DETECÇÃO DA INTRUSÃO UTILIZANDO CLASSIFICAÇÃO BAYESIANA

Roberto Bomeny Maia

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Aprovada por:

Prof. Jorge Lopes de Souza Leão, Dr.Ing.

Prof. Luiz Pereira Caloba, Dr.Ing.

Prof. Flávio Joaquim de Souza, Dr.

RIO DE JANEIRO, RJ - BRASIL

ABRIL DE 2005

MAIA, ROBERTO BOMENY

Detecção da Intrusão Utilizando Classificação Bayesiana [Rio de Janeiro] 2005

XIX, 139 p. 29,7 cm (COPPE/UFRJ, M.Sc., Engenharia Elétrica, 2005)

Tese - Universidade Federal do Rio de Janeiro, COPPE

1. Segurança da Informação

2. Redes de Computadores

3. Modelos Probabilísticos

4. Detecção da Intrusão 5. Redes Bayesianas

I. COPPE/UFRJ II. Titulo (série)

A minha mãe Nadege, minha avó Celina, minha família e amigos.
Obrigado por me terem apoiado nesse desafio.

Agradecimentos

Agradeço a minha avó Celina, pois sem sua atenção no início dos meus estudos jamais teria chegado até aqui.

Ao meu companheiro de mestrado, Alexandre, que me incentivou a finalizar esse trabalho.

Ao meu irmão que me deu inspiração para entrar no mestrado.

Aos meus verdadeiros amigos que foram pacientes o suficiente para suportar a minha dedicação e ausência.

Muito obrigado ao meu orientador, aos colegas, professores e membros da banca pela oportunidade.

Agradeço em especial a Vânia, por estar ao meu lado nesse momento tão importante da minha vida.

Aos meus companheiros de trabalho da Embratel que me ajudaram muito em todo esse caminho.

Obrigado à UFRJ, COPPE e seus funcionários pela colaboração.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

DETECÇÃO DA INTRUSÃO UTILIZANDO CLASSIFICAÇÃO BAYESIANA

Roberto Bomeny Maia

Abril/2005

Orientador: Jorge Lopes de Souza Leão
Programa: Engenharia Elétrica

O crescimento assustador dos sistemas computacionais conectados em rede e a oferta cada vez maior dos serviços oferecidos pela Internet, trazem às instituições o desafio de oferecê-los de maneira confiável, íntegra e disponível. Esses três alicerces definem o conceito de segurança, que tem sido o objeto de pesquisa no intuito de encontrar arquiteturas que consigam identificar as tentativas de uso não autorizado ou indevido e intrusões nos ambientes computacionais. A pesquisa de algoritmos de detecção de intrusão tem evoluído muito nos últimos anos, mas ao mesmo tempo em que são eficientes em detectar os ataques, são ineficientes, pois geram uma quantidade muito grande de alarmes. Assim, o desafio na construção desses algoritmos está em criar um modelo suficientemente robusto e capaz de reduzir a imensa quantidade de eventos gerados, melhorando a qualidade dos alarmes resultantes. O objetivo desse trabalho foi implementar um sistema de detecção de intrusão, chamado NBIS (Naïve Bayes Inference System), usando os métodos de indução probabilística e inferência Bayesiana para desenvolver um classificador que pudesse ser treinado para identificar três tipos diferentes de ataques: Guest, Neptune e Portsweep. Os testes utilizaram os dados simulados gerados pelo Laboratório Lincoln do MIT. Os resultados dessa tese comprovaram a eficiência do modelo proposto do classificador Naïve Bayes na detecção da Intrusão.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

INTRUSION DETECTION USSING BAYESIAN CLASSIFIER

Roberto Bomeny Maia

Abril/2005

Advisors: Jorge Lopes de Souza Leão

Department: Electrical Engineering

The hardwired frightful growth of network computational systems and the offer of services over the Internet bring to companies the challenge to present them with availability, confidentiality and integrity. These three foundations define the security concept, that it has been the object of research in an intention to find architectures that identify the attempts of not authorized or improper use and intrusions in computational environments. The researches of intrusion detention algorithms have growth last years, but at the same time where they are efficient in detecting the attacks, they are inefficient, therefore they generate a great amount of alarms. Thus, the challenge in constructing these algorithms is to create a model enough robust and capable to reduce the immense amount of generated events, improving the quality of the resultant alarms. The objective of this work was to implement a intrusion detection system, call NBIS (Naïve Bayes Inference System), using methods of probabilistic induction and Bayesiana inference to develop a classifier that could be trained to identify three different types of attacks: Guest, Neptune and Portsweep. The tests had used the simulated data generated by MIT Lincoln Laboratory. The results of this thesis had proven the efficiency of the Naïve Bayes classifier model for intrusion detention.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de acrônimos

CERT :	*Computer Emergency Readiness Team*;
IA :	*Inteligência Artificial*;
IDS :	*Intrusion Detection System*;
SDI :	*Sistema de Detecção de Intrusão*;
NBIS :	*Naïve Bayes Inference System*;
HIDS :	*Host Intrusion Detections System*;
NIDS :	*Network Intrusion Detections System* ;
FDP :	*Função de Distribuição de Probabilidade*;
MAP :	*Máximo A Posteriori*;
ML :	*Máximo Likelihood*;
DAG :	*Directed Acyclic Graph*;
DARPA :	*Defense Advanced Research Projects Agency*;
DoS :	*Denial of Service*;
R2L :	*Remote-to-Local*;
U2R :	*User-to-Root*;
FP :	*Falso Positivo*;
FN :	*Falso Negativo*;
RP :	*Real Positivo*;
RN :	*Real Negativo*;

Capítulo 1

Introdução

Com o crescimento significativo dos sistemas computacionais em rede, principalmente os conectados à Internet, foi observado, um aumento exponencial dos eventos de segurança nos últimos 5 anos, conforme observado na Figura 1.1. Com a valorização dos conteúdos digitais, ou seja, a informação, e a oferta cada vez maior de serviços na rede pública, diferentes mecanismos de controle têm sido desenvolvidos com o objetivo de protegê-los.

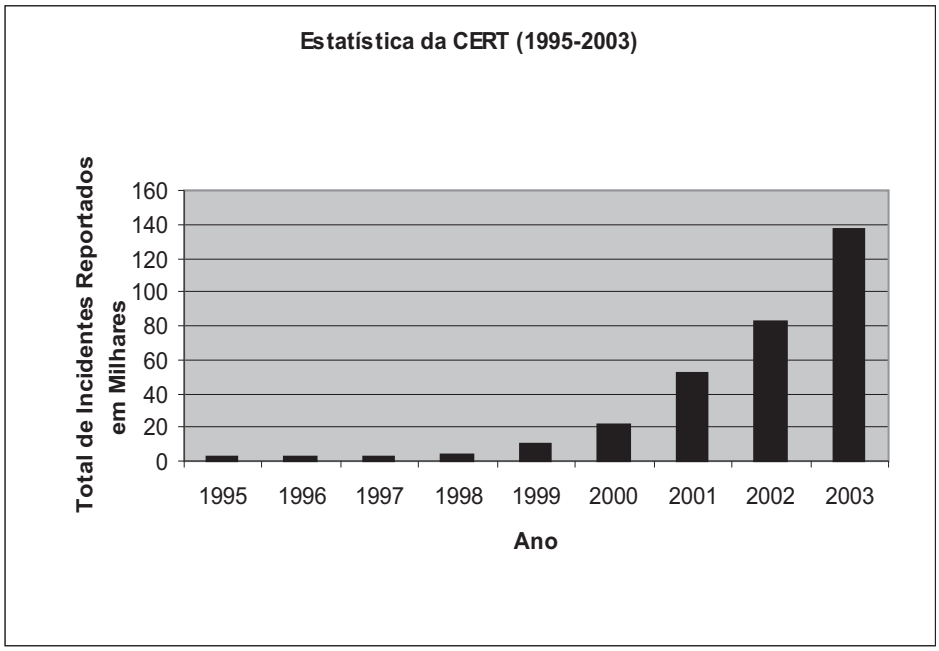


Figura 1.1: Histórico do total de incidentes reportados à CERT de 1995 a 2003

O aprimoramento constante dos ataques e as facilidades computacionais que os atacantes possuem, constitui um cenário desafiador na construção de sistemas que possam detectar, prevenir e proteger os ambientes computacionais atuais.

1.1 Motivação

Segundo o centro nacional de segurança da computação (NCSC - National Computer Security Center) [1] [2] um ambiente computacional ou de rede é considerado seguro desde que sejam estabelecidos mecanismos que garantam a confidencialidade e a integridade dos dados e/ou comunicações sem que as mesmas se tornem indisponíveis aos usuários.

Na ciência da computação existem hoje 3 tipos diferentes de tecnologia que podem ser utilizadas para garantir a segurança de um ambiente computacional, são elas: Firewall, Sistemas de Detecção de Vírus e Detectores de Intrusão.

O Firewall trabalha sobre regras preestabelecidas pela política de segurança permitindo ou bloqueando os tráfegos que passam por ele, sem qualquer julgamento sobre o conteúdo deste. Ele é um elemento ativo na rede e pode influenciar diretamente na disponibilidade da informação em caso de falha.

Os Sistemas de Detecção de Vírus têm a função de verificar o conteúdo de arquivos e mensagens que trafegam na rede à procura de padrões preestabelecidos (assinaturas) de não conformidade, bloqueando-os e comunicando o evento ao usuário. Geralmente estão configurados nas estações de trabalho e nos servidores.

Os Sistemas de Detecção de Intrusão (SDI) são elementos passivos da rede e trabalham como um coletor e analisador de tráfego, procurando por evidências de uso indevido geralmente baseados em regras ou comportamentos.

Detecção de Intrusão é uma área recente de estudo que propõe prover maior segurança aos sistemas computacionais e às redes computacionais atuais, permitindo que elas operem em modo aberto. O objetivo da detecção de intrusão é detectar, preferencialmente em tempo real, o uso não autorizado ou indevido e abusos do ambiente computacional tanto

interno quanto externo [3]. Sua implementação tem sido um desafio nos dias de hoje devido à proliferação das redes que permitem o acesso aos usuários externos, tornando fácil aos invasores evitar qualquer tipo de identificação.

Dentro deste cenário a detecção da intrusão torna-se um poderoso e necessário instrumento para prover maior segurança aos sistemas computacionais e às redes de computadores, permitindo o estabelecimento de uma política que reconheça de maneira rápida e eficiente as novas ameaças e assim realize as devidas alterações nos processos existentes para eliminá-las.

Os sistemas atuais de detecção de intrusão (SDI) são na sua grande maioria baseados em regras ou assinaturas previamente estabelecidas por um especialista. Esse modelo supõe que aos ataques sejam conhecidos e mapeados a priori para que possam ser detectados. Alguns sistemas permitem compor as regras de maneira a detectar pequenas variações das assinaturas conhecidas, mas não possuem qualquer mecanismo para identificar novo tipo de ataque. Existem estudos sendo realizados, propondo algoritmos mais flexíveis e inteligentes que possam detectar ataques conhecidos, variações destes e novos ataques, sem que seja necessária a intervenção do especialista. Este modelo ainda está longe de se tornar realidade, mas os trabalhos com inteligência artificial (IA) têm se mostrado capazes de auxiliar os pesquisadores nessa tarefa.

Os algoritmos existentes para detecção da intrusão, além de detectarem os ataques, o que chamamos de real positivo, também geram uma quantidade muito grande de alarmes de situações que não são evidências de ataque e são denominados de falsos positivos. Assim, o desafio na construção desses algoritmos está em criar um modelo suficientemente robusto e capaz de reduzir a imensa quantidade de eventos gerados, melhorando a qualidade dos alarmes resultantes.

1.2 Objetivo

Nesse cenário desconhecido e altamente mutante da detecção da intrusão o desafio é achar o equilíbrio entre detectar uma evidência real de ataque sem gerar quantidades

absurdas de eventos falsos ou falsos alarmes. Com esse intuito foi implementado e testado nesse trabalho um modelo de detecção da intrusão utilizando-se um classificador com algoritmo de inferência probabilística, chamado Classificador Naïve Bayes.

Esse modelo utiliza como base o teorema de Bayes, cujo nome vem do seu criador Thomas Bayes, um matemático do século XVIII, que teve seu postulato publicado em 1763, 3 anos após a sua morte. A essência da abordagem Bayesiana é prover regras que expliquem como mudar o grau de crença atual em uma hipótese quando da constatação de novas evidências. Sua aplicação tem sido observada em diversas áreas do conhecimento, desde a medicina à computação, sendo considerado um ótimo método para lidar com a incerteza e a imprecisão, que são fenômenos relacionados com a detecção da intrusão.

O objetivo desse trabalho foi implementar um sistema de detecção de intrusão, chamado NBIS (Naïve Bayes Inference System), usando os métodos de indução probabilística e inferência Bayesiana para desenvolver um classificador que pudesse ser treinado para identificar três tipos diferentes de ataques (Guest, Neptune e Portsweep) utilizando dados simulados gerados pelo Laboratório Lincoln do MIT em um projeto apoiado pela agência DARPA cujos resultados serão denominados de agora em diante como DARPA 98 e DARPA 99.

O NBIS foi concebido em 4 módulos, são eles: aquisição e carga de dados, treinamento, classificação e saída de relatórios, que serão detalhados nos capítulos a seguir.

A base de dados do DARPA 98 e DARPA 99, apesar de todos os seus problemas, é freqüentemente utilizada para testes de desempenho em sistemas de detecção de intrusão e foi escolhida nesse trabalho com o intuito de permitir a comparação com outras pesquisas acadêmicas.

Técnicas de mineração de dados (Data Mining) foram necessárias para lidar com a imensa quantidade de informação, 10 Giga bytes de dados comprimidos. Várias transformações e adequações foram necessárias até que se obtivesse u'a massa de dados para treinamento e teste que fosse suficiente para realizar os experimentos.

A classificação utilizada no NBIS é uma variação simplificada da classificação com redes Bayesianas chamada Naïve Bayes que utiliza como fundamento principal a inde-

pendência condicional entre as suas variáveis, as quais compõem os vetores de características utilizadas na detecção dos ataques. A escolha correta desses vetores é fator decisivo no desempenho do SDI, como será visto no decorrer desse trabalho.

Com os resultados dessa tese, acredita-se que se comprove a eficiência do modelo do classificador Naïve Bayes proposto, permitindo que o mesmo possa ser utilizado dentro de uma arquitetura nova ou já existente de detecção da intrusão.

Esse trabalho está organizado da seguinte maneira: o Capítulo 2 apresenta uma visão geral sobre segurança e apresenta os trabalhos desenvolvidos na área de detecção da intrusão; o Capítulo 3 apresenta o teorema de Bayes e o classificador Naïve Bayes. Toda a parte de aquisição e levantamento dos dados, e a definição dos vetores de características são apresentados no Capítulo 4. O Capítulo 5 apresenta a arquitetura do sistema NBIS e o treinamento do classificador Naïve Bayes. No Capítulo 6 são apresentados os resultados dos testes e por fim, no Capítulo 7, a conclusão e os trabalhos futuros.

Capítulo 2

Aspectos Gerais de Segurança

2.1 Introdução

Os requisitos fundamentais de segurança de um ambiente computacional são caracterizados pela confiabilidade, integridade e confidencialidade.

A confiabilidade trata da capacidade do ambiente computacional trabalhar ininterruptamente sem falhas, comportando-se do mesmo modo no qual foi configurado inicialmente, sendo imune a negação de serviço, ou seja, quando a capacidade do sistema atinge um nível inferior ao permitido, não possibilitando o acesso aos recursos oferecidos [4].

A integridade garante que os dados e a comunicação no ambiente computacional não foram alterados ou corrompidos indevidamente, ou seja, se preocupa com a exatidão, fidelidade, não corrupção e a certeza da transferência da informação entre as entidades. Já a confidencialidade garante que os recursos do ambiente computacional estão disponíveis somente para pessoas autorizadas.

Então podemos definir que segurança de sistemas e redes é o meio de garantir a confiabilidade, a integridade e a confidencialidade de um ambiente computacional. O processo de gerenciamento de segurança propõe um modelo suficientemente robusto que garanta esses fundamentos básicos, como apresentado na Figura 2.1.

Como pode ser observada na Figura 2.1 a detecção de problemas é o segundo nível

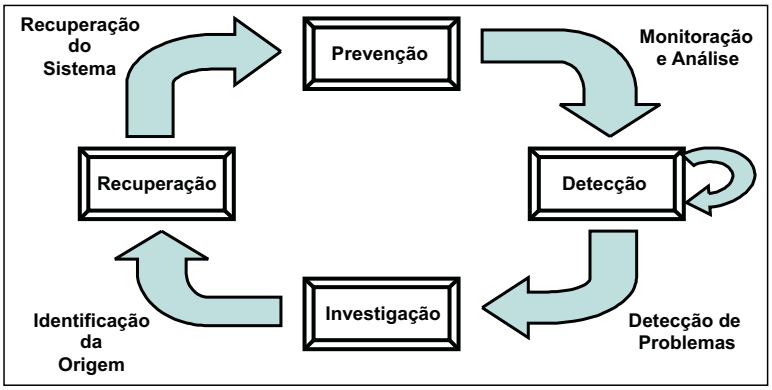


Figura 2.1: Modelo de gerenciamento de segurança

no modelo de segurança e é neste contexto que a detecção de intrusão se aplica. Segundo BACE [5] define-se detecção de intrusão como o processo de monitorar eventos que ocorrem em um sistema computacional ou rede de computadores, analisando-os em busca de evidências de segurança.

Neste capítulo será apresentada uma revisão geral dos sistemas de detecção da intrusão e as tecnologias utilizadas neles e uma visão atual do que está sendo desenvolvido hoje na área.

2.2 Sistemas de Detecção da Intrusão (SDI)

Os sistemas de detecção da intrusão (SDI), cujo inicio do desenvolvimento datam de 1970, é uma evolução dos antigos sistemas de auditoria, os quais definiam um processo de geração, gravação e revisão dos registros cronológicos dos eventos dos sistemas.

O objetivo de um SDI é detectar, preferencialmente em tempo real, o uso não autorizado, indevido e ameaças lógicas, circunscrevendo tais ameaças a sistemas de computação presentes, tanto nos ambientes internos como externos [3].

O funcionamento de um SDI é baseado na premissa de ser possível detectar o comportamento anômalo, pois este será notoriamente diferente de um comportamento considerado padrão, usualmente praticado por um usuário legítimo. Tipicamente, os SDI implementam modelos estatísticos de detecção da anomalia ou são baseados em regras

que detectam o uso indevido.

Em seu artigo DENNING [6] sugere um modelo de sistema especialista de detecção de intrusão em tempo real capaz de detectar interrupções, penetrações e outras formas de abuso. Esse modelo se baseia na hipótese de que uma violação de segurança pode ser detectada monitorando-se os registros de auditoria dos sistemas à procura de padrões de uso anormal. O modelo inclui perfis, para representar o comportamento dos usuários e regras, para adquirir conhecimento sobre esse comportamento a partir dos registros de auditoria com o objetivo de detectar comportamentos anômalos. O modelo é independente do tipo de sistema, ambiente de aplicação, vulnerabilidade do sistema ou tipo de intrusão, provendo a arquitetura genérica para um sistema especialista de detecção da intrusão.

A Figura 2.2 mostra a taxonomia dos sistemas de detecção da intrusão. Segundo SUNDARAM [7] os SDIs são classificados em dois tipos distintos de estratégia de monitoração: baseada em host (HIDS - Host Intrusion Detection System) ou baseada em rede (NIDS - Network Intrusion Detection System).

O HIDS tem sua fonte de dados baseada nos eventos de auditoria gerados a partir do sistema operacional do host., enquanto o NIDS tem seu mecanismo baseado na análise do tráfego da rede, pacote a pacote, podendo também utilizar-se dos eventos de auditoria do sistema operacional. Alguns SDIs se utilizam da máquina de estados do sistema operacional ou do protocolo de rede como fonte de dados para detecção da intrusão.

Os SDIs podem detectar a intrusão para poucos ataques em tempo real ou a posteriori, onde neste último os dados são analisados de forma seqüencial off line (batch). Geralmente os SDIs adotam uma configuração de “quase tempo real” para alguns ataques, ou seja, necessitam agregar informações suficientes durante algum tempo para poder fazer um julgamento.

Ao detectar um ataque o SDI pode simplesmente enviar alertas sobre o evento em questão agindo passivamente ou pode de maneira ativa atuar sobre os dispositivos de proteção fazendo com que os mesmos bloqueiem o ataque. Este modo deve ser utilizado com muito cuidado, pois pode se tornar uma nova vulnerabilidade.

As abordagens de detecção da intrusão são basicamente organizadas em três grandes

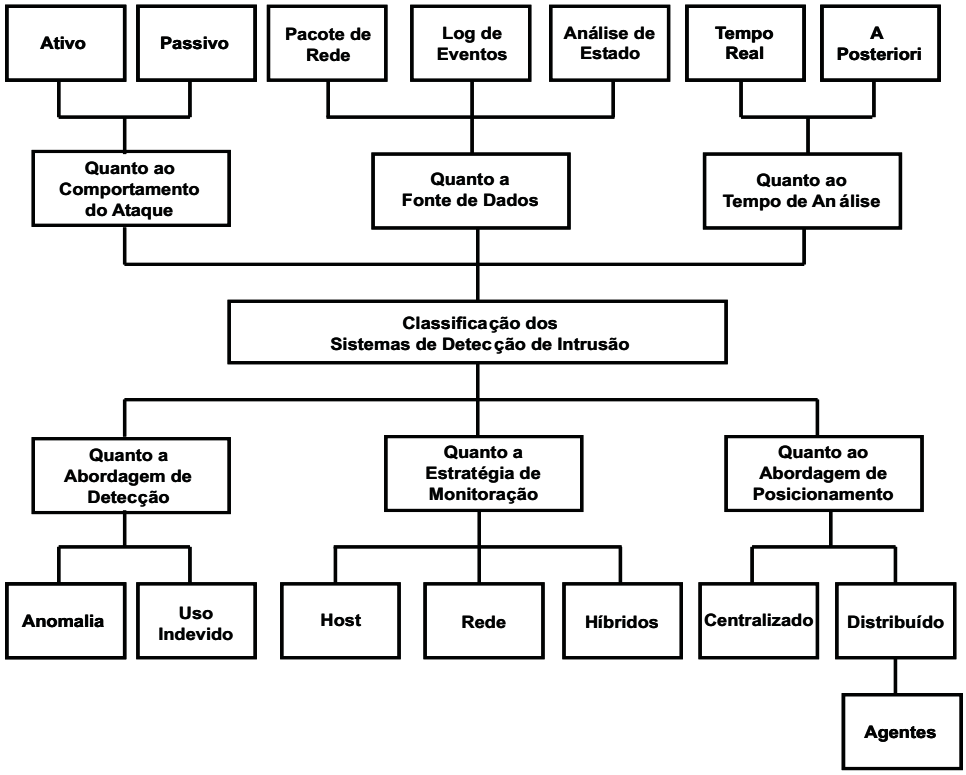


Figura 2.2: Taxonomia dos SDIs

grupos: o primeiro, chamado de detecção por uso indevido (misuse detection) detalhado na Figura 2.3, cuja abordagem busca, a partir de um banco de dados de assinaturas de eventos anteriormente caracterizados como maliciosos, procurar pela presença destas particularidades no tráfego analisado; o segundo grupo é chamado de detecção por anomalia (anomaly detection) no qual, a partir de uma referência de características de comportamento considerados normais, pesquisa-se qualquer desvio que seja significativo dentro de limites pré-estabelecidos pelo especialista, conforme detalhado na Figura 2.4 e, por fim, os sistemas híbridos que basicamente utilizam as duas técnicas anteriormente citadas.

