiros sistemas de detecção em tempo real. A seguir, uma visão detalhada da história dos IDSs segundo [Innella, 2001] e [Bruneau, 2003]:

- 1980 - Com o artigo *"Computer Security Threat Monitoring and Surveillance"* de James Anderson, nasce a noção de detecção de intrusão através de auditoria.
- 1984 - A Dra. Dorothy Denning publicou o trabalho *"An Intrusion Detection Model"* em que apresenta o primeiro modelo para detecção de intrusão, chamado IDES (*Intrusion Detection Expert System*). No mesmo ano o projeto IDES é desenvolvido.
- 1988 - Na Universidade da Califórnia, o projeto Haystack resultou em um IDS baseado em análise de dados de auditoria.
- 1989 - Haystack se torna uma sociedade comercial e lança o Stalker, um IDS baseado em *host* (definição na seção 2.4.1).
- 1990 - Na Universidade da Califórnia, Davis Todd Heberlein introduz a idéia do primeiro IDS baseado em rede (definição na seção 2.4.2): NSM *(Network Security Monitor)*. Heberlein também introduziu a primeira idéia de IDS híbrido.
- 1990 - O IDS baseado em *host* chamado CMDS *(Computer Misuse Detection System)* é desenvolvido pela SAIC *(Science Applications International Corporation)*.
- 1991 - A Força Aérea dos Estados Unidos desenvolve um sistema chamado ASIM *(Automated Security Measurement System)* para monitorar tráfego de rede. Mais tarde o projeto ASIM forma uma companhia comercial chamada Wheel Group.
- 1994 - Wheel Group lançou o NetRanger que foi o primeiro dispositivo IDS baseado em rede comercialmente viável.
- 1997 - A líder de mercado de segurança, ISS *(Internet Security System)*, lança a primeira versão comercial de seu IDS chamado RealSecure.
- 1998 - A Cisco adquire a Wheel Group para fornecer soluções de segurança a seus clientes.
- 1998 - Uma companhia de IDS chamada de Centrax Corporation surge da fusão de pessoas da equipe do projeto Haystack e do projeto CMDS.
- 1998 - Martin Roesch trabalha com um IDS leve para multiplataformas chamado Snort.

- 1998 - Laboratório Lincoln do MIT *(Massachusetts Institute of Technology)* realiza a primeira avaliação de IDS para a DARPA *(Defense Advanced Research Projects Agency)* [DARPA, 1998].

- 1999 - Segunda avaliação de IDS realizada pelo MIT [DARPA, 1999]. Avaliações que foram consideradas as mais objetivas e completas já publicadas.

Desde o ano de 2000 o IDS se consolidou como produto de mercado, e durante essa década as pesquisas na área se expandiram em numerosas propostas e soluções que podem ser encontradas nos principais eventos e simpósios de segurança. Um deles, especialmente dedicado para essa área, é o RAID *(Recent Advances in Intrusion Detection)* que ocorre anualmente desde 1998 [Dacier and Jackson, 1998].

Já no mercado de IDS, em 2003 os líderes eram as empresas *Network Associates*, *Internet Security Systems*, Cisco *Systems* e Symantec [Stiennon, 2004]. Com a evolução dos dispositivos a nova proposta é que o sistema não apenas detecte, mas que também previna uma intrusão: IPS *(Intrusion Prevention System)*.

Em 2006, a Symantec anuncia sua saída do mercado de *firewall* e IPS, mas faz parceria com a Juniper *Networks*. No mesmo ano a Check Point adquire a NFR *Security* e entra no mercado com o produto SmartDefense [Young and Pescatore, 2006]. Ainda em 2006, a IBM adquire a ISS e entra na lista de líderes desse mercado [IBM, 2006].

Recentemente, segundo o Quadrante Mágico de 2010 do Gartner Inc., as empresas líderes em *Network* IPS são [McAfee, 2010]: a McAfee, a SourceFire e a HP[2] . Percebe-se que atualmente os principais dispositivos de IPS implementam métodos que são baseados em regras, assinaturas e inspeção de protocolos de redes.

2.3 Modelos de Classificação de IDSs

Em geral as classificações dos tipos de IDSs estão relacionadas com a classificação dos tipos de ataques. Nesta seção são apresentadas algumas delas e aquela que foi selecionada para ilustrar os diversos tipos de IDS apresentados neste capítulo.

- Howard e Longstaff produziram uma taxonomia [Howard and Longstaff, 1998] que é usada para classificar milhares de ataques utilizando dados do CERT *(Computer Emergency Response Team)*. Basicamente a classificação é feita por: tipo de ataque, ferramentas utilizadas, tipo de acesso, resultado do ataque e objetivo do ataque. Essa taxonomia é útil, tanto para classificar ataques como para classificar IDSs.

- Resultante da parte de uma avaliação de IDS para o Departamento de Defesa dos Estados Unidos, a taxonomia de [Kendall, 1999] proposta por Kristopher Kendall sugere que os ataques devem ser classificados por níveis de privilégios e transições entre os níveis, métodos de transação/exploração e por ações do atacante.

- A taxonomia proposta por Stefan Axelsson [Axelsson, 2000] primeiro classifica os IDSs pela técnica de intrusão, e depois classifica pelas características particulares de cada sistema. Algumas características são: detecção em tempo real; processamento contínuo ou

[2]Em 2009 a empresa HP comprou a 3Com [HP, 2009] e adquiriu o IPS TippingPoint, naquele ano este era um líder no Quadrante Mágico do Gartner Inc.

em lote; fonte dos dados de auditoria; tipo de resposta a intrusões; localização dos dados, segurança do próprio IDS e interoperabilidade com outros IDSs.

- Em 2007, a nova taxonomia proposta por [Tucker et al., 2007] e publicado pelo Emerald *Group* propõe utilizar uma matriz bidimensional, que é composta por: detecção, reconhecimento, identificação, confirmação e prossecução. Essas informações são cruzadas com: arquivo, *host*, rede e *enterprise*. Resultando em uma maneira que facilita a visualização e classificação de IDSs.
- Também em 2007, Wood e Erlinger propõem no documento RFC 4766 [Wood and Erlinger, 2007] um diagrama que ilustra os termos e a interação entre os componentes de um IDS. Esse diagrama é usado como base da classificação de IDS neste documento e foi detalhado na seção seguinte.

2.3.1 Diagrama para Classificação de IDSs

O esforço realizado pelo IDWG *(Intrusion Detection Work Group)* para padronizar a comunicação entre IDSs resultou no documento RFC 4766 [Wood and Erlinger, 2007]. Documento no qual poderá ser encontrado um diagrama usado para representar a troca de mensagens IDMEF (*Intrusion Detection Message Exchange Format*) entre IDSs. Na figura 2.1 apresenta-se o diagrama que foi utilizado para ilustrar a classificação dos tipos de IDSs.

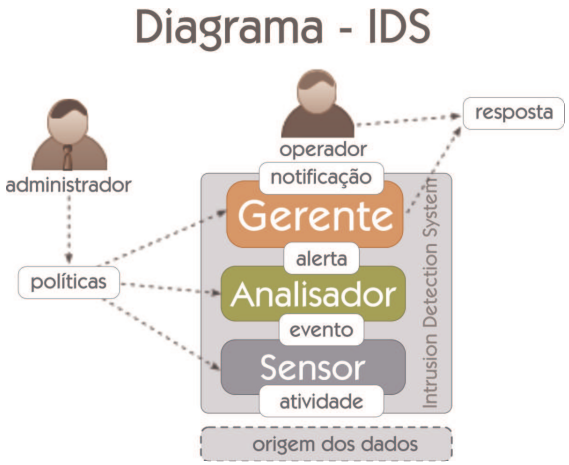

Figura 2.1: Diagrama de IDS - RFC 4766

O **administrador** é o responsável pela configuração do IDS de acordo com as **políticas de segurança** da empresa. Logo, o **operador** é quem verifica os eventos capturados pelo IDS e toma as ações necessárias (**resposta**).

O **sensor** coleta informações (**origem dos dados**) de uma **atividade** não autorizada. Por exemplo, uma sessão de *telnet* é coletada pelo sensor e um **evento** é enviado ao analisador. O **analisador**, por sua vez, analisa o evento para detectar alguma intrusão. Se o evento for identificado como uma ação maliciosa, o analisador gerará um **alerta** ao gerente. Assim o **gerente** fará a gestão de **notificações** de eventos e poderá até mesmo reagir ao incidente (**resposta**).

Nem todo alerta é autêntico, podem ocorrer alertas de **falso positivo**, onde um evento legítimo e não malicioso pode ser identificado como um ataque. Já a ausência de um alerta no caso de um evento malicioso (um ataque), a ocorrência é conhecida como **falso negativo**.

A estrutura apresentada pode ser utilizada como um modelo para identificar uma estrutura de IDS. O sensor, o analisador e o gerente são os mecanismos que podem ser encontrados juntos ou separadamente em qualquer IDS existente. Nas próximas seções, vários IDSs são ilustrados e comparados a esse diagrama.

2.4 Classificação de IDSs pelo Método de Coleta

De acordo com o diagrama da figura 2.1, o componente **sensor** do IDS faz a coleta de informações antes de efetuar uma análise. Os métodos de coleta de informações mais comuns de se encontrar na literatura de IDSs são: baseados em hospedeiro, baseados em rede e, mais recente que estes, baseados em máquinas virtuais. Esses métodos são descritos nas seções seguintes.

2.4.1 IDS baseado em Hospedeiro

O IDS baseado em hospedeiro que também é conhecido como HIDS *(Host-based Intrusion Detection System)*, é um sistema que monitora um único *host* para detectar atividades suspeitas. O HIDS normalmente coleta informações de duas maneiras: pistas de auditoria do sistema operacional, e *logs* do sistema. As pistas de auditoria geralmente são geradas pelo *kernel*, e portanto são mais detalhadas e protegidas do que os *logs* do sistema. Já, os *logs* de sistema são menores e mais fáceis de compreender [Bace and Mell, 2001]. Como ilustrado na figura 2.2, o HIDS é um IDS instalado em um *host* e que através do **sensor** faz a coleta de informações do próprio sistema.

O USTAT [Ilgun, 1993] é um HIDS que estende a abordagem STAT *(State Transition Analysis Technique)* [Ilgun et al., 1995] aplicando a auditoria de trilhas, arquivos e *logs* em sistemas Unix.

O sistema eXpert-BSM [Lindqvist and Porras, 2001] é um HIDS que analisa pistas de auditoria do sistema operacional Solaris da Sun. Esse HIDS contém uma base de conhecimento que foi criada depois de anos de pesquisa e que pode ser utilizada para detectar diversos e específicos tipos de mau uso do sistema.

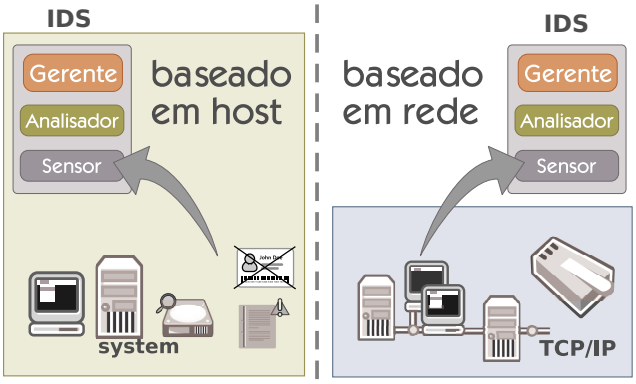

Figura 2.2: Ilustração - IDS baseado em Hospedeiro e IDS baseado em Rede

O Tripware [Hrivnak, 2002] é um HIDS que foi desenvolvido em 1992 na *Purdue University*. A Tripware Inc. foi formada em 1997, e liberou seu *software* como produto em

1999. A função básica do Tripware é verificar a integridade de arquivos e diretórios importantes através de uma base, e gerar um alerta se ocorrer qualquer mudança de acordo com a política aplicada.

Além desses, outros HIDSs conhecidos são [Mandujano, 2004] e [Krawczyk, 2007]: eTrust Audit, LIDS, ISS Proventia *Server* e *Desktop*, OSSEC, Prelude-LML, Samhain, etc.

Algumas vantagens: os HIDS podem detectar ataques em eventos locais que não poderiam ser detectados pela rede; também podem tratar dados criptografados e não são afetados por *switches* de redes. Portanto, o HIDS pode reagir rapidamente à um ataque. Por outro lado, o HIDS é de difícil instalação e manutenção; o próprio HIDS está propenso à ataques; é difícil detectar ataques de rede e sua execução pode interferir no desempenho do próprio *host* [Bace and Mell, 2001].

Sensores de Aplicações

É um subconjunto de HIDS que analisa os eventos de uma aplicação, onde as informações que são coletadas se originam de transações em arquivos de *log* e relatórios da própria aplicação [Bace and Mell, 2001].

Um exemplo é o protótipo em [Almgren and Lindqvist, 2001] que monitora e integra mensagens do servidor *web* Apache para o *framework* EMERALD.

Monitorar a interação entre usuário e aplicação permite tratar mais dados e de maneira mais detalhada. Mas um sensor de aplicação pode ser mais vulnerável que um HIDS se o sistema operacional não tratar de pistas de auditoria [Bace and Mell, 2001].

2.4.2 IDS baseado em Rede

É um sistema conhecido como NIDS *(Network-based Intrusion Detection System)* que faz a coleta das informações da rede para identificar ataques. O NIDS pode ser instalado no modo *inline* ou no modo passivo. No modo *inline*, o NIDS é instalado em um ponto que intercepta o fluxo da rede, atuando como um dispositivo de ponte (*bridge*) e capturando pacotes para detectar uma intrusão. Já no modo passivo, o NIDS é conectado a um *switch* ou *hub* que passa cópias dos pacotes da rede para a sua análise. Para classificar um IDS como um NIDS, basta que o componente **sensor** faça a coleta de pacotes da rede conforme a ilustração da figura 2.2. Em seguida algumas pesquisas na área:

O EMERALD [Porras and Neumann, 1997] é um *framework* IDS criado pela SRI *(Stanford Research Institute) International*. Uma parte NIDS [Valdes and Skinner, 2000] do EMERALD é uma abordagem em redes *bayes* (definição na seção 2.6.3) para detecção em protocolo TCP sem remontagem, mas apenas com verificação de cabeçalhos.

O NetSTAT [Vigna and Kemmerer, 1998] foi uma pesquisa do DARPA que apresentou um NIDS onde o analisador estende a abordagem STAT, a qual é utilizada para criar diagramas de transição de estados que representam intrusões via rede de computadores.

O Snort [Roesch, 1999] é um NIDS que utiliza a biblioteca *PCAP* para capturar e filtrar pacotes de rede. O Snort faz a inspeção dos pacotes da rede utilizando regras e assinaturas de ataques conhecidos. É um NIDS popular na comunidade de *software* livre.

INBOUNDS [Tjaden et al., 2000] é um NIDS desenvolvido pela Universidade de Ohio, e tecnicamente é um *framework* que faz a análise baseada em anomalias. Uma das sua principais funções é analisar os temporizadores e o comprimento das sessões TCP.

Outros NIDSs conhecidos na comunidade e no mercado são [Mandujano, 2004] e [Krawczyk, 2007]: Bro-IDS, Dragon, RealSecure, ISS Proventia, Check Point SmartDefense, IDP da Juniper, Tipping Point, etc.

Algumas vantagens dos NIDSs são [Bace and Mell, 2001]: quando instalado como um elemento passivo fica transparente para o atacante; é fácil de implementar sem interferir no desempenho dos *hosts* e possui independência de plataforma. Por outro lado, algumas desvantagens dos NIDSs estão em tratar dados de redes de alta velocidade; dependência da rede e dos dispositivos, e a dificuldade em tratar dados criptografados. Por não interferir no fluxo da rede, existe a dificuldade em reagir a um ataque.

2.4.3 IDS baseado em Máquina Virtual

O ambiente de virtualização de máquinas permite que o IDS faça coleta de informações de estados dos *hosts* de maneira diferente do convencional. Nesse ambiente a máquina real contém um monitor conhecido como VMM (*Virtual Machine Monitor*) que é uma camada entre o *hardware* e o sistema operacional (ilustração na figura 2.3). É nessa comunicação entre máquina virtual (*virtual machine*) e *hardware* que as informações extraídas pelo VMM (ou *hypervisor*) podem ser utilizadas na detecção de intrusão.

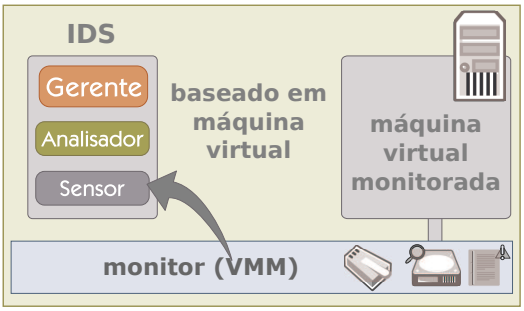

Figura 2.3: Ilustração - IDS baseado em Máquina Virtual

A abordagem de [Garfinkel and Rosenblum, 2003] é uma das primeiras dessa área. Um protótipo chamado de *Livewire* foi criado para testar a arquitetura proposta de VMI (*Virtual Machine Introspection*) que também é conhecida como VMI-*based* IDS. Basicamente o IDS é constituído por um *framework* de políticas, e executa comandos de perguntas ao VMM para poder coletar informações sobre o estado da VM monitorada.

O trabalho de [Kourai and Chiba, 2005] propõe o *HyperSpector framework* que contém três técnicas para detecção de intrusão. Uma delas é a coleta de quadros de rede da VM monitorada para atuar como NIDS. Outra técnica é a montagem remota de disco da VM monitorada para checar a integridade dos arquivos. E, ainda outra, é a técnica de mapeamento de processos da VM monitorada para traçar as chamadas de sistema feitas pelos processos. Para testes, essas três técnicas foram implementadas com auxílio das ferramentas: Snort, Tripware e Truss.

A principal vantagem desses IDSs está em atuar como NIDS e HIDS e monitorar uma máquina virtual de maneira isolada, ou seja, sem a necessidade de serem executados junto ao *host* monitorado. Porém, é possível observar que as desvantagens estão na dependência e nas limitações do ambiente virtualizado.

2.5 Classificação de IDSs pela Estratégia de Análise

Antes de gerar um alerta, o **analisador** do IDS precisa de uma estratégia para determinar se está ocorrendo ou não um ataque. Na literatura é comum encontrar dois tipos de IDSs ao se classificar por estratégia de análise: os que são baseados em mau uso e os que são baseados em anomalia. Mas como complemento destes, nas seções seguintes, também são apresentados os IDSs baseados em especificação e plano de reconhecimento como IDS.

2.5.1 IDS baseado em Mau Uso ou Uso Incorreto

No IDS baseado em mau uso (*misuse*) aplica-se uma técnica onde padrões conhecidos são usados para detectar atividades maliciosas. Os padrões correspondentes aos ataques conhecidos são chamados de assinaturas. Como ilustra a figura 2.4, o **analisador** recebe eventos do **sensor** e faz consultas a uma base de assinaturas para determinar se existe uma correspondência com algum ataque [Lydon, 2004].

A tese de Sandeep Kumar [Kumar, 1995] é o primeiro destaque em IDS baseado em assinaturas. Kumar apresenta um modelo que na sua hierarquia inclui validar eventos através de expressões regulares. Com base na sua tese, o IDIOT IDS foi desenvolvido por estudantes da *Purdue University*.

O STAT [Ilgun et al., 1995] é um exemplo de IDS que faz a representação de padrões de um ataque descrevendo a sequência de ações de uma intrusão através de um diagrama de representação de estados.

LAMBDA [Cuppens and Ortalo, 2000] é a proposta de um modelo geral de regras e assinaturas de ataques para IDSs baseados em mau uso. Foi uma tentativa de colocar uma linguagem declarativa para modelar ataques de maneira global.

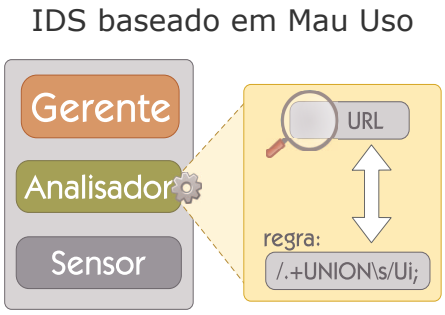

Figura 2.4: Ilustração - IDS baseado em Mau Uso

Grande parte dos IDSs de mercado fazem a combinação de IDSs baseados em redes e assinaturas. Alguns exemplos de IDSs assim são [Krawczyk, 2007]: o Snort e o ISS *Event Policy*.

Uma das vantagens desse método é que não consome tantos recursos como o método baseado em anomalia, além disso, o processamento dos padrões de ataques podem ser otimizados e o número de falsos positivos pode ser controlado apenas com ajustes [Mandujano, 2004].

Entre as desvantagens, exige-se um bom nível de conhecimento de quem cria esses padrões. E outra desvantagem está no padrão de ataque que pode corresponder com uma atividade normal e gerar um grande número de falsos positivos. Mas a principal desvantagem está

na necessidade de atualizar constantemente esses padrões para poderem acompanhar os novos tipos de ameaças [Mandujano, 2004].

2.5.2 IDS baseado em Anomalia ou Comportamento

Conhecido como *Anomaly based IDS* ou *Behavior based IDS*. O objetivo desse IDS é detectar atividades com um comportamento incomum em um *host* ou em uma rede. Assim como ilustra a figura 2.5, o que for diferente de uma atividade normal (legítima), pode ser detectado como um ataque. Portanto, esse tipo de IDS utiliza perfis (*profiles*) que são construídos com dados históricos coletados do ambiente monitorado. Os dados coletados são usados para medir e monitorar as atividades para saber se estas estão ou não fora do normal [Bace and Mell, 2001].

Stefen Hofmeyer, em seu trabalho [Hofmeyr et al., 1998] apresenta o método de IDS baseado em anomalias, onde desenha o perfil de chamadas de sistemas e cria métricas para determinar se uma atividade é normal ou anormal.

Um projeto chamado MINDS *(Minnesota Intrusion Detection System)* [Ertöz et al., 2004] foi implementado e comparado ao método baseado em assinatura do Snort. MINDS é um sistema baseado em mineração de dados (*data mining*) em que o módulo de detecção é alimentado por ferramentas de fluxo de rede, como o NetFlow.

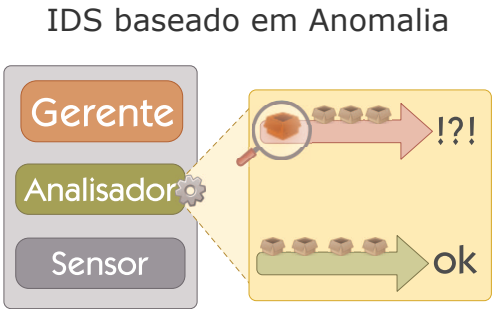

Figura 2.5: Ilustração - IDS baseado em Anomalia ou Comportamento

Entre outros, um sistema de mercado classificado como IDS baseado em anomalias é o Proventia *Anomaly Detection System* da ISS que hoje é um produto da IBM [IBM, 2007].

O IDS baseado em anomalias pode detectar novos ataques pelo simples desvio de comportamento, e sem a necessidade de ter o conhecimento detalhado da intrusão. Os próprios ataques detectados poderão ser tratados como assinaturas para serem utilizados em um IDS baseado em mau uso. Por outro lado, o IDS baseado em anomalias tem as seguintes desvantagens: por detectar atividades anormais, qualquer modificação no comportamento legítimo do *host* ou da rede pode gerar um grande número de falsos alertas; e a coleta de dados para criar o perfil de comportamento pode se tornar difícil e extensiva [Bace and Mell, 2001].

2.5.3 IDS baseado em Especificação

Como ilustra a figura 2.6, cada chamada de sistema feita por uma aplicação deve ser autorizada antes de ser processada. Segundo Andrew Lydon [Lydon, 2004], esse tipo de abordagem requer que as especificações de comportamento normal e de ataques sejam feitas antecipadamente.

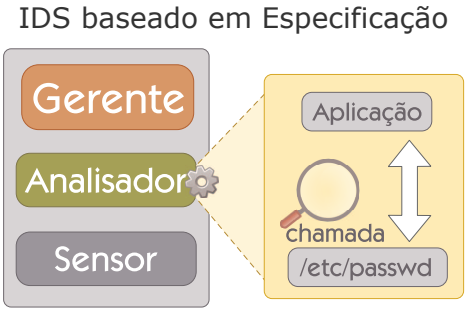

Figura 2.6: Ilustração - IDS baseado em Especificação

O trabalho de Uppuluri e Sekar [Uppuluri and Sekar, 2001] define o IDS baseado em especificação como a combinação de detecção baseada em mau uso com a detecção baseada em anomalia. O trabalho propõe o uso de expressões regulares para criar tais especificações.

Vantagens: se for projetado por um perito, o IDS tem baixo número de falsos positivos. Outra vantagem encontrada é que problemas e ataques podem ser detectados e contidos, mesmo que o administrador não tenha o conhecimento do código-fonte da aplicação que está sendo atacada. Entre as desvantagens, esse tipo de IDS pode tornar o sistema lento; a lista de especificações tende a aumentar, criar gargalos, e nem todos os tipos de ataques podem ser especificados [Lydon, 2004].