Os SDIs de rede geralmente são configurados quanto ao seu posicionamento de modo centralizado, onde existe um SDI para cada seguimento de rede a ser monitorado. Os SDIs de host possuem configurações individualizadas, ou seja, uma para cada estação a ser monitorada. A abordagem distribuída, tanto para NIDS quanto para HIDS, trabalha com agentes que ficam instalados nos nós a serem monitorados, o qual transfere as informações para um SDI central que é responsável pelo processamento.

Segundo FRANK [8] os atuais sistemas de detecção da intrusão podem ser significa-

tivamente melhorados com a utilização de métodos de inteligência artificial (IA), sendo estes aplicados na redução e classificação de dados.

2.3 Detecção por Uso Indevido

A detecção por uso indevido se baseia na representação dos eventos na forma de padrões ou assinaturas os quais permitirão detectar pequenas variações nas características do ataque. Por se basear em regras de assinatura esse modelo não detecta padrões de ataques novos ou desconhecidos. O desafio dos sistemas de detecção por uso indevido é definir regras que englobem todas as possibilidades de ataques e não comprometa a utilização normal do sistema

Este modelo é o mais utilizado atualmente em SDI. Existem vários trabalhos sendo desenvolvidos para criação de sistemas de detecção por uso indevido mais eficientes. As principais linhas de pesquisa baseadas nesse modelo são: Sistemas Especialistas Baseados em Regras (Rule Based Expert Systems), Sistemas Baseados em Raciocínio (Model Based Reasoning Systems), Análise por Transição de Estado (State Transition Analysis), Monitoração de Entrada de Comandos (Key Stroke Monitoring) e Reconhecimento de Padrões (Pattern Matching).

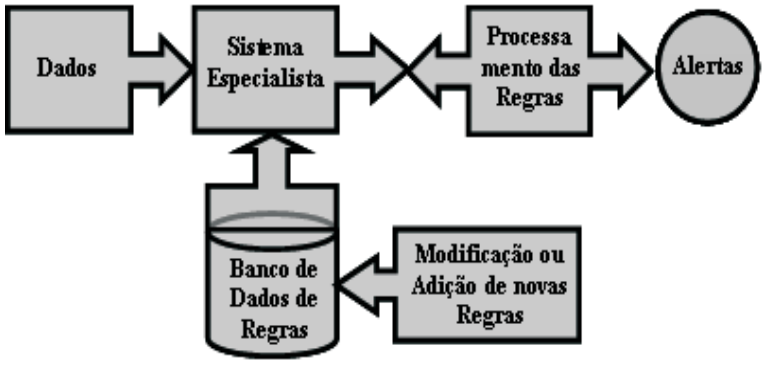


Figura 2.3: Sistemas de detecção da intrusão por uso indevido

2.4 Detecção por Anomalia

A técnica de detecção por anomalia pressupõe que a atividade de um intruso é necessariamente anômala [7]. Isto significa que se estabelecermos um perfil de utilização normal do sistema, qualquer variação deste perfil que tenha significado estatístico e ultrapasse o limiar estabelecido, pode vir a ser um ataque. O desafio desse sistema é estabelecer um limiar no qual não se deixe de detectar tentativas reais, chamadas de falso negativo, e nem detectar tentativas que na verdade são atividades normais, conhecidas como falsos positivos

Esta técnica se assemelha aos métodos heurísticos de detecção de vírus. Existem vários trabalhos sendo desenvolvidos para se criar sistemas de detecção por anomalia, as principais linhas de pesquisa dessa técnica são: aproximação estatística (Statistical Approaches), gerador de padrões por previsão (Predictive Pattern Generation) e redes neurais (Neural Network).

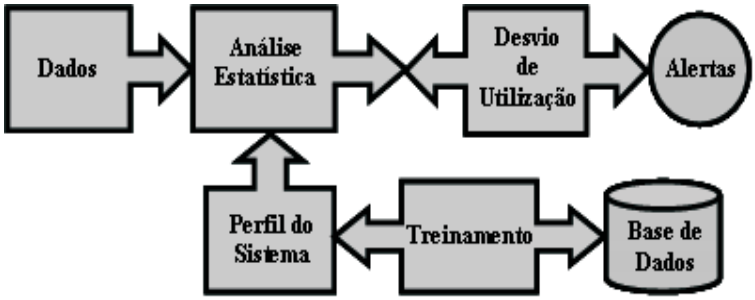


Figura 2.4: Sistemas de detecção da intrusão por anomalia

2.5 Trabalhos Relacionados

Atualmente, a questão da detecção da anomalia encontra-se em aberto, sendo esta área do conhecimento solicitada a elaborar algoritmos suficientemente robustos e capazes de reduzir a imensa quantidade de alertas gerados. Muitos desses alarmes não podem ser considerados evidências de ataques e são denominados como falsos positivos. Assim, o desafio na construção destes modelos está na qualidade dos alarmes resultantes. Hoje

encontramos diversas linhas de pesquisa nessa área, todas têm o objetivo de melhorar o desempenho dos sistemas e diminuir o número de evidências geradas por eles. Abaixo descrevemos alguns estudos nessa área, alguns mais antigos mas que são plenamente utilizados, como os Sistemas Especialistas e outros mais recentes, como os que utilizam Redes Neurais, Lógica Difusa e Redes Probabilísticas.

2.5.1 Sistemas Especialistas ou de Produção

Os sistemas especialistas, classificados como SDI de uso indevido, são definidos como sistemas computacionais capazes de representar e raciocinar sobre algum domínio de conhecimento com o intuito de solucionar problemas e gerar avisos [9]. Os sistemas baseados em regras possuem uma base de regras previamente conhecida que é comparada com os eventos de entrada seguindo um processo de comparação na forma se - então (if-then-else). Se a regra comparada for coincidente, um evento de saída é gerado, caso contrário outra regra é pesquisada e assim sucessivamente. A técnica de sistemas especialistas separa a fase de raciocínio da fase de solução do problema.

A principal desvantagem dessa técnica é a necessidade de manter a base de regras sempre atualizada com as últimas informações de vulnerabilidades, pois há o risco de não se detectar novos ataques. Outros pontos importantes que comprometem esse modelo são: o sistema de busca requer elevado processamento do sistema, a não ordenação das regras pode levar a uma decisão ineficiente e a qualidade das regras que devem ser descritas por um especialista.

Exemplos de sistemas que utilizam essa abordagem são: MIDAS, IDES (Intrusion Detection Expert System), NIDES (Next Generation IDES), DIDS e CMDS. Muitos sistemas comerciais atualmente utilizam esse modelo de detecção e procuram minimizar os problemas de desempenho utilizando técnicas de otimização de hardware.

2.5.2 Sistema Baseado em Modelo de Conclusão

Esta proposta é uma variação da detecção por uso indevido proposto por GARVEY e LUNT [10]. Ele combina o modelo de uso indevido com o raciocínio evidente para suportar conclusões sobre as ocorrências. O modelo procura por seqüências de comportamento de tentativas de intrusão que infiram sobre cenário de intrusão. O sistema é composto por três componentes: O antecipador, que tenta predizer o próximo passo sobre o cenário a ocorrer, o planejador, que traduz a hipótese em um formato compatível com os eventos de auditoria, e o interpretado, que procura pelos dados nos eventos de auditoria. O sistema continua coletando mais evidências da tentativa de intrusão até alcançar um limiar de valor que caracterize uma tentativa de intrusão.

Esta metodologia provê a interdependência do dado com os eventos de auditoria além de reduzir drasticamente o processamento por evento, devido ao seu modo otimizado de procura na base de dados.

A principal desvantagem desse modelo é a responsabilidade que o especialista tem ao criar os cenários. Os cenários devem ser abrangentes e distintos, para não serem confundidos com um comportamento normal.

2.5.3 Análise por Transição de Estado

Nesta técnica, classificada como SDI de uso indevido, o sistema monitorado é representado por um diagrama de transição de estados. Os dados são analisados enquanto o sistema faz a transição de um estado para outro. A transição é realizada se alguma condição Booleana for verdadeira. A proposta de ILGUN [11] e PORRAS e KEMMERER [12] é fazer a transição de um estado seguro para um não seguro baseado nos padrões de ataques conhecidos. Algumas vantagens deste modelo são: detectar ataques cooperativos, detectar ataques que se espalham por múltiplas seções e prever situações que comprometam o sistema e tomar ações pró-ativas. Contudo existem alguns problemas nesse modelo, como: só poderem ser especificados padrões de ataques através de seqüências de eventos e a não detecção de ataques do tipo negação de serviço (DoS), falha de login e variação do uso normal.

Exemplos de sistemas que utilizam essa abordagem são: STAT, USTAT e CP-Net (Colored Petric-Net).

2.5.4 Monitoração de Entrada de Comandos

Esta técnica utiliza a entrada de comandos do usuário, no shell de comando, para determinar a ocorrência de um ataque. O princípio é procurar por seqüências de padrões que indiquem um ataque [7]. Existem muitas desvantagens nesta técnica, como: indisponibilidade de acesso à entrada de comando do usuário, as várias maneiras de se expressar um mesmo ataque, a possibilidade de se utilizar sinônimos para representar um comando no nível do shell e a não análise dos programas executados. Uma melhoria para essa técnica seria monitorar os comandos pelas chamadas ao sistema operacional

2.5.5 Combinações de Padrões

KUMAR e SPAFFORD [13] propuseram um novo sistema de detecção da intrusão, classificada como SDI de uso indevido, baseado em combinação de padrões chamado IDIOT (Intrusion Detection In Our Time). Este modelo codifica assinaturas de intrusões conhecidas em padrões que são comparadas com os eventos de auditoria. Igual ao modelo de análise por transição de estado, este tenta igualar eventos de entrada com padrões que representam cenários de intrusão. A diferença entre os dois está na associação, enquanto que o modelo de análise por transição de estado está associado ao estado, o modelo por combinações de padrões está associado à transição. As vantagens desse modelo são: a facilidade de especificar o que precisa ser comparado, independência de plataforma, excelente desempenho e o potencial de detectar assinaturas do tipo falha de login que o modelo de análise por transição de estado não era capaz. As principais desvantagens são: não permite que se representem padrões mal definidos, alta complexidade em se traduzir os cenários de ataques conhecidos em padrões e a não detecção de conexões não autorizadas e nem ataques do tipo spoofing.

2.5.6 Abordagem Estatística

Esta técnica, classificada como SDI por anomalia, é baseada em perfis que são atualizados conforme a utilização do sistema. O novo perfil gerado é sempre comparado com o anterior para verificar se houve algum desvio significativo na utilização. Caso seja detectada alguma anomalia um evento é gerado. A principal vantagem dessa técnica é o aprendizado do comportamento do usuário com o tempo, mas isso pode levar o sistema a aprender o comportamento de um invasor. Com isso ações que deveriam ser consideradas anômalas serão consideradas normais (falso negativo). O ponto mais importante desta técnica é a escolha dos parâmetros para geração do perfil [7]. Devido a grande quantidade de parâmetros possíveis, torna-se difícil escolher o conjunto de parâmetros que melhor definirão o perfil.

Em seu trabalho DENNING [6] propôs alguns modelos estatísticos, para serem utilizados nos sistemas de detecção da intrusão por anomalia, abordando métricas de tipos diferentes. Entre eles está o modelo operacional, o modelo de média e desvio padrão, o multivariado, o processo Markov e o de séries temporais.

Exemplos de sistemas que utilizam essa abordagem são: Evoluções do IDES, NIDES e MIDAS e Haystack.

2.5.7 Predição da Geração de Padrões

Este método de detecção de intrusão, classificada como SDI por anomalia, tenta prever os eventos futuros baseados nos eventos que já ocorreram e se baseia no trabalho de TENG, CHEN e LU [14] na proposta de uma máquina indutiva baseada em tempo (TIM - Time-based Inductive Machine). Neste modelo são criadas regras que definem cenários do tipo R1 - R2 => (R3 = 30%, R4 = 60%, R5 = 10%). Isto significa que se o evento R2 ocorrer depois de R1 e existe a possibilidade de 30% de que o evento R3 ocorra, 60% que o evento R4 ocorra e 10% que o evento R5 ocorra. Então qualquer seqüência de eventos que não correspondam às regras serão marcadas como uma anomalia. A desvantagem nesse modelo é que pode existir um cenário que não esteja descrito nas regras e gere eventos. Isso pode ser contornado disparando-se avisos sobre qualquer evento que não combine

com pelo menos uma regra. Isso pode gerar uma quantidade alta de falsos positivos. As principais vantagens desse modelo são: o aprendizado das seqüências de padrões pode ser feito automaticamente e atualizados com a entrada de novos eventos de auditoria, a fácil detecção de usuários tentando treinar o sistema e o desempenho na detecção, que ocorre em segundos após a chegada do evento de segurança.

2.5.8 Redes Neurais

A abordagem Neural utiliza técnicas de aprendizagem para caracterizar os comportamentos anômalos. Existem algumas propostas de se utilizar essa técnica em conjunto com os sistemas especialistas para detecção de intrusão por anomalia [15]. O algoritmo de realimentação (backpropagation) da Rede Neural é utilizado para o aprendizado das séries temporais da rede. O objetivo é fazer um sistema capaz de prever o comportamento ou a seqüência deles realizado pelo usuário. Se o comportamento observado for diferente do previsto pelo sistema, um alerta de anomalia será gerado. A grande vantagem desse sistema á a tolerância a espúrios nos dados. Mesmo com dados incompletos ou imprecisos a rede neural irá detectar uma intrusão. Algumas desvantagens são: se a janela de tempo de aprendizado for pequena, a quantidade de falsos positivos aumenta e se for muito grande aumentarão as chances de falsos negativos, a topologia da rede só é determinada após várias tentativas e os usuários podem treinar a rede com uso indevido na fase de treinamento.

2.5.9 Algoritmos Genéticos

Em seus estudos MÉ [16] propôs uma nova técnica utilizando inteligência artificial através de algoritmo genético para a detecção de intrusão por uso indevido. Ele criou o GASSATA (Genetic Algorithm as an Alternative Tool for Security Audit Trail Analysis) que define uma ferramenta de segurança alternativa para análise de eventos de auditoria baseada em algoritmo genético. Sua teoria se baseia na construção de uma matriz de duas dimensões. Uma das diagonais da matriz especifica diferentes ataques conhecidos e o outro eixo os diferentes tipos de eventos derivados da auditoria. Essa matriz representa

os padrões de intrusão. Dado um registro de auditoria monitorado que inclui informações sobre o número de ocorrências de cada evento, GASSATA irá aplicar o algoritmo genético para encontrar possíveis ataques nos registros de auditoria [17].

2.5.10 Baseado em Lógica Difusa

O modelo de detecção de intrusão usando lógica difusa (Fuzzy Logic) aplica as teorias da incerteza e da imprecisão para determinar uma tentativa de intrusão [18]. Esse conceito é muito bem aceito, pois a definição de segurança em si é incerta e imprecisa, portanto Difusa. Este método associado à técnica de redes neurais, conhecido como Neural-Fuzzy, permite a criação dos parâmetros difusos a partir do treinamento da rede, que é um problema encontrado nesse modelo, pois sem o treinamento é necessário que um especialista crie os parâmetros difusos e as regras de inferência.

2.5.11 Agentes Autônomos

CROSBIE e SPAFFORD [19] propuseram a construção de um sistema de detecção da intrusão baseado em agentes autônomos. Ao invés de se ter um único sistema de grande porte de detecção de intrusão, teríamos vários sistemas de pequeno porte trabalhando cooperativamente. Este modelo garante maior eficiência, tolerância a falhas, recuperação, escalabilidade e extensibilidade. Exemplos de sistemas que utilizam essa abordagem são: AAFID (Autonomous Agents for Intrusion Detection) e EMERALD.

2.5.12 Métodos de Mineração de Dados

O método de mineração de dados é usado para extrair automaticamente e adaptativamente padrões normais de utilização a partir dos dados de treinamento. Dois métodos de mineração de dados são propostos por LEE, STOLFO e MOK [20], associação de regras (association rules) e episódios de freqüência seqüencial (serial frequency episode), para coletar dados de eventos de auditoria, seleção de características e análise a posteriori. O método de associação de regras especifica a correlação entre as diferentes características

selecionadas e o método de episódios de freqüência seqüencial representa a ocorrência repetitiva de uma seqüência de padrões em uma seqüência de eventos.

Este modelo pode ser utilizado por qualquer outra técnica, pois a extração das informações da base de dados é independente do mecanismo de detecção de intrusão.

2.5.13 Métodos Probabilísticos

Os métodos probabilísticos já são amplamente utilizados em diversas áreas do conhecimento. Algumas pesquisas recentes têm implementado a abordagem probabilística na detecção da intrusão, mas especificamente a utilização de técnicas de raciocínio probabilístico, como as Redes Bayesianas. Nesse modelo, classificado como detecção por anomalia, utiliza-se a Rede Bayesiana na classificação dos atributos de segurança que podem levar o sistema a uma evidência ou não de intrusão. Esse modelo necessita de treinamento para poder realizar a decisão do sistema. Uma outra abordagem da Rede Bayesiana, chamada Naïve Bayes, tem chamado a atenção pela sua simplicidade e rapidez, sendo usada com sucesso como classificador em sistemas de detecção da intrusão. Como trabalhos recentes nesta área podemos destacar: o trabalho de SABYAL *et al.* [21] que idealizaram um SDI de host utilizando o modelo Naïve Bayes; a análise feita por KRUEGEL *et al.* [22] da utilização do modelo Naïve Bayes na classificação de eventos de chamada do sistema operacional (system calls) na detecção de anomalias; o modelo de detecção em tempo real utilizando redes Bayesiana proposta por PUTTINI *et al.* [23]; o modelo de monitoração adaptativo proposto por VALDES e SKINNER [24]; o sistema ADAM criado por BARBARÁ *et al.* [25] que utiliza técnicas de aprendizado não supervisionadas usando um estimador pseudo-Bayesiano em conjunto com o classificador Naïve Bayes; e o trabalho apresentado por AMOR *et al.* [26] onde é comparado um sistema de detecção da intrusão usando Naïve Bayes e árvore de decisão.

2.5.14 Outros Métodos

Por ser uma área de estudo recente muito se tem feito para amadurecer esses conceitos no meio acadêmico. A seguir listamos algumas linhas de estudo que podem vir

a ser usadas como padrão de mercado no futuro, são elas: análise do comportamento dos estados do protocolo TCP (Specification-based Anomaly Detection), análise através de visualização gráfica do fluxo de tráfego (Graph Clustering and Graph Drawing), utilização de clusterização, redes neurais, VQ (Vector Quantization), SOM (Self-Organizing Map), RBF (Radial Basis Function) e SVM (Support Vector Machines).

Capítulo 3

Classificador Naïve Bayes

3.1 Introdução

Recentemente o interesse pelos métodos de indução probabilísticos tem crescido significantemente devido aos seus vários atrativos como a flexibilidade do modelo, sua resiliência a ruídos e sua aderência à teoria da probabilidade.

Neste capítulo apresentamos um método simples de indução probabilístico denominado classificador Naïve Bayes, baseado no Teorema de Bayes.

3.2 Teorema de Bayes

Se por definição $P(A,B) = P(A|B) \cdot P(B)$ e $P(B,A) = P(B|A) \cdot P(A)$ e se $P(A,B) = P(B,A)$ então:

$$P(A|B) \cdot P(B) = P(B|A) \cdot P(A) \text{ (F-3.1)}$$

Onde, P(A,B) é a probabilidade conjunta, P(A|B) e P(B|A) são probabilidades condicionais e P(A) e P(B) são probabilidades marginais dadas por:

$P(A) = \sum_n P(A|B_n)P(B_n)$ e $P(B) = \sum_n P(B|A_n)P(A_n)$.

Onde, n pode variar de 1 a $+\infty$.

Se passarmos o termo P(B) da Fórmula 3.1 dividindo para o lado direito temos:

$$P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B)}.$$

Essa fórmula defini o teorema de Bayes onde, $P(A)$ é a probabilidade a priori de A. $P(A|B)$ é a probabilidade a posteriori do evento A dado a ocorrência de B. $P(B|A)$ é chamada de probabilidade condicional do evento B dada a ocorrência do evento A, quando conhecemos A e não conhecemos B. Se conhecemos B mas não conhecemos A, então é chamada de verossimilhança. A verossimilhança ou likelihood de $P(B|A)$ é escrita como $L(A|B)$. $P(B)$ é a probabilidade a priori de B e age como uma constante de marginalização fazendo com que o resultado fique sempre entre o intervalo 0 e 1. Com essas definições podemos escrever a regra de Bayes como:

$$prob.posteriori = \frac{verossimilhan\c{c}a \times prob.priori}{constante-de-normaliza\c{c}\tilde{a}o}.$$

A probabilidade a priori P(A) é a probabilidade marginal do evento A levando-se em conta somente as informações conhecidas sobre ele, ou seja, é a distribuição probabilística que expressa a incerteza sobre A antes que dados relevantes sejam levados em consideração. Freqüentemente a probabilidade a priori é a pura estimação subjetiva de um especialista.

A verossimilhança ou likelihood é o inverso da probabilidade condicional. Dado o evento A, usamos a probabilidade condicional $P(B|A)$ para concluir sobre o evento B, mas se o evento B é fornecido, usamos a função de verossimilhança $L(A|B)$ para concluir sobre o evento A. A função de verossimilhança é uma função de probabilidade condicional considerada em função do seu segundo argumento, mantendo-se o primeiro fixo, então temos:

$$a \rightarrowtail P(B|A = a),$$

A probabilidade marginal $P(B)$ é difícil de ser calculada, assim passamos a escrever o teorema como uma proporcionalidade, então temos:

$$P(A_n|B) \sim P(B|A_n) \cdot P(A_n).$$

Como $P(B|A_n)$ é uma probabilidade, a soma sobre todas as hipóteses possíveis deve ser igual a 1, então concluímos que:

$$P(A_n|B) = \frac{P(B|A_n) \cdot P(A_n)}{\sum_m P(B|A_m) \cdot P(A_m)}.$$

Onde, n e m podem variar de 1 a $+\infty$.

Então nesse caso,

$$P(B) = \sum_m P(B|A_m) \cdot P(A_m).$$

é a constante de normalização.

3.3 Introdução ao Classificador Naïve Bayes

O classificador Naïve Bayes pode ser visto como um tipo especial de modelo gráfico probabilístico sendo considerado como um grafo acíclico direcionado (DAG - Directed Acyclic Graph), ou seja, não possui ciclos. Nesse modelo os nós representam as variáveis e os arcos representam a existência de influências casuais diretas entre as variáveis conectadas (Figura 3.1). A intensidade dessas influências são expressas pelas probabilidades condicionais [27].

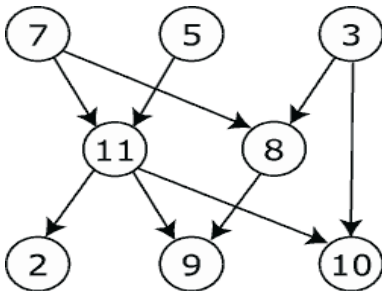


Figura 3.1: Exemplo de um gráfico acíclico direcionado (DAG)

Podemos dizer que o classificador Naïve Bayes pode ser encarado como uma rede Bayesiana com uma estrutura em estrela (Figura 3.2) [28].