2.5.4 Plano de Reconhecimento como IDS

O plano de reconhecimento como IDS é um processo que tenta inferir o objetivo do atacante. Assim como ilustra a figura 2.7 esse é um método útil para determinar a gravidade do ataque e do que foi comprometido. Um exemplo, seria um *hacker* tentar ganhar acesso em quantas máquinas fosse possível e, em contrapartida, um intruso que tenta acessar dados de contabilidade com o intuito de obter ganhos financeiros. Do ponto de vista técnico, o problema mais grave seria o *hacker* conseguir o acesso. Mas do ponto de vista de negócio, o vazamento de informações seria o mais grave. Portanto, o plano de reconhecimento visa identificar esta desconjuntura [Lydon, 2004].

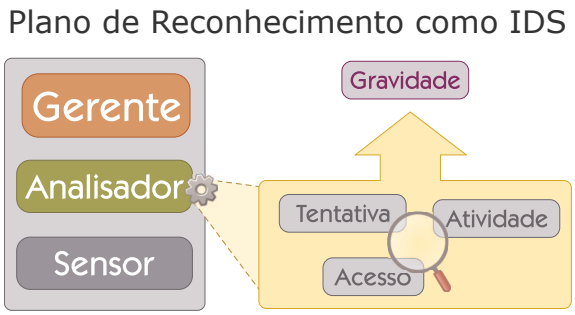

Figura 2.7: Ilustração - Plano de Reconhecimento como IDS

Alguns exemplos de pesquisas sobre esse método são: [Geib and Goldman, 2001] que aborda o assunto e um nível mais alto de granularidade, e [Chirichiello, 2006] que aborda um modelo de *framework* para plano de reconhecimento.

Em geral os planos de reconhecimento são manuais, pois fatores humanos levam vantagem sobre o modo automático. Por outro lado, o plano manual toma tempo do administrador para interpretar um ataque; mas o plano automático muito geral não ajuda na reconstrução do ataque ou predição da ação [Lydon, 2004].

2.6 Classificação de IDSs pela Técnica de Análise

Os IDSs baseados em mau uso precisam de padrões pré-definidos para aplicar a detecção. Já os IDSs baseados em anomalia precisam observar os sistemas alvos (parametrização), construir um modelo de comportamento normal (treinamento), para então poder comparar com o que foi observado antes (detecção). Tanto os IDSs baseados em mau uso como os baseados em anomalia precisam aplicar técnicas para analisar os eventos enviados pelo sensor . A ilustração das técnicas de análise na figura 2.8 foi retirada do artigo de [Garcia-Teodoro et al., 2009]. Nas seções seguintes são apresentadas as técnicas baseadas em: estatística, conhecimento e aprendizagem de máquina.

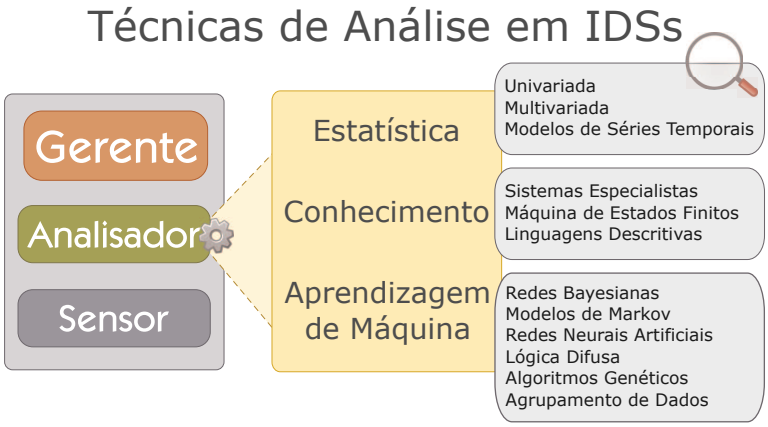

Figura 2.8: Ilustração - Técnicas de Análise em IDSs

2.6.1 IDS baseado em Estatística

É comum encontrar pesquisas na área de IDSs baseados em anomalia que utilizam técnicas estatísticas. Na detecção com base em estatística *(statistical based)* são criados perfis que representam, por exemplo, o comportamento de uma rede. O perfil é baseado em métricas, tais como tráfego, número de pacotes, número de conexões, número de diferentes endereços IPs, etc. Se um evento não atende um perfil, uma pontuação é efetuada para determinar se o evento é irregular, assim o IDS poderá marcar a ocorrência como anômala.

A técnica de análise pode ser simplesmente **univariada** *(univariate)*, onde uma variável é dividida em intervalos e frequências. Ou a técnica de análise pode ser **multivariada** *(multivariate)*, onde se considera as correlações entre duas ou mais métricas. O trabalho de [Ye et al., 2002] aplica a análise multivariada como método de um HIDS. Outro exemplo, é o trabalho de [Qu et al., 2005] que aplica esse método em um NIDS.

Outra técnica de análise se baseia em **modelos de séries temporais** *(time series model)*, que utiliza o intervalo de tempo juntamente com um contador de eventos ou outro tipo de

medida. Se a ocorrência de diversos eventos durante um determinado tempo teve uma frequência maior do que deveria, esses eventos serão detectados como anômalos pelo IDS. O IDES (*Intrusion Detection Expert System*) [Lunt et al., 1988] identifica uma atividade anormal quando esta extrapola um segmento de tempo. Outro trabalho que implementa esse método é o de [Viinikka et al., 2006] que detecta atividades anômalas em séries temporais juntamente com a correlação de alertas.

Uma das vantagens é que esse método não exige conhecimento prévio da atividade normal; pelo contrário, o próprio método aprende o comportamento do sistema a partir de observações. Outra vantagem é que esses métodos estatísticos podem indicar precisamente as atividades maliciosas que ocorrem durante longos períodos de tempo. Entre as desvantagens, os IDSs que utilizam esses métodos estão susceptíveis a serem treinados com os ataques como sendo uma atividade normal. Também existe a dificuldade em atribuir valores aos diferentes parâmetros e, ainda, nem todos os comportamentos podem ser modelados por processos estocásticos [Garcia-Teodoro et al., 2009].

2.6.2 IDS baseado em Conhecimento

Ao aplicar técnicas de análise baseadas em conhecimento *(knowledge based)* o IDS contém uma base de padrões de ataques pré-definidos. Assim, as regras e as assinaturas que foram criadas com o conhecimento de especialistas são utilizados em um IDS baseado em mau uso. Já no IDS baseado em anomalia, esses padrões são criados a partir do comportamento normal do ambiente para detectar ataques através de anomalias.

Os **sistemas especialistas** *(expert systems)* são destinados a classificar os dados auditados de acordo com um conjunto de regras. Essas regras ou especificações são construídas por uma pessoa especialista no assunto que determina o comportamento legítimo do sistema. O modelo então será capaz de detectar ataques e anomalias de acordo com o conhecimento que foi especificado antecipadamente. O IDES [Lunt et al., 1988] (também na seção 2.6.1) é um exemplo de IDS construído como sistema especialista. Dependendo da abordagem, esse método é considerado uma combinação de detecção de anomalia com a detecção de mau uso conforme descrição na seção 2.5.3.

Para criar regras ou especificações é possível utilizar **máquina de estados finitos** *(finite state machine)* ou **linguagens descritivas** *(description languages)*. Um exemplo é o trabalho de [Tseng et al., 2003], onde o comportamento normal do protocolo de roteamento AODV *(Ad-hoc On-Demand Distance Vector)* é representado por uma máquina de estados finitos, permitindo detectar ataques em uma rede móvel. Outro exemplo é o STAT [Ilgun et al., 1995] (também na seção 2.5.1) que representa ataques em máquinas de estados para a detecção de intrusão.

As vantagens desse tipo de IDS estão na robustez e na flexibilidade. Sua principal desvantagem está no desenvolvimento do conhecimento que deve ser de alta qualidade, e que muitas vezes pode ser difícil e demorado para se obter [Garcia-Teodoro et al., 2009].

2.6.3 IDS baseado em Aprendizagem de Máquina

Os métodos de detecção baseados em aprendizagem de máquina *(machine learning based)* estabelecem modelos explícitos ou implícitos de padrões categorizados. Nesse método é comum determinar o comportamento normal através de treinamento realizado sobre uma base

com dados rotulados. No caso de IDS baseado em mau uso, o treinamento deve ser feito sobre uma base de ataques conhecidos. Muitas vezes a aplicação de aprendizagem de máquina pode coincidir com as técnicas estatísticas, mas esse método tem a capacidade de mudar sua estratégia de execução de acordo com novas informações que são adquiridas. A seguir alguns métodos de aprendizagem em IDSs:

a) **Redes Bayesianas:** modelos que utilizam aprendizagem bayesiana para detecção de intrusão têm uma estrutura de detecção probabilística em que uma variável medida pode afetar as outras. Essa informação é usada para criar redes de crenças chamadas de redes bayesianas *(bayes networks)*. As redes bayesianas utilizam um conjunto de probabilidades condicionais que, assim, podem determinar uma intrusão com uma certa probabilidade dado a presença ou ausência de evidências [Mandujano, 2004]. Um exemplo é o trabalho de [Kruegel et al., 2003] que propõe a classificação de eventos para detecção de intrusão usando redes bayesianas. Entre as vantagens, as redes bayesianas têm a capacidade de incorporar tanto o conhecimento quanto os dados. E, também, devido a interdependência entre as variáveis, pode-se fazer a predição de eventos. Entre as desvantagens, as redes bayesianas são altamente dependentes do comportamento do sistema alvo, onde qualquer desvio tende a causar erros de detecção [Garcia-Teodoro et al., 2009].

b) **Modelos de Markov:** uma cadeia de Markov é um conjunto de estados que estão interligados através de probabilidades de transição. Durante a fase de treinamento, essas probabilidades são obtidas através do comportamento normal do sistema. A detecção de anomalia ocorre pela comparação da pontuação (associada à probabilidade) obtida da observação de sequências com um limiar fixo. No caso dos modelos ocultos de Markov, o sistema em questão assume que no processo de Markov os seus estados e as suas transições estão ocultos. Um exemplo é o artigo de [Yeung and Ding, 2003] que trata de um HIDS que implementa modelos de Markov. Não muito diferente das redes bayesianas, esse método é altamente dependente do comportamento do sistema [Garcia-Teodoro et al., 2009].

c) **Redes Neurais Artificiais:** ao simular a operação do cérebro humano (neurônios e as sinapses entre eles), as redes neurais artificiais *(artificial neural networks)* são aplicadas aos IDSs baseados em anomalia devido a flexibilidade e a adaptabilidade às mudanças de ambiente. Uma rede neural contém um conjunto de nós organizados em camadas. As camadas de entrada e saída são conectadas por uma ou mais camadas intermediárias, e o aprendizado ocorre na atualização dos pesos entre os neurônios. Para detectar uma intrusão, a rede neural é usada para predizer se o próximo evento é um ataque ou não através da estimulação dos neurônios [Mandujano, 2004]. Um exemplo é o trabalho de [Ramadas et al., 2003] que apresenta um módulo do IDS INBOUNDS para detecção de anomalias aplicando mapas auto-organizáveis *(self-organizing maps)*. Uma desvantagem é que as redes neurais não oferecem um modelo descritivo de como efetuou a detecção de uma intrusão, sendo assim consideradas como uma caixa preta [Garcia-Teodoro et al., 2009]. Outra desvantagem é o tempo necessário para fazer o treinamento. Porém, as redes neurais fazem bom tratamento de dados que contém ruídos [Mandujano, 2004].

d) **Lógica Difusa:** a lógica difusa *(fuzzy logic)* deriva de um conjunto de teorias *fuzzy* que faz o tratamento da incerteza. As técnicas *fuzzy* são utilizadas na detecção de anomalias principalmente porque os recursos a serem considerados podem ser vistos como variáveis *fuzzy*. O trabalho de [Dickerson and Dickerson, 2000] propõe um IDS chamado FIRE

(Fuzzy Intrusion Recognition Engine) que utiliza a lógica difusa na detecção de anomalias. Para detectar varreduras de portas (*port scans*) e sondas (*probes*) a lógica difusa provou ser efetiva, mas a sua maior desvantagem está no alto consumo de recursos envolvidos [Garcia-Teodoro et al., 2009].

e) **Algoritmos Genéticos:** os algoritmos genéticos *(genetic algorithms)* são categorizados como técnicas de solução aproximada através de busca e otimização, e fazem parte de uma classe particular de algoritmos evolucionários por utilizarem técnicas inspiradas em biologia evolucionária (tais como herança, mutação, seleção e recombinação). Um exemplo é o trabalho de [Li, 2004] que aplica a técnica de algoritmos genéticos como NIDS. Entre as vantagens estão a robustez e a flexibilidade no método de busca que converge para uma solução de múltiplas direções, mesmo quando não há um conhecimento prévio do comportamento do sistema. Porém, sua desvantagem está no alto consumo de recursos envolvidos [Garcia-Teodoro et al., 2009].

f) **Agrupamento de Dados:** as técnicas de agrupamento *(clustering)* funcionam agrupando elementos similares em grupos *(clusters)*, dada uma similaridade ou distância de medida entre esses elementos. Ao fazer o agrupamento, alguns elementos podem não se encaixar em grupo algum, esses são elementos isolados ou *outliers*. Para a detecção de intrusão, os elementos *outliers* podem representar um ataque ou uma anomalia. As abordagens de IDS nessa área variam de acordo com o quanto um *outlier* pode representar um ataque. Um exemplo é o trabalho de [Portnoy et al., 2001] que aplica *clustering* para detecção de intrusão e está detalhado na seção 3.1. Uma vantagem dessa técnica está em determinar a ocorrência de intrusão a partir de dados puros de auditoria, reduzindo o esforço necessário para ajustar o IDS [Garcia-Teodoro et al., 2009]. Outra vantagem está em obter bons resultados a partir de treinamento não-supervisionado. Entre as desvantagens, a principal está na dependência do comportamento do ambiente monitorado.

2.7 Classificação de IDSs pela Arquitetura

Um IDS que monitora no modo autônomo (*standalone*) fica limitado ao ambiente em que está inserido. Já um IDS que faz a monitoração de maneira distribuída, além de fornecer uma visão privilegiada de intrusões no ambiente, também faz a detecção de certos tipos de ataques que seriam mais difíceis de se detectar em um IDS autônomo (ex.: ataques de DDoS). Nesta seção são apresentados e classificados os IDSs distribuídos pela sua arquitetura.

2.7.1 IDS com Análise Centralizada

Nesse tipo de IDS os dados são coletados por sensores de maneira distribuída, mas a análise é efetuada de maneira centralizada (primeira ilustração na figura 2.9). Um exemplo é o DIDS (*Distributed Intrusion Detection System*) [Snapp et al., 1991] que faz a monitoração de eventos de diversos *hosts* com uma análise centralizada. Outro exemplo é o NSTAT [Kemmerer, 1998] que aplica a análise STAT (abordagem na seção 2.5.1) em um servidor que recebe dados de auditoria de diversos *hosts* distribuídos.

A principal vantagem desse tipo de IDS está em centralizar o armazenamento e a análise dos eventos, mas seu funcionamento é melhor em um ambiente onde o volume e a

frequência desses eventos são baixos. Entre as desvantagens está o problema de escalabilidade, pois o servidor central pode sofrer com gargalos e, além de representar um ponto de falha, pode se tornar o alvo de ataques [Peng et al., 2007].

2.7.2 IDS com Análise Descentralizada

Nesse tipo de IDS tanto os dados são coletados de maneira distribuída como a análise também pode ser distribuída. Essa arquitetura apresenta diversas vantagens tais como: escalabilidade, sobrevivência em caso de falhas e isolamento de ataques. Nesta seção são apresentados alguns tipos de IDSs com análise descentralizada:

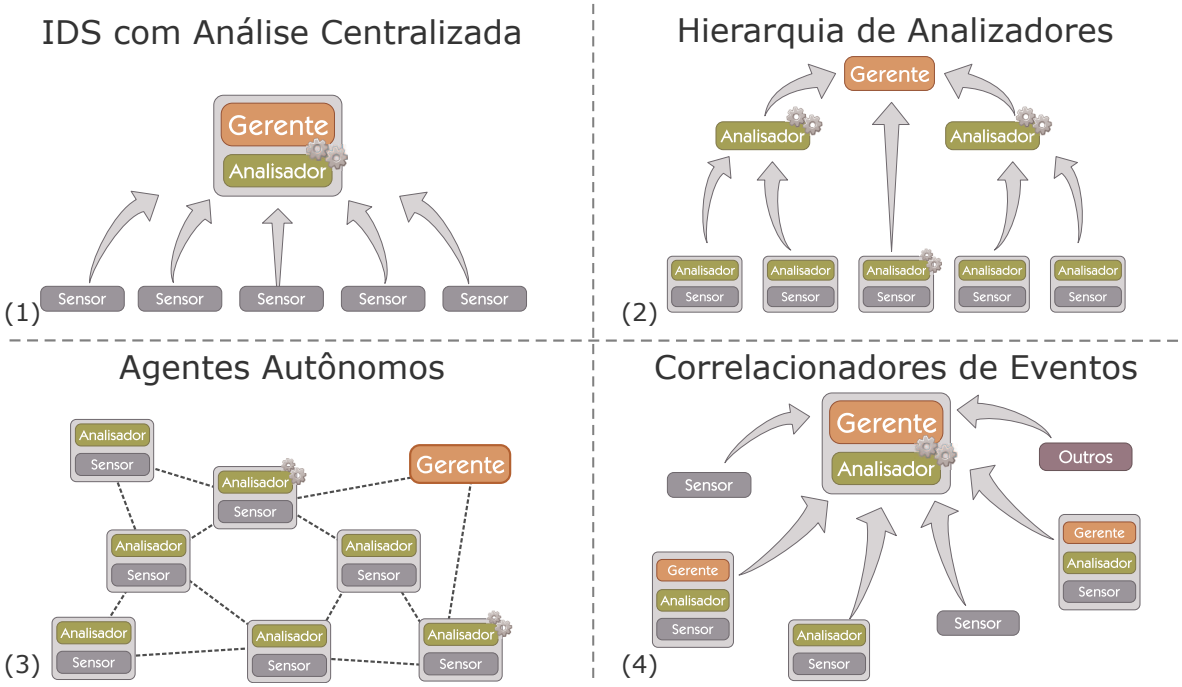

Figura 2.9: Ilustração - Arquiteturas de IDSs Distribuídos

a) **Hierarquia de Analisadores:** este ambiente é composto por diversos agentes de IDS instalados em pontos estratégicos da rede, em servidores ou aplicações. Na ocorrência de alertas de intrusão, o agente irá reportar aos analisadores de um nível hierárquico mais alto (ilustração na figura 2.9 (2)). Um exemplo é o *framework* EMERALD [Porras and Neumann, 1997] onde os analisadores atuam em um ambiente de larga escala e distribuídos hierarquicamente. Esse *framework* aborda a detecção de intrusão distribuída, sobrevivência da rede, isolamento de ataques e resposta automática. Outro exemplo é o GrIDS (*Graph-based Intrusion Detection System*) [Cheung et al., 1999] que agrega informações de comunicações na rede de forma hierárquica e que permite a inspeção visual da ocorrência de ataques através de grafos. Um ponto importante a ser observado nesse tipo de IDS é: se a topologia utilizada convergir para um nó raiz, o IDS pode ser sobrecarregado ou sofrer ataques de negação de serviço.

b) **Agentes Autônomos:** os agentes autônomos são utilizados como um *framework* de IDS para a detecção de ataques de maneira distribuída (ilustração na figura 2.9 (3)). Basicamente um agente autônomo é apenas um programa com capacidade de aprendizagem e que

em geral utilizam técnicas de inteligência artificial ou implementam uma analogia biológica [Lydon, 2004]. Um exemplo é o AAFID *(Autonomous Agents for Intrusion Detection)* [Balasubramaniyan et al., 1998] que propõe uma arquitetura distribuída com pequenas entidades independentes, que são agentes para detectar comportamento anormal ou malicioso. Outro exemplo é o IDA *(Intrusion Detection Agent)* [Asaka et al., 1999] que é uma tentativa de delegar IDS para agentes autônomos visando atingir escalabilidade. Outro exemplo, ainda, é o CDIS *(Computer Defense Immune System)* [Williams et al., 2001] que faz a análise de tráfego da rede através de agentes que formam um sistema computacional imunológico e distribuído. Entre as vantagens, os sistemas autônomos podem atuar como IDSs independentes. Mas as abordagens sobre esse tipo de IDS, segundo [Lydon, 2004], são gerais e deixam de especificar detalhes de implementação.

c) **Correlacionadores de Eventos:** o objetivo desse tipo de IDS é coletar dados de várias fontes e até mesmo de outros IDSs, e fazer uma correlação entre as informações para detectar atividades maliciosas. Além disso, essa arquitetura permite a fusão dos eventos para reduzir o número de alertas a serem vistos pelo operador (ilustração na figura 2.9 (4)). Um exemplo é o *framework* EMERALD [Porras and Neumann, 1997] (citado anteriormente nesta seção) que faz a correlação hierárquica de alertas, e permite identificar os alertas que fazem parte de um mesmo ataque. Outros correlacionadores são [Krawczyk, 2007]: Prelude Correlator, OSSIM, RSA enVision, Novell Sentinel, entre outros. O correlacionamento de informações é um mecanismo eficiente para fornecer uma visão de ataques que não poderiam ser detectados por IDSs isolados. Mas poderá se tornar lento e ineficiente quando não é otimizado para tratar um grande número de informações.

2.8 Considerações

Neste capítulo foram tratados assuntos sobre IDSs de uma maneira geral. Durante a pesquisa vários pontos foram observados com relação aos trabalhos realizados nessa área e são relatados a seguir:

- Falsos positivos: este é o ponto mais observado nos estudos de IDSs, pois em qualquer método quanto menos frequentes forem os falsos positivos, maior é a possibilidade de ocorrerem alertas legítimos. Quando se trata de um IDS comercial, adaptar o IDS ao ambiente de uma empresa não é uma tarefa fácil, pois o administrador tem que tratar de muitos falsos positivos para que o IDS atinja seu objetivo.

- Falsos negativos: mesmo que um IDS esteja adaptado para evitar alertas de falsos positivos, ainda resta evitar os falsos negativos que são ataques legítimos e não identificados pelo IDS. Esse é um ponto crítico de um IDS, pois o seu objetivo principal é detectar uma intrusão e acaba não fazendo. Talvez por causa de seu método estar obsoleto ou por não estar preparado para novos tipos de ataques. Um IDS bem configurado é capaz de evitar muitos falsos negativos.

- Consumo de recursos: também é possível observar a preocupação dos autores com o consumo de recursos, tanto de rede como de *hardware*. A implementação errada de um IDS pode resultar em gargalos e até a negação de serviços. A topologia também é um

ponto discutido, principalmente em IDSs que comunicam eventos e alertas através da rede.

- Alertas em tempo real: o tempo dispensado para detectar uma intrusão e gerar um alerta é um fator importante para que o operador possa reagir a um ataque rapidamente (resposta a incidente).

Este capítulo também permitiu situar em que contexto se encontra a arquitetura proposta neste trabalho: a) a coleta de requisições HTTP é feita através da rede, portanto é uma proposta classificada como NIDS; b) a estratégia de análise é baseada em anomalias; c) os algoritmos de agrupamento de dados são utilizados como técnica de análise baseada em aprendizagem de máquina, e d) a arquitetura não é distribuída.

Capítulo 3

Trabalhos Relacionados

Neste capítulo são apresentados os trabalhos que estão relacionados com a arquitetura proposta. A seção 3.1 trata dos IDSs baseados em anomalia que implementam algoritmos de agrupamento de dados, e na seção 3.2 são apresentados diversos trabalhos de IDSs baseados em anomalia que tratam de ataques aos servidores *web* e ao protocolo HTTP. Ao final, na seção 3.3, são feitas as considerações sobre a pesquisa realizada neste capítulo.

3.1 IDSs baseados em Anomalia que implementam Agrupamento de Dados

Os algoritmos de agrupamento de dados (*data clustering*) são utilizados em IDSs baseados em anomalia para extraírem um modelo de comportamento do ambiente a ser monitorado. Esse modelo é construído através do agrupamento de informações similares e que acabam formando grupos (*clusters*). Então esses grupos são utilizados no ambiente de produção para auxiliar na identificação de intrusões. Nesta seção são apresentados os trabalhos relacionados com foco no modo como os agrupamentos são feitos e como os grupos são utilizados na detecção de intrusão.

a) ***Intrusion Detection with Unlabeled Data Using Clustering*** (Detecção de Intrusão com Dados Não-rotulados utilizando Agrupamento) [Portnoy et al., 2001]: é um artigo onde se propõe um algoritmo de detecção de anomalias para encontrar intrusões sobre dados não rotulados, ou seja, onde a aprendizagem e a detecção são feitas no modo não-supervisionado. Os testes foram efetuados sobre a base [KDD, 1999] que foi divida em dez partes, onde quatro partes foram selecionadas e alternadas para treinamento e detecção de intrusão. Foi utilizada uma variação do algoritmo de agrupamento de ligação simples (*single-linkage*) e os grupos foram formados tomando como base a distância euclidiana entre as conexões TCP extraídas dessa base. **Assim os rótulos foram atribuídos após o agrupamento e, dado um valor de percentual N, os grupos com maior número de instâncias representavam atividades normais e os grupos com menor número de instâncias representavam ataques.** No momento de aplicar a detecção, as conexões TCP foram comparadas com os grupos rotulados, e se o grupo mais próximo estava rotulado como anômalo, a conexão seria classificada como ataque. Esse trabalho se destacou entre os demais de sua época por conseguir bons resultados com treinamento não-supervisionado.

b) ***Data Mining for Network Intrusion Detection*** (Mineração de Dados para Detecção de Intrusão em Rede) [Dokas et al., 2002]: nesse trabalho os autores fizeram testes de detecção de intrusão utilizando algumas técnicas de mineração de dados sobre a base [KDD, 1999]. Em três dessas técnicas foram utilizados algoritmos de agrupamento para a detecção de *outliers*. Essas técnicas são similares entre si, onde basicamente se estabelece um limiar para determinar as atividades de rede que estão dentro de um grupo. **Assim os grupos mais densos são classificados como normais e, durante a detecção de intrusão, os *outliers* são classificados como ataques.** As técnicas utilizadas foram: agrupamento por vizinho mais próximo (*nearest neighbor clustering*); agrupamento utilizando a medida de Mahalanobis, e o agrupamento com LOF (*Local Outlier Factor*) que é a técnica mais promissora segundo os autores.

c) ***A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data*** (Uma Estrutura Geométrica para a Detecção de Anomalias no modo Não-supervisionado: Detectando Intrusões em Dados Não-rotulados) [Eskin et al., 2002]: onde se propõe um *framework* para detectar ataques onde os dados não-rotulados são representados em um espaço de características (*feature space*). Assim os dados são representados geometricamente como pontos em um espaço multidimensional. Então os pontos que estão em regiões mais densas são considerados atividades normais. Já os pontos que estão mais distantes (*outliers*) são rotulados como ataques. Nesse trabalho os testes são realizados com algoritmos de agrupamento e de classificação sobre a base [KDD, 1999].

d) ***Y-Means: A Clustering Method for Intrusion Detection*** (Método de Agrupamento para Detecção de Intrusão) [Guan et al., 2003]: nesse trabalho é proposto um método heurístico chamado *Y-Means* (baseado no *K-Means*) para ser utilizado na detecção de intrusão. O algoritmo *K-Means* (definição detalhada na seção 4.1.2) inicia com um número K de centróides[1] pré-definidos, e a cada iteração os elementos são agrupados e um novo centróide é calculado. Não obstante, o algoritmo *Y-Means* também faz o agrupamento de informações, mas além disso faz a divisão e mesclagem de grupos de acordo com um limiar pré-definido. Assim, a cada grupo formado se estabelecem as áreas confiantes (*confident area*) que são utilizadas na detecção de intrusão para separar as atividades normais dos *outliers*. Os *outliers*, por sua vez, são classificados como ataques. Ao final, esse algoritmo obteve bons resultados com testes realizados sobre a base [KDD, 1999].

e) ***A Machine Learning Approach to Anomaly Detection*** (Um Abordagem de Aprendizagem de Máquina para Detecção de Anomalia) [Mahoney et al., 2003]: nesse trabalho são apresentados dois métodos para detecção de anomalia, um utilizando aprendizagem de regras e outro utilizando agrupamento. Os autores apresentam um algoritmo chamado CLAD *(Clustering for Anomaly Detection)* onde o treinamento é realizado sobre dados não rotulados. E, para detectar intrusões, os *outliers* são determinados de acordo com a densidade e a distância dos outros grupos.

f) **MINDS - *Minnesota Intrusion Detecion System*** (Sistema de Detecção de Intrusão de Minnesota) [Ertöz et al., 2004]: nesse trabalho os autores descrevem como desenvolveram um IDS baseado em anomalia para inspecionar fluxos de rede. A parte da arquitetura do MINDS

[1] Centróide (*centroid*) é o elemento central ou o centro de referência de um grupo.

que faz a inspeção baseada em anomalia implementa o agrupamento com LOF, onde os *outliers* são classificados como anomalia. Então a tarefa de identificar um ataque com base na anomalia ficava a cargo de um analista humano. Ao final, os testes realizados com o MINDS se mostraram satisfatórios detectando mais ataques em comparação ao IDS Snort.

g) ***Unsupervised Anomaly Detection in Network Intrusion Detection Using Clusters*** (Detecção de Anomalia no modo Não-supervisionado em Detecção de Intrusão em Rede usando Grupos) [Leung and Leckie, 2005]: nesse trabalho os autores propõem o algoritmo de agrupamento fpMAFIA (*frequent-pattern*) para ser utilizado na detecção de intrusão. Esse algoritmo faz o agrupamento baseado em grades e densidade de maneira similar ao algoritmo de agrupamento pMAFIA. Durante o agrupamento os dados são representados como pontos em um espaço bidimensional. Assim os pontos que estão nos grupos criados são rotulados como normais, e um pequeno percentual dos pontos que não se encaixam nesses grupos são rotulados como anormais (ataques). Os testes foram realizados sobre a base [KDD, 1999] e o fpMAFIA obteve melhores taxas de detecção de intrusão na comparação com os outros algoritmos.

h) ***Distributed Intrusion Detection based on Clustering*** (Detecção de Intrusão Distribuída baseada em Agrupamento) [Zhang et al., 2005]: nesse trabalho é proposto um IDS distribuído, onde um algoritmo de agrupamento é executado nos agentes que coletam informações da rede. Então os agentes selecionam os grupos menores e classificam como anomalias. Assim o IDS central também aplica um algoritmo de agrupamento sobre essas anomalias e tenta identificar aquelas que realmente representam ataques. Os autores fizeram testes sobre a base [KDD, 1999] e concluíram que esse é um IDS viável.

i) ***A Comparison Between the Silhouette Index and the Davies-Boudin Index in Labelling IDS Clusters*** (Uma Comparação entre os Índices de Silhouette e Davies-Bouldin em Rotulamento de Grupos em IDS) [Petrović, 2006]: nesse trabalho o autor faz a comparação do uso dos índices de Silhouette e Davies-Bouldin para atribuir rótulos após o agrupamento de informações. Esses algoritmos são normalmente utilizados para atribuir um índice de qualidade após a realização de um agrupamento. Por exemplo, grupos internamente densos e distantes entre si podem representar um bom agrupamento. Mas nesse trabalho a idéia principal é detectar os ataques em massa, por exemplo: negação de serviço (DoS). Assim, os grupos que são identificados como muito densos por esses índices, são rotulados como ataques. Os testes foram realizados sobre a base [KDD, 1999] onde os testes com o índice de Silhouette foram satisfatórios mas não obtiveram melhor performance do que o índice de Davies-Bouldin.

j) ***Clustering-based Network Intrusion Detection*** (Detecção de Intrusão em Rede baseada em Agrupamento) [Zhong et al., 2007]: nesse trabalho o método de treinamento e o método de detecção de intrusão são semelhantes aos apresentados no trabalho de [Portnoy et al., 2001] (item "a" desta seção). De acordo com os autores, o treinamento é realizado no modo não-supervisionado utilizando algoritmos de agrupamento. Após o agrupamento das informações, o grupo com maior número de instâncias é selecionado e rotulado como normal, então esse grupo será utilizado como o centróide principal. Assim os demais grupos e instâncias serão ordenados de acordo com a sua distância ao centróide principal. Então de acordo com um percentual η, as instâncias próximas a esse centróide serão rotuladas como normais, e as restantes receberão o rótulo de ataque. Nesse trabalho os autores realizaram testes sobre a

base [DARPA, 1998] utilizando os algoritmos de agrupamento *K-Means*, *Online K-Means*, SOM (*Self-Organizing Maps*), Neural-Gas e MOSG (*Mixture-Of-Spherical Gaussians*), este com o algoritmo EM (*Expectation-Maximization*). Nos testes também foi utilizado o algoritmo de classificação SVM (*Support Vector Machines*), mas as melhores taxas de detecção foram obtidas com o algoritmo *Online K-Means*.

k) ***Labelling Clusters in an Anomaly based IDS by means of Clustering Quality Indexes*** (Rotulando Grupos em IDS baseado em Anomalia através de Índices de Qualidade de Agrupamento) [Storløkken, 2007]: nesse trabalho, semelhante ao de [Petrović, 2006], o autor considera que os grupos compactos representam ataques em massa. Assim, após aplicar o algoritmo de agrupamento foram utilizados os índices de Dunn e C-Index para identificar e rotular os grupos mais densos como ataques. Nesse trabalho foi utilizada a base [KDD, 1999] para a detecção de ataques de negação de serviço (DoS).

l) ***Mining Common Outliers for Intrusion Detection*** (Minerando *Outliers* Comuns para Detecção de Intrusão) [Singh et al., 2009]: propõe o algoritmo COD (*Common Outlier Detection*) onde uma intrusão é detectada quando uma requisição *outlier* ocorre em um sistema S_1 e o mesmo *outlier* ocorre em um sistema diferente S_2. O IDS foi testado em uma base de registros de acessos *web*.

Nesta seção foram apresentados diversos trabalhos relacionados com a arquitetura proposta neste trabalho e um resumo do que foi pesquisado pode ser visualizado na tabela 3.1. Nessa tabela, de maneira resumida, a segunda coluna diz como o trabalho relacionado identifica os ataques durante o agrupamento, e na terceira coluna diz como o agrupamento é avaliado para identificar ataques.

Tabela 3.1: Comparação entre os trabalhos relacionados de agrupamento de dados

Trabalho	**Durante o agrupamento**	**Após o agrupamento**
[Portnoy et al., 2001]	-	grupos menores são rotulados como ataques
[Dokas et al., 2002]	*outliers* são ataques	-
[Eskin et al., 2002]	*outliers* são ataques	-
[Guan et al., 2003]	*outliers* são ataques	-
[Mahoney et al., 2003]	*outliers* são ataques	-
[Ertöz et al., 2004]	*outliers* são ataques	-
[Leung and Leckie, 2005]	*outliers* são ataques	-
[Zhang et al., 2005]	agente detecta *outliers*	central analisa *outliers* buscando ataques
[Petrović, 2006]	grupos compactos	índices auxiliam na detecção de ataques em massa
[Zhong et al., 2007]	-	ataques estão distantes do centróide principal
[Storløkken, 2007]	grupos compactos	índices auxiliam na detecção de ataques em massa
[Singh et al., 2009]	detecção de *outliers*	*outliers* comuns em sistemas diferentes são ataques
Arquitetura Proposta	-	grupos com baixa reputação podem ser ataques

3.2 IDSs baseados em Anomalia para Serviços *Web* e Protocolo HTTP

Um meio de identificar intrusões em serviço *web* é através da monitoração das mensagens HTTP (*HyperText Transfer Protocol*). Nesta seção são apresentados diversos trabalhos de IDSs baseados em anomalia que tratam desse tipo de mensagem para a detecção de ataques.

a) ***Anomaly Detection of Web-based Attacks*** (Detecção de Anomalia baseada em Ataques *Web*) [Kruegel and Vigna, 2003]: nesse trabalho a detecção de ataques é feita através do URI (*Uniform Resource Identifier*). O URI é coletado das requisições (com o método GET) e segmentado em: caminho de recurso e seus parâmetros. Com essas informações um treinamento é realizado para criar perfis e valores de limiares (*threshold*). Então esses perfis são utilizados no momento da detecção de intrusão onde uma pontuação de anomalia é calculada para decidir se uma requisição é normal ou intrusiva. Diversos modelos de detecção foram aplicados: tamanho do atributo, distribuição dos caracteres, inferência estrutural (NFA), localizador de *token*, presença ou ausência de atributos e ordenação dos atributos. Testes foram realizados com três bases de registros de acessos coletados de servidores Apache. Através de experimentos a proposta dos autores foi capaz de detectar ataques de *buffer overflow*, mudança de diretório, XSS, validação de entrada e *code red*. Mais tarde o mesmo trabalho [Kruegel et al., 2005] foi apresentado com alguns complementos e com base nesse modelo uma arquitetura foi proposta por [Robertson et al., 2006].

b) ***Learning DFA Representations of HTTP for Protecting Web Applications*** (Aprendizagem de Representações DFA em HTTP para Proteção de Aplicações *Web*) [Ingham et al., 2007]: nesse trabalho é proposto um método de detecção por anomalia onde as requisições normais de HTTP fazem parte da aprendizagem utilizando indução DFA (*Deterministic Finite Automata*). Esse modelo DFA é criado a partir de *tokens* extraídos de requisições HTTP, os quais são verificados a cada requisição para calcular o quanto esta pode ser uma anomalia (ataque). Além disso também são tratados assuntos de compactação e generalização de modelos DFAs; aprendizagem com dados que se modificam com o tempo, e ainda apresenta heurísticas para a redução de falsos positivos. Mais tarde um *framework* para testes desse tipo de IDS é proposto na tese de [Ingham, 2007] e, entre os algoritmos testados nesse *framework*, os que são baseados em *tokens* (DFA e *n-grams*) obtiveram melhores resultados [Ingham and Inoue, 2007].

c) ***Boosting Web Intrusion Detection Systems by Inferring Positive Signatures*** (Impulsionando Sistemas de Detecção de Intrusão *Web* pela Inferência de Assinaturas Positivas) [Bolzoni and Etalle, 2008]: nesse trabalho os autores apresentam o Sphinx, um tipo de IDS baseado em anomalia que gera automaticamente um modelo de assinaturas positivas a partir de treinamento efetuado com requisições HTTP livres de ataques (aprendizagem supervisionada). As assinaturas são formadas por expressões regulares que podem ser interpretadas e modificadas pelo operador do IDS. Ao se utilizar esse modelo na detecção de intrusão, as requisições que não forem regulares poderão ser consideradas como ataques.

d) ***URI Anomaly Detection using Similarity Metrics*** (Detecção de Anomalia em URI usando Métricas de Similaridade) [Yahalom, 2008]: nesse trabalho o autor propõe dois algoritmos de detecção: o primeiro faz comparação dos URIs e aplica uma pontuação para determinar

se é anômalo ou não. O segundo algoritmo aplica agrupamento de dados para treinamento, e depois utiliza esses grupos para fazer a detecção de anomalia. Para fazer as comparações de similaridade entre URIs foram testados dois tipos de medidas de similaridade: NCD *(Normalized Compression Distance)* e *n-grams*.

e) ***Integrated Detection of Attacks Against Browsers, Web Applications and Databases*** (Detecção Integrada de Ataques contra Navegadores, Aplicações *Web* e Banco de Dados) [Criscione et al., 2009]: nesse trabalho os autores apresentam a ferramenta Masibty para a detecção de ataques *web* baseado em anomalias, onde o treinamento é aplicado no modo não-supervisionado. A proposta é de um sistema de prevenção de ataques tanto do lado do cliente como no lado do servidor atuando como *proxy* reverso. As informações são extraídas das requisições e respostas HTTP que consistem em: URI, parâmetros de consulta e contexto da sessão (sequência, *cookies*, identificadores). Um algoritmo de agrupamento é utilizado sobre URLs no momento do treinamento (ou no momento da detecção) e os *outliers* são descartados como possíveis ataques. Esse IDS também implementa mecanismos probabilísticos onde uma pontuação de anomalia é atribuída para cada ação, e a combinação dessas pontuações pode levar ao IDS gerar um alerta de ataque. Além disso, diversos mecanismos são implementados para detectar tipos específicos de ataques como SQL *Injection* e XSS.

f) ***Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic*** (*Spectrogram*: Um Modelo de Mistura de Cadeias de Markov para a Detecção de Anomalia em Tráfego *Web*) [Song et al., 2009]: nesse trabalho os autores apresentam um IDS baseado anomalia para a detecção de ataques de injeção de código, onde o modo de treinamento é supervisionado. Para detectar ataques, são analisados os parâmetros que são passados via método GET e via método POST. E, tanto para o treinamento como para a detecção de intrusão, a técnica de análise empregada utiliza modelos de cadeias de Markov e *n-grams*. O IDS foi testado e obteve bons resultados na detecção de ataques de XSS, RFI e SQL *Injection*.

g) ***HMM-Web: a framework for the detection of attacks against Web applications*** (HMM-Web: um *framework* para a detecção de ataques contra aplicações *Web*) [Corona et al., 2009]: Nesse trabalho os autores propõem um *framework* para um IDS baseado em anomalia que implementa modelos ocultos de Markov (HMM) como técnica de análise. O IDS consiste em mecanismos de HMM para tratar cada atributo enviado através de consultas URI. Assim o módulo de decisão avalia, através de cálculos de probabilidade, se uma consulta enviada para uma aplicação deve ser tratada como ataque ou não. De acordo com os autores, a solução proposta foi eficiente tanto na fase de treinamento como na fase de detecção de ataques de SQL *Injection* e XSS. Mais tarde, esse método também fez parte do trabalho de [Ariu, 2010]. E no trabalho de [Corona, 2010] o autor apresenta o *Web Guardian* como uma evolução do HMM-Web, onde um módulo acoplado ao WAF (*ModSecurity*) do Apache permite uma inspeção mais avançada de HTTP, HTTPS e requisições com o método POST.

h) ***Protecting a Moving Target: Addressing Web Application Concept Drift*** (Protegendo um Alvo em Movimento: Endereçando o Conceito de Movimento em Aplicação *Web*) [Maggi et al., 2009]: esse trabalho leva em conta que as aplicações *web* não são estáticas, e que as mudanças que ocorrem são responsáveis pelo aumento de falsos positivos nos IDSs

que monitoram essas aplicações. Então um método foi proposto para identificar as mudanças de um sítio *web* a partir das respostas dos servidores HTTP, ou seja, as URLs fornecidas por uma resposta HTTP podem ser mais confiáveis nas próximas requisições. Outro exemplo seria a captura e a análise dos formulários HTML, onde os campos servem como informação confiável na requisição de POST que seguirá. Em testes com a ferramenta *webanomaly* o número de alertas foi reduzido significativamente quando foi aplicado o conceito de movimento (*drift*). Esse método também faz parte dos trabalhos de [Robertson, 2009] e [Maggi, 2009].

i) ***HMMPayl: an application of HMM to the analysis of the HTTP Payload*** (HMMPayl: uma aplicação de HMM para análise da carga útil do HTTP) [Ariu and Giacinto, 2010]: nesse trabalho é apresentado um IDS baseado em anomalia que implementa modelos ocultos de Markov (HMM) para a análise e detecção de intrusão sobre sequência de *bytes* da carga útil do HTTP. Permitindo, assim, a detecção de diversos ataques que são direcionados para a aplicação *web* como também para o servidor HTTP. Esse método também faz parte do trabalho de [Ariu, 2010].

j) ***Effective Anomaly Detection with Scarce Training Data*** (Detecção de Anomalia com Escassez de Dados para Treinamento) [Robertson et al., 2010]: nesse trabalho é proposto um método para detecção de anomalia mesmo com escassez de dados, esse tipo de escassez ocorre por causa de requisições que são legítimas mas que aparecem com menos frequência ou com menos informações. Para solucionar esse problema, os autores propõem um método de treinamento que resulta em um modelo com perfis estáveis e aprendizagem contínua. Para criar esses perfis são implementadas as técnicas de análise com agrupamento de dados (*clustering*) e modelos ocultos de Markov (HMM). Ao realizar os testes, os autores concluíram que quanto mais estáveis são os perfis, menor é a taxa de falsos positivos. Esse método também faz parte dos trabalhos de [Robertson, 2009] e [Maggi, 2009].

k) ***Detection of Server-side Web Attacks*** (Detecção de Ataques *Web* lado do Servidor) [Corona and Giacinto, 2010]: nesse trabalho os autores apresentam uma arquitetura de IDS onde cada atributo do cabeçalho de uma requisição HTTP é tratado separadamente e analisado por um sensor de anomalia. Assim cada sensor é composto por um módulo de decisão estatística onde um atributo de uma requisição é avaliada como anômala ou não. Então, se ao menos um atributo for marcado como anômalo, a requisição será considerada um ataque. Para auxiliar na detecção em tempo real, os modelos de detecção são criados a partir de treinamentos que implementam algoritmo de agrupamento de ligação simples (*single linkage*), modelos estatísticos e modelos ocultos de Markov (HMM).

Os trabalhos que implementam detecção de intrusão baseada em anomalia para servidores *web* diferem entre si pois apresentam arquiteturas diferentes e que muitas vezes combinam diversas técnicas de análise para a detecção de ataques. Para simplificar essa visão, os trabalhos relacionados e a arquitetura proposta são apresentados na tabela 3.2.

Tabela 3.2: Comparação entre os trabalhos relacionados de detecção de ataques *web*

Trabalho	**Dados para análise**	**Técnicas de análise**
[Kruegel and Vigna, 2003]	parâmetros de consulta do URI (GET)	diversos modelos probabilísticos
[Ingham et al., 2007]	requisições são segmentadas em *tokens*	indução DFA
[Bolzoni and Etalle, 2008]	parâmetros das requisições	expressões regulares positivas
[Yahalom, 2008]	URI das requisições	agrupamento e medidas de similaridade
[Criscione et al., 2009]	requisições e respostas (sessão)	agrupamento e modelos probabilísticos
[Song et al., 2009]	requisições (GET/POST)	mistura de cadeias de Markov
[Corona et al., 2009]	parâmetros das requisições (GET)	modelos ocultos de Markov
[Maggi et al., 2009]	requisições e respostas (fonte HTML)	mesmas de [Kruegel and Vigna, 2003]
[Ariu and Giacinto, 2010]	*bytes* da carga útil	modelos ocultos de Markov
[Robertson et al., 2010]	parâmetros das requisições	agrupamento e HMM
[Corona and Giacinto, 2010]	parâmetros das requisições	agrupamento, estatística e HMM
Arquitetura Proposta	origem e parâmetros das requisições	agrupamento de dados

3.3 Considerações

Este capítulo focou os IDS baseados em anomalia, e durante a pesquisa foi possível observar alguns pontos em relação a esses tipos de IDS:

- Normalização dos dados: se os dados não são normalizados poderão não somente ocasionar erros nos modelos de detecção mas também aumentar o tempo computacional no momento do treinamento.

- Supervisão do aprendizado: seria desejável que os métodos pudessem aprender automaticamente o comportamento dos sistemas de maneira não-supervisionada *(unsupervised)*, alguns trabalhos propõe esse método. Porém, se houver uma base de treinamento que contenha ataques, poderá aumentar a probabilidade de que estes sejam detectados como se fossem atividades normais. Logo, os métodos que são treinados com dados rotulados, ou melhor, de maneira supervisionada, apresentam melhores resultados.

- Momento do aprendizado: existe uma preocupação com o momento em que o treinamento ocorre, pois se for efetuado em uma base *offline*, os modelos criados para detecção poderão se tornar obsoletos com o passar do tempo. Logo, se o treinamento for aplicado no modo *online*, o método se torna mais complexo e tem como complemento os mesmos problemas da aprendizagem não-supervisionada.

- Escassez de dados para aprendizado: outra preocupação está em definir a quantidade de dados necessários para efetuar um bom treinamento. Poucos dados no momento do treinamento permitem que atividades legítimas e raras sejam classificadas como anomalia, resultando em falsos positivos.

- Taxa de detecção: os métodos baseados em anomalia apresentam a vantagem de obter não somente boas taxas de detecção mas também de serem eficientes para identificar novos tipos de intrusões.

- Falsos positivos: este é um ponto bastante discutido nesta área e é considerado uma das desvantagens deste modelo de detecção baseado em anomalia, pois quando esses métodos

são aplicados em ambientes em que há muitas atividades legítimas que são novas ou diferentes, acabam gerando um grande número de alarmes de falsos positivos.

Uma breve comparação entre os trabalhos relacionados e a arquitetura proposta pode ser visualizada nas tabelas 3.1 e 3.2. E, nos itens seguintes, alguns pontos podem ser observados em relação à contribuição deste trabalho:

- Um dos últimos trabalhos comparados com o Snort é o MINDS de [Ertöz et al., 2004] na seção 3.1. Atualmente o Snort é capaz de detectar diversos tipos de ataques e continua sendo utilizado tanto na comunidade de *software livre* como no mercado. A colaboração desta dissertação está em realizar uma comparação de um IDS baseado em anomalia com o Snort em um ambiente real e constituído de ataques *web*.

- Os trabalhos de [Portnoy et al., 2001] e [Zhong et al., 2007] (na seção 3.1) fazem uma avaliação do agrupamento para rotular os grupos como normais ou ataques após o agrupamento. Os demais trabalhos diferem desses pois o algoritmo de agrupamento pode determinar em tempo de execução o que é um ataque (*outlier*) ou não. Uma colaboração da arquitetura proposta está em atribuir rótulos de reputação para os grupos após o agrupamento.

- Os demais trabalhos (na seção 3.2) que tratam de detecção de ataques *web* diferem desta proposta principalmente porque esta utiliza os endereços IPs (os que originaram as requisições HTTP) como parâmetro no momento do treinamento do IDS. No módulo de treinamento, os índices de popularidade e confiabilidade são calculados de acordo com as informações das requisições juntamente com os endereços IPs. Sendo assim, a atribuição de rótulos de reputação aos grupos com base nesses índices permitem que o modelo de detecção de intrusão gere baixas taxas de falsos positivos.

Capítulo 4

Arquitetura Proposta

O objetivo principal desta arquitetura é construir um IDS baseado em anomalia que com agrupamento de dados possa efetuar a detecção de intrusão em serviços *web* e HTTP. Em geral, os IDSs baseados em anomalia precisam de duas etapas: uma para treinamento e outra para detectar intrusão com os modelos criados durante o treinamento. Da mesma maneira, a arquitetura desta proposta foi dividida em duas partes principais: uma para treinamento não-supervisionado na seção 4.1 e outra para detecção de intrusão na seção 4.2.