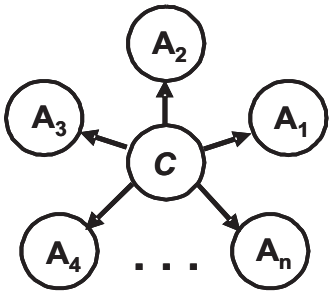


Figura 3.2: Estrutura em estrela do Classificador Naïve Bayes

A Rede Bayesiana é uma das ferramentas que lida com informações incertas baseada na teoria da probabilidade. Ela é a representação de uma distribuição de probabilidade conjunta sobre todas as variáveis representadas pelos nós do grafo. Se $X_{(1)}, \cdots, X_{(n)}$ são variáveis e *pai(A)* é o nó pai de A, então a distribuição conjunta de $X_{(1)}$ até $X_{(n)}$ é representado pelo produto das distribuições das probabilidades $p(X_i|pai(X_i))$ para $i = 1$ até n. Se o nó X não possuir pai, então ele é dito um nó incondicional.

O Naïve Bayes tem diversas vantagens devido a sua estrutura simplificada. Sua inferência, no nosso caso a classificação, é alcançada em tempo linear, enquanto a inferência em uma rede Bayesiana tradicional com estrutura genérica é um problema "NP-completo". A estrutura do Naïve Bayes pode ser facilmente atualizada permitindo adicionar ou remover evidências no modelo. Entretanto, sua suposição de independência condicional entre as variáveis nem sempre é verdadeira podendo trazer influências negativas ao resultado da inferência.

3.3.1 Modelo do Classificador

O classificador Naïve Bayes [29] [30] [31] [32] é um método de classificação antigo e bem conhecido e utiliza a abordagem probabilística para associar uma classe a um objeto ou caso baseado nos valores dos atributos usados para descrevê-los. Ele calcula a probabilidade condicional para cada classe e então decide pela a classe mais provável para um determinado objeto ou caso.

Seja C a classe de atributo com domínio finito de m classes, onde $dom(C) = \{c_1, \cdots, c_m\}$. Seja U o conjunto dos outros atributos usados para descrever o caso ou o objeto,

onde $U = \{A_1, \cdots, A_n\}$. Esses atributos podem ser simbólicos $dom(A_k) = \{a_{k,1}, \cdots, a_{k,mk}\}$, ou numéricos $dom(A_k) = \mathbb{R}$. Utilizando essa notação o caso ou objeto pode ser descrito pelo vetor de atributos instanciados $(a_1, \cdots, a_n)$ do atributo $(A_1, \cdots, A_n)$.

Para um dado vetor de atributos instanciados $(a_1, \cdots, a_n)$ o classificador Naïve Bayes tenta calcular a probabilidade condicional $P(C = c_i | A_1 = a_1, \cdots, A_n = a_n)$ para todo o conjunto c_i e então decide pela a classe baseado na maior probabilidade. Para esse calculo é necessário armazenar a distribuição de probabilidade de classe para cada ponto do domínio de atributos, o que cresce exponencialmente com o aumento desse número, tornando o cálculo impraticável. Para evitar esse problema o classificador Naïve Bayes explora as regras do teorema de Bayes em conjunto com a independência condicional entre as variáveis. Utilizando-se a regra de Bayes a probabilidade condicional é invertida, ou seja:

$$P(C = c_i | A_1 = a_1, \cdots, A_n = a_n) = \frac{f(A_1 = a_1, \cdots, A_n = a_n | C = c_i) \cdot P(C = c_i)}{f(A_1 = a_1, \cdots, A_n = a_n)}.$$

Para que essa inversão seja sempre possível a função de densidade probabilística tem que ser obrigatoriamente positiva, ou seja:

$$f(A_1 = a_1, \cdots, A_n = a_n) > 0.$$

Existem duas observações importantes a serem feitas sobre a fórmula acima. Na primeira o denominador da equação pode ser descartado, pois para o caso ou objeto em análise este valor é fixo e não tem qualquer influência na escolha da classe. Se for necessário, essa influência pode ser restaurada normalizando-se a distribuição pelas classes, usando a seguinte equação:

$$f(A_1 = a_1, \cdots, A_n = a_n) = \sum_{j=1}^{m} f(A_1 = a_1, \cdots, A_n = a_n | C = c_i) \cdot P(C = c_i).$$

Então:

$$P(C = c_i | A_1 = a_1, \cdots, A_n = a_n) = \frac{P(C = c_i)}{p_0} f(A_1 = a_1, \cdots, A_n = a_n | C = c_i). \quad \text{(F-3.2)}$$

Onde $p_0 = f(A_1 = a_1, \cdots, A_n)$ é a constante de normalização.

Na segunda observação, como podemos constatar, a inversão não trouxe qualquer ganho em relação ao espaço de amostra que continua sendo exponencial. Para eliminar esse problema utilizamos a abordagem da independência condicional entre as variáveis. Para explorar essa abordagem, primeiramente aplicamos a regra da probabilidade seqüencial, ou seja:

Se temos uma amostra da observação X, onde:

$$P(\theta|X) \propto P(\theta) \cdot L(\theta|X). \text{ Eq: 3.1}$$

e uma segunda observação Y, independente da primeira, onde:

$$P(\theta|X,Y) \propto P(\theta) \cdot L(\theta|X,Y). \text{ Eq: 3.2}$$

Como X é independente em relação a Y, temos:

$$P(X,Y|\theta) = P(X|\theta) \cdot P(Y|\theta) \text{ ou Eq: 3.3}$$

$$L(X,Y|\theta) = L(X|\theta) \cdot L(Y|\theta). \text{ Eq: 3.4}$$

Então podemos concluir que:

$$P(\theta|X,Y) \propto P(\theta) \cdot L(\theta|X) \cdot L(\theta|Y) \text{ Eq: 3.5}$$

$$P(\theta|X,Y) \propto P(\theta|X) \cdot L(\theta|Y). \text{ Eq: 3.6}$$

Onde $P(\theta|X)$ é a probabilidade a posteriori para a observação X, que é tratada como probabilidade a priori para a observação Y.

Aplicando a inferência seqüencial na Fórmula 3.2 temos:

$$P(C = c_i|A_1 = a_1, \cdots, A_n = a_n) =$$

$$= \frac{P(C=c_i)}{p_0} f(A_j = a_j|A_{j-1} = a_{j-1}, \cdots, A_1 = a_1, C = c_i)$$

$$\cdots$$

$$= f(A_2 = a_2|A_1 = a_1, C = c_i)$$

$$= f(A_1 = a_1|C = c_i)P(C = c_i).$$

Em seguida, aplicamos a independência condicional que diz: dado o valor do atributo da classe, qualquer atributo A_j é independente de qualquer outro, ou seja, assumimos que sabendo a classe é suficiente para determinar a densidade de probabilidade do valor a_j, sem precisar saber o valor de qualquer outro atributo. Isso simplifica significantemente a fórmula inicial, pois os termos multiplicadores A_{j-1}, são cancelados na expressão acima. Com isso temos:

$$P(C = c_i|A_1 = a_1, \cdots, A_n = a_n) = \frac{P(C=c_i)}{p_0} \prod_{j=1}^{n} f(A_j = a_j|C = c_i)$$

Essa é a fórmula principal do classificador Naïve Bayes. Para um atributo simbólico A_j as probabilidades condicionais $P(A_j = a_j|C = C_i)$ são armazenadas em uma tabela simples $mx(m_j - 1)$, pois só temos uma condição para cada atributo, o que torna o cálculo viável. Para atributos numéricos, é geralmente suposto que a distribuição de densidade probabilística é uma distribuição normal, então só precisamos armazenar a média ou valor esperado $\mu_j(c_i)$ e a variância $\sigma_j^2(c_i)$. Se necessário, os atributos numéricos podem ser discretizados e usados como atributos simbólicos [33].

3.3.2 Treinamento do Classificador

A aprendizagem do classificador Naïve Bayes pode ser feita usando-se métodos de treinamento supervisionado. Em muitas aplicações práticas, para a estimação de parâmetros no modelo Naïve Bayes é utilizado o método de máxima verossimilhança.

A verossimilhança máxima, é um método de estimação de ponto que utiliza como estimador, de um parâmetro populacional não observável, um membro do espaço de parâmetros que maximize a função de verossimilhança. Se X é uma variável aleatória

observável e p denota o parâmetro populacional não observável a ser estimado, então a probabilidade de uma saída observável $X = x$ em função de p mantendo-se x fixo é a função de verossimilhança

$$L_{X=x}(p) = P(X = x|p).$$

Então, o valor de p que maximiza a função $L(p)$ é a verossimilhança máxima estimada de p.

Para aplicar o método da máxima verossimilhança do classificador Naïve Bayes basta estimar a densidade condicional probabilística $f(A_j = a_j|C = c_i)$. Para atributos simbólicos temos:

$$P(A_j = a_j|C = c_i) = \frac{N(A_j=a_j,C=c_i)}{N(C=c_i)}.$$

Onde $N(C = c_i)$ é o número de amostras que pertencem à classe c_i e $N(A_j = a_j, C = c_i)$ é o número de amostras que além de terem amostras pertencentes à classe c_i tenham o valor a_j para o atributo A_j. Para assegurar que a probabilidade será sempre positiva, classes não representadas são removidas. Se um atributo não correr para uma determinada classe sua probabilidade é definida por $\frac{1}{2}N$ ou a uniforme a priori de $1/N$, onde N é o número total de amostras, e adicionada à distribuição estimada, a qual é então normalizada.

Para um atributo numérico Aj a função de estimação de verossimilhança máxima padrão para parâmetros de uma distribuição normal é:

$$\mu_{c_i} = \frac{1}{N(C=c_i)} \sum_{k=1}^{N(C=c_i)} a_j(k)$$

para a média ou valor esperado, onde $a_j(k)$ é o valor do atributo A_j na "k-esima"amostra caso pertença a classe c_i, e para a variância temos:

$$\sigma_j^2(c_i) = \frac{1}{N(C=c_i)} \sum_{k=1}^{N(C=c_i)} (a_j(k) - \mu_{c_i})^2.$$

3.3.3 Inferência do Classificador

A inferência no classificador Naïve Bayes combina o modelo probabilístico e o de regra de decisão. A regra mais comum usada para inferência Naïve Bayes é máxima a posteriori, ou seja, escolhe a classe que obtiver maior densidade de probabilidade condicional a posteriori. Esse tipo de inferência é conhecido como regra de decisão MAP (Máximo A Posteriori). A decisão é então dada por:

$$Classe(f_1, \cdots, f_n) = argmax_{c_i} \frac{P(C=c_i)}{p_0} \prod_{j=1}^{n} f(A_j = a_j | C = c_i).$$

Se utilizarmos a função de distribuição normal $P(x)$ com média ou valor esperado θ e variância ϕ dada por,

$$P(x) = \frac{1}{\sqrt{2\pi\phi}} e^{\frac{-(x-\theta)^2}{2\phi}}$$

para descrever o comportamento dos atributos das classes do classificador, tal que sua normal seja representada por $x \sim N(\theta, \phi)$ e similarmente a função de probabilidade a priori da variável aleatória θ, que é a média da função de verossimilhança, tal que sua normal seja representada por $\theta \sim N(\theta_0, \phi_0)$, então temos:

$$P(x|\theta) = \frac{1}{\sqrt{2\pi\phi}} e^{\frac{-(x-\theta)^2}{2\phi}},$$

$$P(\theta) = \frac{1}{\sqrt{2\pi\phi_0}} e^{\frac{-(\theta-\theta_0)^2}{2\phi_0}}.$$

Aplicando essas equações ao teorema de Bayes, temos:

$$P(\theta|x) = \frac{1}{\sqrt{2\pi\phi}} e^{\frac{-(x-\theta)^2}{2\phi}} \cdot \frac{1}{\sqrt{2\pi\phi_0}} e^{\frac{-(\theta-\theta_0)^2}{2\phi_0}}. \text{ (F-3.3)}$$

Em alguns casos, a função de probabilidade a priori $P(\theta)$ não é representada por uma equação de distribuição normal e sim por uma equação linear como, por exemplo, a freqüência relativa da variável aleatória dada por:

$$P(\theta) = \frac{n_{c_i}}{n}.$$

Onde: n_{c_i} é a quantidade de amostras da classe c_i e n é o número total de amostras.

Aplicando-se essa propriedade na Fórmula 3.3 temos:

$$P(\theta|x) = \frac{1}{\sqrt{2\pi\phi^2}} e^{\frac{-(x-\theta)^2}{2\phi^2}} \cdot \frac{n_{c_i}}{n}.$$

Nesse caso a escolha da classe mais provável utilizando-se o método MAP é o valor de x que maximize $P(\theta|x)$.

3.4 Conclusão

O classificador Naïve Bayes é simples, rápido e de fácil implementação. Apesar de usar a argumentação da independência probabilística, o que em muitos casos não é verdadeiro, ele tem sido amplamente usado em várias áreas de pesquisa e com muito sucesso. Variações do seu modelo têm sido propostas e observamos nos resultando melhoras nos resultados da classificação, mas sempre trazendo ônus para o desempenho do sistema que suporta o classificador no sentido do consumo dos recursos do mesmo. Dentre alguns trabalhos podemos citar: FRIEDMAN e GOLDSZMIDT [34] propuseram a interligação entre as variáveis, quebrando o paradigma da independência condicional e trazendo maior liberdade para o modelo; um classificador possibilístico foi apresentado por BORGELT e GEBHARDT [35], o qual é muito mais simples de ser implementado, mas não consegue lidar com variáveis contínuas; e LANGLEY e SAGE [32] apresentaram uma simplificação do modelo Naïve Bayes baseado em "Greedy".

Esse trabalho se limita a utilizar o Classificador Naïve Bayes como demonstrado nesse capítulo.

Capítulo 4

Aquisição e Levantamento dos Dados

4.1 Introdução

Para realizar a implementação do NBIS foi necessária a criação de um ambiente de teste onde fosse possível validar as hipóteses presentes neste trabalho. Para realizar tais testes se fez necessária obter uma base de dados de tráfego que contivesse além das evidências de ataque, tráfego livre dessas ocorrências. Esse conjunto de dados deveria estar organizado de tal modo que as indicações de ataques estivessem bem documentadas e com referências precisas da sua ocorrência no tempo.

Existem duas bases de tráfego conhecidas que são usadas para essa finalidade, o 1999 KDD [36] e o DARPA 98 [37] e o DARPA 99 [38]. O 1999 KDD foi desenvolvido pelo Departamento de Informática e Ciência da Computação da Universidade de Irvine na Califórnia (UCI - University of California, Irvine) baseado no trabalho realizado pela agência DARPA, mas utilizando outras métricas de marcação das informações dos arquivos de tráfego. A base DARPA 98 e DARPA 99, criado pelo Laboratório Lincoln do Instituto de Tecnologia de Massachusetts (MIT Lincoln Labs), foi a base escolhida para realizar as experiências nesse trabalho devido a boa documentação disponível no site [39] e também por ser um trabalho padrão amplamente usado pelos pesquisadores da área de segurança, facilitando assim a comparação entre outros trabalhos realizados pela comunidade acadêmica.

Nesse capítulo é apresentado todo o processo de aquisição de dados, preparação das informações da base do DARPA 98 e DARPA 99 para carga no banco de dados e a definição dos vetores de características que serão usados para os testes.

4.2 A Base DARPA 98 e DARPA 99

Avaliação de Detecção de Intrusão DARPA 98 e DARPA 99 foi um trabalho realizado em conjunto pelo Grupo de Tecnologia da Informação de Sistemas (IST - Information Systems Technology) do Laboratório Lincoln do MIT, a Agência de Pesquisa de Projetos Avançados de Defesa (DARPA - Defense Advanced Research Projects Agency) e o Laboratório de Pesquisa da Força Aérea dos Estados Unidos (AFRL/SNHS) com o objetivo de criar um ambiente capaz de simular uma rede local tanto no comportamento das aplicações como no tráfego, na qual ataques pudessem ser submetidos e avaliados pelos sistemas de detecção de intrusão.

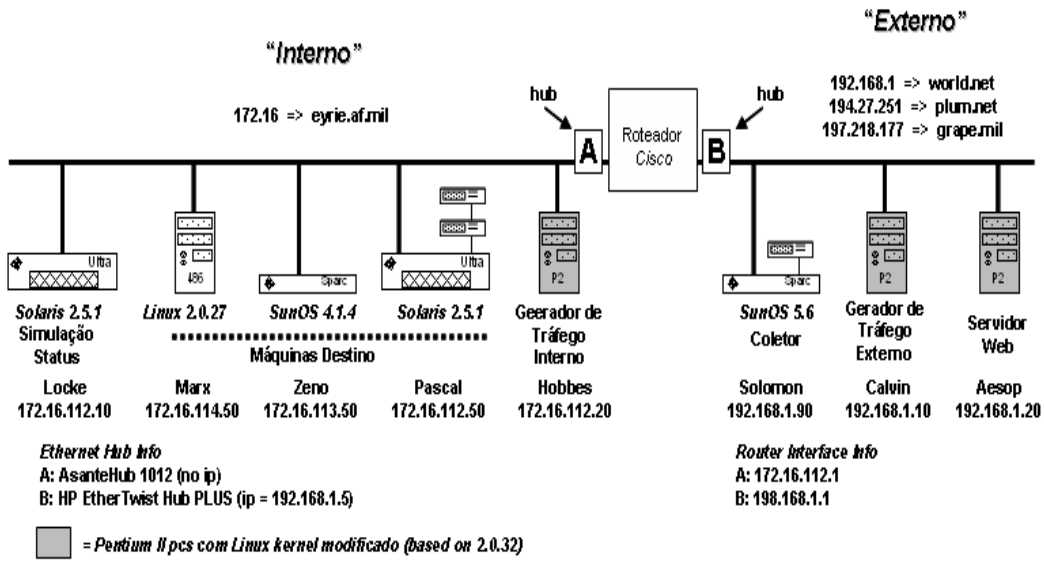


Figura 4.1: Diagrama da rede da avaliação do DARPA 98

A primeira avaliação do DARPA foi feita em 1998 [37] na qual foram feitas simulações de tráfego normal equivalente a uma rede contendo 100 usuários com perfis diferentes e 1000 estações conforme a Figura 4.1. Mais de 300 instâncias de 32 diferentes tipos de ataques foram lançados durante 9 semanas, 7 para coleta de dados de treinamento

e 2 para os dados de teste. A geração de tráfego normal junto com os ataques se fez necessários para possibilitar a ocorrência de falsos alarmes, denominado falsos positivos. Diversos serviços foram injetados na rede tais como: HTTP, FTP, SMTP, IRC, Telnet, SNMP, dentre outros. Os dispositivos de segurança foram omitidos na configuração da rede, pois o objetivo é o teste dos sistemas de detecção de intrusão.

Os ataques foram realizados do lado externo da infra-estrutura da rede local e estão listados na Tabela 4.1. Eles foram classificados em quatro grupos: probes, negação de serviço (DoS – Denial of service), remoto para local (R2L – Remote-to-Local) e usuário para root (U2R – User-to-Root).

Classe dos Ataques	DoS	R2L	U2R	Probe
Tipos de Ataques	Apache2	Dictionary	Anypw	InsideSniffer
	Arppoison	FTPwrite	Casesen	IPSweep
	Back	Guest	Eject	Portsweep
	Crashiis	HTTPtunnel	Ffbconfig	Ls Domain
	Dosnuke	Imap	Fdformat	Mscan
	Land	Named	Loadmodules	NtInfoScan
	Mailbomb	Ncftp	Ntfsdos	Nmap
	Neptune	Netbus	Perl	Queso
	Ping da Morte	Netcat	Ps	ResetScan
	Process Table	Phf	Sechole	Saint
	Selfping	Ppmacro	Xterm/Xterm1	Satan
	Smurf	Sendmail	Yaga	
	SSHprocesstable	SSHtrojan		
	Syslogd	Xlock		
	TCPreset	Xsnoop		
	Teardrop	GuessTelnet		
	UDPstorm	GuessPop/Ftp		

Tabela 4.1: Lista dos ataques por classe do DARPA 98

A segunda versão do DARPA veio em 1999 [38] com alguns aprimoramentos, mas com os mesmos conceitos. Alguns ataques novos foram incorporados e os arquivos de tráfegos foram separados em dois, um de entrada e outro de saída. A principal diferença foi a inclusão da plataforma Microsoft Windows no cenário da rede. As modificações apresentadas nessa nova versão não interferiram com a evolução desse trabalho em relação aos dados do DARPA 98.

Como resultado das simulações do DARPA 98 e DARPA 99 vários arquivos de coleta foram gerados. Os arquivos usados nesse trabalho foram: arquivo bruto de tráfego chamado de dumpfile, e o arquivo contendo a lista detalhada das conexões com informações sobre os eventos de ataques efetuados, chamado de listfile. Cada arquivo contém informação de um dia (24 horas) de tráfego.

O arquivo bruto de tráfego (dumpfile) é gerado por um coletor de tráfego (sniffer) que coleta todo tráfego que passa pelo segmento de rede e guarda em arquivo no formato bruto com todas as informações referentes aos protocolos vistos no segmento de rede.

O arquivo de eventos (listfile) contém uma linha para cada conexão ou pacote (TCP, UDP ou ICMP) com as seguintes informações: identificador único para cada conexão referente ao dia da coleta, data, hora, duração, nome do serviço e protocolo, porta de serviço de origem, porta de serviço de destino, endereço IP de origem, endereço IP de destino, identificador de ataque (0 para normal e 1 para ataque) e o nome do ataque.

4.2.1 Críticas à Base DARPA 98 e DARPA 99

Em seu artigo Testando Sistemas de Detecção de Intrusão: Uma Crítica à Avaliação de Sistemas de Detecção de Intrusão do DARPA 1998 e DARPA 1999 realizada pelo Laboratório Lincoln, MCHUGH [40] faz um levantamento detalhado sobre as fragilidades da abordagem feita nessa avaliação. Dentre as mais importantes e relevantes para esse trabalho temos: erro na data e hora dos arquivos de dados, erro na marcação dos ataques, falta de arquivos de marcação dos ataques de 1998, falta de uma marcação única das conexões, lista de ataques incompleta e sem informações sobre os mesmos e muitos outros como veremos a seguir.

A falta de detalhamento dos procedimentos realizados na avaliação e do racional utilizado para essa escolha tornam o trabalho muito obscuro podendo influenciar na interpretação dos resultados.

A taxonomia dos ataques usados foi desenvolvida privilegiando o ponto de vista do atacante e isso pode ter introduzido influências na avaliação dos ataques.

A quantidade e qualidade de tráfego normal e de fundo gerados para a avaliação, além de não corresponderem ao cenário descrito, podem ter influenciado nos resultados dos testes, onde os valores de falso positivo podem ter sido inferiores aos que se encontraria utilizando-se tráfego real. A mesma crítica pode ser feita em relação ao tráfego de ataque que não foi distribuído em conjunto com o tráfego de fundo, prejudicando a análise dos resultados.

4.3 Módulo de Carga de Dados

Segundo DIHUA [41] o processador de dados (data mining) deve ser entendido como uma estrutura utilizada com o objetivo de classificar e descrever as características principais encontradas em cada conexão ou sessão de rede. Para tanto, esta estrutura utiliza uma coleção de programas que validam as regras de captura dos comportamentos presentes nas informações coletadas.

Como resultado das operações da prospecção de dados, é esperado, segundo BLOEDORN *at al.* [42], que estes sistemas sejam capazes de produzir um sumário dos dados suficientemente claro onde sejam facilitadas as operações de extração e classificação dos dados.

Como requisito fundamental, segundo DENNING [6] o prospector de dados deve ser independente de qualquer sistema, ambiente de aplicação, vulnerabilidade sistêmica ou tipo de intrusão, mas deve ser concebido e implementado com o objetivo de prover um modelo de propósito geral.

Observando as questões anteriormente citadas, foi construído um modelo apresentado na Figura 4.2, onde o primeiro componente, fase de pré-carga, tem a função de gerar um

arquivo intermediário a partir do arquivo bruto de tráfego (dumpfile) para em seguida ser normalizado, fase de adequação, e então ser carregado na base de dados, fase de carga de dados. Esse mesmo procedimento é realizado para o arquivo de eventos (listfile). Nesse processo entramos com o arquivo bruto de dados e saímos com uma base de dados organizada.

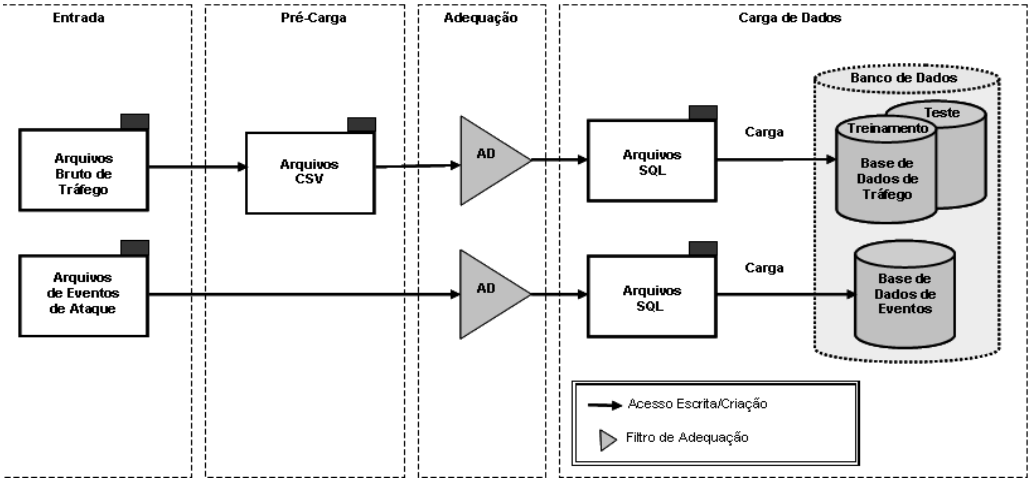


Figura 4.2: Diagrama em blocos do módulo de carga de dados

Nesse trabalho os arquivos utilizados para carga no bando de dados foram: o conjunto de dados de treinamento do DARPA 98 e o conjunto de dados de teste do DARPA 99. Não se utilizou o mesmo conjunto de dados porque os arquivos de lista de ataques para o conjunto de dados de teste do DARPA 98 não apresentaram com as marcações dos ataques, necessárias para a mensuração e comparação dos resultados.

O processo de carga de dados foi a parte mais sensível do trabalho, pois qualquer inconsistência nessa fase traria repercussões desastrosas para os outros módulos do sistema. Foi necessário criar dois procedimentos separados para a carga de dados, um para o arquivo bruto de tráfego e outro para o arquivo de eventos como mostrado na Figura 4.2. Nos dois casos foi necessária a normalização dos dados de entrada antes da carga na base de dados.

4.3.1 Carga dos Dados de Tráfego

A carga do arquivo bruto de tráfego é feita processando-se o arquivo dumpfile, para o conjunto de dados de treinamento e de teste, mais de 10GBytes comprimidos, gerando-se um novo arquivo no formato CSV contendo as informações das conexões. Entende-se por conexão o processo de conversação fim a fim entre dois nós de rede [43]. Nesse processo de conexão existem sinalizações de início, meio e fim da conversação. O processo de criação do arquivo CSV é feito pelo software TCPTrace [44] com uma pequena alteração no seu código fonte para salvar a informação de tempo nos campos de data e hora. O TCPTrace é uma ferramenta de uso gratuito que tem a função de mapear o protocolo TCP, descartando os demais, o que não causou problemas uma vez que todos os ataques escolhidos para o teste utilizam o protocolo TCP. A principal função desta ferramenta é construir sessões TCP a partir dos diferentes pacotes presentes no tráfego bruto. Para cada sessão é criada uma linha no arquivo CSV com todas as informações sobre a determinada conexão.

Anteriormente à fase de carga de dados algumas informações geradas pelo TCPTrace são normalizadas e salvas em um arquivo no formato SQL para carga no banco de dados. Nesse processo são inseridos o identificador único para cada sessão carregada na base de dados (SessionID) e os campos: status do ataque (Status), nome do ataque (AttackName) e nome da classe do ataque (AttackClass). A data e hora são formatadas em um único campo de tempo linear transformado em segundos referenciando-se a data base de 01/01/1998 ou 01/01/1999 a 0:00 hora (Start_Time). Na base de dados do conjunto de dados de teste foi necessário incluir mais um campo para identificar o sentido do fluxo do tráfego, ou seja, “entrante” ou “sainte” (Fluxo). A Tabela 4.2 lista todos os campos disponíveis no banco de dados de tráfego.

O primeiro campo (SessionID) é o identificador único para cada conexão referente ao dia de coleta, o segundo campo (Session) é o identificador da sessão referenciada no arquivo de tráfego, seguido das informações de endereço IP de origem (Host_a) e destino (Host_b), porta de serviço de origem (Port_a) e destino (Port_b), data (Dia, Mês, Ano) e hora da conexão (Hora, Minuto e Segundo) e o tempo linear (Start_Time). Em seguida encontramos o total de pacotes enviados (total_packtes_a2b) e recebidos (to-

SessionID	Session	host_a
host_b	port_a	port_b
dia	mês	ano
hora	minuto	segundo
Start_Time	total_packets_a2b	total_packets_b2a
resets_sent_a2b	resets_sent_b2a	ack_pkts_sent_a2b
ack_pkts_sent_b2a	pure_ acks_sent_a2b	pure_acks_sent_b2a
sack_pkts_sent_a2b	sack_pkts_sent_b2a	dsack_pkts_sent_a2b
dsack_pkts_sent_b2a	max_sack_blks/ack_a2b	max_sack_blks/ack_b2a
unique_bytes_sent_a2b	unique_bytes_sent_b2a	actual_data_pkts_a2b
actual_data_pkts_b2a	actual_data_bytes_a2b	actual_data_bytes_b2a
rexmt_data_pkts_a2b	rexmt_data_pkts_b2a	rexmt_data_bytes_a2b
rexmt_data_bytes_b2a	zwnd_probe_pkts_a2b	zwnd_probe_pkts_b2a
zwnd_probe_bytes_a2b	zwnd_probe_bytes_b2a	outoforder_pkts_a2b
outoforder_pkts_b2a	pushed_data_pkts_a2b	pushed_data_pkts_b2a
SYN/FIN_pkts_sent_a2b	SYN/FIN_pkts_sent_b2a	req_1323_ws/ts_a2b
req_1323_ws/ts_b2a	adv_wind_scale_a2b	adv_wind_scale_b2a
req_sack_a2b	req_sack_b2a	sacks_sent_a2b
sacks_sent_b2a	urgent_ data_pkts_a2b	urgent_data_pkts_b2a
urgent_data_bytes_a2b	urgent_data_bytes_b2a	mss_requested_a2b
mss_requested_b2a	max_segm_size_a2b	max_segm_size_b2a
min_segm_size_a2b	min_segm_size_b2a	avg_segm_size_a2b
avg_segm_size_b2a	max_win_adv_a2b	max_win_adv_b2a
min_win_adv_a2b	min_win_adv_b2a	zero_win_adv_a2b
zero_win_adv_b2a	avg_win_adv_a2b	avg_win_adv_b2a
initial_window_bytes_a2b	initial_window_bytes_b2a	initial_window_pkts_a2b
initial_window_pkts_b2a	ttl_stream_length_a2b	ttl_stream_length_b2a

missed_data_a2b	missed_data_b2a	truncated_data_a2b
truncated_ data_b2a	truncated_packets_a2b	truncated_packets_b2a
data_xmit_time_a2b	data_xmit_time_b2a	idletime_max_a2b
idletime_max_b2a	hardware_dups_a2b	hardware_dups_b2a
throughput_a2b	throughput_b2a	Status
AttackName	AttackClass	*Fluxo

Tabela 4.2: Lista dos campos de informação do arquivo de tráfego. * Somente para o conjunto de dados de teste do DARPA 99

tal_packtes_b2a) pela origem e pelo destino. Os demais campos são as sinalizações do protocolo TCP e podem servir de vetores de característica para o classificador. Os 3 últimos campos da base fazem referência ao tipo de tráfego, se normal ou com ataque. Se a sessão for normal o campo status terá o valor 0, o campo AttackName conterá a string "Não Ataque" e o campo AttackClass terá a string "Sem Classe". Caso seja uma evidência de ataque o campo Status será igual a 1 e os campos AttackName e AttackClass conterão a informação sobre o ataque em questão. O último campo Fluxo só é usado na base do conjunto de dados de teste do DARPA 99 para referenciar a direção do tráfego, se de entrada conterá a string "IN" e de saída a string "OUT".

Registros de Treinamento		
	Total	Relação sobre o Total de Registros
Total de Registros	2.398.703	
Não Ataques	829.580	34,58%
Ataques	1.569.123	65,42%

Tabela 4.3: Levantamento quantitativo dos registros na base de dados de tráfego de treinamento

Após a carga de todos os arquivos brutos de tráfego na base de dados de treinamento e de teste realizou-se o levantamento quantitativo das informações na base, que são mostradas na Tabelas 4.3, 4.4, 4.5 e 4.6.

Registros de Teste		
	Total	Relação sobre o Total de Registros
Total de Registros	1.196.733	
Não Ataques	1.109.694	92,73%
Ataques	87.039	7,27%

Tabela 4.4: Levantamento quantitativo dos registros na base de dados de tráfego de teste

Registros de Treinamento			
Tipos de Ataques	Quantidade	Tipos de Ataques	Quantidade
ejectfail	1	eject	11
format_clear	1	anomaly	11
warez	1	rootkit	16
ffb_clear	1	warezmaster	19
format-fail	1	land	35
dict_simple	1	Guest telnet	50
spy	4	dict	884
perlmagic	4	ipsweep	923
phf	5	nmap	1.031
format	6	warezclient	1.654
ftp-write	6	back	2.216
imap	6	portsweep	11.617
loadmodule	8	satan	23.948
ffb	10	neptune	1.526.643
multihop	10		
Total de Registros de Ataques = 1.569.123			

Tabela 4.5: Levantamento quantitativo dos registros e eventos de ataque na base de dados de tráfego de treinamento

Registros de Treinamento			
Tipos de Ataques	Quantidade	Tipos de Ataques	Quantidade
xterm1	1	sechole	9
ffbconfig	2	ps	9
imap	2	phf	11
land	2	secret	13
ls	2	httptunnel	24
selfping	2	guest	27
sendmail	2	queso	28
fdformat	3	guesspop	30
loadmodule	3	ncftp	38
named	3	ntinfoscan	45
netbus	3	ppmacro	64
sshtrojan	3	guesstelnet	67
perl	3	guessftp	80
xsnoop	3	dict	86
ftpwrite	4	back	161
dosnuke	4	portsweep	262
netcat	4	mscan	484
xterm	4	sshprocesstable	753
ipsweep	5	processtable	1113
xlock	5	mailbomb	1351
eject	8	apache2	1727
crashiis	9	satan	8488
casesen	9	neptune	72.075
Total de Registros de Ataques = 87.039			

Tabela 4.6: Levantamento quantitativo dos registros e eventos de ataque na base de dados de tráfego de teste

4.3.2 Carga dos Dados de Eventos de Ataque

A marcação dos eventos de ataque na base de tráfego precisou de levantamento detalhado a priori, pois as informações de tempo do arquivo de eventos não coincidiam com as conexões da base de dados. A forma encontrada para fazer esse relacionamento foi procurar na base de tráfego pelo conjunto: porta de serviço e endereço IP de origem e destino. Para isso as informações dos eventos de ataque foram carregadas no banco de dados para facilitar a manipulação dos dados. Uma normalização dos dados foi necessária para formatar os dados para carga no banco, como pode ser visto na Figura 4.2. A Tabela 4.7 lista todos os campos disponíveis no banco de dados de eventos.

Day	Month	Year
Hour	Minute	Second
host_a	port_a	host_b
port_b	AttackName	*Flow

Tabela 4.7: Campos do banco de dados de eventos. * Somente para o conjunto de dados de teste do DARPA 99

Após a carga dos eventos na base de dados, um programa fazia a comparação do conjunto, porta de serviço e endereço IP de origem e destino da base de eventos com a base de tráfego para cada dia carregado. O resultado era armazenado no arquivo de relatório com as seguintes informações: Dia, Mês, Ano, Hora, Minuto, Segundo, Host_a, Port_a, Host_b, Port_b, Sessão. Com esse conjunto realizava-se uma comparação manual das duas informações e se o evento fosse dado como inválido era então descartado. Se a informação fosse constatada como um evento de ataque real, então a marcação dos campos status e do nome do ataque e classe eram feitas na base de dados de tráfego baseado no número da sessão. Para o conjunto de dados de teste do DARPA 99 o campo de fluxo (Flow) também era levado em consideração na busca dos dados na base de tráfego.

4.4 Definição das Classes e dos Tipos de Ataque

Como pudemos observar no processo de carga dos dados a marcação dos eventos de ataque foram identificados pelo campo Status e AttackName do arquivo de lista de eventos, onde 0 significa não haver ataque e 1 a presença de ataque seguido do nome do mesmo. Foram escolhidas somente duas classes para cada tipo de ataque selecionado. A classe Ataque, que define a presença de um evento de ataque sob análise e a classe NAtaque, que define a não presença do mesmo, podendo nesse caso ser um tráfego normal ou um outro tipo de ataque.

A escolha dos ataques para realização do trabalho se baseou principalmente na distribuição dos mesmos na base de dados. Levou-se em consideração a possibilidade deles serem percebidos como tráfego de rede e tivessem relevância no cenário atual de segurança. Tentou-se selecionar um ataque de cada classe, mas a classe U2R não possui ataques com perfil de rede, sua presença só é percebida na análise de detecção de intrusão de host (HIDS). As Tabelas 4.8 e 4.9 listam os ataques que foram definidos para os testes junto com os valores quantitativos da ocorrência na base de dados de treinamento e de teste e o percentual em relação ao total geral de registros e ao total de registros de evidência de ataque.

Registros de Treinamento			
Tipos de Ataques	Quantidade de Registros	Relação sobre o Total de Registros	Relação sobre o Total de Registros c/ Ataque
Guest (R2L)	50	0,002%	0,003%
Portsweep (Probe)	11.617	0,48%	0,74%
Neptune (DoS)	1.526.643	63,64%	97,29%

Tabela 4.8: Lista dos ataques selecionados com o levantamento quantitativo na base de treinamento

Foi escolhido um ataque de cada classe passível de análise através de tráfego de rede. Como pode ser observado há uma predominância do ataque Neptune tanto na base de

Registros de Teste			
Tipos de Ataques	Quantidade de Registros	Relação sobre o Total de Registros	Relação sobre o Total de Registros c/ Ataque
GuessTelnet (R2L)	67	0,0056%	0,0770%
Portsweep (Probe)	262	0,0219%	0,3010%
Neptune (DoS)	72075	6,0226%	82,8077%

Tabela 4.9: Lista dos ataques selecionados com o levantamento quantitativo na base de teste

treinamento como na base de teste. O ataque Portsweep possui uma alta ocorrência na base de treinamento, mas o mesmo comportamento não é observado na base de testes. Já o ataque Guest, que na base de teste recebeu o nome de GuessTelnet, possui pouca ocorrência nas duas bases. Veremos a seguir mais detalhadamente de cada um dos ataques selecionados e como eles foram representados para a classificação.

4.4.1 Ataque Guest

Segundo KENDALL [45] o ataque Guest é uma variação do ataque de Dicionário sendo classificado como um ataque de acesso remoto para um usuário local (R2L – Remote to Local). É um ataque no qual o atacante tenta obter acesso a uma estação através de tentativas sucessivas a possíveis usuários e senhas. Como Guest é geralmente o nome de uma conta de usuário criada para permitir o acesso de visitantes ao sistema, muitas vezes vindo pré-configurada sem senha ou com uma senha default fraca, torna-se a primeira tentativa de vulnerabilidade praticada pelos invasores [46].

O ataque Guest é geralmente observado em infra-estruturas que utilizam os serviços Telnet ou rlogin, estes serviços de rede têm a finalidade de prover acesso remoto a usuários. Nesse trabalho utilizamos como referência o ataque Guest sobre o serviço Telnet na porta de serviço 23.

Esse tipo de ataque pode ser analisado por dois tipos de detectores de intrusão, de host

ou de rede. Os SID de hosts detectam este ataque através da análise dos logs de tentativas sucessivas de acesso à conta Guest, bastando para isso estabelecer um limiar para geração dos alarmes. Quando esse tipo de ataque é investigado por um SID de rede, não se tem acesso aos logs do sistema, então tem que se encontrar uma característica no tráfego que evidencie esse comportamento.

Os ataques Guest simulados na Avaliação de Detecção de Intrusão DARPA 98 e DARPA 99 foram feitos utilizando-se um programa em Perl que tentava acessos sucessivos à conta Guest utilizando uma senha baseada em permutações de nome de usuários.

A Tabela 4.10 descreve detalhes sobre as informações contidas na base de dados de tráfego de treinamento e teste sobre esse ataque.

Guest	Quantidade de Ataques	Quantidade de Registros	Relação sobre o Total de Registros	Relação sobre o Total de Registros c/ Ataque
Dados de Treinamento	1	50	0,0021%	0,0032%
Dados de Teste	4	67	0,0056%	0,0770%

Tabela 4.10: Quantificação do Ataque Guest

A quantidade de ataques desse tipo tanto na base de treinamento quanto na de teste é muita baixa, inferior a 1%, o que prejudicou muito a análise dos vetores de características e a realização dos testes. Como o comportamento desse ataque é bem conhecido, pode-se fazer um simulador que gerou dados sintetizados com o objetivo de aumentar o número de amostras e conseqüentemente melhorar o treinamento.

4.4.2 Definição dos Vetores de Características

Observando-se o comportamento do ataque Guest verificou-se que a diferença do seu comportamento para o normal é a sua ocorrência no tempo, ou seja, enquanto em uma situação normal ocorrem no máximo 3 a 5 conexões dentro de um minuto, durante um ataque Guest observamos a ocorrência de um número superior a 30 conexões nesse mesmo

espaço de tempo.

Para representar esse comportamento utilizamos as seguintes características: total de conexões (TxG) na porta 23 (Telnet) e o total de sinais Reset (RST) do protocolo TCP (resets_sent_a2b), no sentido da origem para o destino, no intervalo de 5 minutos. O comportamento dessas duas variáveis podem ser observadas nas Figuras 4.3 e 4.4 onde temos a quantidade de cada uma na presença ou não de ataque. Os dados dos gráficos foram gerados com as informações da base de dados de treinamento e de um sintetizador aleatório de dados que gerou mais informações baseadas nas existentes para melhorar a análise.

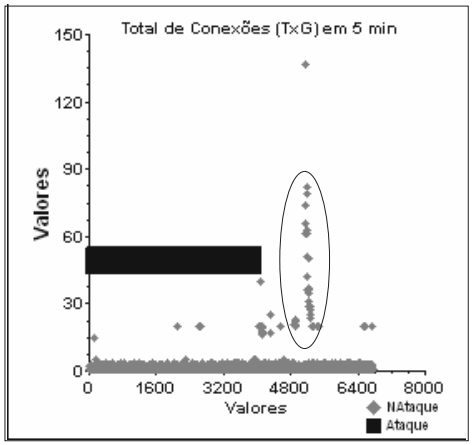


Figura 4.3: Comportamento da característica do número total de conexões (TxG) em 5 minutos

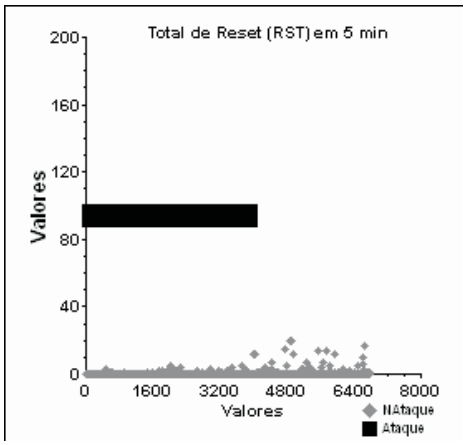


Figura 4.4: Comportamento da característica do número total de Reset (RST) em 5 minutos

Conforme evidenciado pelas Figuras 4.3 e 4.4 a presença deste ataque é facilmente identificada pelo número total de conexões que é significativamente maior que nos tráfegos considerados normais, o mesmo comportamento é observado com o número total de Resets. Esse comportamento já era esperado, pois um número elevado de TxG indica uma alta taxa de conexões ou tentativa de conexão e associado a um alto valor de RST indica uma taxa elevada de desconexões, coincidindo com o padrão de uma tentativa de acesso a uma conta de usuário mal sucedida . Fazendo-se uma análise mais criteriosa sobre os pontos na vertical da Figura 4.3 (sinalizado com um círculo), observamos que os mesmos representam evidências de outros ataques que possuem as mesmas características, como o ataque Dicionário (Dict) e Neptune que estão presentes na mesma porta de serviço utilizada pelo ataque Guest.

4.4.3 Ataque Neptune

O ataque Neptune, também conhecido como ataque SYN Flood, é classificado como um ataque de negação de serviço (DoS – Denial of service). Segundo KENDALL [45] toda implementação TCP/IP é vulnerável a esse tipo de ataque ou pelo mesmo em algum grau. O objetivo desse ataque é fazer solicitações sucessivas de conexões a um serviço. Como a solicitação de uma conexão causa a abertura de um processo para que o mesmo seja atendido, e esse mecanismo ocupa recursos do sistema, se forem feitas muitas solicitações simultaneamente isso pode levar o serviço ao esgotamento fazendo com que o mesmo pare de responder a novas solicitações, temporariamente ou permanentemente. Ou seja, negando a oferta do serviço que é o objetivo principal deste tipo de ataque [47]. O sintoma desse ataque no host atacado depende basicamente do serviço nele configurado.

Esse ataque pode ser detectado por um SDI de rede ou de host. No primeiro observa-se o número de conexões SYN destinadas a uma mesma estação. No segundo monitora-se o tamanho da tabela de conexões até um limite pré-estabelecido.

A simulação do ataque Neptune na Avaliação de Detecção de Intrusão DARPA 98 e DARPA 99 foi realizada utilizando-se um programa em linguagem C da bugtraq o qual cria um número configurável de conexões simultâneas para um host destino especificado.

A Tabela 4.11 descreve detalhadamente as informações contidas na base de dados de tráfego de treinamento e teste sobre esse ataque.

Neptune	Quantidade de Ataques	Quantidade de Registros	Relação sobre o Total de Registros	Relação sobre o Total de Registros c/ Ataque
Dados de Treinamento	13	1.526.643	63,6%	97,3%
Dados de Teste	4	72.075	6,0%	82,8%

Tabela 4.11: Quantificação do ataque Neptune

Podemos observar pelos números que o ataque netuno é o mais quantitativo de todos os ataques, tanto na base de treinamento quanto na base de teste. Ele está presente em 63,6% dos registros de treinamento, mais que a quantidade de registros de tráfego normal (não ataque). O mesmo não acontece na base de teste, mas mesmo assim ele é bem representativo em relação aos registros de ataques, 82,8%. Esses números facilitaram muito a escolha dos vetores de características.