As requisições HTTP são os dados de entrada na **arquitetura para treinamento**, onde oito campos são extraídos de seus cabeçalhos e são separados (seção 4.1.1) e colocados em listas para que se possa efetuar o agrupamento (seção 4.1.2). Antes que o resultado do agrupamento seja utilizado como modelo de detecção de intrusão, os grupos resultantes devem passar por uma avaliação e receber rótulos de reputação (seção 4.1.3). É nesse módulo de avaliação dos grupos que os endereços IPs e suas requisições servem como parâmetros para calcular os índices de popularidade e de confiabilidade de cada grupo. Assim a junção ponderada desses índices permitirá que cada grupo receba um rótulo de reputação, rótulo qual servirá como parâmetro na tomada de decisão na arquitetura de detecção de intrusão.

Na **arquitetura para detecção de intrusão** as requisições HTTP são coletadas e uma inspeção é aplicada para verificar se estão incompletas ou se contém alguma violação de protocolo (seção 4.2.1). Após a primeira inspeção, cada requisição é tratada separadamente onde oito campos do cabeçalho HTTP são separados e comparados um a um com a lista de centróides gerada anteriormente. Se o valor de um campo corresponder com uma distância próxima de um centróide dado um limiar pré-definido, então este campo receberá a mesma reputação do grupo deste centróide (ou assinatura). Caso contrário, o campo receberá a pior reputação. Com a correlação da reputação de cada campo, o IDS determinará se a requisição HTTP é normal, suspeita ou intrusiva (seção 4.2.2).

Ao final deste capítulo, na seção 4.3 são feitas algumas considerações em relação à arquitetura proposta.

4.1 Arquitetura para Treinamento

Nesta arquitetura, o treinamento é realizado no modo não-supervisionado *(unsupervised learning)*, onde um número limitado de requisições são coletadas sem a necessidade de estarem rotuladas como normais ou ataques. Porém, é importante que o treinamento seja feito em uma base onde a maioria das requisições devem representar o acesso normal. Espera-se,

pois, que o módulo de avaliação dos grupos consiga separar acessos legítimos de acessos suspeitos já no momento do treinamento (seção 4.1.3). A arquitetura proposta para treinamento têm três módulos: tratamento das requisições, agrupamento de dados, avaliação dos grupos, e é ilustrada na figura 4.1.

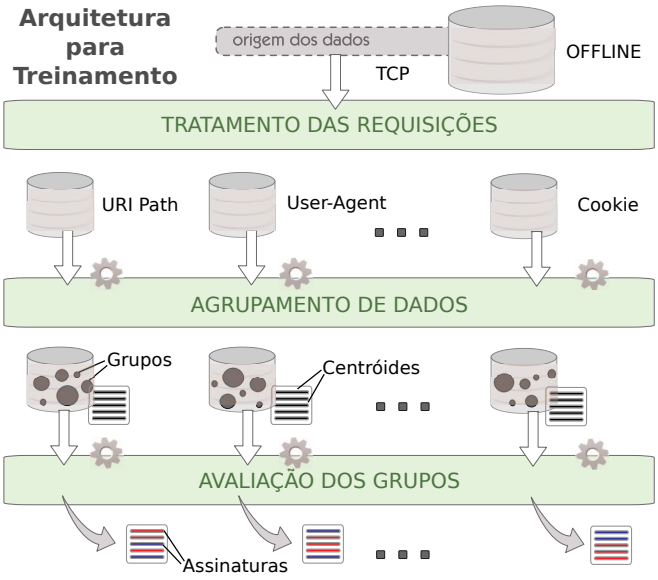

Figura 4.1: Ilustração - Arquitetura para treinamento

4.1.1 Módulo de Tratamento das Requisições

Conforme a figura 4.1, os segmentos TCPs são coletados de uma base (ex.: arquivo PCAP) que foi gerada no ambiente que se deseja monitorar. Esses segmentos carregam partes de requisições HTTP que deverão ser remontadas ou, ainda, requisições HTTP agrupadas que deverão ser divididas. Inspecionar as requisições incompletas ou que violam regras do protocolo também são funções do módulo de inspeção de requisições, portanto esses detalhes foram descritos na seção 4.2.1.

O protocolo HTTP *(HyperText Transfer Protocol)* está definido no documento RFC 2616 [Fielding et al., 1999]. As requisições HTTP são criadas do lado do cliente, que na maioria das vezes utiliza um navegador *web (browser)*. As requisições HTTP são enviadas para um servidor *web* (ex.: Apache) solicitando o acesso a um recurso específico (ex: index.html). Na figura 4.2 é apresentado um exemplo de uma requisição HTTP com o método mais comum a ser utilizado para acessos *web*, o método GET. A primeira linha do cabeçalho HTTP é composta pelo método (1), localização do recurso (2) e a versão do protocolo (3). Nas linhas subsequentes, todas separadas com os caracteres especiais CR/LF (\r\n), acompanham diversos atributos referentes a solicitação do recurso. Já na figura 4.3 é apresentado um exemplo de uma requisição utilizando o método POST que, além de conter o cabeçalho, também contém o corpo com os dados enviados pelo navegador.

Ao juntar as informações que estão nas figuras 4.2 e 4.3 é possível observar que foram destacados oito campos das requisições HTTP que são coletados por este módulo. E são os seguintes: (1) método *(Method)*, (2) caminho do recurso *(UPath)*, (3) consulta ao recurso *(UQuery)*, (4) nome de domínio ou endereço IP do sítio *web (Host)*, (5) agente do usuário

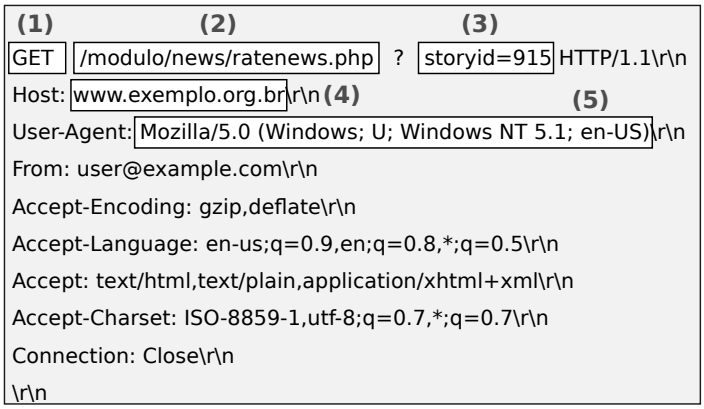

Figura 4.2: Exemplo - Requisição HTTP (GET)

(User-Agent), (6) caminho da referência *(RPath)*, (7) *Cookie* e (8) o conteúdo *(Content)*. Esses campos foram escolhidos por terem relação com os ataques identificados e rotulados na base de requisições utilizada neste trabalho.

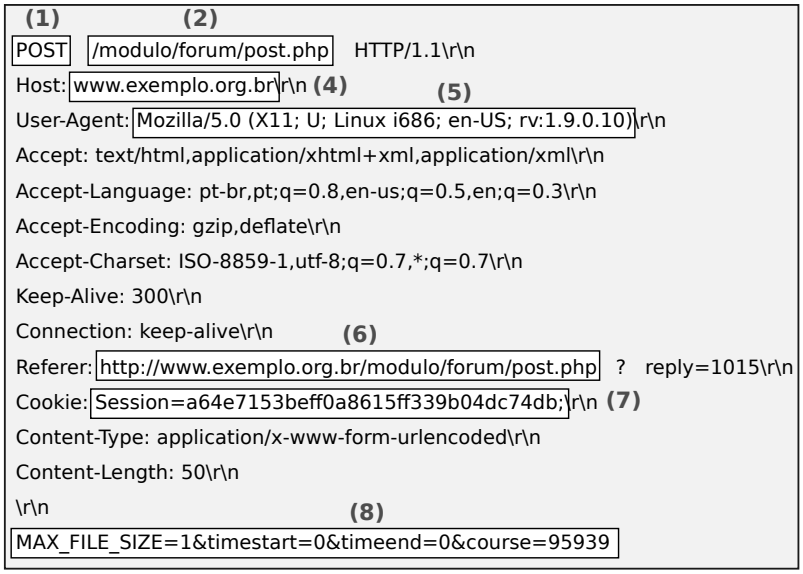

Figura 4.3: Exemplo - Requisição HTTP (POST)

É importante observar que este módulo irá preparar as informações antes de enviar para o módulo de agrupamento, e o formato dessas informações podem onerar a execução do algoritmo de agrupamento. Um exemplo é o campo *Content* que pode carregar grandes arquivos de textos ou figuras. A solução, no entanto, é utilizar medidas de distância numérica ou implementar algum algoritmo de compressão ou normalização. Durante os testes neste trabalho, o campo *Content* foi normalizado apenas substituindo caracteres especiais por caracteres legíveis.

Ao todo são criadas oito listas (ilustração na figura 4.4), uma para cada campo, preenchidas com seus valores de acordo com um número limitado de requisições HTTP. Em cada linha dessas listas é adicionado o endereço IP do *host* que construiu a requisição, pois com base nesses endereços IPs será possível calcular a popularidade dos grupos que serão criados conforme a seção 4.1.3 .

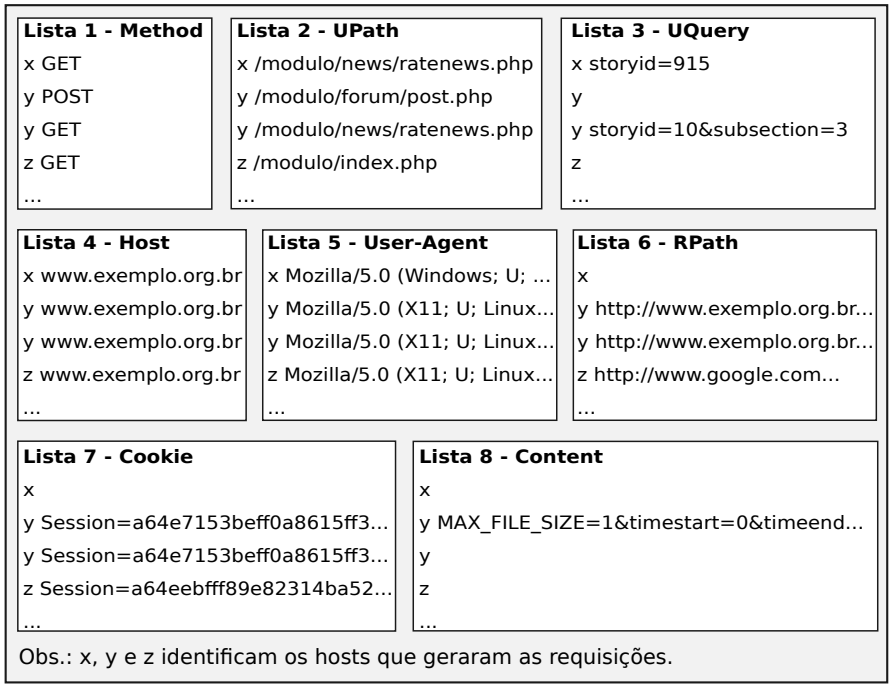

Figura 4.4: Exemplo - Requisições HTTP em listas

4.1.2 Módulo de Agrupamento de Dados

Neste módulo, cada uma das listas criadas conforme a seção anterior serão utilizadas separadamente para extrair grupos. Para realizar tal tarefa é necessário fazer a seleção: do algoritmo de agrupamento, da medida de similaridade e do índice de validação do agrupamento.

a) Algoritmo de Agrupamento

Para fazer o agrupamento de dados (*data clustering*) qualquer algoritmo de agrupamento pode ser utilizado se atender ao seguinte pré-requisito: o algoritmo deve estabelecer áreas limítrofes para os grupos ou ao menos fornecer informações sobre os elementos centrais (conhecidos como centróides). Isso é necessário pois esses valores de limiares ou centróides serão reutilizados como assinaturas no momento da detecção de intrusão. Para esse fim poderão ser encontrados diversos algoritmos de agrupamento que foram reunidos e detalhados em livros tais como o de [Tan et al., 2005] e o de [Gan et al., 2007]. Neste trabalho o algoritmo *K-Means*[1] foi selecionado porque tem a vantagem de ser simples e, além de ser popular na comunidade científica, converge em poucas iterações para um resultado estável [Tan et al., 2005].

O *K-Means* faz parte dos algoritmos particionais de agrupamento de dados, onde se estabelece um parâmetro inicial que são os k centróides, parâmetro o qual representa a quantidade de grupos que se deseja como resultado. Cada elemento é associado ao seu centróide mais próximo, formando, assim, os grupos. O centróide é recalculado a cada iteração de acordo com os elementos associados ao seu grupo, podendo ser esse a média ou o elemento central. Esses passos são repetidos até que não haja mais mudanças de centróides ou que não hajam mais

[1] O algoritmo foi modificado para limitar o tamanho dos grupos com o intuito de melhorar a performance na comparação de assinaturas (explicação na primeira etapa da seção 4.2.2).

trocas de elementos entre os grupos [Tan et al., 2005]. A figura 4.5 apresenta uma ilustração do agrupamento *K-Means*.

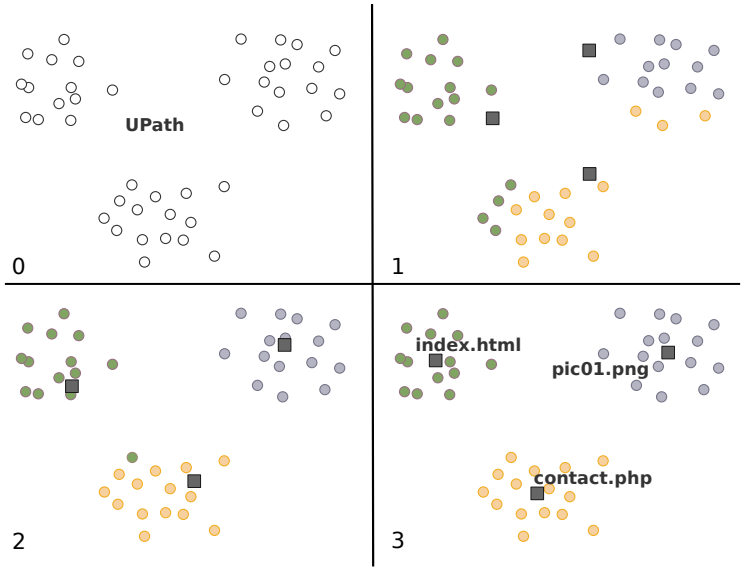

Figura 4.5: Ilustração - Agrupamento *K-Means* em 3 iterações

b) Medidas de Similaridade

Outro ponto a ser levado em consideração neste módulo são as medidas de distâncias a serem utilizadas, pois durante o agrupamento os valores que são similares entre si deverão pertencer ao mesmo grupo. Abaixo algumas medidas que poderão ser utilizadas:

- A medida mais simples a ser utilizada pode ser feita apenas com a quantidade de caracteres ou *bytes* dos campos extraídos das requisições HTTP. Assim é possível fazer a comparação numérica entre esses campos aplicando diversas medidas, tais como: *Manhattan*, *Euclidean*, *Mahalanobis* entre outras que poderão ser encontradas com mais detalhes no livro de [Gan et al., 2007].

- Outra medida que pode ser utilizada é a comparação entre vetores, onde o conteúdo dos campos do tipo texto (*string*) é separado em *tokens* através da técnica conhecida como *n-grams*. Um exemplo do uso dessa técnica seria determinar um valor para *n* igual a 3 e a palavra "acessar" ficaria dividida da seguinte maneira: ace, ces, ess, ssa e sar. A transformação do texto em *tokens* possibilita a comparação entre vetores[2] utilizando cálculos matemáticos, tais como *dice*, *cosine*, *jaccard*, etc.

- Ainda existem outras medidas entre *strings* que podem ser utilizadas, e uma delas é a distância de edição (*edit distance*) onde um cálculo é efetuado simulando a quantidade de caracteres necessários para serem inseridos/removidos/substituídos como se o objetivo fosse tornar duas *strings* iguais. Um exemplo seria calcular a distância entre as palavras

[2]Os vetores são criados de acordo com a disposição dos *tokens* e a quantidade de ocorrências desses dentro de um texto.

"carro" e "cargo", onde a substituição de uma letra resultaria na distância 1 (um) entre essas palavras. Uma medida de distância assim é a de Levenshtein, mas existem outras que aplicam técnicas similares: Monge-Elkan, Smith-Waterman, Jaro-Winkler entre outras.

Neste trabalho, a medida de distância utilizada é calculada pelo algoritmo de Jaro-Winkler, pois esse algoritmo resultou em melhores agrupamentos em testes preliminares que foram realizados durante esta pesquisa. Mais detalhes sobre essas e outras medidas de distância entre *strings* podem ser encontradas com mais detalhes no trabalho de [Cohen et al., 2003]. Nesse mesmo trabalho, os autores fazem comparação entre as medidas e destacam o algoritmo Jaro-Winkler que obteve bons resultados e boa performance nas medidas de distância sobre bases de nomes diversos.

Uma exceção está na medida de similaridade que deve ser aplicada na lista construída com o campo *Method*, visto que esse campo contém valores limitados conforme os documentos RFC 2616 [Fielding et al., 1999] e RFC 4918[3] [Dusseault, 2007]. Sendo assim, para essa lista basta apenas uma comparação simples entre as *strings*: se forem iguais a distância é 0 (zero), mas se forem diferentes a distância é 1 (um).

c) Índice de Validação do Agrupamento

Ao aplicar algoritmos de agrupamento em um conjunto de dados é possível que resulte em apenas um grupo grande e com todos os elementos, como também é possível que cada elemento resulte em um grupo por si só. De uma maneira ou de outra os resultados obtidos por algoritmos de agrupamento podem ser bons ou ruins, e dependem tanto do algoritmo escolhido como do tipo de dado e do resultado que se espera. A escolha de um algoritmo de agrupamento para atingir um resultado específico é considerada uma tarefa difícil, esse é um problema conhecido como o dilema do usuário (*user's dilemma*) que pode ser conferido no artigo de [Jain, 2010]. Para amenizar esse problema, são utilizados os índices de validação do agrupamento (*clustering validity index*) que geralmente consideram o melhor agrupamento aquele que contém grupos densos (*dense*) internamente e distantes uns dos outros (*scatter*). Exemplos de índices que poderão ser utilizados neste módulo são: Davies-Bouldin, Dunn's Index, SD Index, entre outros que podem ser encontrados no livro de [Gan et al., 2007]. Neste trabalho os agrupamentos foram escolhidos de acordo com os índices obtidos com o método de Davies-Bouldin, esse método foi utilizado por ser simples e de execução rápida.

Após realizar o agrupamento, as oito listas de grupos resultantes juntamente com os centróides de cada grupo deverão ser enviados para o módulo de avaliação de grupos conforme a ilustração na figura 4.1.

4.1.3 Módulo de Avaliação dos Grupos

Este módulo faz parte da contribuição científica desta dissertação e o principal trabalho relacionado é o de [Portnoy et al., 2001] (descrito na seção 3.1). Neste módulo o rótulo é atribuído ao grupo de acordo com a sua reputação, diferentemente de [Portnoy et al., 2001] que rotula cada grupo como normal ou ataque. O processo de avaliação de grupos foi dividido em quatro etapas que estão descritas nesta seção juntamente com um exemplo que pode ser visualizado na figura 4.6.

[3] O documento RFC 4918 trata de extensões HTTP para WebDAV e torna obsoleto o documento RFC 2518 [Goland et al., 1999].

Este módulo foi criado a partir de observações experimentais durante a verificação de ataques na base de requisições. Durante essa análise foi possível observar que as informações das requisições que eram mais comuns (ou **populares**) tinham uma chance menor de serem consideradas ataques, e que aquelas que não eram comuns podiam ser consideradas **confiáveis** se o *host* de origem também fosse confiável. Assim as informações poderiam ser agrupadas, e através de cálculos empíricos determinar um índice de popularidade e um índice de confiabilidade para cada grupo. Ao final, esses índices foram combinados para calcular um índice de reputação, e o processo todo resultou em um método heurístico para atribuir rótulos de **reputação** para os grupos.

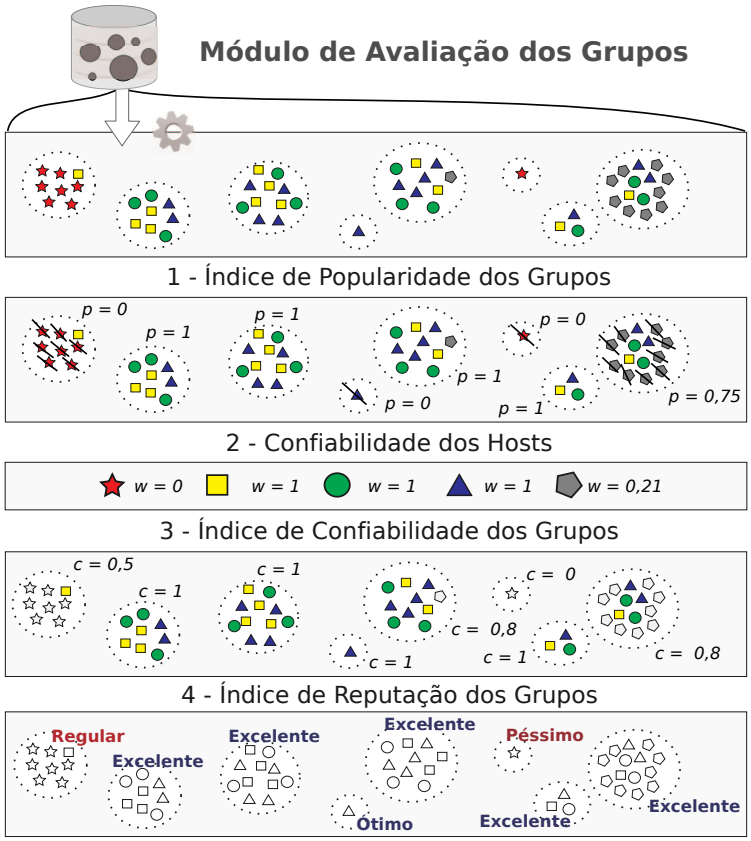

Figura 4.6: Ilustração - As quatro etapas para avaliação dos grupos

Etapa 1 - Índice de Popularidade dos Grupos

O objetivo de calcular o índice de popularidade é determinar o quanto um grupo está equilibrado em relação a diversidade de *hosts* que o acessaram. Isso significa que quanto melhor for a distribuição dos acessos realizados pelos *hosts* melhor será o índice de popularidade. Porém dois problemas podem ocorrer:

- **Problema 1.** Um *host* pode efetuar tantos acessos ao mesmo grupo o quanto ele puder, aumentando significativamente o número de instâncias e desequilibrando o grupo (ex.: acessos realizados por *crawler*).

- **Problema 2.** Um grupo que poderia ter um bom índice de popularidade mas não têm, isso por causa de diversos *hosts* que realizaram poucos senão apenas um acesso dentro do grupo.

- **Solução para os problemas 1 e 2.** Uma solução proposta para esses problemas é estabelecer uma linha de corte (l) cujo valor representa um percentual de acessos que não pode ser ultrapassado por *host* algum. Então os *hosts* que ultrapassarem esse limite terão seus valores de acessos ignorados. Já os *hosts* que não ultrapassarem esse limite irão colaborar para que o índice de popularidade do grupo seja melhorado.

Sendo assim, a linha de corte (l) passa a ser um parâmetro que deve ser definido antes de calcular o índice de popularidade, e esse valor deve ficar entre 0 (zero) e 1 (um) representando o percentual máximo de acessos que um *host* pode ter efetuado dentro de um grupo. Assim o quanto mais o valor de l for próximo de zero, mais *hosts* serão necessários para tornar o grupo popular. Por exemplo, se l fosse igual a 0,01 (1%), seriam necessários no mínimo 100 *hosts* com quantidade de acessos iguais para tornar um grupo popular. Em outro exemplo, se l fosse igual a 0,5 (50%), seriam necessários apenas 2 *hosts* com quantidade de acessos iguais para tornar um grupo popular.

Algoritmo 1 Cálculo do índice de popularidade

```
q_i = ∅
z = true
while z do
  z = false
  for i = 1; i ≤ n; i = i + 1 do
    if q_i > 0 or q_i = ∅ then
      q_i = m_i / m
    end if
    if q_i > l then
      q_i = 0
      m = m - m_i
      z = true
    end if
  end for
end while
Calcular p dado q (equação 4.1)
```

O algoritmo 1 apresenta uma proposta simples para que a linha de corte (l) possa funcionar conforme o que foi proposto. O valor de i identifica o *host* e o valor de m identifica a quantidade total de acessos efetuados ao grupo, portanto, o valor de m_i representa o total de acessos do *host* i dentro do grupo. Já a variável q_i representa o percentual de acessos efetuados pelo *host* i dentro do grupo. A variável z é booleana e serve para manter o laço de repetição enquanto houver algum valor de q_i zerado, isso significa que o índice de popularidade só pode ser calculado se na última iteração nenhum *host* ultrapassar o percentual estabelecido na linha de corte. Outra estrutura de repetição faz com que todos os acessos realizados pelos *hosts* de

i até n sejam analisados. Ao final do algoritmo, o índice de popularidade (p) é calculado de acordo com os valores obtidos de q, e está detalhado na equação 4.1.

$$p = \frac{1}{n}\sum_{i=1}^{n} a \text{ onde } \begin{cases} a = 0, & \text{se } q_i = 0 \\ a = 1, & \text{se } q_i > 0 \end{cases} \tag{4.1}$$

Para exemplificar como é feito o cálculo do índice de popularidade uma ilustração é apresentada tanto na figura 4.6 como na figura 4.7. As figuras geométricas nestas ilustrações representam uma informação que foi acessada através de uma requisição, por exemplo: uma consulta feita através de uma URL. E os diferentes formatos das figuras geométricas representam os diferentes *hosts* que originaram esses acessos (com base no endereço IP).

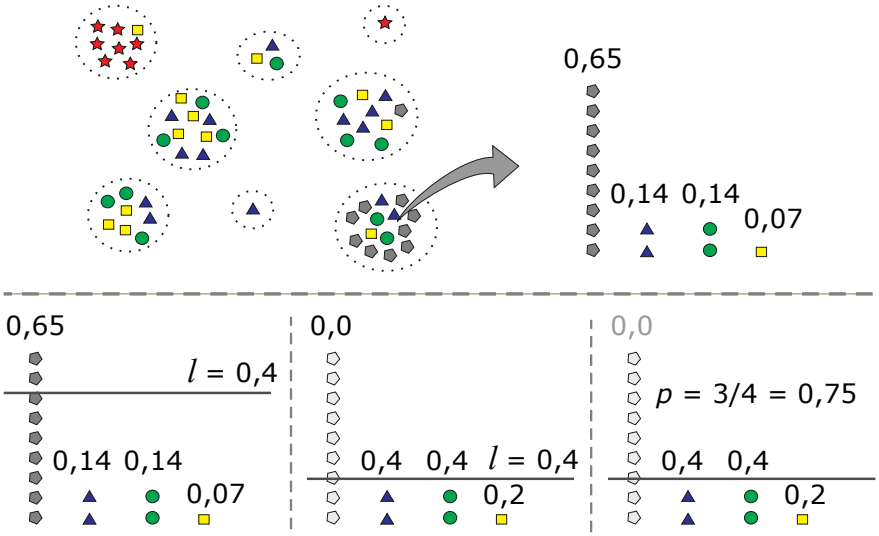

Figura 4.7: Ilustração - Cálculo do índice de popularidade

Etapa 2 - Confiabilidade dos *Hosts*

Nesta etapa, os *hosts* que popularizaram os grupos deverão receber um grau de confiança, o qual será utilizado como base para calcular a confiabilidade dos grupos. Para isso utiliza-se a equação 4.2 , onde para cada *host* (i) é feito o somatório (v_i) de todos os índices de popularidade (p_j) dos grupos que esse *host* acessou. A tupla (i,j) apresentada na equação 4.2 representa uma instância de acesso do *host* i ao grupo j, e a variável m é a quantidade total de grupos no agrupamento.

$$v_i = \sum_{j=1}^{m} a \text{ onde } \begin{cases} a = 0, & \text{se } \nexists \text{ (i,j)} \\ a = p_j, & \text{se } \exists \text{ (i,j) e } q_{ij} > 0 \end{cases} \tag{4.2}$$

- **Problema 3.** Um problema que pode ocorrer nesta etapa é que se a quantidade de acessos realizados pelos *hosts* for muito menor que a quantidade de grupos populares, a confiabilidade dos *hosts* pode ficar baixa e prejudicar o cálculo do índice de confiabilidade na terceira etapa.

- **Solução para o problema 3.** Uma solução que foi implementada, porém precisa de mais estudos, é a maximização da confiabilidade dos *hosts* em comparação ao índice do *host* mais confiável.

Sendo assim, a confiabilidade dos *hosts* (w_i) será maximizada calculando as três equações: 4.3, 4.4 e 4.5. Nessas equações a variável u representa o somatório de todos os índices de popularidade (p). Na equação 4.4 a variável g representa o índice do *host* mais confiável, e na equação 4.5 é calculada para todos os *hosts* a sua confiabilidade de acordo com os valores obtidos de g e de u.

$$u = \sum_{j=1}^{m} p_j \tag{4.3}$$

$$g = max\left(\frac{v_i}{u}\right) \tag{4.4}$$

$$w_i = \frac{v_i}{g \cdot u} \tag{4.5}$$

Uma ilustração de como calcular a confiabilidade dos *hosts* é apresentada tanto na figura 4.6 como na figura 4.8.

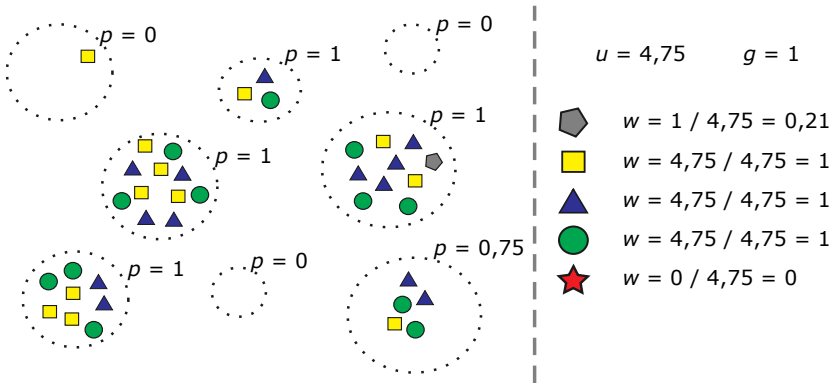

Figura 4.8: Ilustração - Cálculo da confiabilidade dos *hosts*

Etapa 3 - Índice de Confiabilidade dos Grupos

O cálculo deste índice de confiabilidade visa principalmente resolver o problema abaixo:

- **Problema 4.** Durante a avaliação de agrupamento os grupos com um menor número de instâncias e que representam acessos legítimos (ex.: URL pouco acessada) podem aumentar o índice de falsos positivos no momento da detecção de intrusão.

- **Solução para o problema 4.** Uma solução para esse problema está em calcular um índice de confiabilidade do grupo de acordo com a confiabilidade dos *hosts* que efetuaram os acessos.

Sendo assim, após a identificação da confiabilidade de cada *host*, é possível calcular o índice de confiabilidade (c) de cada grupo. Para calcular este índice, basta somar a confiabilidade w_i dos *hosts* e dividir pela quantidade h de *hosts* que participaram desse grupo. Esse cálculo pode ser visualizado na equação 4.6.

$$c = \frac{1}{h} \sum_{i=1}^{h} w_i \tag{4.6}$$

As figuras 4.6 e 4.9 apresentam uma ilustração para exemplificar o cálculo do índice de confiabilidade.

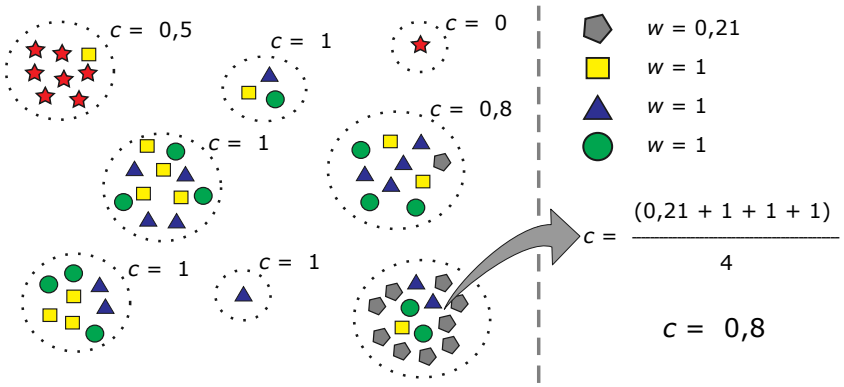

Figura 4.9: Ilustração - Cálculo do índice de confiabilidade

Etapa 4 - Índice de Reputação dos Grupos

Uma vez que se obtém os índices de popularidade e de confiabilidade dos grupos, calcula-se o índice de reputação (r) estipulando os fatores de ponderação, desde que a soma de y_p e y_c seja igual a 4.

$$r = y_p \cdot p + y_c \cdot c \tag{4.7}$$

A idéia em aplicar uma escala de 0 (zero) a 4 (quatro) para ponderação está em dar ênfase a um índice mais do que ao outro. Como é possível observar nas etapas anteriores, é mais custoso ao índice de confiabilidade atingir valores próximos de 1 (um). Se fosse utilizada a escala de 0 (zero) a 3 (três), esta permitiria apenas ponderar desta maneira: $y_p = 1$ e $y_c = 2$. Logo na escala de 0 (zero) a 4 (quatro) é possível ainda dar mais importância para a confiabilidade sem ter que zerar a popularidade: $y_p = 1$ e $y_c = 3$. A seguir alguns exemplos que são sugeridos para serem utilizados dependendo do ambiente que será monitorado:

- Ponderação $y_p = 1$ e $y_c = 3$: um exemplo seria o uso do campo *Method*, pois o método GET é mais popular que o POST, mas é necessário que o acesso via POST seja pelo menos confiável.

- Ponderação $y_p = 4$ e $y_c = 0$: neste caso seria um bom exemplo utilizar com o campo *User-Agent*, pois é mais importante que os navegadores sejam populares do que confiáveis, visto que no lado cliente o índice de confiabilidade só aumenta se os *hosts* utilizarem mais de um navegador.

- Ponderação $y_p = 0$ e $y_c = 4$: um exemplo neste caso seria no campo *Content* de uma requisição, onde só importa que o conteúdo seja confiável.
- Ponderação $y_p = 2$ e $y_c = 2$: neste caso, o fator de ponderação está equilibrado, e um exemplo seria o uso do campo *Referer*, onde é importante que o *link* de origem de referência da requisição seja popular e confiável.

Consequentemente o resultado do índice de reputação (r) também ficará na escala de zero (0) a quatro (4) e uma sugestão é utilizar essa escala empírica para definir uma reputação conforme apresentado na tabela 4.1.

Tabela 4.1: Escala de reputação para atribuição de rótulos

Escala (r)	**Reputação**
3,0 ⊢ 4,0	Excelente
2,0 ⊢ 3,0	Ótima
1,5 ⊢ 2,0	Boa
1,0 ⊢ 1,5	Regular
0,1 ⊢ 1,0	Ruim
0,0 ⊢ 0,1	Péssima

Para ilustrar o cálculo do índice de reputação, um exemplo é apresentado tanto na figura 4.6 como na figura 4.10.

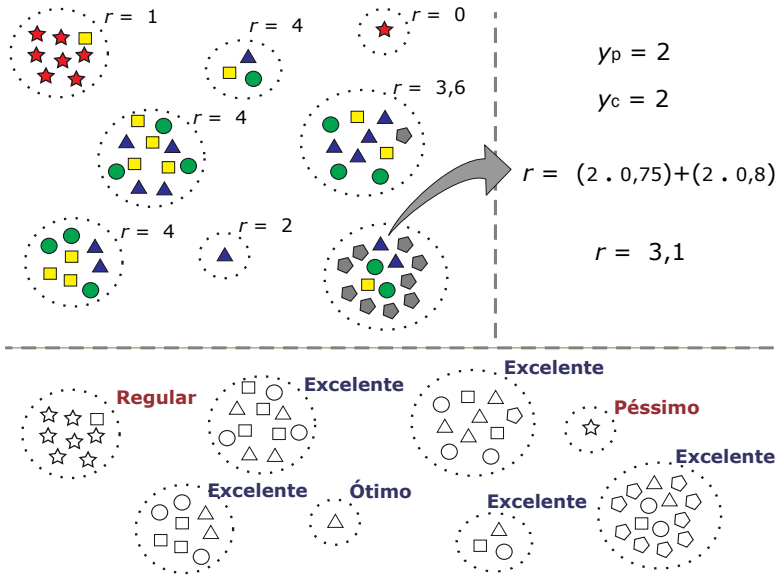

Figura 4.10: Ilustração - Cálculo do índice de reputação

Ao final, as oito listas recebidas estarão com seus grupos rotulados conforme a sua reputação. E os centróides desses grupos poderão ser utilizados como assinaturas no módulo de detecção de intrusão.

4.2 Arquitetura para Detecção de Intrusão

Após a arquitetura para treinamento fornecer assinaturas que representam o comportamento do ambiente a ser monitorado, a arquitetura de detecção de intrusão irá aplicar essas assinaturas no ambiente em produção para identificar ataques. Esse é o momento em que o IDS será colocado à prova, pois é esperado que a taxa de detecção de ataques seja a melhor possível e o que índice de falsos positivos seja próximo de zero. Esta arquitetura é composta por dois módulos (ilustração na figura 4.11). O módulo de inspeção de requisições irá determinar se as requisições estão incompletas ou se violam alguma regra de protocolo. E o módulo de detecção de intrusão irá inspecionar em tempo real os rótulos atribuídos aos campos de uma requisição e efetuar a tomada de decisão.

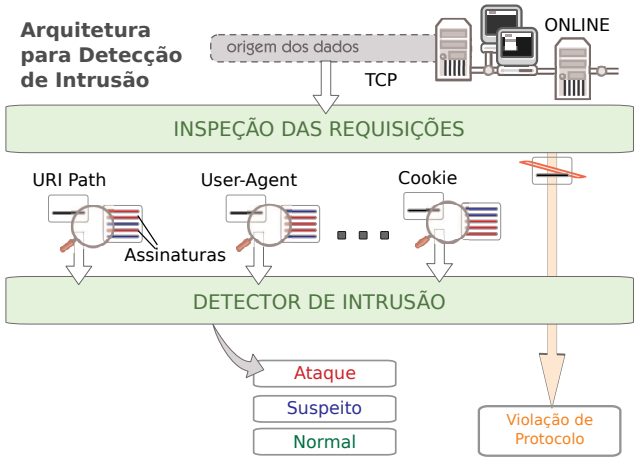

Figura 4.11: Ilustração - Arquitetura para Detecção de Intrusão

4.2.1 Módulo de Inspeção das Requisições

Este módulo deverá coletar os segmentos TCP de uma fonte *online* assim como ilustra a figura 4.11 (ex.: interface *Ethernet*). Então as diversas requisições que irão chegar fragmentadas ou agrupadas em segmentos TCP deverão ser tratadas por alguma biblioteca ou até mesmo por outro IDS. A requisição incompleta que ficar além de um tempo pré-determinado na memória *(buffer)*, esta deverá ser descartada. Após remontar as requisições, diversas verificações serão efetuadas para determinar se a requisição está violando alguma regra de protocolo.

A estrutura do cabeçalho da requisição HTTP contém: uma linha inicial, linhas opcionais de cabeçalho e uma linha em branco (CR/LF). O corpo da requisição deverá conter informações que o cliente deseja enviar (as figuras 4.2 e 4.3 ilustram essas linhas).

A linha inicial de uma requisição deve conter o método, o caminho de acesso ao recurso e a versão do protocolo. Deverão ser aceitos apenas os métodos estabelecidos no documento RFC 2616 [Fielding et al., 1999]: OPTIONS, GET, HEAD, DELETE, TRACE, CONNECT, POST e PUT. Se o servidor *web* suportar serviços WebDAV, os métodos definidos nos documentos RFC 2518 [Goland et al., 1999] e RFC 4918 [Dusseault, 2007] também deverão ser levados em conta. O caminho é a parte da URL que deve ficar entre o método e a versão do protocolo HTTP. No final da linha, o protocolo HTTP poderá assumir uma das duas versões: HTTP/1.0 ou HTTP/1.1.

As linhas seguintes do cabeçalho da requisição são opcionais, onde cada linha contém um campo que é preenchido com valores referentes ao recurso a ser acessado, ou carregam informações sobre a origem do acesso. Uma das diferenças na definição do HTTP/1.0 para o HTTP/1.1 é que no mínimo o campo *Host* deve acompanhar a requisição para permitir um URI completo. A informação preenchida em cada linha do cabeçalho HTTP deverá ter o nome do campo e os valores do campo separados pelo caractere de dois pontos ":". Ao final de cada linha deve haver os caracteres CR/LF e o documento RFC também define que as aplicações devem tratar de linhas separadas por LF. Exemplo: "User-Agent:Firefox\n".

O conteúdo do corpo HTTP poderá carregar as informações que o usuário deseja enviar através da requisição. Podendo essa informação ser: os campos preenchidos de um formulário ou até mesmo um arquivo para *upload*. O preenchimento de qualquer conteúdo no corpo da requisição requer que uma linha no cabeçalho chamada de *Content-length* especifique de que tamanho será esse corpo. Os métodos POST e PUT são utilizados para enviar requisições com conteúdo, entretanto o documento RFC não proíbe que os outros métodos também sejam usados para enviar conteúdo.

Ao final, depois de descartar e alertar as requisições que violaram regras de protocolo, então as requisições que atenderem a este módulo terão seus oito campos extraídos para serem verificados pelo módulo de detecção de intrusão.

4.2.2 Módulo de Detecção de Intrusão

Uma a uma, as requisições recebidas pelo módulo anterior serão inspecionadas para verificar se contém ataques, assim ilustra a figura 4.11. Os algoritmos e os limiares estabelecidos na arquitetura de treinamento irão influenciar diretamente nas decisões que ocorrerem neste módulo. Cada requisição deverá passar por duas etapas que estão descritas a seguir.

1) Comparação com as Assinaturas

Cada campo da requisição HTTP será comparado com a lista de centróides fornecidas pela arquitetura de treinamento. Para efetuar as comparações existem duas situações possíveis:

a) Em uma situação, o campo da requisição deverá ser comparado **com todos** os centróides, então o campo receberá o mesmo rótulo de reputação de acordo com o centróide mais próximo ou similar. Esse é o tipo de comparação realizado no trabalho de [Portnoy et al., 2001].

b) Em outra situação, o campo da requisição será comparado com os centróides somente **até encontrar um** que esteja mais próximo (ou similar) de acordo com um limiar (*threshold*) pré-definido, então o campo receberá o mesmo rótulo de reputação de acordo com esse centróide. Se mesmo comparando com todos os centróides não for encontrado um que atenda aos requisitos, então o campo receberá o rótulo da pior reputação possível. Esse é o caso utilizado nos experimentos deste trabalho.

Para identificar a situação em que ocorre o melhor desempenho vai depender da quantidade de centróides a serem utilizados em cada situação. Embora, por exemplo, se a quantidade de centróides for a mesma nas duas situações, "b)" levará vantagem pois permite que a comparação seja feita somente até encontrar o centróide mais próximo, ao contrário de "a)" que

precisa comparar com todos os centróides. Além disso, o índice de popularidade pode ser utilizado para ordenar a lista de centróides, reduzindo ainda mais a quantidade de comparações a serem feitas.

Ao final desta etapa, cada campo da requisição HTTP receberá uma reputação (excelente, ótima, boa, regular, ruim ou péssima), e na segunda etapa deste módulo uma decisão será tomada em relação a requisição.

2) Tomada de Decisão

O modelo de tomada de decisão proposto nesta etapa foi obtido através de uma observação simples durante os experimentos: a maioria das requisições que não continham ataques tinham seus campos rotulados com a reputação boa, ótima ou excelente. Então foi utilizada a equação 4.8, onde d é o valor da tomada de decisão dado a soma da pontuação de reputação conforme a tabela 4.2.

$$d = \sum_{i=1}^{n} \text{pontuação}\,(r_i) \tag{4.8}$$

Tabela 4.2: Escala de reputação para tomada de decisão

Reputação (*r*)	**Pontuação**
Excelente	0
Ótima	0
Boa	0
Regular	1
Ruim	2
Péssima	4

O valor de r representa a reputação do campo i da requisição HTTP. Assim é possível considerar que toda requisição em que o valor de d ficar igual a 0 (zero) não representa perigo de ataque. Qualquer pontuação abaixo do limite deverá considerar a requisição como suspeita. E, se o valor de d for superior ou igual ao limite, a requisição será considerada intrusiva conforme a tabela 4.3.

Tabela 4.3: Valores para a tomada de decisão

Decisão	**Requisição**
$d = 0$	Normal
$0 < d < 8$	Suspeita
$d \geq 8$	Intrusiva

Assim, as requisições que são normais poderão ser ignoradas pelo IDS. As demais requisições, principalmente as de ataque, deverão ser inspecionadas pelo operador para que ele possa atuar com uma resposta a incidente. Para ilustrar, duas requisições são apresentadas na figura 4.12, onde a primeira ilustra uma requisição normal sendo analisada pelo IDS, já a segunda é uma ilustração de uma requisição intrusiva gerada por um *bot* conhecido como *Casper* [Romang, 2010].

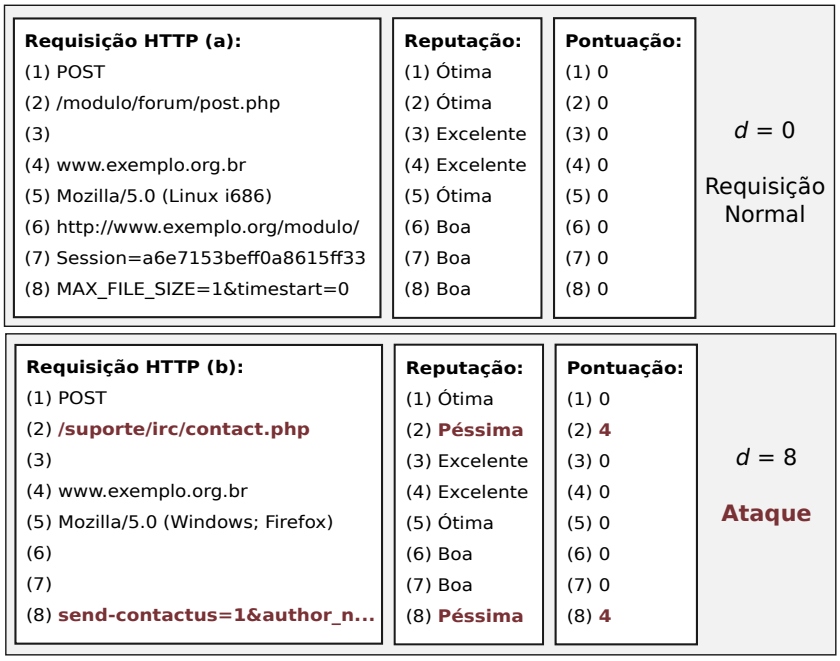

Figura 4.12: Ilustração - Tomada de decisão nas requisições HTTP

4.3 Considerações

Neste capítulo foi apresentada a arquitetura do IDS proposto para detecção baseada em anomalia com a utilização de algoritmos de agrupamento para a inspeção de ataques aos servidores *web* e HTTP. Abaixo, algumas considerações sobre esta arquitetura:

- No momento do treinamento, modelos de comportamento poderão ser extraídos mesmo que a base não seja rotulada. Apesar de ser uma aprendizagem não-supervisionada, é importante que a base para treinamento contenha em sua maioria requisições de acesso normal ao servidor *web* para evitar que modelos normais de comportamento sejam criados a partir de ataques.

- É preciso levar em conta a limitação da arquitetura para tratar uma grande quantidade de requisições para o treinamento, pois isso está relacionado diretamente com a complexidade e o tempo de execução do algoritmo de agrupamento que foi escolhido.

- O módulo de avaliação de grupos desta arquitetura que faz parte da contribuição científica e deverá tratar cada grupo criado de maneira diferente. Pois os *hosts* que efetuaram o acesso a esses grupos irão afetar em como os grupos menores *(outliers)* e os grupos maiores serão rotulados. Assim, nem todo *outlier* poderá ser considerado um modelo de intrusão, melhorando significativamente na redução da taxa de falsos positivos.

- Outro ponto a ser observado é o modelo simples da tomada de decisão. Pois durante os testes este modelo se mostrou eficiente, sem a necessidade da implementação de algoritmos de classificação.

Capítulo 5

Cenário de Validação

Para que o IDS baseado em anomalia proposto no capítulo anterior pudesse ser comparado ao IDS Snort, foi necessária a preparação de um cenário para validação e testes. O primeiro requisito para compor este cenário foi a seleção de um conjunto de requisições (*dataset*). Para isso foram coletadas as requisições HTTP de dois servidores *web* com sítios públicos e acessíveis pela *Internet*. Essas requisições, que estão detalhadas na seção 5.1, foram inspecionados manualmente e revelaram diversos tipos de ataques. Cada ataque encontrado foi rotulado para que pudesse servir como unidade de medida na comparação entre os IDSs. Assim, a seção 5.2 classifica e detalha estes ataques.

Em seguida, na implementação do IDS foi utilizada a linguagem de programação Perl. Assim, todos os algoritmos necessários foram desenvolvidos ou foram importados de módulos CPAN[1]. A implementação do algoritmo de agrupamento juntamente com as medidas de similaridade são assuntos da seção 5.3. Na seção 5.4, o IDS Snort precisou ser configurado para a detecção de ataques HTTP e foram feitos os ajustes para realizar os testes. Já para a comparação entre os IDSs foi selecionado o índice de Medida-F *(F-Measure)* na seção 5.5. E ao final deste capítulo, na seção 5.6, são feitas algumas considerações sobre este cenário de validação.

5.1 Conjunto de Requisições HTTP

Para realizar os testes da proposta foi selecionado o protocolo HTTP *(Hyper-Text Transfer Protocol)* com serviços *web*, pois atende aos seguintes requisitos: grande volume de informações, diferentes origens de acesso, diversos tipos de variáveis de acesso e serviços; e além disso, porque se tornou um serviço popular na *Internet* e muito visado por atacantes. Para confirmar isso, nas estatísticas do sítio CERT.br (Centro de Estudos, Resposta e Tratamento de Incidentes de Segurança do Brasil) as notificações de ataques aos servidores *web* não passavam de 1% até 2007 e, em 2010, passou para mais de 6% do total de ataques [CERT.br, 2010]. Outro sítio que é conhecido na comunidade de *hackers* é o Zone-H, o qual recebeu notificações de mais de 500.000 ataques em 2009, e passou para aproximadamente 1.500.000 ataques *web* em 2010 [Zone-H, 2010].