4.4.4 Definição dos Vetores de Características

Como dito anteriormente o ataque Neptune tem uma característica marcante, a alta quantidade de solicitações de conexão de uma origem externa para um destino interno à rede em um curto espaço de tempo. Inicialmente pensou-se que somente o número total de conexões (TxG) e o número total de sinais FIN, SYN e RST, denominado FSR [48], no intervalo de tempo seriam suficientes para detectar o comportamento desse ataque. O FSR é a soma dos três sinais (FSR = FIN+SYN+RST) utilizados por uma conexão TCP/IP, o SYN marca o início de uma sessão, o FIN o término e o RST o encerramento da mesma. Esses três sinais juntos fazem a indicação de uma conexão fim a fim. Uma quantidade elevada de FSR pode indicar uma anomalia de comportamento do tráfego, como por exemplo, um ataque.

As Figuras 4.5 a 4.16 mostram o comportamento desses sinais na base de dados de

treinamento.

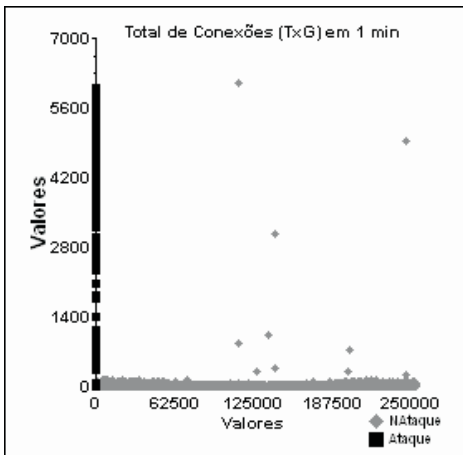


Figura 4.5: Total de conexões em 1 minuto em toda a base de treinamento

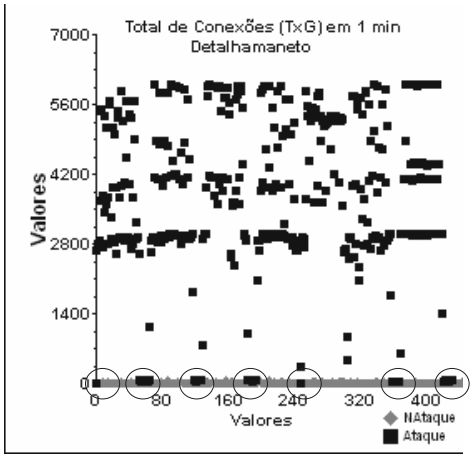


Figura 4.6: Detalhamento do total de conexões em um seguimento da base de treinamento

Observamos na Figura 4.6 que na presença de ataque o número total de conexões (TxG) em 1 minuto é muito superior ao do tráfego normal. Algumas indicações de ataques se misturam no tráfego normal (sinalizado com um circulo) indicando uma variação desse ataque que se espalha no tempo para não ser percebido.

Observamos que os ataques marcados nos gráficos 4.5, 4.6, 4.7 e 4.8 não podem ser identificados somente pela presença do FSR e do número de conexões, pois se confundem com o tráfego normal. Por esse motivo foram acrescentados mais 3 vetores de características, são eles: o total do sinal ACK enviado da origem para o destino no intervalo de 1 minuto (ACK_a2b), o total de pacotes recebidos e transmitidos em 1 minuto (BPP) e a média desse total pelo número de conexões (MBPP).

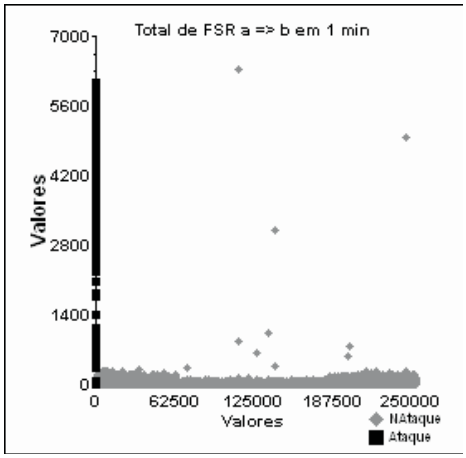


Figura 4.7: Total de FSR da origem para o destino (FSR_a2b) em toda a base de treinamento

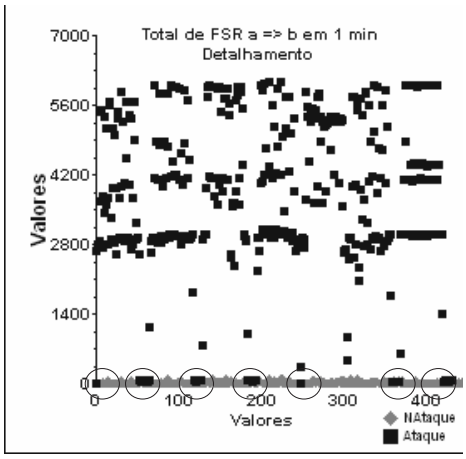


Figura 4.8: Total de FSR da origem para o destino (FSR_a2b) em um seguimento da base de treinamento

O sinal ACK vindo de uma origem externa em direção à rede interna indica que alguma estação que iniciou uma conexão está respondendo normalmente a uma conversação. A não existência desse sinal em uma conversação já iniciada indica que alguém pediu uma conexão mas não continuou o processo. Isso pode ocorrer por vários motivos, um deles seria a rede estar ocupada. Mas a ocorrência sucessiva desse fenômeno associada a um mesmo endereço IP de origem é uma forte evidência de um ataque Neptune em andamento, como pode ser visto nas Figuras 4.11 e 4.12.

O total de pacotes transmitidos e recebidos (BPP) por si só não é uma boa evidência de ataque, mas usado em conjunto com a média (MBPP) em relação ao total de conexões

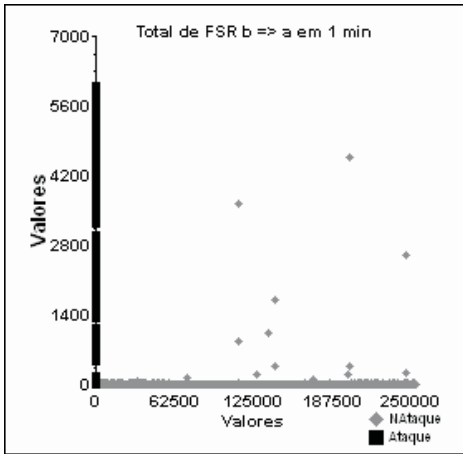


Figura 4.9: Total de FSR do destino para a origem (FSR_b2a) em toda a base de treinamento

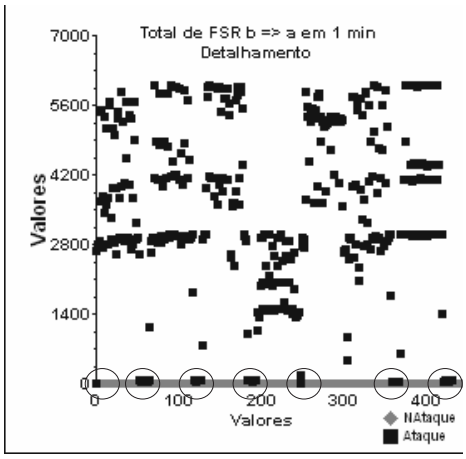


Figura 4.10: Total de FSR do destino para a origem (FSR_b2a) em um seguimento da base de treinamento

(TxG) torna-se muito útil nesse processo como podemos ver nas Figuras 4.13, 4.14, 4.15 e 4.16.

A Figura 4.16 mostra claramente o comportamento do ataque Neptune em relação à quantidade de informações enviadas e recebidas, praticamente nenhuma. Isso é um contra-senso, pois se existe uma solicitação de conexão porque não existe tráfego? Essa é uma das características dos ataques DoS.

Como o Neptune é predominante na base de treinamento, não existe quase tráfego normal interferindo no seu comportamento, somente alguns pontos isolados que podem

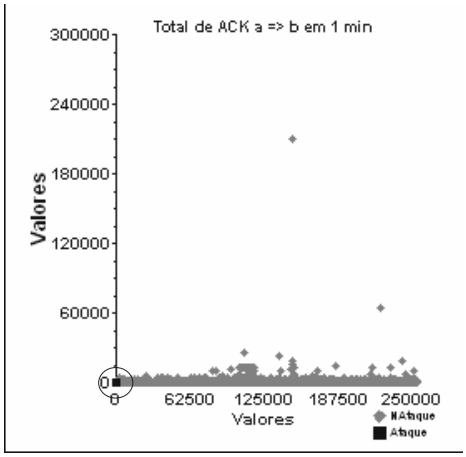


Figura 4.11: Total de ACK da origem para o destino (ACK_a2b) na base de treinamento

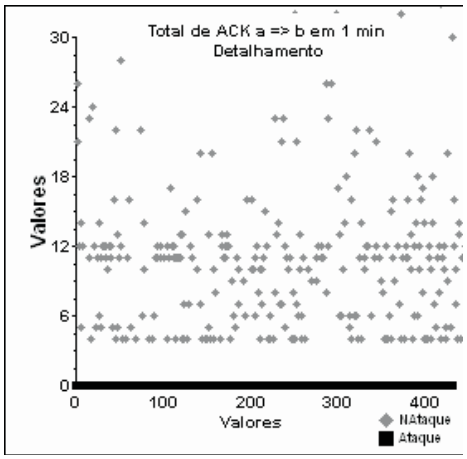


Figura 4.12: Total de ACK da origem para o destino (ACK_a2b) em um seguimento da base de treinamento

ser observados nas Figuras 4.5, 4.7 e 4.9. Essa falta de informação de tráfego normal pode atrapalhar o treinamento do sistema.

Os 6 vetores de características apresentados até aqui são os que irão representar esse ataque no sistema de classificação Naïve Bayes.

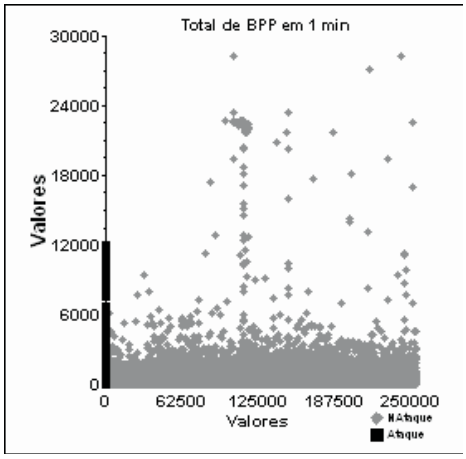


Figura 4.13: Total de pacotes recebidos e transmitidos em 1 minuto (BPP) na base de treinamento

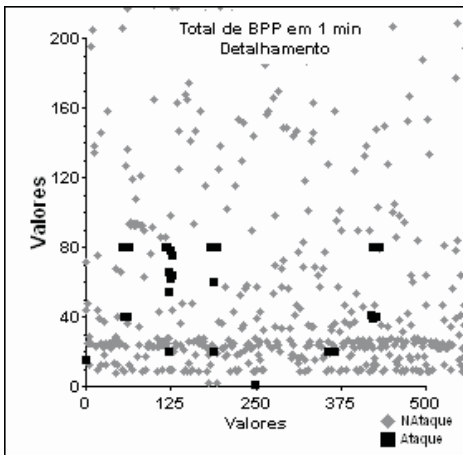


Figura 4.14: Total de pacotes recebidos e transmitidos em 1 minuto (BPP) em um seguimento da base de treinamento

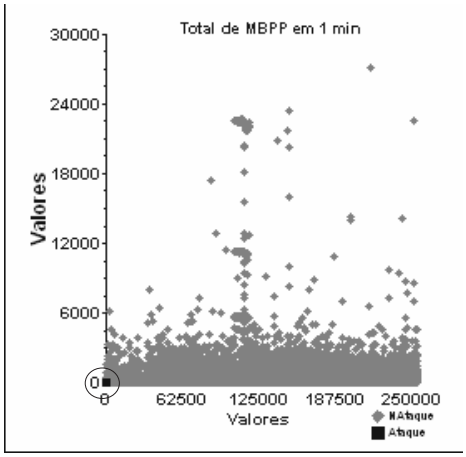


Figura 4.15: Média de pacotes por conexão (MBPP) na base de treinamento

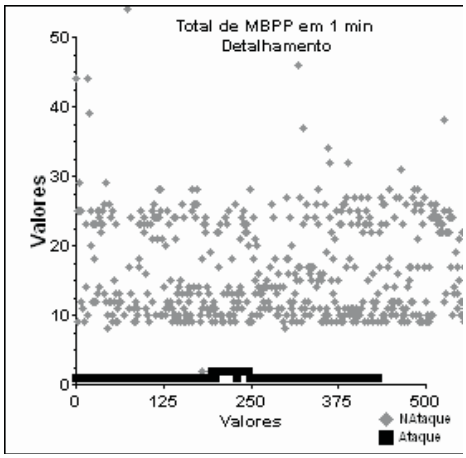


Figura 4.16: Média de pacotes por conexão (MBPP) em um seguimento da base de treinamento

4.4.5 Ataque Portsweep

O Portsweep é um ataque da família dos Port Scanners ou varredores de portas. Segundo os estudos desenvolvidos por STANIFORD [49], a varredura de portas pode ser caracterizada pela ocorrência de pacotes destinados a N combinações diferentes de portas de serviço oriundas de uma mesma fonte e detectável em M segundos.

A estratégia no uso da varredura de portas objetiva identificar os serviços disponíveis em um servidor, e assim explorar as possíveis vulnerabilidades, presentes nas diferentes implementações que suportam as facilidades oferecidas.

Conforme observado por NORTHCUTT [50], a estratégia de execução de um processo de varredura de portas pode ser reunida em dois grupos. O primeiro é denominado varredura horizontal, que busca uma porta de serviço específica a partir de uma lista de endereços IP, conhecido como IPSweep. O segundo é chamado de varredura vertical ou Portsweep, que a partir de um único endereço IP, tenta detectar todos os serviços disponíveis na estação.

As técnicas de varreduras de portas podem ser classificadas como: aberta, meio aberta, secreta e ociosa. Na varredura aberta, é realizada uma conexão completa, ou seja, um sinal SYN é enviado pela fonte, e como resposta é esperado um sinal SYN/ACK do receptor. Este comportamento se realizará no cenário onde o serviço solicitado esteja disponível na estação destino. Caso contrário, é esperada uma sinalização RST/ACK, indicando assim a indisponibilidade do mesmo.

A varredura meio aberta é similar à varredura aberta, sendo que na primeira, quando a porta estiver disponível, é enviado o pacote de retorno RST ao invés de um ACK.

No modo secreto, envia-se um pacote com várias informações de sinais de controle do protocolo (flags) inconsistentes, como SYN e ACK juntos como solicitação de conexão ao invés de somente um SYN.

A varredura ociosa utiliza um terceiro agente, usado com o objetivo de receber os pacotes de retorno, produzidos pelo receptor. Esta técnica permite ocultar a identificação do originador da varredura, tornando-se assim uma estratégia mais eficiente para a identifi-

cação de serviços.

Esse ataque pode ser detectado por um SDI de rede analisando a quantidade de conexões solicitadas em diferentes portas de serviços em um curto espaço de tempo.

A simulação do ataque Portsweep na Avaliação de Detecção de Intrusão DARPA 98 e DARPA 99 foi realizada utilizando-se programas usuais de mercado como, NMap, Satan, Saint dentre outros.

A Tabela 4.12 descreve detalhadamente as informações contidas na base de dados de tráfego de treinamento e teste sobre esse ataque.

Portsweep	Quantidade de Ataques	Quantidade de Registros	Relação sobre o Total de Registros	Relação sobre o Total de Registros c/ Ataque
Dados de Treinamento	14	11.617	0,48%	0,74%
Dados de Teste	13	262	0,022%	0,30%

Tabela 4.12: Quantificação do ataque Portsweep

Podemos observar pelos números que o ataque Portsweep não tem um quantitativo representativo de ataques, tanto na base de treinamento quanto na base de teste. Ele está presente em apenas 0,48% dos registros de treinamento e 0,02% na base de teste. Os números melhoram em relação aos registros de ataques, 0,74% para o treinamento e 0,3% para teste. Se levarmos em consideração a grande quantidade de ataques Neptune existentes, como visto anteriormente, esses números não são tão ruins, permitindo a escolha dos vetores de características.

4.4.6 Definição dos Vetores de Características

Observamos que o processo de conexão utilizado pelos programas de varredura apresentam como característica serem de duração muito curta, geralmente restrito a um ou dois pacotes. Assim, segundo os estudos de DICKERSON (2000), conclui-se que os res-

pectivos tráfegos que definem o evento de segurança investigado fica melhor caracterizado pelo conjunto chamado SDP (Source IP, Destination IP e Destination Port), constituído pelo endereço IP de origem, endereço IP de destino e porta de serviço de destino. A quantidade de ocorrências dos SDPs no tempo nos dá uma visão tridimensional (Figura 4.17) bem clara do comportamento de uma varredura [18].

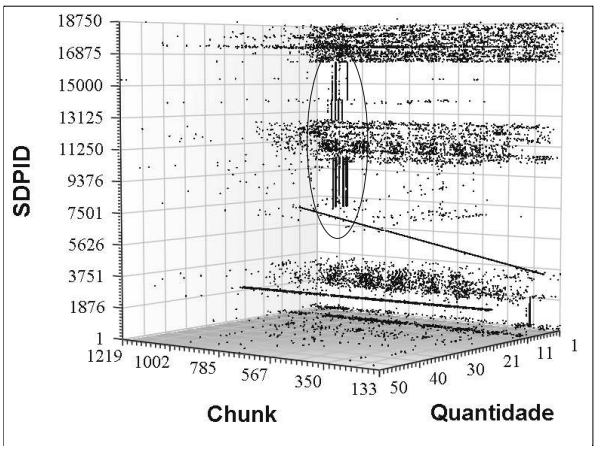


Figura 4.17: Evidência de ataques Portsweep (sinalizado por um círculo) através do SDP em função da quantidade de pacotes e o intervalo de tempo (chunk)

Outra característica é a média de pacotes enviados e recebidos em relação ao número total de conexões em um intervalo de tempo (MBPP).

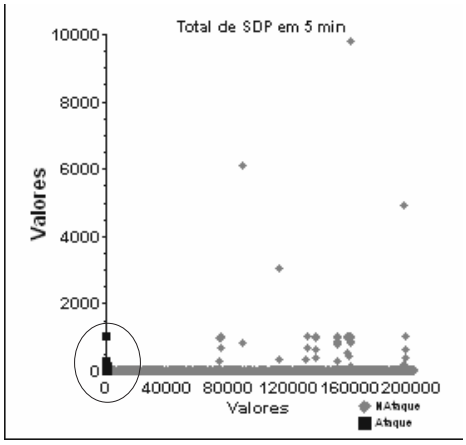


Figura 4.18: Total de SDPs em 5 minutos na base de treinamento

As Figuras 4.18 a 4.23 mostram o comportamento desses vetores de características na base de dados de treinamento.

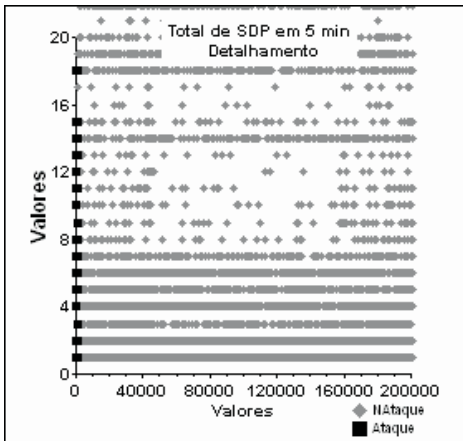


Figura 4.19: Total de SDPs em 5 minutos em um seguimento da base de treinamento

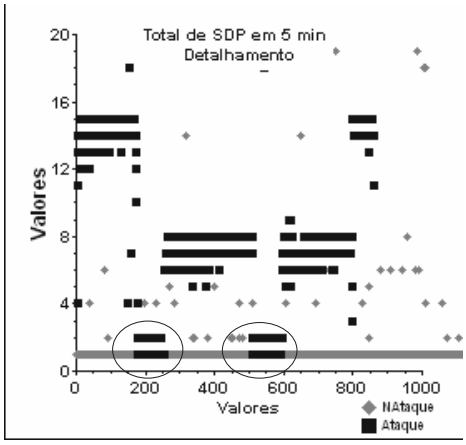


Figura 4.20: Total de SDPs em 5 minutos em um seguimento da base de treinamento

As Figuras 4.18 e 4.21 fornecem uma visão geral do comportamento do tráfego normal em relação aos vetores de características SDP e MBPP. O ataque está concentrado na parte inferior esquerda dos gráficos (sinalizado com um círculo) que estão detalhados nos gráficos seguintes. No gráfico 4.20 fica evidenciado pela linha horizontal cravada no valor 1 na parte inferior do gráfico, o comportamento normal da característica SDP. Os ataques são as outras linhas horizontais superiores, variando de valores. Essas linhas evidenciam o acesso sucessivo a diferentes portas de serviços em um curto espaço de tempo. Existem alguns ataques que aparecem junto com o tráfego normal (sinalizado com círculos). Esse comportamento é explicado por variações do ataque no qual o tempo de varredura é ampliado para não ser detectado. Esse tipo de variação pode ser detectado aumentando-se o tempo do acúmulo dos vetores de características, como pode ser observado nas Figuras 4.24 e 4.25.

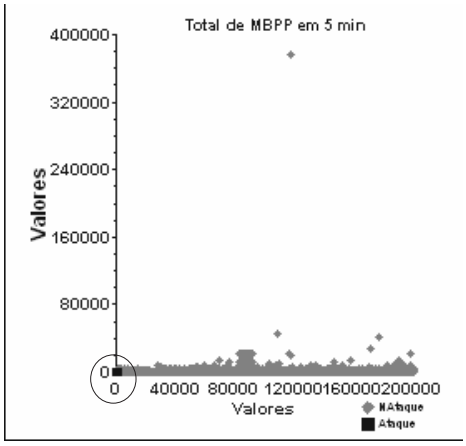


Figura 4.21: Média de pacotes por conexão em 5 minutos (MBPP) na base de treinamento

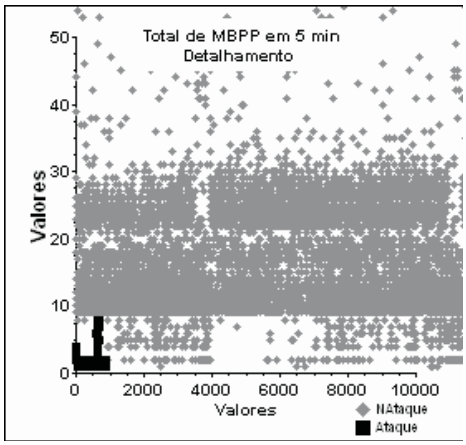


Figura 4.22: Detalhamento da média de pacotes por conexão em 5 minutos (MBPP) em um seguimento da base de treinamento

Na Figura 4.23 observamos dois comportamentos fora do padrão do ataque (sinalizados com círculos). Fazendo-se uma investigação mais detalhada observou-se que no total de pacotes recebidos e transmitidos havia não somente o ataque em curso, mas um acesso à porta de serviço 80 (HTTP) entre o host atacante e o atacado. Apesar de não ser um comportamento comum, pode-se encontrar essa situação em um cenário real de rede.