As bases públicas mais comuns e mais utilizadas por pesquisadores para avaliarem seus IDSs são sem dúvida, as bases do DARPA *(Defense Advanced Research Projects Agency)*

[1]CPAN (*Comprehensive Perl Archive Network*): é um repositório público de algoritmos e módulos criados na linguagem *Perl*.

de 1999 [DARPA, 1999] e, também, as bases da competição internacional de 1999 - *Knowledge Discovery and Data Mining Tools Competition* [KDD, 1999]. Pode-se dizer que são bases de mais de uma década bem documentadas mas que não contemplam as novidades no mundo *web* e a diversidade de ataques que existem atualmente. Portanto, havia uma expectativa de que uma competição conhecida como CDX 2009 *(Cyber Defence Exercise)* organizada pelo ITOC *(Information Technology Operations Center) - U.S. Army* - viesse a oferecer uma base nova com diversos tipos de ataques e que substituiria o [DARPA, 1999]. A competição ocorreu, mas infelizmente não foram fornecidos os dados completos devido a problemas técnicos [CDX, 2009] e, na sequência, na competição de 2010 os dados não foram coletados. Ainda uma outra base que foi coletada de uma competição CTF [iCTF, 2008] foi verificada. Mas devido ao número de ataques ser maior do que o de acessos normais, resultaria em mau treinamento para o IDS baseado em anomalia.

Na falta de uma base pública, recorreu-se aos servidores *web* hospedados na empresa CELEPAR (Companhia de Informática do Paraná) [CELEPAR, 2010] para realizar os experimentos neste trabalho. Para formar esta base, foram selecionados dois servidores *web* que não fossem críticos e que não trafegassem informações confidenciais. Através de um *switch* posicionado entre os servidores *web* e o *firewall*, foram coletados os pacotes IPs entre as 14:00 do dia 09/12/2010 e as 20:00 do dia 21/12/2010 (GMT). Ao todo foram gerados 200 arquivos do tipo *pcap (Packet Capture*) com mais de 5 milhões de requisições. Os detalhes sobre estas bases são apresentados a seguir.

WS1 - Servidor *Web* I

O primeiro servidor *web* contém um sítio muito acessado pela comunidade brasileira, sendo um domínio DNS *(Domain Name Service)* principal e diversos outros domínios associados. O sítio principal tem sua base em portal do tipo Drupal [Drupal, 2010] e ocorre constante atualização de notícias.

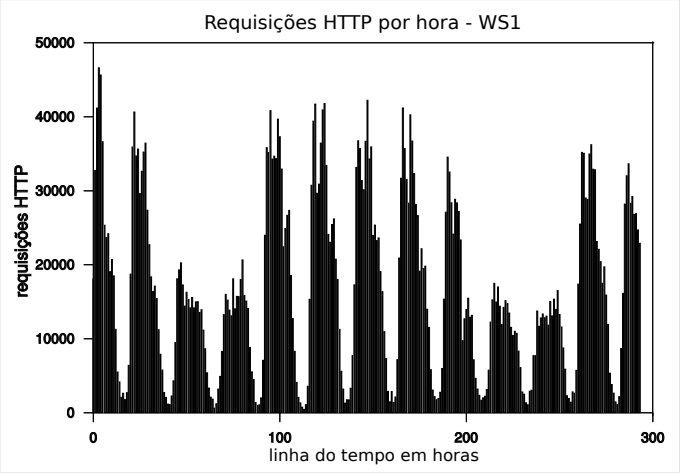

Figura 5.1: Gráfico - Requisições HTTP de WS1

A quantidade de requisições HTTP que foram coletadas estão representadas no gráfico 5.1. Para rotular os ataques a base de requisições foi subdividida em duas partes: entre as 14:00 do dia 09/12/2010 e as 13:00 do dia 15/12/2010, e a segunda parte entre as 13:00 do dia

15/12/2010 e as 20:00 do dia 21/12/2010 (GMT). Na primeira parte cada ataque foi rotulado manualmente, para que precisamente os IDSs fossem testados. A segunda parte não foi rotulada para poder avaliar a quantidade de ataques e falsos positivos que os IDSs irão alertar em uma base desconhecida. Com mais detalhes na tabela 5.1, os ataques representam apenas 0,003% das requisições da primeira parte.

Tabela 5.1: Requisições HTTP de WS1

WS1(parte 1)	Requisições/IP	Total IPs	Requisições/Hora	Total Requisições
Normais	47	53.134	17.397	2.522.535
Ataques	61	136	57	8.293
Total	48	53.210	17.454	2.530.828
WS1(parte 2)	Requisições/IP	Total IPs	Requisições/Hora	Total Requisições
Sem rótulo	47	52.109	16.197	2.429.569
WS1	Requisições/IP	Total IPs	Requisições/Hora	Total Requisições
Total	49	100.724	16.872	4.960.397

WS2 - Servidor *Web* II

Já o segundo servidor *web* hospeda dois sítios populares com alguns subdomínios de DNS. A base desses sítios foi desenvolvida com o portal XOOPS [XOOPS, 2010], a qual tem centenas de usuários que participam de fóruns e, também, ocorre a constante atualização de notícias. Da mesma maneira que a base de WS1, esta base também foi dividida em duas partes.

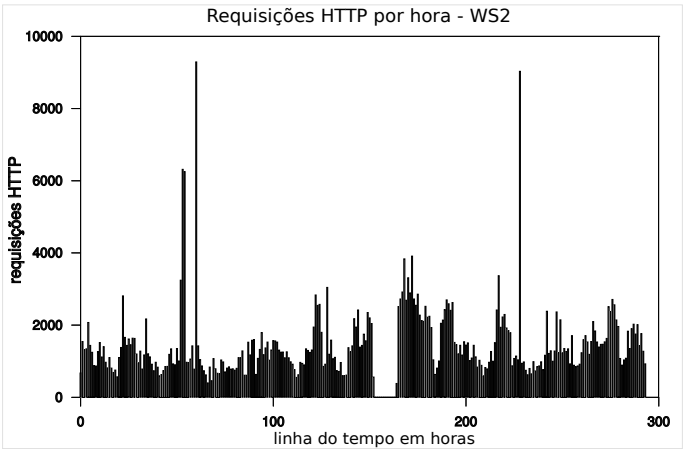

Figura 5.2: Gráfico - Requisições HTTP de WS2

Percebe-se no gráfico 5.2 que o servidor ficou sem comunicação entre as 23:00 do dia 15/12/2010 e as 09:00 do dia 16/12/2010 (GMT), devido a manutenção. No restante, esse servidor foi o que mais sofreu ataques. Na primeira parte das requisições rotuladas, conforme a tabela 5.2, os ataques correspondem a 5% dessas requisições.

Tabela 5.2: Requisições HTTP de WS2

WS2(parte 1)	Requisições/IP	Total IPs	Requisições/Hora	Total Requisições
Normais	43	4.121	1.225	178.267
Ataques	22	478	76	10.366
Total	42	4.445	1.301	188.633
WS2(parte 2)	Requisições/IP	Total IPs	Requisições/Hora	Total Requisições
Sem rótulo	55	4.266	1.552	232.828
WS2	Requisições/IP	Total IPs	Requisições/Hora	Total Requisições
Total	54	7.801	1.434	421.461

5.2 Ataques Rotulados

No processo de coleta de ataques dos servidores WS1 e WS2, somente na primeira parte foram identificados mais de 18 mil ataques, sendo que apenas aproximadamente 8 mil foram considerados não repetidos. O servidor WS1 apresentou menor número de ataques, mas representa uma base importante para detectar ataques específicos em meio aos diversos acessos normais. Já o servidor WS2 foi mais visado por atacantes e sofreu não só ataques *web*, mas também o ataque de *spammers* e uma varredura por vulnerabilidades da ferramenta OpenVAS [OpenVAS, 2011]. Portanto, os ataques mais comuns nessas bases foram mapeados e classificados em 10 tipos principais e alguns desses estão relacionados com os ataques *web* descritos no sítio OWASP [OWASP, 2011]. A tabela 5.3 mostra a quantidade de ataques classificados por tipo e, nas seções seguintes, são apresentados detalhes sobre cada tipo de ataque.

Tabela 5.3: Classificação dos Ataques

WS1	**Quantidade**	**Sem Repetição**	**WS2**	**Quantidade**	**Sem Repetição**
A01	16	14	A01	163	155
A02	31	22	A02	400	227
A03	66	6	A03	25	21
A04	7.926	10	A04	56	34
A05	6	5	A05	47	19
A06	0	0	A06	251	245
A07	91	2	A07	137	44
A08	147	44	A08	822	90
A09	9	9	A09	8.458	7.080
A10	1	1	A10	7	7
Total	**8.293**	**113**	**Total**	**10.366**	**7.922**

A01 - Injeção de Código de SQL

O ataque por injeção de código SQL *(SQL Injection)* consiste em enviar uma consulta *(query)* da aplicação cliente com o objetivo de obter informação sensível do banco de dados da aplicação do servidor. Além disso, esses ataques podem até modificar a base *(Insert, Delete, Update)* e executar operações de administração ou fazer com que o sistema gerenciador de banco de dados (DBMS) execute comandos no sistema operacional. Uma subclassificação

desse ataque é o *Blind SQL Injection* que busca por retorno de erros do servidor para coletar informações sobre o sítio atacado. Detalhes sobre esses ataques podem ser encontrados no sítio [OWASP, 2011]. A figura 5.3 ilustra dois ataques, onde no segundo exemplo o ataque foi ofuscado em hexadecimal para poder burlar sistemas de proteção.

```
Exemplo 1:
GET /index.php?option=zzz&author=62+union+select+1,concat_ws%280x3a,username,password

Exemplo 2:
GET /modulos/novo/vertopico.php?post=7757%20%61%6E%64%20%36%3D%35
  =
GET /modulos/novo/vertopico.php?post=7757 and 6=5
```

Figura 5.3: Ilustração - Injeção de Código SQL (A01)

A02 - Injeção de Código Remoto

Injeção de código é um nome genérico para diversos tipos de ataques que dependem da injeção de código para ser executado pela aplicação remota. Esse código malicioso pode ser enviado por URI ou até mesmo por *cookie* explorando a validação ruim do servidor *web* para a entrada e saída de dados. A figura 5.4 ilustra esse ataque e mais detalhes podem ser encontrados no sítio [OWASP, 2011].

```
Exemplo 1:
GET /arquivos/pagina=http://10.76.14.148/cmd.txt???

Exemplo 2:
GET /index.php?id=ftp://usuario:123456@ftp.zzz.br/bypass.img?

Exemplo 3:
GET /phpmyadmin/config/config.inc.php?p=phpinfo();
```

Figura 5.4: Ilustração - Injeção de Código Remoto (A02)

A03 - Travessia de Caminho

O ataque de travessia de caminho *(path traversal)* busca por arquivos armazenados no sistema que podem conter informações de acesso como senhas ou código-fonte. Esse ataque utiliza *dot-dot-slash* (../) para encontrar arquivos e pastas que estão fora da raiz do sítio *web* mas com acesso limitado pelo sistema operacional [OWASP, 2011]. A figura 5.5 ilustra dois exemplos desse ataque.

```
Exemplo 1:
GET /arquivos/pagina=../../../../../../../../../../../../../etc/passwd

Exemplo 2:
GET /index.php?opcao=zzz&tarefa=../../../../../../../../../../../../../../../proc/self/environ%00
```

Figura 5.5: Ilustração - Travessia de Caminho (A03)

A04 - Busca Forçada

O ataque é descrito pelo sítio [OWASP, 2011] como a tentativa de enumerar recursos e acessos não referenciados na aplicação *web*, de maneira a descobrir algum arquivo ou diretório vulnerável. Por exemplo: um arquivo temporário pode conter *backup* do código-fonte, páginas ocultas para acesso de administração, entre outros. Uma ferramenta que faz esse tipo de busca é o Nikto [Nikto, 2011]. A figura 5.6 ilustra alguns exemplos.

```
GET /senhas/
GET /manager/html
GET /wp-login.php
GET /administrator/index.php
GET /phpmyadmin/
GET /index.asp.bkp
...
```

Figura 5.6: Ilustração - Busca Forçada (A04)

A05 - Busca por Erro

Um tipo de ataque encontrado com frequência na base dos dois servidores *web* é o uso do apóstrofo isolado (’ ou %27). Embora possa ser encontrado como um tipo de ataque de injeção de SQL, um teste revelou que esse ataque encontrou uma página que retornava um erro: 500 - *Internal Server Error*. Então esse ataque foi classificado separadamente e alguns exemplos podem ser visualizados na figura 5.7.

```
Exemplo 1:
GET /secao.php?id='

Exemplo 2:
GET /acesso.php?id=%27

Exemplo 3:
GET /modulo/imprime.php?formulario='1&amp;amp;topico='234&amp;amp;inicio='0
```

Figura 5.7: Ilustração - Busca por Erro (A05)

A06 - Abuso de Formulários

Este é um ataque originado por *spammers* que pretendem fazer propaganda de produtos tanto para o sítio local quanto para os sítios de busca. Um *spam* pode ser postado em uma resposta em um fórum ou em comentário de notícias, e muitas vezes o *spammer* cria um usuário legítimo no sistema obrigando os administradores a reforçarem suas políticas de moderação. No servidor WS2 foram encontrados diversos POSTs de *spammers* tanto para os fóruns como até mesmo para o formulário de busca do sítio. De maneira resumida a figura 5.8 faz uma ilustração desse ataque.

```
POST /form.php
...
content: _TOKEN_REQUEST=...&skipValidation..=.&dohtml=.reply=1&subject=Re:&id=62
&topic...&message=.Wow.+put+stuff.+found+on+this+site+www.zzz.br+buy+Cover
%5B/url%5D&submit=Enviar
```

Figura 5.8: Ilustração - Abuso de Formulários (A06)

A07 - Ataque de *Malwares*

Centenas de tentativas de ataques para os servidores *web* foram identificados como originados por estações infectadas por *malwares*, mais especificamente um *bot* conhecido como *Casper* [Romang, 2010]. Esse *malware* tenta explorar vulnerabilidade de sítios que utilizam o portal e107 [e107, 2011]. A figura 5.9 ilustra duas requisições, uma com método POST e outra com método GET.

```
Exemplo 1:
POST /suporte/irc/contact.php
...
content: send-contactus=1&author_name=%5Bphp%5Decho('casper'.php_uname().'kae')
%3Bdie%28%29%3B%5B%2Fphp%5D.

Exemplo 2:
GET /forum/main.php?website=%7Cecho%20%22casper%22;echo%20%22kae%22;%7C
User-Agent: MaMa CaSpEr
```

Figura 5.9: Ilustração - Ataque de *Malwares* (A07)

A08 - Abuso de *Proxy* Aberto

Outro ataque que foi encontrado em grande quantidade nas bases de requisições foram as tentativas de utilizar o servidor *web* como se fosse um *proxy* aberto. Se o servidor *web* estiver mal configurado, o atacante pode até mesmo explorar o *proxy* aberto para enviar *spam* via a porta 25 SMTP [Apache, 2009]. A figura 5.10 apresenta três exemplos desse tipo de ataque.

```
Exemplo 1:
GET http://www.zzz.cn
Host: zzz.cn

Exemplo 2:
CONNECT mailz.zzz.br:25
Host:

Exemplo 3:
POST 10.23.9.7/proxy5/check.php
Host: 10.23.9.7
```

Figura 5.10: Ilustração - Abuso de *Proxy* Aberto (A08)

A09 - Varredura por Vulnerabilidades

As ferramentas automatizadas de varreduras *(scan)* executam diversos tipos de ataques para descobrir vulnerabilidades em um servidor ou em uma aplicação. O sistema OpenVAS [OpenVAS, 2011], com as assinaturas de ataques que herdou do Nessus [Nessus, 2011],

foi o responsável pela maior parte dos diferentes tipos de ataques efetuados ao servidor WS2. Segundo o sítio do OpenVAS, são mais de 20 mil assinaturas de diversos ataques e na base de requisições foram encontrados 8.458 ataques via requisições HTTP. Também foram encontrados 9 ataques diferentes no servidor WS1 que foram causados por ferramenta de varredura não identificada.

A10 - Outros Ataques

Outros ataques específicos que não exploram caminhos comuns como o URI ou o conteúdo das requisições HTTP foram identificados e classificados no mesmo grupo de ataques:

- Um ataque foi identificado no servidor WS1 enviando diversas vezes o cabeçalho de *User-Agent* na mesma requisição.
- Dois ataques foram identificados explorando o cabeçalho de *Referer* no qual contém pseudo-HTML para explorar alguma ferramenta que trata de *logs*.
- Quatro ataques de *PHP Injection* tentam explorar alguma vulnerabilidade através de requisições com método POST.
- Um ataque que explora o cabeçalho de *Cookie* com a intenção de sequestrar uma sessão de um sítio de rede social.

5.3 Desenvolvimento do IDS

O IDS baseado em anomalia da arquitetura proposta foi desenvolvido utilizando a linguagem Perl. O propósito original da linguagem Perl era de apenas ser utilizada para manipulação de textos, mas hoje faz parte de uma gama de aplicações tais como sistemas de administração, desenvolvimento *web*, programação de rede, entre outras [PerlDoc, 2011]. Além disso, o repositório CPAN [CPAN, 2011] disponibiliza módulos e *scripts* em Perl para suportar diversas atividades. O IDS que foi desenvolvido faz uso de alguns desses módulos e detalhes sobre a sua implementação são apresentados a seguir.

1) Coleta de Requisições HTTP

A primeira parte desenvolvida do IDS foi feita para atender os módulos descritos nas seções 4.1.1 e 4.2.1. Foi necessário tratar da coleta e remontagem de pacotes IPs e segmentos TCPs, tarefa que foi confiada à extensão da biblioteca LibNIDS [Wojtczuk, 2010] disponível no CPAN e conhecida como Net::LibNIDS [Bergman, 2004]. Essa biblioteca permite coletar dados de uma interface de rede ou de um arquivo do tipo PCAP, e foi disponibilizada justamente para prestar o suporte básico para o desenvolvimento de um IDS baseado em rede. Assim, restou implementar apenas o tratamento das requisições HTTP que poderiam estar incompletas, agrupadas ou poderiam violar definições estabelecidas nos documentos RFC 2616, RFC 2518 e RFC 4918.

2) Algoritmo de Agrupamento

Para atender o que foi descrito nas seções 4.1.2 e 4.1.3 foi necessário desenvolver o algoritmo de agrupamento. Em testes preliminares foi testado o algoritmo AHC *(Agglomerative Hierarchical Clustering)*, mas o algoritmo *K-Means* foi selecionado devido a sua convergência rápida para uma configuração estável [Tan et al., 2005]. Porém, o algoritmo *K-Means* foi modificado para fazer o seguinte: o uso de medidas de similaridade entre *strings* e o uso de limiares de distância máxima entre um elemento do grupo e seu centróide. Mais detalhes sobre essa escolha podem ser encontrados nas seções 4.1.2 e 4.2.2.

3) Medidas de Distância de Similaridade

Foram desenvolvidas medidas de distância de similaridade entre *strings* para atender os requisitos das seções 4.1.2 e 4.2.2. Duas medidas de distância baseadas em *tokens*, *cosine* e *jaccard*, foram desenvolvidas e testadas com *n-grams* (com $n = 1$). Já para os testes com a medida de distância de edição *(edit distance)*, ou Levenshtein, foi utilizado o módulo LevenshteinXS [Mistrut and Goldberg, 2003]. Mas um teste preliminar com 100 elementos para cada um dos campos (UPath, UQuery e User-Agent) levou a escolha da medida de distância Jaro-Winkler que, além do módulo disponível no CPAN [Weng, 2009], os critérios de similaridade para os campos UPath e UQuery resultaram em melhores agrupamentos. Portanto, os experimentos foram realizados com a medida de distância Jaro-Winkler, tanto para o agrupamento como na detecção de intrusão servindo como critério de distância para as assinaturas (ou centróides). Mais detalhes desses tipos de medidas de similaridade poderão ser encontrados no trabalho de [Cohen et al., 2003].

4) Avaliação dos Agrupamentos

Antes de executar o algoritmo de agrupamento é necessário estabelecer um número k de grupos ou um número t de limiar máximo de distância. Esses valores podem influenciar diretamente na qualidade do agrupamento e afetar o módulo de detecção de intrusão. Sendo assim, antes de fazer o agrupamento descrito na seção 4.1.2 uma pequena amostra foi utilizada com diferentes distâncias para selecionar uma que aproximasse de um bom agrupamento. O índice Davies-Bouldin leva em conta que grupos que são densos *(dense)* internamente e bem separados entre si *(scatter)* representam o agrupamento ideal. Abaixo, na equação 5.1, a fórmula para o cálculo desse índice que foi utilizado nos experimentos deste trabalho. Outros índices podem ser encontrados no livro de [Gan et al., 2007].

$$\textit{Davies-Bouldin} = \frac{1}{n} \sum_{i=1, i \neq j}^{n} \max \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right) \tag{5.1}$$

5.4 Preparação do IDS Snort

O Snort é um IDS que, além de ser baseado em regras, também permite que *preprocessors* sejam desenvolvidos para decodificar e inspecionar os pacotes antes de aplicar as regras. Basicamente, o Snort pode ser utilizado para detectar diversos tipos de ataques através

de diferentes protocolos. Por esse motivo, o Snort foi configurado para tratar somente de ataques via requisições HTTP para que a comparação ao IDS proposto fosse possível. Apenas três *preprocessors* foram ativados com seus valores padrões:

- *Frag3*: desfragmentação de pacotes IPs.
- *Stream5*: inspeção *stateful* e remontagem de *streams* TCPs.
- *HTTP_Inspect*: para normalização de requisições HTTP e detecção de anomalia.

Tabela 5.4: Grupos e quantidade de regras do Snort e da Emerging Threats

Grupos	**Regras HTTP ou 80 TCP**	**Total de Regras**
backdoor	98	607
dos	5	17
exploit	34	147
scan	5	11
shellcode	25	26
web-misc	31	61
web-iis	4	4
web-cgi	7	9
web-php	3	3
emerging-scan	116	169
emerging-dos	4	18
emerging-exploit	86	220
emerging-web_server	186	206
emerging-web_specific_apps	4.703	4.715
Total	**5.307**	**6.213**

Além disso, foram utilizadas as assinaturas do sítio do Snort e também da comunidade *Emerging Threats* [EmergingThreats, 2010]. A tabela 5.4 apresenta os principais grupos de regras que foram ativados nos experimentos, esses grupos continham 5.307 assinaturas para detecção de ataques *web* e HTTP. A versão 2.9.0 do Snort foi utilizada para realizar os experimentos, e detalhes de sua configuração podem ser consultados no manual encontrado no próprio sítio do Snort [Snort, 2011].

5.5 Critério de Comparação de IDSs

Para comparação entre os IDSs é necessária uma medida que permita comparar o número de verdadeiros positivos, falsos positivos e falsos negativos. No livro de [Rijsbergen, 1979] é possível encontrar a primeira menção da Medida-F *(F-Measure)* que permite esse tipo de comparação, e nas equações 5.2, 5.3 e 5.4 estão as fórmulas utilizadas para a comparação entre os IDS:

$$\text{Precisão} = \frac{vp}{vp + fp} \tag{5.2}$$

$$\text{Revisão} = \frac{vp}{vp + fn} \tag{5.3}$$

$$\text{Medida-}F = 2 \cdot \frac{\text{precisão} \cdot \text{revisão}}{\text{precisão} + \text{revisão}} \tag{5.4}$$

A Medida-F faz um equilíbrio no cálculo da precisão com uma revisão. Os valores utilizados para essa medida são:

- Verdadeiros positivos (vp): ataques que foram detectados.
- Falsos positivos (fp): atividades legítimas que foram detectadas como ataques.
- Falsos negativos (fn): ataques que ocorreram mas não foram detectados.

Outro ponto importante que foi utilizado para fazer a comparação entre os IDSs foi a sumarização dos alertas. A sumarização foi feita para que os ataques repetidos fossem contabilizados apenas uma vez, e os ataques que continham variações fossem rotulados separadamente cada um como um ataque diferente. Por exemplo, na tabela 5.3 o ataque A04 no servidor WS1 se repetiu por milhares de vezes mas foi sumarizado em apenas dez. Em outro exemplo, na mesma tabela, os ataques A01 eram similares mas com diversas variações para injeção de SQL, assim poucos ataques foram sumarizados. Outro ajuste foi realizado para sumarizar os alertas do Snort, visto que duas bases de assinaturas foram utilizadas e poderia ocorrer casos em que assinaturas similares detectassem o mesmo ataque. A sumarização também foi aplicada aos alerta de falsos positivos e de falsos negativos.

5.6 Considerações

Este capítulo apresentou como o cenário proposto na seção anterior foi implementado para a realização de experimentos. De acordo com as observações, foram levantados alguns pontos em relação a este capítulo:

- A seleção de uma base de dados que contivesse ataques rotulados se tornou uma tarefa difícil, pois as que são populares datam há mais de 10 anos. Assim uma base foi coletada de servidores *web* da empresa CELEPAR e rotulada manualmente.
- Na base de requisições utilizada, os ataques foram identificados e classificados em 10 tipos principais. Esses ataques rotulados foram utilizados para fazer a comparação do IDS proposto com o IDS Snort.
- O desenvolvimento do IDS foi feito com a linguagem Perl utilizando a biblioteca Lib-NIDS. Os algoritmos *K-Means* e Jaro-Winkler foram selecionados para agrupamento e medida de similaridade respectivamente, e o índice de Davies-Bouldin foi selecionado para determinar a qualidade dos agrupamentos realizados.
- O IDS Snort foi otimizado para detectar somente os ataques nas requisições *web*, de maneira tal para que fosse comparado com o IDS baseado em anomalia.
- Ao final, a Medida-F foi selecionada por permitir uma comparação entre os IDSs com base em verdadeiros positivos, falsos positivos e falsos negativos.

Capítulo 6

Resultados dos Experimentos

Este capítulo apresenta os resultados dos experimentos realizados com o IDS da arquitetura proposta. Na primeira seção (6.1) é apresentada a configuração inicial dos limiares de distância e os valores dos fatores de ponderação que foram utilizados no IDS. Na segunda seção (6.2) são apresentados alguns dados sobre o treinamento do IDS. Na seção 6.3, diversos valores para a de linha de corte (l) foram testados para encontrar a melhor detecção do IDS. Depois de selecionar o melhor modo de detecção, o IDS proposto é comparado ao IDS Snort na seção 6.4. Como complemento, na seção 6.5, as taxas de alarmes falsos positivos são analisadas em testes adicionais com o IDS. Para finalizar, na seção 6.6, são feitas as considerações sobre os experimentos que foram realizados.

6.1 Configuração Inicial do IDS

Para a configuração inicial do IDS um subconjunto de requisições foi coletado apenas para estabelecer os valores de limiares de distância de similaridade. O objetivo era coletar 100.000 requisições das primeiras 24 horas para se obter esses limiares, isso foi possível nos testes com a primeira parte de WS1 (3,9% dessas requisições). Porém, a mesma quantidade de requisições de WS2 não foi possível de se obter, pois nas primeiras 24 horas eram um pouco mais do que 30.000 requisições, então somente estas foram utilizadas para os testes (15,9% da primeira parte de WS2). Para encontrar os melhores valores de limiares foram levados em conta os índices de Davies-Bouldin obtidos sobre o resultado dos agrupamentos. Os limiares testados representavam um percentual dissimilaridade[1] de distância entre *strings* de 10% (0,1) a 30% (0,3) (incrementando de um em um por cento), e o algoritmo Jaro-Winkler foi utilizado para calcular as distâncias. Ao final, os limiares de distâncias apresentados na tabela 6.1 foram utilizados nos experimentos realizados com o IDS proposto.

Tabela 6.1: Seleção de limiares de distância de dissimilaridade

Servidor *Web*	**Host**	**UPath**	**UQuery**	**User-Agent**	**Cookie**	**RPath**	**Content**
WS1	0,22	0,14	0,10	0,13	0,18	0,14	0,18
WS2	0,14	0,20	0,15	0,13	0,19	0,14	0,18

[1] O percentual de dissimilaridade é apenas o inverso do percentual de similaridade, ou seja, se o percentual de dissimilaridade for próximo de zero (0%) significa que os elementos são similares.

Para os fatores de ponderação de popularidade (y_p) e de confiabilidade (y_c) foram utilizados os valores apresentados na tabela 6.2. Para o campo *User-Agent* a reputação leva em conta somente o índice de popularidade, porque o índice de confiabilidade privilegia os agentes cujos *hosts* confiáveis variaram de agentes durante os acessos (ex.: *hosts* do tipo *proxy*). No campo *RPath* foi considerado que a origem da referência é importante tanto para a popularidade como para a confiabilidade dos grupos. Nos demais campos foi levado em conta que é preciso ter mais confiabilidade do que popularidade, mais detalhes sobre estes valores estão na quarta etapa da seção 4.1.3.

Tabela 6.2: Seleção de fatores de ponderação para popularidade e confiabilidade

Ponderação	**Method**	**Host**	**UPath**	**UQuery**	**User-Agent**	**Cookie**	**RPath**	**Content**
y_p	1	1	1	1	4	1	2	1
y_c	3	3	3	3	0	3	2	3

6.2 Treinamento do IDS

No trabalho de [Portnoy et al., 2001] o treinamento do IDS foi realizado com 10% dos dados, e depois o modelo treinado foi utilizado na detecção de intrusão em outras partes da base utilizada. Porém, neste trabalho os experimentos foram realizados com 3%, 6%, 9%, 12% e 15% da primeira parte do conjunto de requisições de cada servidor, o objetivo era observar qual percentual se aproximaria melhor de um bom modelo de detecção nestes servidores.

Tabela 6.3: Quantidade de requisições utilizadas para treinamento do IDS

Servidores *web*	**T-3%**	**T-6%**	**T-9%**	**T-12%**	**T-15%**
WS1(parte 1)	75.924	151.849	227.774	303.699	379.624
WS2(parte 1)	5.658	11.317	16.976	22.635	28.294

Na tabela 6.3, cada coluna apresenta a quantidade de requisições coletadas para fazer o treinamento do IDS. Na sequência, as tabelas 6.4 e 6.5 apresentam a quantidade de centróides gerados após aplicar o algoritmo de agrupamento em cada um dos treinamentos realizados.

Tabela 6.4: Quantidade de centróides após o agrupamento (WS1 - parte 1)

WS1(parte 1)	**T-3%**	**T-6%**	**T-9%**	**T-12%**	**T-15%**
Method	5	5	5	5	6
Host	2	2	7	7	7
UPath	192	250	306	376	488
UQuery	39	49	66	80	103
User-Agent	43	51	63	80	93
Cookie	14	18	19	19	23
RPath	31	35	41	43	49
Content	7	8	8	9	9
Total	**333**	**418**	**515**	**619**	**778**

Tabela 6.5: Quantidade de centróides após o agrupamento (WS2 - parte 1)

WS2(parte 1)	**T-3%**	**T-6%**	**T-9%**	**T-12%**	**T-15%**
Method	3	4	5	5	5
Host	9	14	15	16	16
UPath	35	39	44	51	49
UQuery	65	88	107	127	130
User-Agent	31	42	43	54	55
Cookie	23	31	32	30	43
RPath	11	13	16	18	21
Content	8	14	18	20	20
Total	**185**	**245**	**280**	**321**	**339**

6.3 Seleção de Melhor Detecção do IDS

Para a avaliação do agrupamento é necessário estabelecer uma linha de corte *(l)* com o propósito de criar grupos populares, o valor estabelecido para essa linha de corte interfere diretamente nos modelos criados para a detecção de intrusão e afeta as taxas de detecção do IDS. Sendo assim, os modelos de detecção foram treinados e avaliados com diferentes valores para l e aplicados nas 24 horas conseguintes de requisições com o intuito de identificar os melhores índices de medida-F. Esse subconjunto continha 532.160 requisições (21%) da parte 1 do servidor WS1, e 56.323 requisições (29,8%) da parte 1 do servidor WS2.

Os resultados dos testes efetuados com o servidor WS1 que estão na tabela 6.6 apontam para o valor de linha de corte l igual a 0,35 como melhor opção para detecção de intrusão. Estes valores foram obtidos calculando a medida-F (detalhes na seção 5.5) onde o número de alertas de ataques são reduzidos (63 ataques não repetidos) em comparação ao servidor WS2. O número maior de falsos positivos do que verdadeiros positivos foi o que colaborou para que esses índices ficassem baixos.

Tabela 6.6: Resultados de índice de Medida-F (WS1 - parte 1)

WS1(parte 1)	**T-3%**	**T-6%**	**T-9%**	**T-12%**	**T-15%**
$l = 0,05$	0,1143	0,0478	0,0404	0,0429	0,0467
$l = 0,10$	0,1332	0,0503	0,3226	0,0472	0,0462
$l = 0,15$	0,1441	0,2682	0,3237	0,4000	0,0499
$l = 0,20$	0,1921	0,3310	0,3462	0,4074	0,4889
$l = 0,25$	0,1687	0,2945	0,3488	**0,4335**	0,4889
$l = 0,30$	0,2792	0,3310	0,3516	0,3077	0,5146
$l = 0,35$	0,3116	**0,3416**	0,3516	0,1739	**0,5658**
$l = 0,40$	0,3116	**0,3416**	**0,3571**	0,1753	0,5600
$l = 0,45$	**0,3264**	**0,3416**	**0,3571**	0,1767	0,5600

Tabela 6.7: Resultados de índice de Medida-F (WS2 - parte 1)

WS2(parte 1)	**T-3%**	**T-6%**	**T-9%**	**T-12%**	**T-15%**
$l = 0,05$	0,9153	**0,9544**	**0,9571**	**0,9199**	**0,9108**
$l = 0,10$	0,9491	0,9543	0,9529	0,9139	0,9099
$l = 0,15$	0,9497	0,9499	0,9517	0,9088	0,8991
$l = 0,20$	0,9496	0,7674	0,9512	0,9092	0,9040
$l = 0,25$	**0,9545**	0,7606	0,9474	0,2972	0,9015
$l = 0,30$	0,9489	0,7631	0,9472	0,2672	0,8724
$l = 0,35$	0,9489	0,7604	0,9131	0,2675	0,8719
$l = 0,40$	0,9487	0,7604	0,9131	0,2675	0,8718
$l = 0,45$	0,9483	0,7604	0,9131	0,2672	0,8717

Tabela 6.8: Resultados de índice de Medida-F (WS2 - parte 1) - sem o OpenVAS

WS2(parte 1)	**T-3%**	**T-6%**	**T-9%**	**T-12%**	**T-15%**
$l = 0,05$	0,5085	**0,7414**	**0,7887**	**0,7313**	**0,7616**
$l = 0,10$	0,6944	0,7373	0,7506	0,6321	0,6175
$l = 0,15$	0,6954	0,6551	0,7100	0,4249	0,2880
$l = 0,20$	0,6934	0,6718	0,6222	0,4241	0,3080
$l = 0,25$	**0,7020**	0,4122	0,3072	0,3103	0,2872
$l = 0,30$	0,4582	0,4225	0,2995	0,2812	0,2817
$l = 0,35$	0,4582	0,3576	0,2420	0,2660	0,2633
$l = 0,40$	0,4569	0,3576	0,2420	0,2660	0,2629
$l = 0,45$	0,4398	0,3576	0,2420	0,2084	0,2055

Já os resultados obtidos com o servidor WS2 e que estão na tabela 6.7 apresentaram ótimos índices, porém esses índices foram alcançados devido a grande parte dos ataques serem originados pela ferramenta OpenVAS e que elevaram a taxa de precisão no cálculo da medida-F. Para uma melhor análise, na tabela 6.8 os alertas referentes aos ataques realizados pela ferramenta OpenVAS foram ignorados, ou seja, 90,8% dos ataques não foram considerados. Ao final, a linha de corte (l) com valor de 0,05 foi selecionada para fazer a comparação com o IDS Snort.

6.4 Comparação do IDS Proposto com o IDS Snort

O IDS proposto é baseado em anomalias e o IDS Snort é baseado em regras, e ambos foram preparados para detectar intrusão em ambiente *web*. O IDS Snort não precisou de treinamento mas necessitou de uma base de assinaturas de ataques construída por especialistas. O IDS proposto não precisou de uma base de assinaturas, mas precisou ser treinado em um subconjunto de 15% da base de requisições. O treinamento foi realizado nas primeiras 24 horas, e a detecção de intrusão foi feita com os dois IDSs nas horas seguintes da parte 1 dos servidores WS1 e WS2. Em detalhes, foram inspecionadas 1.969.692 requisições (77,8%) da primeira parte do servidor WS1, e 157.440 requisições (83,4%) da primeira parte do servidor WS2.

Tabela 6.9: Comparação com o IDS Snort por Ataque (WS1 - parte 1)

(*l*=0,35)	V.Positivos	F.Negativos	Detecção	Snort	V.Positivos	F.Negativos	Detecção
A01	8	6	**57,1%**	A01	7	7	50,0%
A02	20	2	**90,9%**	A02	12	10	54,5%
A03	4	2	**66,7%**	A03	2[a]	4	33,3%
A04	7	3	**70,0%**	A04	1	9	10,0%
A05	4	1	**80,0%**	A05	0	5	0,0%
A06	0	0	0,0%	A06	0	0	0,0%
A07	2	0	**100,0%**	A07	0	2	0,0%
A08	23	21	**52,3%**	A08	0	44	0,0%
A09	6	3	**66,7%**	A09	0	9	0,0%
A10	0	1	0,0%	A10	0	1	0,0%
Total	74	39	**65,5%**	Total	22	91	19,4%

[a] - Ataques detectados pelo *preprocessor http_inspect*.

Os resultados apresentados na tabela 6.9 mostram que o IDS baseado em anomalia obteve melhores taxas de detecção do que o IDS Snort. Mas na tabela 6.10 é possível observar que o IDS proposto obteve um número mais alto de falsos positivos, onde o operador do IDS iria encontrar um alerta legítimo a cada 4,22 alertas. Porém, o IDS proposto conseguiu superar o IDS Snort no índice de medida-F.

Tabela 6.10: Comparação com o IDS Snort - Medida-F (WS1 - parte 1)

(WS1 - parte 1)	V.Positivos	F.Positivos	F.Negativos	Precisão	Revisão	Medida-F
IDS(*l*=0.35)	74	239	39	0,2364	0,6549	**0,3474**
Snort	22	4	91	0,8462	0,1947	0,3165

De acordo com a tabela 6.11, o IDS proposto apresenta melhores taxas de detecção que o IDS Snort na base de requisições do servidor WS2. Mas nessa tabela também é possível observar que o IDS Snort foi mais preciso na detecção de ataques de injeção de SQL (A01), visto que é um ataque bem conhecido e com várias assinaturas escritas. Já por outro lado, é possível observar que o IDS proposto obteve uma boa taxa de detecção de ataques de varredura (A09), demonstrando que este IDS está apto para detectar ataques novos. O IDS Snort também obteve

uma boa taxa de detecção de ataques de varredura, mas vale ressaltar que a maior parte desses ataques foram detectados com uma assinatura criada para o *User-Agent* do Nessus/OpenVAS.

Tabela 6.11: Comparação com o IDS Snort por Ataque (WS2 - parte 1)

(*l*=0,05)	V.Positivos	F.Negativos	Detecção	Snort	V.Positivos	F.Negativos	Detecção
A01	0	155	0,0%	A01	101	54	**65,2%**
A02	204	23	**89,9%**	A02	119	108	52,4%
A03	19	2	**90.5%**	A03	2[a]	19	9,5%
A04	25	9	**73,5%**	A04	11	23	32,4%
A05	6	13	**31,6%**	A05	0	19	0,0%
A06	24	221	**9,8%**	A06	0	245	0,0%
A07	11	33	25,0%	A07	11	33	25,0%
A08	76	14	**84,4%**	A08	6	84	6,7%
A09	5.840[b]	1.240	**82,5%**	A09	4.535[c]	2.545	64,1%
A10	0	7	0,0%	A10	4	3	**57,1%**
Total	6.205	1.717	**78,3%**	Total	4.789	3.133	60,4%

[a] - Ataques detectados pelo *preprocessor http_inspect*.
[b] - 108 ataques foram detectados e barrados pelo módulo de inspeção de requisições.
[c] - 22 ataques foram detectados pelo *preprocessor http_inspect*.

Tabela 6.12: Comparação com o IDS Snort - Medida-F (WS2 - parte 1)

(**WS2** - parte 1)	V.Positivos	F.Positivos	F.Negativos	Precisão	Revisão	Medida-F
IDS(*l*=0,05)	6.205	176	1.717	0,9724	0,7833	**0,8677**
Snort	4.789	1	3.133	0,9998	0,6045	0,7535

De acordo com os resultados do IDS baseado em anomalia na tabela 6.12, o operador encontraria um alerta de falso positivo a cada 36,2 alertas, mas isso ocorre devido a grande quantidade de ataques do OpenVAS (A09) que foram detectados. Se levar em conta que a melhor detecção deveria aproximar o valor do índice de medida-F para um (1), os dois IDSs tiveram uma melhora na detecção no servidor WS2 em relação ao servidor WS1, porém, a quantidade de ataques detectados do tipo varredura foi o que colaborou para a melhora desses índices. Ao final, o IDS proposto obteve um índice de medida-F melhor do que o IDS Snort.

6.5 Testes Adicionais com o IDS

O objetivo dos testes adicionais foi observar os resultados do IDS proposto onde o treinamento e a detecção de intrusão foram feitos no modo não-supervisionado. Para estes experimentos foram utilizados os mesmos valores de configuração descritos na seção 6.1. Para o treinamento foram utilizadas 364.435 requisições (15%) da segunda parte do servidor WS1, e 34.924 requisições (15%) da segunda parte do servidor WS2. Para o valor de linha de corte foram utilizados os mesmos valores que foram selecionados na seção 6.3, e a quantidade de centróides criados após o agrupamento são apresentados na tabela 6.13.

Tabela 6.13: Quantidade de centróides após o agrupamento (WS1 e WS2 - parte 2)

Servidor *web*	Method	Host	UPath	UQuery	User-Agent	Cookie	RPath	Content
WS1(p2) *l*=0,35	5	7	495	109	93	23	49	9
WS2(p2) *l*=0,05	4	17	50	122	48	35	19	18

Depois do treinamento, os modelos criados foram aplicados na detecção de intrusão nos servidores *web*. Foram inspecionadas 1.825.063 requisições (75,1%) da segunda parte do servidor WS1, e 196.648 requisições (84,4%) da segunda parte do servidor WS2. Os resultados estão na tabela 6.14 onde os resultados obtidos na parte 1 são similares aos obtidos na parte 2. Nesta tabela também é possível observar que na parte 2 do servidor WS2 também ocorreu uma varredura com a ferramenta OpenVAS resultando no aumento do número de verdadeiros positivos.

Tabela 6.14: Resultados obtidos com o IDS nas partes 1 e 2 dos servidores *web*

Servidor *web*	V.Positivos	F.Positivos	Precisão
WS1(parte 1) *l*=0,35	74	239	0,3474
WS1(parte 2) *l*=0,35	177	284	0,3839
WS2(parte 1) *l*=0,05	6.205	176	0,9724
WS2(parte 2) *l*=0,05	6.677	249	0,9640

Já na tabela 6.15 é possível observar que a taxa de falsos positivos (FP) também é similar entre as detecções feitas na parte 1 e parte 2 dos servidores *web*. Essa taxa também mostra que o número de falsos positivos a serem analisados pelo operador tornam esse IDS viável em um ambiente de produção.

Tabela 6.15: Taxas de falsos positivos do IDS nos servidores WS1 e WS2

Servidor *web*	FP Sumarizados	Total de FP	Requisições	Taxa de FP
WS1(parte 1) *l*=0,35	239	401	1.969.692	0,0203%
WS1(parte 2) *l*=0,35	284	383	1.825.063	0,0209%
WS2(parte 1) *l*=0,05	176	597	157.440	0,3791%
WS2(parte 2) *l*=0,05	249	651	196.648	0,3310%

6.6 Considerações

Este capítulo apresentou os principais resultados obtidos nos experimentos com o IDS proposto. O IDS Snort obteve baixo número de falsos positivos, uma característica de IDSs baseados em regras que estão bem configurados com o ambiente de monitoração. Já o IDS baseado em anomalia obteve bons resultados nas taxas de detecção de ataques. Alguns pontos foram observados durante os experimentos e são considerados a seguir:

- É possível observar que as requisições que foram coletadas pelos servidores WS1 e WS2 são diferentes em diversos aspectos: quantidade de requisições, quantidade de ataques,

quantidade de centróides e os diferentes valores selecionados para *l*. Mesmo com bases de requisições diferentes os índices obtidos foram satisfatórios, porém, isso não quer dizer que o IDS proposto irá resultar em bons índices para qualquer tipo de sítio e servidor *web*, seriam necessários mais experimentos em diferentes ambientes.

- Os melhores índices de detecção no servidor WS1 foram obtidos com um treinamento sobre no mínimo de 15% da base de requisições. Já no servidor WS2 desde os 3% já era possível de se obter bons índices.

- Durante os testes foi observado que os valores baixos para *l* resultaram em melhores taxas de detecção de ataques, mas em contrapartida o índice de falsos positivos aumentou. Já com valores mais altos para *l* resultaram em baixos índices de falsos positivos, mas a taxa de detecção também reduziu. Assim o índice de medida-F foi calculado para encontrar o melhor valor de *l* para a detecção de intrusão.

- Também foi possível observar que a taxa de falsos positivos não é expressiva em comparação com a quantidade de requisições que foram inspecionadas, demonstrando que o método aplicado atingiu resultados satisfatórios.

- O IDS baseado em anomalia obteve melhores taxas de detecção na maioria das comparações. Mas a aplicação da medida-F tornou mais justa a comparação entre os IDSs devido aos índices de falsos positivos e falsos negativos. Ao final o IDS proposto obteve melhores índices de medida-F do que o IDS Snort.

Capítulo 7

Conclusão e Trabalhos Futuros

Este capítulo apresenta a conclusão deste trabalho (seção 7.1) e as sugestões para os trabalhos futuros (seção 7.2).

7.1 Conclusão

Apresentou-se aqui uma proposta de arquitetura de IDS baseado em anomalia que utiliza agrupamento de dados para a detecção de ataques em servidores *web*. A comparação com o IDS Snort foi aplicada em um cenário real e com ataques recentes. Nas duas bases de requisições que foram utilizadas, o IDS proposto obteve melhores taxas de detecção na maioria dos ataques. Para o servidor WS1 foi necessário um treinamento com um mínimo de 15% da quantidade total de requisições para se obter um bom índice de medida-F. Para o servidor WS2 um pouco de treinamento (3%) já resultava em um bom modelo de detecção, mas os testes realizados com treinamento em 15% da base apresentaram resultados igualmente satisfatórios. Ao final, o IDS proposto obteve melhores índices de medida-F do que o IDS Snort. Porém, ao detalhar os resultados, o número de ataques detectados pelo IDS proposto foi maior, mas os menores índices de falsos positivos foram obtidos com o IDS Snort.

A colaboração principal deste trabalho está no método heurístico proposto para uma avaliação mais apurada dos grupos de informações criados após a aplicação de um algoritmo de agrupamento. Nos trabalhos relacionados os grupos recebem um rótulo de normalidade ou, no caso de grupos menores (*outliers*), um rótulo de ataque. Neste trabalho foi apresentado como calcular um índice de popularidade e um índice de confiabilidade para cada grupo levando em conta se os *hosts* que efetuaram as requisições eram confiáveis, assim também foi apresentado uma maneira de utilizar esses índices para calcular um índice de reputação, o qual foi utilizado para atribuir rótulos de reputação para os grupos. Ao implementar este método foi possível observar que o número de falsos positivos não foi expressivo, se levar em conta que das milhões de requisições inspecionadas pouco mais de 2 mil foram detectadas como alarme falso.

7.2 Sugestões para Trabalhos Futuros

Para fazer os experimentos com a arquitetura proposta, uma amostra de milhões de requisições HTTP foram coletadas de dois servidores *web*. Com essa amostra foram realizados diversos testes que demonstraram que o IDS proposto é promissor, porém essa amostra não

pode ser considerada a melhor representação de todos os ambientes e sítios *web* existentes. Por isso, uma sugestão é que o IDS proposto seja testado em diferentes conjuntos de requisições de diferentes ambientes *web*. E outra sugestão é a criação de um novo conjunto de requisições constituído de ataques rotulados e recentes, e que possa ser disponibilizado para o público, visto que durante esta pesquisa uma grande dificuldade foi a obtenção de uma base de requisições com essas características.

Mais uma sugestão é a de realizar mais experimentos modificando alguns métodos e ferramentas utilizados nesta arquitetura: testes com outros algoritmos de agrupamento, experimentar outros algoritmos de cálculo de distância de similaridade e, ainda, validação de agrupamento com outros índices. Sugere-se também estudar um meio de melhorar o método heurístico proposto, pois durante as observações experimentais notou-se que o índice de confiabilidade dos *hosts* pode ter uma variação indesejada e afetar negativamente a avaliação do agrupamento. Outra sugestão, ainda, é estudar um meio de implementar uma tomada de decisão mais elaborada para melhorar a detecção de intrusão (por exemplo: estruturas de condição, algoritmos de classificação, etc.), visto que a tomada de decisão utilizada neste trabalho é simples.

Para dar continuidade ao processo de aprendizagem e detecção, também sugere-se um trabalho futuro para que esta arquitetura de IDS possa ser melhorada com o intuito de realizar uma aprendizagem contínua (*active learning*).

Uma última sugestão é implementar o método heurístico proposto em ambientes em que se deseja fazer auditoria de informações visando buscar intrusos após a invasão, visto que este método foi criado durante as observações e atribuições de rótulos na base de requisições.

Referências Bibliográficas

[Almgren and Lindqvist, 2001] Almgren, M. and Lindqvist, U. (2001). Application-Integrated Data Collection for Security Monitoring. In *Recent Advances in Intrusion Detection (RAID 2001)*, LNCS, pages 22–36, Davis, California. Springer.

[Apache, 2009] Apache (2009). ProxyAbuse (http://wiki.apache.org/httpd/ProxyAbuse - acesso em 19/01/2011).

[Ariu, 2010] Ariu, D. (2010). *Host and Network based Anomaly Detectors for HTTP Attacks*. Phd thesis, University of Cagliari, Cagliari (Italy).