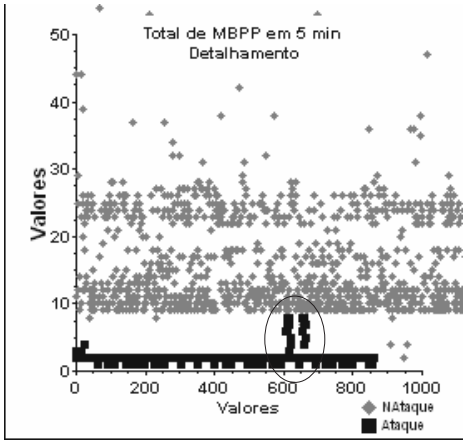


Figura 4.23: Detalhamento da média de pacotes por conexão em 5 minutos (MBPP) em um seguimento da base de treinamento

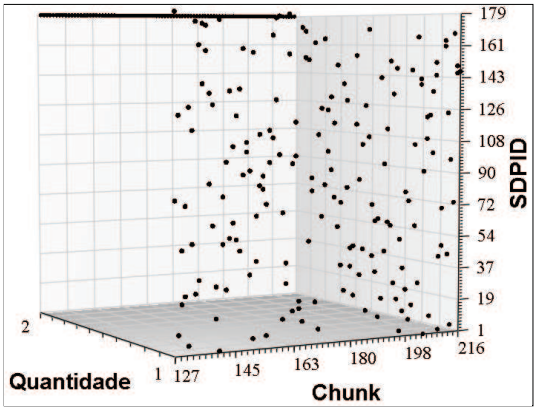


Figura 4.24: Ataque Portsweep com tempo de varredura ampliado

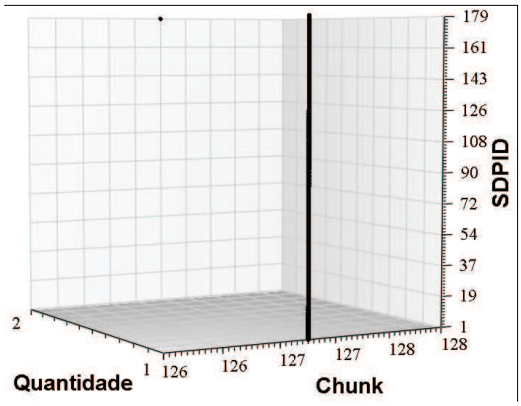


Figura 4.25: Visão da Figura 4.24 com intervalo maior de tempo dentro um mesmo intervalo de tempo (chunk)

Capítulo 5

Implementação do NBIS

5.1 Introdução

O Sistema de Inferência Naïve Bayes (NBIS – Naïve Bayes Inference System) é a proposta de um novo sistema de detecção de intrusão em redes de computadores baseado no modelo probabilístico com o intuito de contribuir na melhora do desempenho dos sistemas SDI hoje existentes.

Nesse capítulo iremos apresentar a implementação do sistema de detecção de intrusão proposto nesse trabalho. Passaremos por todas as etapas que compõem o sistema até a decisão final gerada pelo classificador.

5.2 Objetivo do NBIS

O NBIS foi idealizado com o objetivo de estudar e analisar condições específicas de segurança em redes de computadores. O NBIS se posiciona no cenário de segurança como um sistema probabilístico de detecção de intrusão baseado em assinatura.

A implementação permitiu avaliar o seu desempenho na detecção de alguns tipos de ataques ou anomalias de comportamento de tráfego e compará-lo com outro modelo baseado em lógica Fuzzy [18] e sua derivação Neuro-Fuzzy. Sua arquitetura é baseada no

modelo probabilístico de inferência Bayesiana usando o classificador Naïve Bayes para a classificação e detecção dos ataques.

5.3 Arquitetura do Sistema

A arquitetura do NBIS utiliza o modelo de inferência probabilístico com inferência seqüencial Bayesiana, distribuição normal dos vetores de característica e classificação probabilística através do classificador Naïve Bayes.

O NBIS possui 4 blocos fundamentais, como descrito na Figura 5.1, que formam a arquitetura do sistema que vai desde a aquisição dos dados até o módulo de classificação e decisão. Os blocos são:

1. Módulo de Carga de Dados
2. Módulo de Treinamento
3. Módulo Classificador
4. Módulo de Saída do Sistema

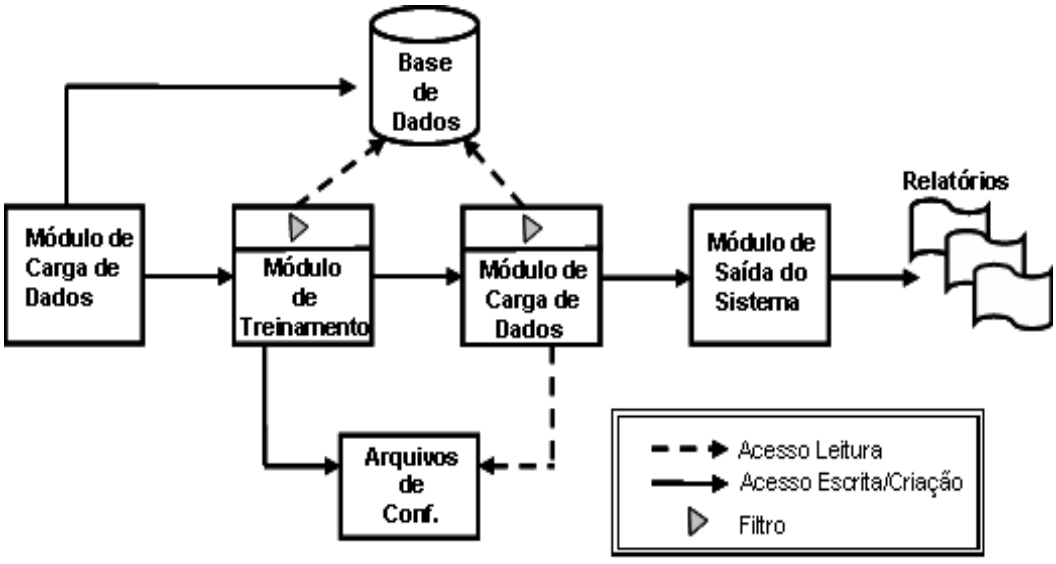


Figura 5.1: Diagrama em bloco do sistema NBIS

5.3.1 Módulo de Carga de Dados

O Módulo de Carga de Dados foi apresentado no Capítulo 4 e tem a responsabilidade de fazer a carga e normalização dos dados de treinamento e teste para uso pelos outros módulos.

5.3.2 Módulo de Treinamento

O treinamento é a parte mais importante da arquitetura, pois um treinamento mal feito pode colocar em risco toda a arquitetura do sistema. O Módulo de Treinamento é formado por programas que acessam a base de dados do conjunto de treinamento, carregado pelo Módulo de Carga de Dados.

O processo de treinamento ocorre em duas etapas (Figura 5.2). Na primeira é gerado o arquivo com os dados de treinamento baseados nos vetores de características definidas para cada tipo de ataque, como foi apresentado no Capítulo 4, a partir do conjunto de dados de treinamento.

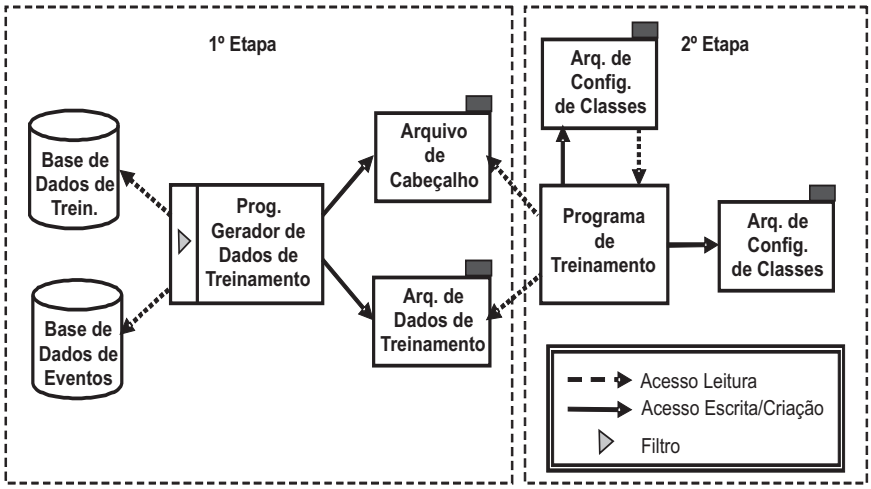


Figura 5.2: Diagrama em bloco do módulo de treinamento

Na segunda etapa é feito o treinamento do sistema, que consiste em gerar os arquivos de configuração a partir do arquivo de dados de treinamento, gerado no passo anterior, que serão usados pelo Módulo Classificador.

O processo de geração do arquivo de dados de treinamento consiste em ler as in-

formações do banco de dados de treinamento, filtrá-las, processá-las e ordená-las na seqüência correta para a carga pelo programa de treinamento. Antes da fase de processamento é feita uma filtragem nos dados de entrada. Essa filtragem funciona como um pré-classificador e depende do tipo de ataque que se está analisando. Para cada seqüência que é processada é verificado se os dados geram uma saída de indicação de ataque, ou seja, se aquele conjunto de dados é uma evidência real ou não de ataque. Isso é feito verificando-se a existência de marcação de ataque na base de dados de eventos, descrita no Capítulo 4, para aquele conjunto de dados usados para gerar os vetores de característica. A partir dessa informação é inserida ao final da seqüência de dados a indicação da classe à qual esse conjunto de dados pertence, podendo ser a classe Ataque ou Não Ataque, representada por NAtaque. É gerado ao final dessa fase o arquivo de cabeçalho, contendo a identificação de cada vetor de característica e da classe por último.

O treinamento utiliza o arquivo de dados de treinamento como entrada e gera a base de dados das classes do classificador na saída. Como estamos representando os dados de entrada a partir de uma distribuição normal, essa base de dados conterá as informações sobre a média ou valor esperado e variância para cada distribuição de densidade probabilística de verossimilhança, ou seja, para cada classe. Além disso a base de dados possui informações sobre a distribuição de probabilidade a priori e os valores mínimo e máximo de cada classe. A Figura 5.3 mostra um exemplo da base de dados das classes do classificador para o ataque do tipo Guess.

O programa de treinamento utiliza como apoio outros dois arquivos de configuração. O arquivo de domínio, que contém as informações dos valores máximo e mínimo de cada vetor de característica, o qual é gerado e utilizado pelo próprio programa de treinamento. O arquivo de cabeçalho (header) possui a informação sobre o nome do campo de cada vetor de característica e da classe e a ordem em que eles aparecem no arquivo de dados de treinamento.

O método de treinamento implementado no NBIS é o treinamento supervisionado de verossimilhança máxima, onde se calcula para cada conjunto classe e vetor de característica o valor que maximiza a função de densidade probabilística de verossimilhança. Como foi dito anteriormente, utilizou-se o conceito da distribuição normal para representar a dis-

```
/*----------------------------------------
  domains
----------------------------------------
dom(TRST) = ZZ [0, 99];
dom(TxG) = ZZ [1, 137];
dom(Decisao) = { NAtaque, Ataque };

/*----------------------------------------
  naive Bayes classifier
----------------------------------------
nbc(Decisao) = {
  params = 0, maxllh;
  prob(Decisao) = {
    NAtaque: 6769 (62.8%),
    Ataque : 4003 (37.2%) };
  prob(TRST|Decisao) = {
    NAtaque: N(0.111981, 1.31262) [6769],
    Ataque : N(94.4434, 8.21882) [4003] };
  prob(TxG|Decisao) = {
    NAtaque: N(2.52918, 48.8221) [6769],
    Ataque : N(49.4949, 8.3454) [4003] };
};

/*----------------------------------------
  number of attributes: 3
  number of tuples    : 10772
----------------------------------------
```

Figura 5.3: A base de dados das classes do ataque Guest possui 3 vetores de características (atributos), sendo um deles a definição das classes. A distribuição de probabilidade normal é apresentada na parte central da figura.

tribuição dos valores de vetores de característica. Isso significa que utilizamos a função de densidade probabilística de distribuição normal explicada no Capítulo 3. O resultado do treinamento é o valor da distribuição normal para cada classe e vetor de característica no formato $N \sim (\mu\ ,\ \sigma^2)$ mais a quantificação dos vetores utilizados. Para o cálculo da média ou valor esperado e a variância são utilizadas as seguintes fórmulas pelo programa de treinamento:

$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$ para o cálculo da média ou valor esperado e

$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$ para o valor da variância.

Onde: x_i é o valor do atributo ou vetor de característica de uma determinada classe e n é a quantidade total de amostras dessa mesma classe.

5.3.3 Módulo Classificador

A inferência e a classificação no NBIS são feitas em um único passo, conforme o diagrama em bloco da Figura 5.4. Nesse processo o programa de classificação Naïve Bayesiano lê a base de dados das classes do classificador, gerado na fase de treinamento, e as informações do banco de dados de teste contendo os novos valores de entrada a serem classificados. Antes do processamento é feita uma filtragem nos dados de entrada. Essa filtragem funciona como uma "pré-classificação"separando as informações pertinentes ao ataque em questão que serão analisadas pelo classificador Naïve Bayes. Esse processo reduz significativamente a quantidade de dados a ser analisada pelo classificador.

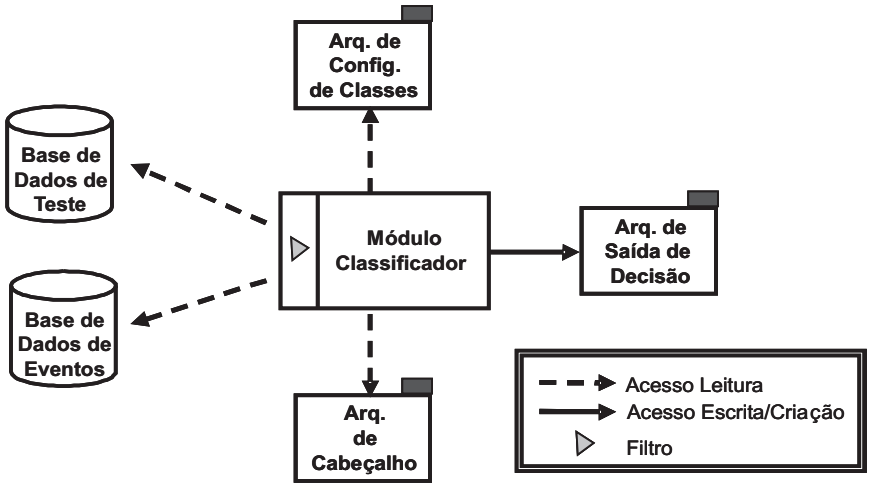


Figura 5.4: Diagrama em bloco da classificação

Para que fosse possível medir o desempenho do classificador foi necessário marcar a seqüência de dados de teste com a classe correta antes da classificação. Esse processo é semelhante ao realizado na primeira fase do treinamento, mas sem qualquer modificação da base de dados das classes do classificador. Com isso é possível verificar se a classificação realizada foi correta ou não.

A inferência usada pelo NBIS é a inferência Bayesiana com seqüencialmente. Neste modelo a probabilidade a posteriori é usada como probabilidade a priori no próximo passo de inferência. Para cada classe é calculada a probabilidade a posteriori da nova evidência dados os valores de treinamento. A definição da classe é feita usando-se o método de máximo a posteriori, conhecido como MAP (Máximo A Posteriori). Sendo assim o novo

atributo ou vetor de característica será classificado na classe em que ocorrer o maior valor de probabilidade a posteriori. O calculado do MAP é feito usando-se o teorema de Bayes. Para isso precisamos da probabilidade a priori de cada classe. O cálculo da probabilidade a priori de uma classe é dada pela razão entre a freqüência da ocorrência de amostras de uma mesma classe e o número total de amostra e é calculada pela seguinte fórmula:

$$Priori = \frac{n_{c_i}}{n}$$

Onde: n_{c_i} é a quantidade de vetores de característica da classe c_i e n é o número total de amostras da base.

Outra informação fornecida pelo classificador é a confiança. Esse valor expressa o quanto de certeza o classificador tem sobre aquela determinada classificação, estando ela certa ou errada. Se esse valor for igual a 1, então o classificador está dizendo que tem 100% de confiança de estar certo na sua resposta. Se esse valor for igual a 0, então o classificador tem 0% de confiança na sua classificação, ou seja, não tem certeza alguma. Geralmente isso ocorre quando os valores de entrada estão fora do raio de atuação do classificador. A confiança é calculada pela razão entre o valor da probabilidade a posteriori da classe escolhida pela soma das probabilidades a posteriori de todas as classes como segue:

$$Confiança = \frac{Posteriori_{cl_i}}{\sum_{i=1}^{n}(Posteriori_{cl_n})}$$

Onde: $Confiança$ é a confiança do classificados, $Posteriori_{cli}$ é a probabilidade da classe cl_i escolhida e o denominador é a soma da probabilidade de todas as n classes.

O arquivo de saída da classificação é semelhante ao arquivo de dados de treinamento com duas colunas a mais, uma com o resultado da classificação e outra com o valor da confiança.

5.3.4 Módulo de Saída do Sistema

Ao final do processo de inferência e classificação o Módulo Classificador gera os arquivos de saída de decisão que são usados para gerar os relatórios de medidas de de-

sempenho do sistema.

5.4 Avaliação de Desempenho

Para medir o desempenho do sistema foi necessário criar um padrão que pudesse aferir o trabalho realizado. O padrão de medida de mercado para sistemas de detecção de intrusão usa como base a análise quantitativa de alarmes de intrusão ou de tentativa de intrusão verdadeiros, denominados Real Positivos (RP), e a quantidade de indicação de intrusão falsas, mas que foram confundidas como verdadeiras, denominado Falso Positivo (FP). Dessas duas medidas podemos obter o Falso Negativo (FN), calculado retirando-se do número total de amostras de ataque às indicações verdadeiras (RP), e o Real Negativo, calculada retirando-se do número total de amostras sem evidência de ataque as erroneamente detectadas. Os valores úteis para avaliar o classificador são descritos abaixo:

1. Total de Amostras Analisadas (TAA): Indica o total das amostras contidas na base de dados que foram usadas para o teste e analisadas pelo classificador.

2. Real Positivo (RP): Quantidade de eventos com indicação de ataque e que foram comprovadas, indicando que o sistema acertou na classificação.

3. Falso Positivo (FP): Quantidade de eventos com indicação de ataque e que não foram comprovados, ou seja, o sistema errou na classificação.

4. Real Negativo (RN): Quantidade de eventos sem indicação de ataque e que foram comprovadas que não eram, indicando que o sistema acertou na classificação.

5. Falso Negativo (FN): Quantidade de eventos sem indicação de ataque e que foram comprovadas como sendo, ou seja, o sistema errou na classificação.

6. Total de Ataques (TAt): Indica a quantidade real de ataques presentes na base de dados usada para o teste. O TAt pode ser achado somando-se a quantidade da Real Positivos com a quantidade de Falso Negativos (TAt = RP + FN).

7. Percentual Referencial de Ataque (PRA): Indica o percentual de ataques presentes na base em análise. É calculado pela razão entre o total de ataques e o número total de amostras efetivamente analisadas (PRA = TAt /TAA).

8. Percentual de Acerto Referencial (PAR): Indica a eficiência do classificador. É calculado pela razão entre a quantidade de Real Positivo e o número total de ataques presentes na amostra (PAR = RP/TAt).

9. Percentual de Acerto (PA): Indica o quanto o classificador acertou. É calculado pela razão da soma dos acertos e o número total de amostras efetivamente analisadas (PA = RN + RP/TAA).

10. Percentual de Erro (PE): Indica o quanto o classificador errou. É calculado pela razão da soma dos erros e o número total de amostras analisadas (PE = FP + FN/TAA).

5.5 Ferramentas Utilizadas

A arquitetura do NBIS foi desenvolvida utilizando-se programas escritos em Perl, descritos no Anexo I, e Banco de Dados MySQL sobre plataforma Linux. Para a carga e normalização dos dados brutos do DARPA 98 e DARPA 99 foram usados vários programas escritos em Perl. No treinamento e classificação foi usada uma coleção de softwares desenvolvidos por Cristian Borgelt do Departamento de Engenharia de Linguagem e Processamento do Conhecimento da Universidade de Magdeburg na Alemanha. O sistema chamado de indutor de classificação Bayesiana (BCI – Bayesian Classifier Induction), descrito no Anexo I, foi apresentado em um dos seus trabalhos [35] e em seu livro [28]. Ele permite fazer classificações baseadas em Redes Bayesianas ou Naïve Bayesiana. Nesse trabalho utilizou-se a linha Naïve Bayesiana, devido a característica do problema como apresentado no Capítulo 4.

O BCI foi todo desenvolvido em linguagem "C"e seu código fonte disponibilizado. Ele é composto por 3 utilitários principais. O *dom*, o *bci* e o *bcx*. O *dom* é usado para gerar o arquivo de domínio das classes usado na fase de treinamento. O *bci* é o programa de treinamento e usa o arquivo de domínio como base junto com o arquivo de dados de trei-

namento como visto anteriormente nesse capítulo. O *bcx* é o classificador propriamente dito, ele é responsável por calcular os valores das probabilidades usadas na classificação Naïve Bayesiana e gerar o arquivo de saída com o resultado da classificação.

Os utilitários do BCI foram incorporados a programas escritos em Perl que tornaram o sistema unificado conforme apresentado pela arquitetura do sistema.

5.6 Treinamento do Sistema

Aplicando-se a teoria apresentada na arquitetura do sistema e as informações levantadas na fase de aquisição e levantamento de dados, temos todas as ferramentas para iniciar o treinamento do sistema para cada um dos ataques apresentados. Para isso utilizamos o programa de treinamento do BCI (Bayes Classifier Induction) apresentado anteriormente.

5.6.1 Treinamento do Ataque Guest

O ataque Guest é representado na base de dados de treinamento somente por 1 ataque com 50 registros. Devido a essa escassez de dados, foi necessária a criação de dados sintetizados para melhorar a decisão do sistema. Esses dados foram criados por um programa gerador de dados aleatórios que foi ajustado a fim de criar valores para os dois vetores de características dentro dos patamares observados como tráfego sobre ataque e normal. Mesmo utilizando os dados sintetizados o mais recomendável seria levantar mais dados de tráfego real, principalmente sob ataque para melhorar o treinamento do sistema.

O levantamento dos vetores de características para o treinamento do ataque Guest é feito através de um programa em Perl que acessa a base de dados de treinamento separando-a em blocos de 5 minutos e procura pela ocorrência da porta de serviço de destino 23 (Telnet) nesse intervalo. Se existirem registros com essa característica então contabiliza-se o número total de conexões (TxG) e o total de Resets (RST) enviados pelo atacante. Esse processo se repete para todos os endereços IPs de origem única, ou seja, todas as conexões cujo endereço IP de origem seja o mesmo, até o fim da lista de IPs ou o fim do intervalo de tempo, percorrendo toda a base.

Os vetores de características são então salvos no arquivo de treinamento e a classe à qual pertencem identificada através das indicações “Ataque” e “NAtaque” respectivamente no campo Decisão no final de cada linha. Pelos resultados obtidos os dois vetores de características foram suficientes para caracterizar esse ataque e foram usadas para treinar o classificador Naïve Bayes considerando a arquitetura mostrada na Figura 5.5.

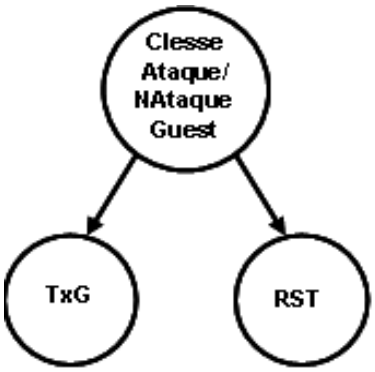


Figura 5.5: Arquitetura do classificador Naïve Bayes para o ataque Guest com os 2 vetores de características e as 2 classes Ataque e NAtaque.

Após esse levantamento realiza-se o treinamento utilizando o programa BCI. Esse programa lê todos os campos do arquivo de treinamento e calcula as funções de distribuição de probabilidade normal para cada vetor de característica e para cada classe. A saída desse programa é mostrada na Figura 5.6 que é a base de dados das classes usado pelo classificador.

5.6.2 Treinamento do Ataque Neptune

O ataque Neptune tem uma representatividade muito elevada na base de dados de treinamento, os 13 ataques possuem 1.526.643 registros, representando 63,6% do total da base.

O levantamento dos vetores de características para o treinamento do ataque Neptune é feito através de um programa em Perl que acessa a base de dados de treinamento separando-a em blocos de 1 minuto. Dentro desse intervalo de tempo procura-se pelo conjunto endereço IP de origem e endereço IP destino únicos, ou seja, todas as conexões cujo endereço IP de origem e destino sejam os mesmos, e analisam-se todas as sessões

```
/*-----------------------------------------
  domains
-----------------------------------------
dom(TRST) = ZZ [0, 99];
dom(TxG) = ZZ [1, 137];
dom(Decisao) = { NAtaque, Ataque };

/*-----------------------------------------
  naive Bayes classifier
-----------------------------------------
nbc(Decisao) = {
  params = 0, maxllh;
  prob(Decisao) = {
    NAtaque: 6769 (62.8%),
    Ataque : 4003 (37.2%) };
  prob(TRST|Decisao) = {
    NAtaque: N(0.111981, 1.31262) [6769],
    Ataque : N(94.4434, 8.21882) [4003] };
  prob(TxG|Decisao) = {
    NAtaque: N(2.52918, 48.8221) [6769],
    Ataque : N(49.4949, 8.3454) [4003] };
};

/*-----------------------------------------
  number of attributes: 3
  number of tuples    : 10772
-----------------------------------------
```

Figura 5.6: Resultado do treinamento Naïve Bayes para o ataque Guest.

desse conjunto, calculando-se o número total de conexões (TxG), a quantidade de FSR da origem para o destino (FSR_a2b) e do destino para a origem (FSR_b2a), o total de sinais ACKs da origem para o destino (ACK_a2b), o total de pacotes transmitidos e recebidos (BPP) e a sua razão pelo número total de conexões (MBPP = BPP/TxG). Esse processo se repete até o fim da lista de IPs ou o fim do intervalo de tempo, percorrendo a base toda.

Os vetores de características são então salvos no arquivo de treinamento e a classe a qual pertencem identificada através das indicações “Ataque” e “NAtaque” no campo Decisão no final de cada linha. Pelos resultados obtidos os seis vetores de características foram suficientes para caracterizar esse ataque e foram usados para treinar o classificador Naïve Bayes considerando a arquitetura mostrada na Figura 5.7.

Após esse levantamento realiza-se o treinamento utilizando o programa BCI. Esse programa lê todos os campos do arquivo de treinamento e calcula as funções de distribuição de probabilidade normal para cada vetor de característica e para cada classe. A saída desse programa é mostrada na Figura 5.8 que é a base de dados das classes usado pelo classificador.

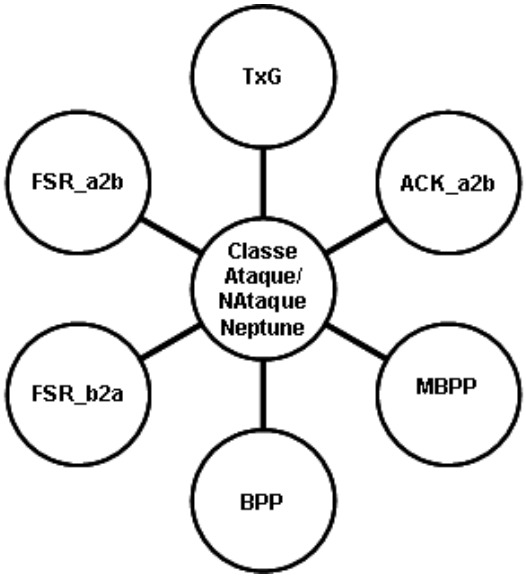


Figura 5.7: Arquitetura do classificador Naïve Bayes para o ataque Neptune com os 6 vetores de características e as 2 classes Ataque e NAtaque.

5.6.3 Treinamento do Ataque Portsweep

O ataque Portsweep tem uma representatividade significativa na base de dados de treinamento, com 14 ataques representando 0,48% do total da base e 0,74% dos registros de ataque. Se expurgarmos o ataque Netuno da base esses valores passam para 1,33% e 27,35% respectivamente.

O levantamento dos vetores de características para o treinamento do ataque Portsweep é feito através de um programa em Perl que acessa a base de dados de treinamento separando-a em blocos de 5 minutos. Dentro desse intervalo de tempo procura-se pelo conjunto endereço IP de origem e endereço IP destino únicos, ou seja, todas as conexões cujo endereço IP de origem e destino sejam os mesmos, e analisam-se todas as sessões desse conjunto, calculando-se o total de portas acessadas por este par de endereços nesse intervalo (SDP) e a razão do número total de pacotes transmitidos e recebidos pelo número total de conexões (MBPP = BPP/TxG). Esse processo se repete até o fim da lista de IPs ou o fim do intervalo de tempo, percorrendo a base toda.

Os vetores de características são então salvos no arquivo de treinamento e a classe à qual pertencem identificada através das indicações “Ataque” e “NAtaque” no campo

```
/*-----------------------------------------
  domains
-----------------------------------------*/
dom("TxG") = ZZ [1, 9069];
dom("FSR_a2b") = ZZ [0, 9071];
dom("FSR_b2a") = IR [0, 6001];
dom("ACK_a2b") = ZZ [0, 210376];
dom("BPP") = ZZ [1, 376065];
dom("MBPP") = ZZ [1, 376065];
dom(Decisao) = { NAtaque, Ataque };

/*-----------------------------------------
  naive Bayes classifier
-----------------------------------------*/
nbc(Decisao) = {
  params = 0, maxllh;
  prob(Decisao) = {
    NAtaque: 254359 (99.8%),
    Ataque : 432 (0.2%) };
  prob("TxG"|Decisao) = {
    NAtaque: N(3.42846, 660.706) [254359],
    Ataque : N(3533.9, 3.25894e+06) [432] };
  prob("FSR_a2b"|Decisao) = {
    NAtaque: N(6.44402, 826.722) [254359],
    Ataque : N(3544.55, 3.26797e+06) [432] };
  prob("FSR_b2a"|Decisao) = {
    NAtaque: N(0.705143, 181.526) [254359],
    Ataque : N(3298.91, 3.37976e+06) [432] };
  prob("ACK_a2b"|Decisao) = {
    NAtaque: N(53.0078, 441635) [254359],
    Ataque : N(0.0925926, 3.69513) [432] };
  prob("BPP"|Decisao) = {
    NAtaque: N(103.485, 1.39199e+06) [254359],
    Ataque : N(4018.22, 5.73788e+06) [432] };
  prob("MBPP"|Decisao) = {
    NAtaque: N(61.1082, 1.08983e+06) [254359],
    Ataque : N(1.12037, 0.105881) [432] };
};

/*-----------------------------------------
  number of attributes: 7
  number of tuples    : 254791
-----------------------------------------*/
```

Figura 5.8: Resultado do treinamento Naïve Bayes para o ataque Neptune.

Decisão no final de cada linha. Pelos resultados obtidos os seis vetores de características foram suficientes para caracterizar esse ataque e foram usadas para treinar o classificador Naïve Bayes considerando a arquitetura mostrada na Figura 5.9.

Após esse levantamento realiza-se o treinamento utilizando o programa BCI. Esse programa lê todos os campos do arquivo de treinamento e calcula as funções de distribuição de probabilidade normal para cada vetor de característica e para cada classe. A saída desse programa é mostrada na Figura 5.10 que é a base de dados das classes usado pelo classificador.

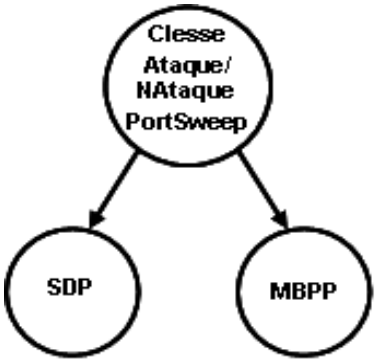


Figura 5.9: Arquitetura do classificador Naïve Bayes para o ataque Portsweep com os 2 vetores de características e as 2 classes Ataque e NAtaque.

5.7 Classificação do Sistema

Essa é a última etapa do sistema. Após o treinamento ter sido realizado e os arquivos de configuração das classes terem sido criados pode-se iniciar a fase de classificação dos ataques usando a base de dados de tráfego de teste. Para isso utilizamos o programa de classificação do BCI (Bayes Classifier Induction) apresentado anteriormente. O detalhamento da classificação será visto no próximo capítulo, onde serão apresentados os resultados e as análises dos mesmos.

```
/*-----------------------------------------------
  domains
-----------------------------------------------
dom(SDP) = ZZ [1, 1031];
dom(MBPP) = ZZ [1, 376065];
dom(Decisao) = { NAtaque, Ataque };

/*-----------------------------------------------
  naive Bayes classifier
-----------------------------------------------
nbc(Decisao) = {
  params = 0, maxllh;
  prob(Decisao) = {
    NAtaque: 151893 (99.5%),
    Ataque : 767 (0.5%) };
  prob(SDP|Decisao) = {
    NAtaque: N(1, 0) [151893],
    Ataque : N(14.3638, 1964.12) [767] };
  prob(MBPP|Decisao) = {
    NAtaque: N(56.2387, 1.15124e+06) [151893],
    Ataque : N(1.91917, 0.0743002) [767] };
};

/*-----------------------------------------------
  number of attributes: 3
  number of tuples    : 152660
-----------------------------------------------
```

Figura 5.10: Resultado do treinamento Naïve Bayes para o ataque Portsweep.

Capítulo 6

Análise dos Resultados

6.1 Introdução

Para todos os ataques a classificação é feita executando-se um programa similar ao programa de treinamento só que ao invés de gerar o arquivo de dados de treinamento para cada conjunto de vetor de característica é feita a sua classificação. O resultado do classificador é então comparado com a informação de presença ou não de ataque na base de dados de teste. As informações são contabilizadas e apresentadas na saída na forma de relatório seguindo as recomendações da avaliação de desempenho apresentada no Capítulo 5.

Neste capítulo serão apresentados os resultados da classificação para cada ataque com as suas devidas análises que servirão de insumo para a conclusão do trabalho.

6.2 Classificação do Ataque Guest

Na base de teste o ataque Guest está representado por 4 ataques com 67 registros no total, representando uma quantidade muito pequena em relação ao restante da base. As Figuras 6.1 a 6.4 mostram com detalhes o comportamento desses ataques na base de teste.

Podemos observar na Figura 6.2 que existe uma grande intercessão entre o ataque e o

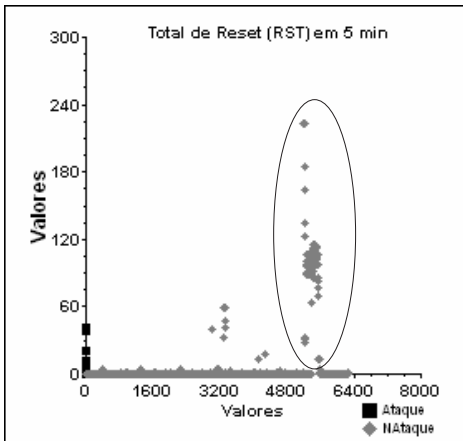


Figura 6.1: Comportamento do vetor de característica RST na base de teste.

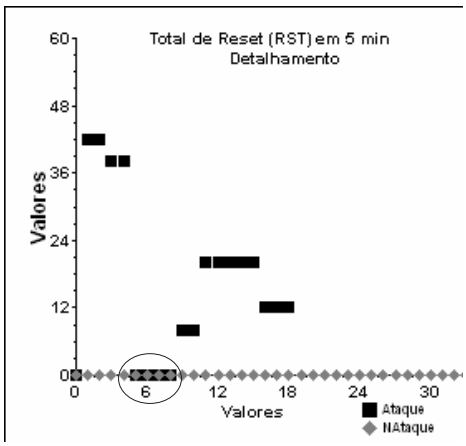


Figura 6.2: Comportamento do vetor de característica RST na base de teste com detalhamento do ataque.

tráfego normal (sinalizado com um círculo), que deve prejudicar a classificação. O comportamento marcado com um círculo na Figura 6.1 não parece ser normal, provavelmente é um outro tipo de ataque na mesma porta de serviço que o Guest utiliza.

O vetor de característica TxG esta bem comportado como podemos observar na Figura 6.4, com grande diferenciação entre o tráfego normal e o ataque. Na Figura 6.3, podemos observar o mesmo comportamento visto no gráfico anterior, confirmando a suspeita da existência de outro ataque.

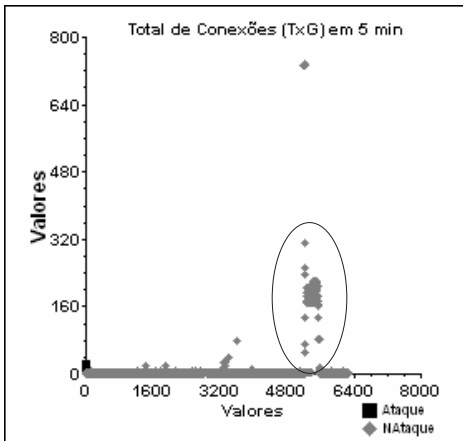


Figura 6.3: Comportamento do vetor de característica TxG na base de teste.

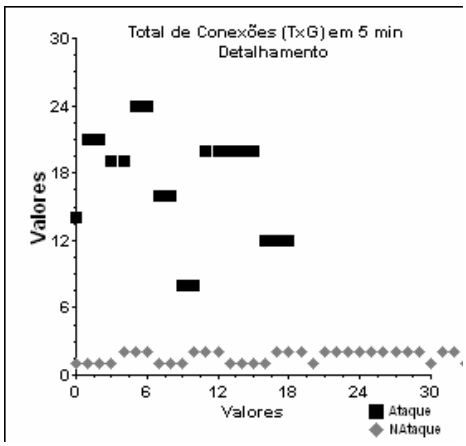


Figura 6.4: Comportamento do vetor de característica TxG na base de teste com detalhamento do ataque.

6.2.1 Resultados

A Tabela 6.1 apresenta o resultado da classificação do ataque Guest. O primeiro ponto importante a se observar é a quantidade total de ataques (TAt) que apresenta a quantidade de 8 ataques, enquanto que realmente só existem 4, como foi apresentado anteriormente. Essa diferença ocorre devido ao particionamento da base em intervalos de tempo independentes realizado pelo sistema. Então se um ataque tem início em um intervalo de tempo e se propaga para o próximo, o sistema poderá contá-lo como 2 ou mais ataques ao invés de 1.

Dos 8 ataques encontrados pelo sistema na base somente 2 foram detectados e estão

identificados pelo número de Real Positivo (RP), os outros 6 não foram detectados como evidência de ataque, ficando contabilizados como Falsos Negativos (FN). O sistema também detectou 9 ataques que não eram evidência do ataque Guest e foram identificados como Falsos Positivos (FP). Em uma análise mais detalhada desses eventos, evidenciou-se a presença de outros ataques similares ao Guest, como dito anteriormente. O número alto de Real Negativo (RN) indica que o classificador não está confundindo tráfego normal com ataques.

A Figura 6.5 oferece uma visão do cenário de classificação com as duas classes e os pontos para classificação em torno delas.

Em relação à quantidade de amostras analisadas o sistema teve um bom desempenho com 99,05% de taxa de acerto (PA) e 0,95% de taxa de erro (PE), apesar da taxa de acerto referencial (PAR) ter sido baixa, 25%.

Ataque Guest	
Índices	Valor
Total de Amostras Analisadas (TAA)	1.580
Total de Ataques (TAt = RP + FN)	8
% Referencial de Ataque (PRA=TAt/TAA)	0,51%
Total de Falso Positivo (FP)	9
Total de Falso Negativo (FN)	6
Total de Real Positivo (RP)	2
Total de Real Negativo (RN)	1.563
% de Acerto (PA=RP+RN/TAA)	99,05%
% de Erro (PE=FP+FN/TAA)	0,95%
% de Acerto Referencial (PAR=RP/TAt)	25%

Tabela 6.1: Resultado da classificação do ataque Guest.

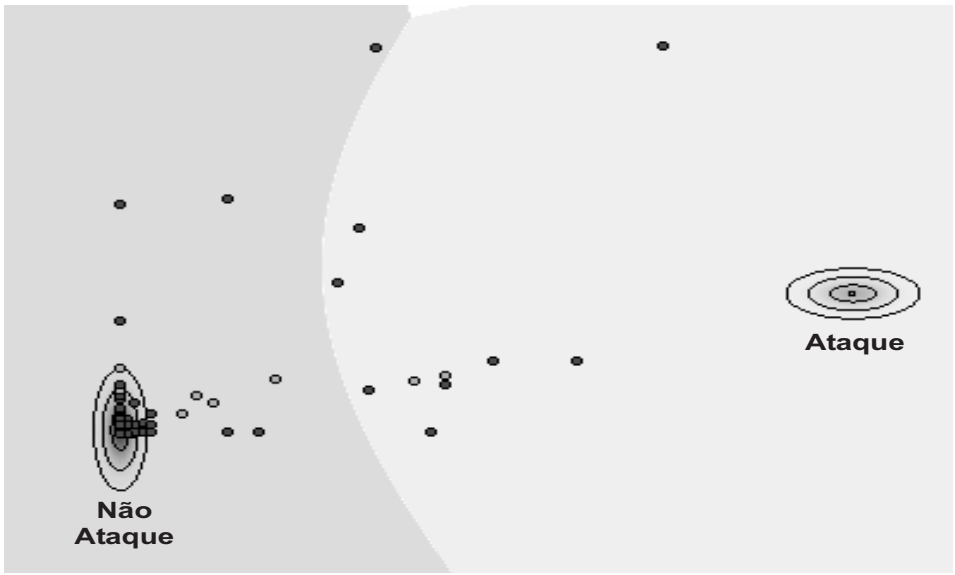


Figura 6.5: Representação gráfica dos pontos do conjunto de dados de teste em relação às classes.

6.3 Classificação do Ataque Neptune

Na base de teste o ataque Neptune está representado por 4 ataques com 72.075 registros no total, representando uma quantidade razoável de informação. As Figuras 6.6 a 6.17 mostram com detalhes o comportamento desses ataques na base de teste.

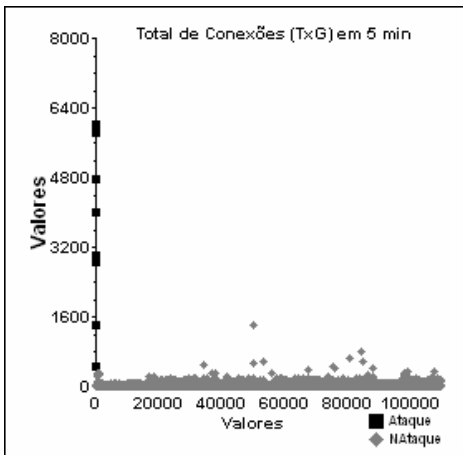


Figura 6.6: Comportamento do vetor de característica TxG em toda a base de teste.

Podemos observar pelas Figuras 6.6 a 6.11 que os vetores de características possuem comportamentos muito semelhantes e que existe uma grande diferenciação entre o ataque e o tráfego normal.

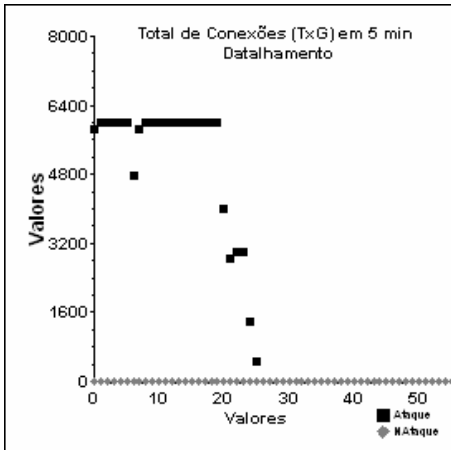


Figura 6.7: Comportamento do vetor de característica TxG em toda a base de teste com detalhamento do ataque.

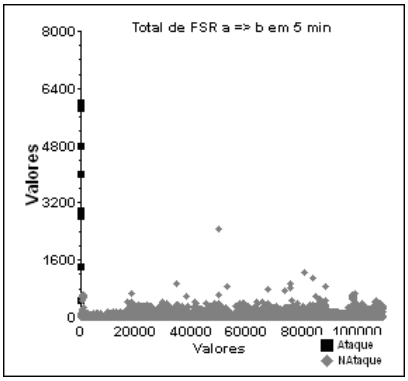


Figura 6.8: Comportamento do vetor de característica FSR_a2b em toda a base de teste.

Podemos observar nas Figuras 6.12 e 6.13 que existe uma grande diferenciação entre o ataque e o tráfego normal para esse vetor de característica, o qual se mantêm sempre em zero quando na presença de ataque.

Podemos observar nas Figuras 6.14 e 6.15 que existe uma grande diferenciação entre o ataque e o tráfego normal quando da presença do ataque, mas no comportamento geral eles se confundem.

Podemos observar nas Figuras 6.16 e 6.17 que existe uma grande diferenciação entre o tráfego normal com ataque que se mantêm sempre em zero.

Pelas características apresentadas em todos os gráficos acima fica bem evidenciado o comportamento do ataque se diferenciando do tráfego normal. É possível que o classifi-

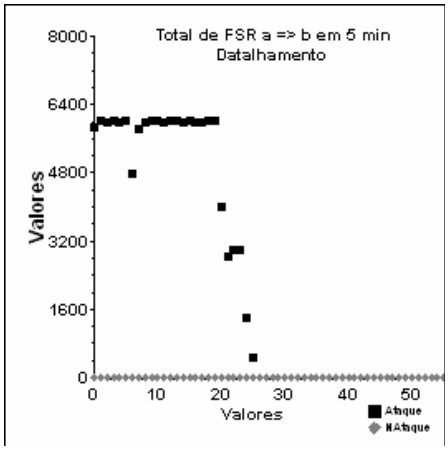


Figura 6.9: Comportamento do vetor de característica FSR_a2b em toda a base de teste com detalhamento do ataque.

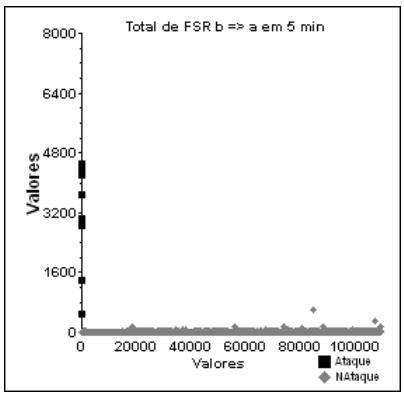


Figura 6.10: Comportamento do vetor de característica FSR_b2a em toda a base de teste.

cador se confunda com outros tipos de ataques, pois existem ataques com comportamento muito similar ao do Neptune presentes na base.