[Ariu and Giacinto, 2010] Ariu, D. and Giacinto, G. (2010). HMMPayl: an application of HMM to the analysis of the HTTP Payload. *Journal of Machine Learning Research - Proceedings Track*, 11:81–87.

[Asaka et al., 1999] Asaka, M., Taguchi, A., and Goto, S. (1999). The Implementation of IDA: An Intrusion Detection Agent System. In *Proceedings of the 11th Annual FIRST Conference on Computer Security Incident Handling and Response (FIRST'99)*.

[Axelsson, 2000] Axelsson, S. (2000). Intrusion Detection Systems: A Survey and Taxonomy. Technical Report 99-15, Dept. of Computer Engineering, Chalmers University of Technology.

[Bace and Mell, 2001] Bace, R. and Mell, P. (2001). *Intrusion Detection Systems*. National Institute of Standards and Technology (NIST).

[Balasubramaniyan et al., 1998] Balasubramaniyan, J. S., Garcia-Fernandez, J. O., Isacoff, D., Spafford, E. H., and Zamboni, D. (1998). An Architecture for Intrusion Detection Using Autonomous Agents. In *ACSAC*, pages 13–24.

[Bergman, 2004] Bergman, A. (2004). Net::LibNIDS (http://search.cpan.org/~abergman/Net-LibNIDS-0.01/LibNIDS.pm - acesso em 23/12/2010).

[Bolzoni and Etalle, 2008] Bolzoni, D. and Etalle, S. (2008). Boosting Web Intrusion Detection Systems by Inferring Positive Signatures. In Meersman, R. and Tari, Z., editors, *OTM Conferences (2)*, volume 5332 of *Lecture Notes in Computer Science*, pages 938–955. Springer.

[Bruneau, 2003] Bruneau, G. (2003). The History and Evolution of Intrusion Detection. *SANS InfoSec Reading Room - Intrusion Detection*.

[CDX, 2009] CDX (2009). ITOC Research: CDX Datasets (http://www.itoc.usma.edu/research/dataset/ - acesso em 23/12/2010).

[CELEPAR, 2010] CELEPAR (2010). CELEPAR - Informática do Paraná (http://www.celepar.pr.gov.br/ - acesso em 23/12/2010).

[CERT.br, 2010] CERT.br (2010). Estatísticas do CERT.br (http://www.cert.br/stats/incidentes/ - acesso em 23/12/2010).

[Cheung et al., 1999] Cheung, S., Crawford, R., Dilger, M., Frank, J., Hoagland, J., Levitt, K., Rowe, J., Staniford-Chen, S., Yip, R., and Zerkle, D. (1999). The Design of GrIDS: A Graph-Based Intrusion Detection System. In *Proceedings of the 19th National Information Systems Security Conference*, pages 361–370.

[Chirichiello, 2006] Chirichiello, A. (2006). A Logical Framework for Plan Recognition for Intrusion Detection.

[Cohen et al., 2003] Cohen, W. W., Ravikumar, P., and Fienberg, S. E. (2003). A Comparison of String Distance Metrics for Name-Matching Tasks. In *Proceedings of IJCAI-03 Workshop on Information Integration*, pages 73–78.

[Corona, 2010] Corona, I. (2010). *Detection of Web-based Attacks*. Phd thesis, University of Cagliari, Cagliari (Italy).

[Corona et al., 2009] Corona, I., Ariu, D., and Giacinto, G. (2009). HMM-Web: A Framework for the Detection of Attacks Against Web Applications. In *ICC*, pages 1–6. IEEE.

[Corona and Giacinto, 2010] Corona, I. and Giacinto, G. (2010). Detection of Server-side Web Attacks. *Journal of Machine Learning Research - Proceedings Track*, 11:160–166.

[CPAN, 2011] CPAN (2011). CPAN - Comprehensive Perl Archive Network (http://www.cpan.org/ - acesso em 11/01/2011).

[Criscione et al., 2009] Criscione, C., Salvaneschi, G., Maggi, F., and Zanero, S. (2009). Integrated Detection of Attacks Against Browsers, Web Applications and Databases. In *Proceedings of the 2009 European Conference on Computer Network Defense*, EC2ND '09, pages 37–45, Washington, DC, USA. IEEE Computer Society.

[Cuppens and Ortalo, 2000] Cuppens, F. and Ortalo, R. (2000). LAMBDA: A Language to Model a Database for Detection of Attacks. In *Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection*, RAID '00, pages 197–216, London, UK. Springer-Verlag.

[Dacier and Jackson, 1998] Dacier, M. and Jackson, K. (1998). RAID98: First International workshop on the Recent Advances on Intrusion Detection.

[DARPA, 1998] DARPA (1998). 1998 DARPA Intrusion Detection Evaluation Data Set (http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1998data.html - acesso em 27/12/2010).

[DARPA, 1999] DARPA (1999). 1999 DARPA Intrusion Detection Evaluation Data Set (http://www.ll.mit.edu/mission/communications/ist/corpora/ideval/data/1999data.html - acesso em 29/12/2010).

[Dickerson and Dickerson, 2000] Dickerson, J. E. and Dickerson, J. A. (2000). Fuzzy Network Profiling for Intrusion Detection. In *Proceedings of NAFIPS 19th International Conference of the North American Fuzzy Information Processing Society, Atlanta*, pages 301–306.

[Dokas et al., 2002] Dokas, P., Ertoz, L., Kumar, V., Lazarevic, A., Srivastava, J., and Tan, P. N. (2002). Data Mining for Network Intrusion Detection. In *Proceedings of the NSF Workshop on Next Generation Data Mining,* Baltimore.

[Drupal, 2010] Drupal (2010). Drupal - Open Source CMS (http://drupal.org/ - acesso em 29/12/2010).

[Dusseault, 2007] Dusseault, E. L. (2007). RFC 4918 - HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV).

[e107, 2011] e107 (2011). e107 CMS (http://e107.org - acesso em 19/01/2011).

[EmergingThreats, 2010] EmergingThreats (2010). Emerging Threats.net Open rulesets (http://rules.emergingthreats.net/ - acesso em 23/12/2010).

[Ertöz et al., 2004] Ertöz, L., Eilertson, E., Lazarevic, A., Tan, P. N., Kumar, V., Srivastava, J., and Dokas, P. (2004). *MINDS - Minnesota Intrusion Detection System*. MIT Press.

[Eskin et al., 2002] Eskin, E., Arnold, A., Prerau, M., Portnoy, L., and Stolfo, S. (2002). A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data. In *Applications of Data Mining in Computer Security*.

[Fielding et al., 1999] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and Lee, T. B. (1999). RFC 2616 - HTTP/1.1, the Hypertext Transfer Protocol. http://w3.org/Protocols/rfc2616/rfc2616.html.

[Gan et al., 2007] Gan, G., Ma, C., and Wu, J. (2007). *Data Clustering: Theory, Algorithms, and Applications*. Society for Industrial and Applied Mathematics, Philadephia, PA.

[Garcia-Teodoro et al., 2009] Garcia-Teodoro, P., Díaz-Verdejo, J. E., Maciá-Fernández, G., and Vázquez, E. (2009). Anomaly-based Network Intrusion Detection: Techniques, Systems and Challenges. *Computers & Security*, 28(1-2):18–28.

[Garfinkel and Rosenblum, 2003] Garfinkel, T. and Rosenblum, M. (2003). A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *NDSS*. The Internet Society.

[Gartner, 2011] Gartner (2011). Magic Quadrant Research Metodology (http://www.gartner.com/technology/research/methodologies/research_mq.jsp - acesso em 15/01/2011).

[Geib and Goldman, 2001] Geib, C. W. and Goldman, R. P. (2001). Plan Recognition in Intrusion Detection Systems. In *DARPA Information Survivability Conference and Exposition (DISCEX)*.

[Goland et al., 1999] Goland, Y., Whitehead, E., Faizi, A., Carter, S., and Jensen, D. (1999). RFC 2518 - HTTP Extensions for Distributed Authoring – WEBDAV.

[Guan et al., 2003] Guan, Y., Ghorbani, A. A., and Belacel, N. (2003). Y-means: A Clustering Method for Intrusion Detection. In *Proceedings of Canadian Conference on Electrical and Computer Engineering*, pages 1083–1086.

[Hofmeyr et al., 1998] Hofmeyr, S. A., Forrest, S., and Somayaji, A. (1998). Intrusion Detection Using Sequences of System Calls. *Journal of Computer Security*, 6(3):151–180.

[Howard and Longstaff, 1998] Howard, J. D. and Longstaff, T. A. (1998). A Common Language for Computer Security Incidents.

[HP, 2009] HP (2009). HP to Acquire 3Com (http://www.hp.com/hpinfo/newsroom/press/2009/091111xa.html - acesso em 15/01/2011).

[Hrivnak, 2002] Hrivnak, A. (2002). Host Based Intrusion Detection: An Overview of Tripware and Intruder Alert.

[IBM, 2006] IBM (2006). IBM to Acquire Internet Security Systems (http://www-03.ibm.com/press/us/en/pressrelease/20164.wss, acesso em 05/12/2010).

[IBM, 2007] IBM (2007). IBM Proventia Network Anomaly Detection System (ADS), (http://www-935.ibm.com/services/uk/index.wss/offering/iss/y1026942, acesso em 05/12/2010).

[iCTF, 2008] iCTF (2008). The 2008 iCTF Data (http://ictf.cs.ucsb.edu/data/ictf2008/ - acesso em 23/12/2010).

[Ilgun, 1993] Ilgun, K. (1993). USTAT: A Real-Time Intrusion Detection System for UNIX. In *Proceedings of the 1993 IEEE Symposium on Security and Privacy*, SP '93, pages 16–28, Washington, DC, USA. IEEE Computer Society.

[Ilgun et al., 1995] Ilgun, K., Kemmerer, R. A., and Porras, P. A. (1995). State Transition Analysis: A Rule-Based Intrusion Detection Approach. *IEEE Transactions on Software Engineering*, 21:181–199.

[Ingham, 2007] Ingham, K. L. (2007). *Anomaly Detection for HTTP Intrusion Detection: Algorithm Comparisons and the Effect of Generalization on Accuracy*. PhD thesis, The University of New Mexico.

[Ingham and Inoue, 2007] Ingham, K. L. and Inoue, H. (2007). Comparing Anomaly Detection Techniques for HTTP. In *Proceedings of the 10th international conference on Recent advances in intrusion detection*, RAID'07, pages 42–62, Berlin, Heidelberg. Springer-Verlag.

[Ingham et al., 2007] Ingham, K. L., Somayaji, A., Burge, J., and Forrest, S. (2007). Learning DFA representations of HTTP for protecting web applications. *Computer Networks*, 51(5):1239–1255.

[Innella, 2001] Innella, P. (2001). The Evolution of Intrusion Detection Systems (http://www.securityfocus.com/infocus/1514 - acesso em 05/12/2010). *SecurityFocus*.

[Jain, 2010] Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8):651–666.

[KDD, 1999] KDD (1999). KDD Cup 1999 databases (http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html - acesso em 23/12/2010).

[Kemmerer and Vigna, 2002] Kemmerer, R. and Vigna, G. (2002). Intrusion Detection: A Brief History and Overview. *IEEE Computer*, pages 27–30. Special publication on Security and Privacy.

[Kemmerer, 1998] Kemmerer, R. A. (1998). NSTAT: A Model-based Real-time Network Intrusion Detection System. Technical report, Santa Barbara, CA, USA.

[Kendall, 1999] Kendall, K. (1999). A Database of Computer Attacks for the Evaluation of Intrusion Detection Systems. In *DARPA Off-line Intrusion Detection Evaluation, Proceedings DARPA Information Survivability Conference and Exposition (DISCEX)*, volume 2, pages 12–26.

[Kourai and Chiba, 2005] Kourai, K. and Chiba, S. (2005). HyperSpector: Virtual Distributed Monitoring Environments for Secure Intrusion Detection. In *Proceedings of the 1st ACM/USENIX international conference on Virtual execution environments*, VEE '05, pages 197–207, New York, NY, USA. ACM.

[Krawczyk, 2007] Krawczyk, P. (2007). Intrusion Detection and Prevention Tree (http://ipsec.pl/kryptografia/intrusion-detectionprevention-systems-classification-tree.html, acesso em 05/12/2010).

[Kruegel et al., 2003] Kruegel, C., Mutz, D., Robertson, W., and Valeur, F. (2003). Bayesian Event Classification for Intrusion Detection. In *Proceedings of the 19th Annual Computer Security Applications Conference*, ACSAC '03, pages 14–23, Washington, DC, USA. IEEE Computer Society.

[Kruegel and Vigna, 2003] Kruegel, C. and Vigna, G. (2003). Anomaly Detection of Web-based Attacks. In *Proceedings of the 10th ACM conference on Computer and communications security*, CCS '03, pages 251–261, New York, NY, USA. ACM.

[Kruegel et al., 2005] Kruegel, C., Vigna, G., and Robertson, W. (2005). A Multi-model Approach to the Detection of Web-based Attacks. *Computer Networks*, 48:717–738.

[Kumar, 1995] Kumar, S. (1995). *Classification and Detection of Computer Intrusions*. PhD thesis, West Lafayette, IN, USA.

[Leung and Leckie, 2005] Leung, K. and Leckie, C. (2005). Unsupervised Anomaly Detection in Network Intrusion Detection using Clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science - Volume 38*, ACSC '05, pages 333–342, Darlinghurst, Australia. Australian Computer Society, Inc.

[Li, 2004] Li, W. (2004). Using Genetic Algorithm for Network Intrusion Detection. In *Proceedings of the United States Department of Energy Cyber Security Group 2004 Training Conference*, pages 24–27.

[Lindqvist and Porras, 2001] Lindqvist, U. and Porras, P. A. (2001). eXpert-BSM: A Host-Based Intrusion Detection Solution for Sun Solaris. In *ACSAC*, pages 240–251. IEEE Computer Society.

[Lunt et al., 1988] Lunt, T. F., Jagannathan, R., Lee, R., Listgarten, S., Edwards, D. L., Neumann, P. G., Javitz, H. S., Valdes, A., Lunt, T. F., Jagannathan, R., Lee, R., Listgarten, S., Edwards, D. L., Neumann, P. G., Javitz, H. S., and Valdes, A. (1988). IDES: The Enhanced Prototype - A Real-Time Intrusion-Detection Expert System. Technical report, SRI International, 333 Ravenswood Avenue, Menlo Park.

[Lydon, 2004] Lydon, A. (2004). Compilation for Intrusion Detection Systems. Master's thesis, College of Engineering and Technology of Ohio University, Ohio.

[Maggi, 2009] Maggi, F. (2009). *Integrated Detection of Anomalous Behavior of Computer Infrastructures*. PhD thesis, Politecnico di Milano - Dipartimento di Elettronica e Informazione, Milano.

[Maggi et al., 2009] Maggi, F., Robertson, W., Kruegel, C., and Vigna, G. (2009). Protecting a Moving Target: Addressing Web Application Concept Drift. In *Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*, RAID '09, pages 21–40, Berlin, Heidelberg. Springer-Verlag.

[Mahoney et al., 2003] Mahoney, M. V., Chan, P. K., and Arshad, M. H. (2003). A Machine Learning Approach to Anomaly Detection. Technical Report CS-2003-06, Department of Computer Science, Florida Institute of Technology, Melbourne, FL.

[Mandujano, 2004] Mandujano, S. (2004). A Multiagent Approach to Outbound Intrusion Detection.

[McAfee, 2010] McAfee (2010). McAfee positioned as a Leader in Gartner Magic Quadrant for Network Intrusion Prevention Systems (http://resources.mcafee.com/content/NIPS_MQ_2010, acesso em 18/01/2011).

[Merriam-Webster, 2011] Merriam-Webster (2011). Merriam-Webster Dictionary (http://www.merriam-webster.com/dictionary/ - acesso em 15/01/2011).

[Mistrut and Goldberg, 2003] Mistrut, D. and Goldberg, J. (2003). Text::LevenshteinXS (http://search.cpan.org/dist/Text-LevenshteinXS/LevenshteinXS.pm - acesso em 23/12/2010).

[Nessus, 2011] Nessus (2011). Nessus Vulnerability Scanner (http://www.nessus.org/ - acesso em 05/01/2011).

[Nikto, 2011] Nikto (2011). Nikto Open Source web server scanner (http://www.cirt.net/nikto2 - acesso em 05/01/2011).

[OpenVAS, 2011] OpenVAS (2011). OpenVAS - Open Vulnerability Assessment System Community Site (http://www.openvas.org/ - acesso em 05/01/2011).

[OWASP, 2011] OWASP (2011). Open Web Application Security Project (http://www.owasp.org/ - acesso em 05/01/2011).

[Peng et al., 2007] Peng, T., Leckie, C., and Ramamohanarao, K. (2007). Information Sharing for Distributed Intrusion Detection Systems. *Journal of Network and Computer Applications*, 30(3):877–899.

[PerlDoc, 2011] PerlDoc (2011). Perl Programming Documentation (http://perldoc.perl.org/perlintro.html - acesso em 11/01/2011).

[Petrović, 2006] Petrović, S. (2006). A Comparison Between the Silhouette Index and the Davies-Bouldin Index in Labelling IDS Clusters. *Proceedings of the 11th Nordic Workshop of Secure IT*, pages 53–64.

[Porras and Neumann, 1997] Porras, P. A. and Neumann, P. G. (1997). EMERALD: Event Monitoring Enabling Responses to Anomalous Live Disturbances. In *Proceedings of the 1997 National Information Systems Security Conference (NISSC)*, pages 353–365.

[Portnoy et al., 2001] Portnoy, L., Eskin, E., and Stolfo, S. (2001). Intrusion Detection with Unlabeled Data Using Clustering. In *Proceedings of ACM Workshop on Data Mining Applied to Security*.

[Qu et al., 2005] Qu, G., Hariri, S., and Yousif, M. (2005). Multivariate Statistical Analysis for Network Attacks Detection. In *Proceedings of the ACS/IEEE 2005 International Conference on Computer Systems and Applications*, AICCSA '05, page 9, Washington, DC, USA. IEEE Computer Society.

[Ramadas et al., 2003] Ramadas, M., Ostermann, S., and Tjaden, B. (2003). Detecting Anomalous Network Traffic with Self-organizing Maps. In *Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection, LNCS*, pages 36–54. Springer Verlag.

[Rijsbergen, 1979] Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth.

[Robertson, 2009] Robertson, W. (2009). *Detecting and Preventing Attacks Against Web Applications*. PhD thesis, UC Santa Barbara.

[Robertson et al., 2010] Robertson, W., Maggi, F., Kruegel, C., and Vigna, G. (2010). Effective Anomaly Detection with Scarce Training Data. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA.

[Robertson et al., 2006] Robertson, W., Vigna, G., Kruegel, C., and Kemmerer, R. A. (2006). Using Generalization and Characterization Techniques in the Anomaly-based Detection of Web Attacks. In *Proceedings of the 13 th Symposium on Network and Distributed System Security (NDSS)*.

[Roesch, 1999] Roesch, M. (1999). Snort - Lightweight Intrusion Detection for Networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, Berkeley, CA, USA. USENIX Association.

[Romang, 2010] Romang, E. (2010). ByroeNet / Casper Bot Search – e107 RCE scanner (http://eromang.zataz.com/2010/07/13/byroenet-casper-bot-search-e107-rce-scanner/ - acesso em 19/01/2011).

[Singh et al., 2009] Singh, G., Masseglia, F., Fiot, C., Marascu, A., and Poncelet, P. (2009). Mining Common Outliers for Intrusion Detection. In Guillet, F., Ritschard, G., Zighed, D. A., and Briand, H., editors, *EGC (best of volume)*, volume 292 of *Studies in Computational Intelligence*, pages 217–234. Springer.

[Snapp et al., 1991] Snapp, S. R., Brentano, J., Dias, G. V., Goan, T. L., Heberlein, L. T., lin Ho, C., Levitt, K. N., Mukherjee, B., Smaha, S. E., Grance, T., Teal, D. M., and Mansur, D. (1991). DIDS (Distributed Intrusion Detection System) - Motivation, Architecture, and An Early Prototype. In *Proceedings of the 14th National Computer Security Conference*, pages 167–176.

[Snort, 2011] Snort (2011). Snort Users Manual (http://www.snort.org/docs - acesso em 15/01/2011).

[Song et al., 2009] Song, Y., Keromytis, A. D., and Stolfo, S. J. (2009). Spectrogram: A Mixture-of-Markov-Chains Model for Anomaly Detection in Web Traffic. In *NDSS*. The Internet Society.

[Stiennon, 2004] Stiennon, R. (2004). Gartner's Magic Quadrant for Intrusion Detection Systems, 2H03.

[Storløkken, 2007] Storløkken, R. (2007). Labelling Clusters in an Anomaly based IDS by means of Clustering Quality Indexes. Master's thesis, Faculty of Computer Science and Media Technology - Gjøvik University College, Box 191, N-2802 Gjøvik, Norway.

[Tan et al., 2005] Tan, P.-N., Steinbach, M., and Kumar, V. (2005). *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Tjaden et al., 2000] Tjaden, B., Welch, L., Ostermann, S., Chelberg, D., Balupari, R., Bykova, M., Mitchell, A., Lissitsyn, D., Tong, L., Masters, M., Werme, P., Marlow, D., Chapell, B., and Iv, P. I. (2000). INBOUNDS: The Integrated Network-Based Ohio University Network Detective.

[Tseng et al., 2003] Tseng, C., Balasubramanyam, P., Ko, C., Limprasittiporn, R., Rowe, J., and Levitt, K. (2003). A Specification-based Intrusion Detection System for AODV. In *Proceedings of the 1st ACM workshop on Security of ad hoc and sensor networks*, pages 125–134. ACM Press.

[Tucker et al., 2007] Tucker, C., Furnell, S., Ghita, B., and Brooke, P. (2007). A New Taxonomy for Comparing Intrusion Detection Systems. *Internet Research*, 17:88–98.

[Uppuluri and Sekar, 2001] Uppuluri, P. and Sekar, R. (2001). Experiences with Specification-Based Intrusion Detection. In *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, RAID '00, pages 172–189, London, UK. Springer-Verlag.

[Valdes and Skinner, 2000] Valdes, A. and Skinner, K. (2000). Adaptive, Model-based Monitoring for Cyber Attack Detection. In Debar, H., Me, L., and Wu, F., editors, *Recent Advances in Intrusion Detection (RAID 2000)*, number 1907 in Lecture Notes in Computer Science, pages 80–92, Toulouse, France. Springer-Verlag.

[Vigna and Kemmerer, 1998] Vigna, G. and Kemmerer, R. A. (1998). NetSTAT: A Network-Based Intrusion Detection Approach. In *Proceedings of the 14th Annual Computer Security Applications Conference*, page 25, Washington, DC, USA. IEEE Computer Society.

[Viinikka et al., 2006] Viinikka, J., Debar, H., Mé, L., and Séguier, R. (2006). Time Series Modeling for IDS Alert Management. In *Proceedings of the ACM Symposium on InformAtion, Computer and Communications Security(AsiaCCS'06)*, pages 102–113, Taipei, Taiwan.

[Weng, 2009] Weng, S.-C. (2009). Text::JaroWinkler (http://search.cpan.org/~scw/Text-JaroWinkler-0.1/JaroWinkler.pm - acesso em 23/12/2010).

[Williams et al., 2001] Williams, P. D., Anchor, K. P., Bebo, J. L., Gunsch, G. H., and Lamont, G. D. (2001). CDIS: Towards a Computer Immune System for Detecting Network Intrusions. In *RAID '00: Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection*, pages 117–133, London, UK. Springer-Verlag.

[Wojtczuk, 2010] Wojtczuk, R. (2010). LibNIDS (http://libnids.sourceforge.net/ - acesso em 23/12/2010).

[Wood and Erlinger, 2007] Wood, M. and Erlinger, M. (2007). RFC 4766 - Intrusion Detection Message Exchange Requirements. RFC 4766 (Informational).

[XOOPS, 2010] XOOPS (2010). XOOPS CMS (Content Management System) (http://www.xoops.org/ - acesso em 29/12/2010).

[Yahalom, 2008] Yahalom, S. (2008). URI Anomaly Detection using Similarity Metrics. Master's thesis, Tel-Aviv University, Tel-Aviv.

[Ye et al., 2002] Ye, N., Member, S., Emran, S. M., Chen, Q., and Vilbert, S. (2002). Multivariate Statistical Analysis of Audit Trails for Host-Based Intrusion Detection. *IEEE Transactions on Computers*, 51:810–820.

[Yeung and Ding, 2003] Yeung, D.-Y. and Ding, Y. (2003). Host-Based Intrusion Detection Using Dynamic and Static Behavioral Models. *Pattern Recognition*, 36:229–243.

[Young and Pescatore, 2006] Young, G. and Pescatore, J. (2006). Magic Quadrant for Network Intrusion Prevention System Appliances, 2H06.

[Zhang et al., 2005] Zhang, Y.-F., Xiong, Z.-Y., and Wang, X.-Q. (2005). Distributed Intrusion Detection based on Clustering. *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, 4:2379–2383 Vol. 4.

[Zhong et al., 2007] Zhong, S., Khoshgoftaar, T., and Seliya, N. (2007). Clustering-based Network Intrusion Detection. *International Journal of Reliability, Quality and Safety Engineering*, 14(2):169–187.

[Zone-H, 2010] Zone-H (2010). Zone-H.org - Unrestricted information - Yearly Monthly Daily attacks (http://www.zone-h.org/stats/ymd - acesso em 23/12/2010).