6.3.1 Resultados

A Tabela 6.2 apresenta o resultado da classificação do ataque Neptune. O primeiro ponto importante a se observar é a quantidade total de ataques que está marcando 26 e como foi apresentado anteriormente só existem 4. Essa diferença ocorre devido ao particionamento da base em intervalos de tempo independentes pelo sistema. Então se um ataque tem início em um intervalo de tempo e se propaga para o próximo, o sistema poderá contá-lo como 2 ou mais ataques ao invés de 1. Principalmente por esse ataque

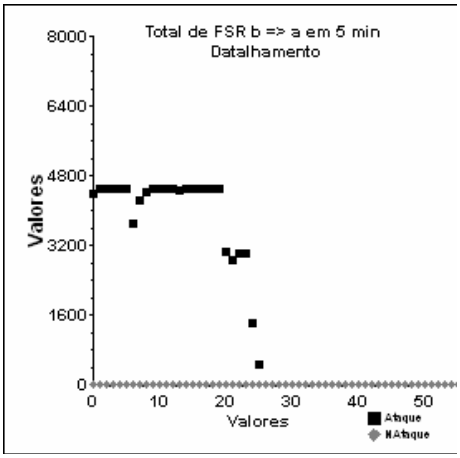


Figura 6.11: Comportamento do vetor de característica FSR_b2a em toda a base de teste com detalhamento do ataque.

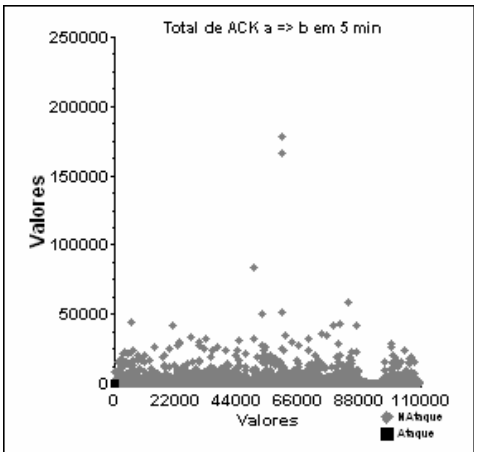


Figura 6.12: Comportamento do vetor de característica ACK_a2b em toda a base de teste.

ter uma característica de longa duração sua presença foi detectada em vários intervalos de tempo. Essa diferença apesar de ser grande não prejudica a análise, pois comparados a um sistema real podemos dizer que os ataques foram identificados, só que para o mesmo ataque alguns alarmes foram gerados a mais.

Dos 26 ataques encontrados pelo sistema na base todos foram detectados e estão identificados pelo percentual de Real Positivos (RP). O sistema também detectou 10 ataques que não eram evidência do ataque Neptune e foram identificados como Falso Positivo (FP). Em uma análise mais detalhada esses ataques foram evidenciados como outros ataques similares ao Neptune, como previsto anteriormente. O número alto de Real Negativo (RN) indica que o classificador não está confundindo tráfego normal com ataques.

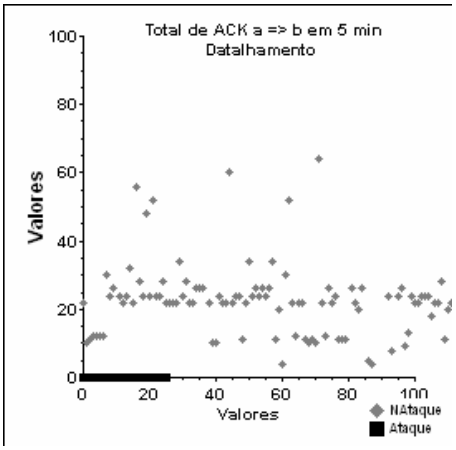


Figura 6.13: Comportamento do vetor de característica ACK_a2b em toda a base de teste com detalhamento do ataque.

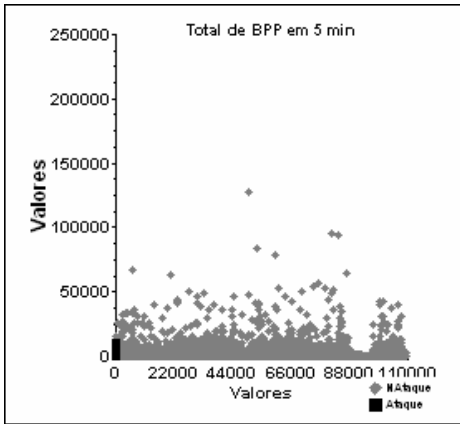


Figura 6.14: Comportamento do vetor de característica BPP em toda a base de teste.

A Figura 6.18 oferece uma visão do cenário de classificação com as duas classes e os pontos para classificação em torno delas.

Em relação à quantidade de amostras analisadas o sistema teve um excelente desempenho com 99,9% de taxa de acerto (PA) e somente 0,01% de taxa de erro (PE), e com a taxa de acerto referencial (PAR) de 100%.

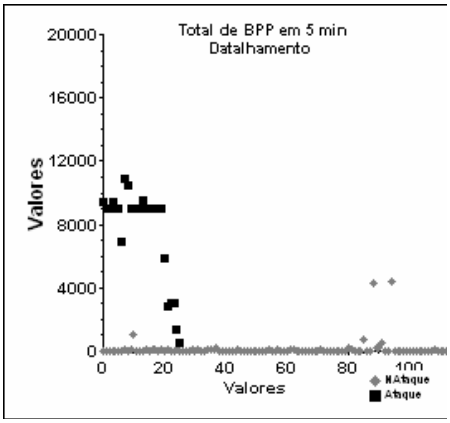


Figura 6.15: Comportamento do vetor de característica BPP em toda a base de teste com detalhamento do ataque.

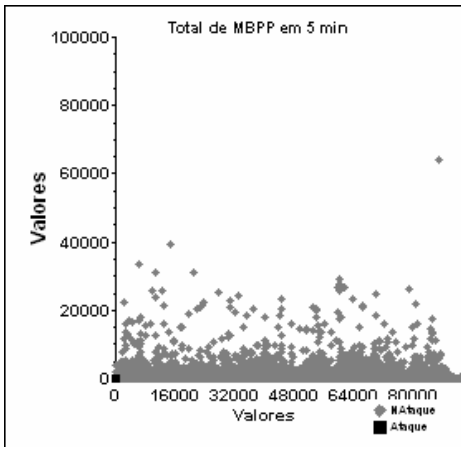


Figura 6.16: Comportamento do vetor de característica MBPP em toda a base de teste.

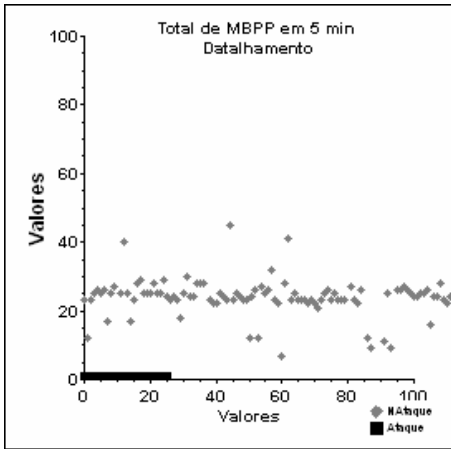


Figura 6.17: Comportamento do vetor de característica MBPP em toda a base de teste com detalhamento do ataque.

Ataque Neptune	
Índices	Valor
Total de Amostras Analisadas (TAA)	10.163
Total de Ataques (TAt = RP = FN)	26
% Referencial de Ataque (PRA=TAt/TAA)	0,26%
Total de Falso Positivo (FP)	10
Total de Falso Negativo (FN)	0
Total de Real Positivo (RP)	26
Total de Real Negativo (RN)	10.127
% de Acerto (PA=RP+RN/TAA)	99,90%
% de Erro (PE=FP+FN/TAA)	0,098%
% de Acerto Referencial (PAR=RP/TAt)	100%

Tabela 6.2: Resultado da classificação do ataque Neptune.

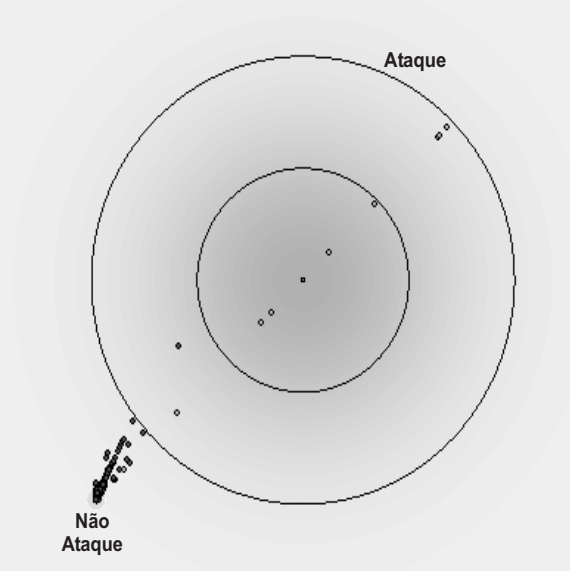


Figura 6.18: Representação gráfica dos pontos do conjunto de dados de teste em relação às classes.

6.4 Classificação do Ataque Portsweep

Na base de teste o ataque Portsweep está representado por 13 ataques com 262 registros no total, representando uma quantidade baixa de informação. As Figuras 6.19 a 6.24 mostram com detalhes o comportamento desses ataques na base de teste.

Pelas características apresentadas nos gráficos fica bem evidenciado o comportamento do ataque se misturando com o tráfego normal. É possível que o classificador gere um número expressivo de Falsos Positivos, pois com certeza ele irá se confundir com o tráfego normal e com outros ataques, pois existem tráfegos auto-similares ao ataque Portsweep.

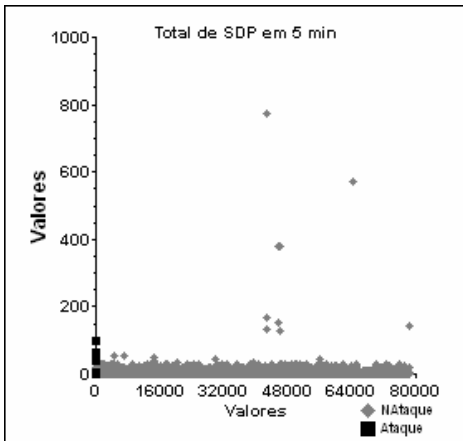


Figura 6.19: Comportamento do vetor de característica SDP em toda a base de teste.

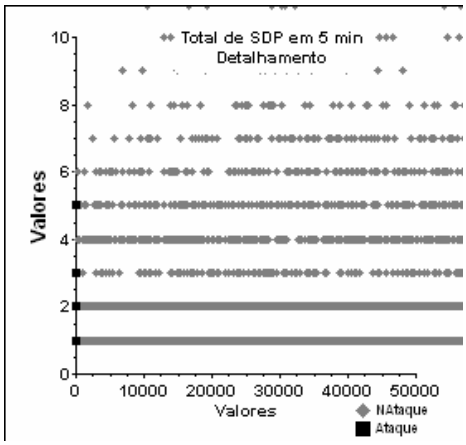


Figura 6.20: Comportamento do vetor de característica SDP em toda a base de teste com detalhamento do ataque.

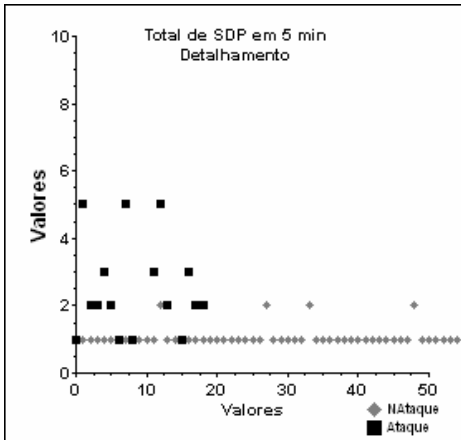


Figura 6.21: Comportamento do vetor de característica SDP em toda a base de teste com detalhamento do ataque.

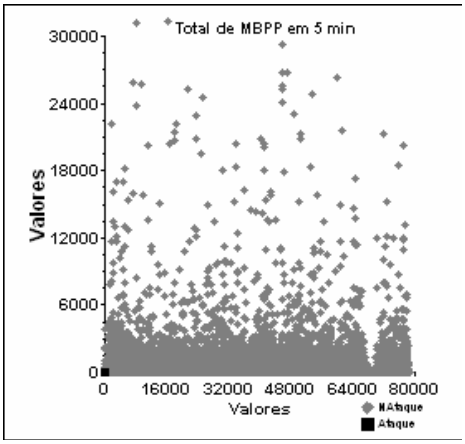


Figura 6.22: Comportamento do vetor de característica MBPP em toda a base de teste.

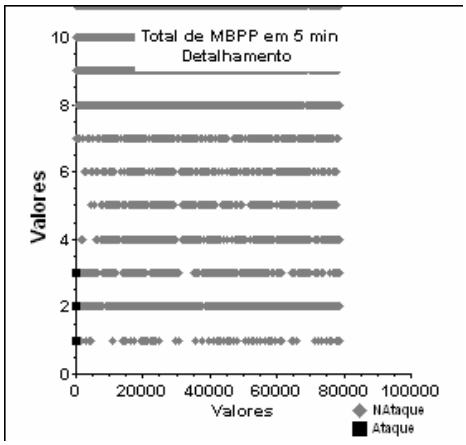


Figura 6.23: Comportamento do vetor de característica MBPP em toda a base de teste com detalhamento do ataque.

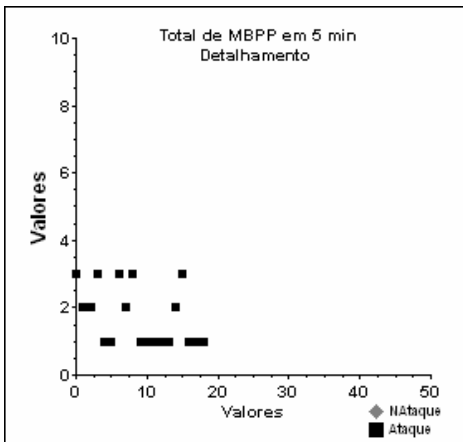


Figura 6.24: Comportamento do vetor de característica MBPP em toda a base de teste com detalhamento do ataque.

6.4.1 Resultados

A Tabela 6.3 apresenta o resultado da classificação do ataque Portsweep. O primeiro ponto importante a se observar é a quantidade total de ataques (TAt) que apresenta a quantidade de 19 ataques, enquanto que realmente só existem 13, como foi apresentado anteriormente. Essa diferença ocorre devido ao particionamento da base em intervalos de tempo independentes realizado pelo sistema. Então se um ataque tem início em um intervalo de tempo e se propaga para o próximo, o sistema poderá contá-lo como 2 ou mais ataques ao invés de 1

Ataque Portsweep	
Índices	Valor
Total de Amostras Analisadas (TAA)	2.470
Total de Ataques (TAt = RP + FN)	19
% Referencial de Ataque (PRA=TAt/TAA)	0,77%
Total de Falso Positivo (FP)	1.109
Total de Falso Negativo (FN)	1
Total de Real Positivo (RP)	18
Total de Real Negativo (RN)	1.342
% de Acerto (PA=RP+RN/TAA)	55,06%
% de Erro (PE=FP+FN/TAA)	44,94%
% de Acerto Referencial (PAR=RP/TAt)	94,74%

Tabela 6.3: Resultado da classificação do ataque Portsweep.

Dos 19 ataques encontrados pelo sistema na base 18 foram detectados e estão identificados pelo percentual de Real Positivos (RP). O sistema deixou de detectar 1 ataque que era uma evidência real do ataque Portsweep sendo identificados como Falso Negativo (FN). O alarmante foi a quantidade de Falsos Positivos (FP), contabilizados em 1109. Esse comportamento era esperado conforme foi explicado anteriormente.

Em relação à quantidade de amostras analisadas o sistema teve um desempenho mediano com 55,06% de taxa de acerto (TA) e 44,94% de taxa de erro (TE) apesar da taxa de acerto referencial (PAR) ter sido de 94,74%.

Capítulo 7

Conclusão e Trabalhos Futuros

7.1 Conclusão

Pelos resultados obtidos no Capítulo 6 e pela experiência adquirida durante o trabalho de aquisição dos dados, levantamento dos vetores de característica e os testes de classificação o classificador Naïve Bayes mostrou-se um excelente modelo para realizar a tarefa de classificação aplicada à detecção da intrusão. Sua simplicidade de implementação e a rapidez de processamento foram fatores relevantes no processo.

Podemos observar pelos números finais que para alguns tipos de ataque o comportamento do classificador não foi o esperado. Alguns fatores específicos e outros gerais influenciaram para esses resultados e serão detalhados a seguir.

Os problemas mencionados no Capítulo 4 sobre as falhas no desenvolvimento do DARPA 98 e DARPA 99 prejudicaram o resultado do trabalho, principalmente os problemas de marcação dos ataques que nem sempre estavam bem correlacionados com os dados dos tráfegos, o que possivelmente nos levou a inserir erros na base de dados de treinamento e teste como: a não marcação de tráfegos com evidência de ataques e marcação errônea de tráfegos normais como ataque. A falta de informações sobre como foram realizados os ataques prejudicou a análise do comportamento dos mesmos e conseqüentemente a definição dos vetores de características. Outros fatores mencionados no capítulo 4 podem ter influenciado nos resultados, mas sua extensão não pode ser mensurada, pois

é intrínseco ao modelo de teste realizado pelo Laboratório Lincoln do MIT.

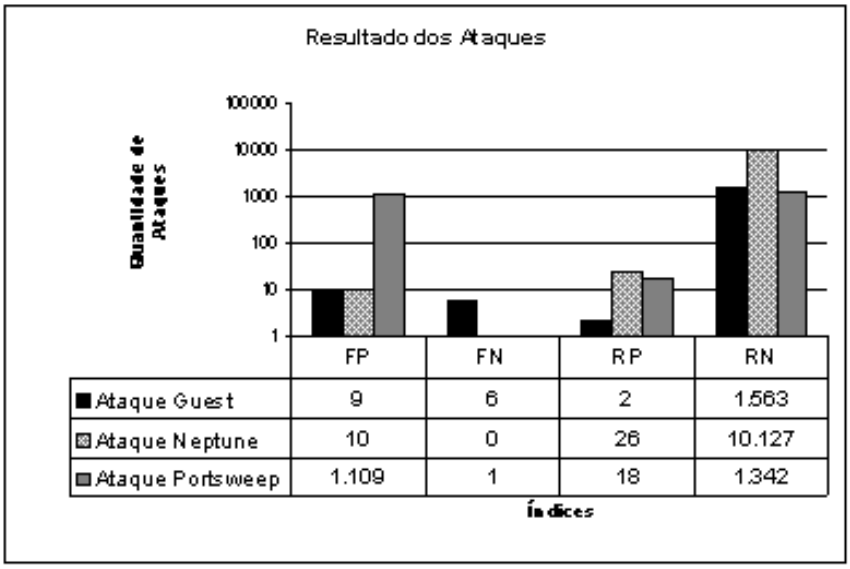
Foram observadas algumas questões sobre os dados de treinamento que influenciaram nos resultados de alguns ataques, mas que devem ser consideradas para todos, como a análise prévia dos dados estabelecendo-se limites superiores e/ou inferiores para que os mesmos não prejudiquem o resultado final como um todo. Um exemplo deste problema pode ser observado pela Figura 4.21 dos dados de treinamento do vetor de característica MBPP do ataque Portsweep que possui um ponto entre 350.000 e 400.000, enquanto que a maioria dos dados tem seu comportamento entre 0 e 5.000. A mesma evidência pode ser observada no vetor de característica ACK_a2b e BPP do ataque Neptune (Figuras 4.11 e 4.13).

Como o classificador Naïve Bayes necessita de treinamento supervisionado é de suma importância que existam dados de treinamento suficientes para que as funções de densidade probabilísticas possam cobrir o espaço amostral de modo a refletir os comportamentos possíveis de cada ataque. Pode ser observado pelos números da tabela 4.8 que os ataques Guest e Portsweep não possuem dados de treinamento suficientes para um bom aprendizado, refletindo diretamente sobre os resultados apresentados no Capítulo 6.

O resultado do ataque Guest (Figura 7.1) não foi satisfatório, pois obteve-se um percentual de acerto em relação à indicação real de ataque de 25%, ou seja, dos 8 ataques somente 2 foram detectados. Devido à quantidade de acerto nos registros que não têm evidência de ataque (Real Negativo), ou seja, tráfego normal, o percentual de acerto geral foi de 99,05%, considerado excelente. A quantidade de 9 falsos positivos (FP) é muito boa e indica que o classificador não confundiu o tráfego normal com evidências de ataques. A detecção deste tipo de ataque foi prejudicado pela falta de dados de treinamento, que se tentou compensar com a geração de dados sintetizados, mas como estes não refletem todas as variações de comportamento do ataque, a melhora não foi significativa. A falta de informação sobre os ataques nos dados de teste não permitiu que uma análise mais detalhada dos falsos negativos (FN) pudesse ser feita. Uma validação dos vetores de características na presença de uma quantidade maior de dados de treinamento é aconselhada para melhorar o desempenho do classificador nesse tipo de ataque.

A classificação do ataque Neptune foi excelente, 100% de acerto real, ou seja, todos

os 26 ataques indicados na base de teste foram detectados. Dos 10.127 registros de tráfego normal somente 10 foram indicados como evidência falsa de ataque Neptune (Falso Negativo). Esses números comprovam que uma boa escolha dos vetores de características em conjunto com uma quantidade satisfatória do conjunto de dados de treinamento é fator determinante para uma boa classificação. Os resultados podem ser observados na Figura 7.1.



	FP	FN	RP	RN
Ataque Guest	9	6	2	1.563
Ataque Neptune	10	0	26	10.127
Ataque Portsweep	1.109	1	18	1.342

Figura 7.1: Resultado final dos ataques

O resultado da detecção do ataque Portsweep não foi satisfatório (Figura 7.1), apesar do excelente resultado de 94,74% de acerto referência, ou seja, dos 19 ataques da base de teste 18 foram detectados. O mau resultado é referenciado à grande quantidade de erros de classificação, dos 1.343 registros de tráfego normal 1109 foram identificados com evidência falsa de ataque, ou seja, quase 83% de erro. É importante ressaltar que dos 1.109 registros de tráfego (FP) todos eram realmente normais e não evidências de outros ataques. Esse problema se deve a grande intercessão das informações de tráfego normal com evidência de ataque nos vetores de características. Essas evidências estão sinalizadas nos referidos gráficos dos vetores de características dos dados de treinamento (Figuras 4.18 a 4.23). Neles podemos observar que houve a influência de uma informação espúria no vetor de característica MBPP que fez com que a função de distribuição de probabilidade (FDP) ficasse com o valor de variância muito fora do real. A FDP do vetor de característica SDP para a classe não ataque ficou com média 1 e variância 0, definindo

uma decisão abrupta, o que não é aconselhável para um classificador. Uma tentativa para melhorar o classificador foi feita modificando-se o programa de classificação para não analisar tráfegos nas portas utilizadas com oferta de serviços na rede como: HTTP (80), SMTP (25), FTP (21 e 22), SSH (22). Essa alteração na filosofia de análise fez com que o número de falsos positivos diminuísse para praticamente 12% contra os 83% anteriores, mas em compensação o número de ataques detectados diminuiu quase 50%. Como solução final para o problema essa alternativa não se mostrou eficaz.

A solução para diminuir a quantidade de falsos positivos do ataque Portsweep é a melhoria dos vetores de características, ou com a substituição de alguns deles ou com a inclusão de outros que possam separar claramente no espaço amostral a evidência de ataque do tráfego normal.

Em paralelo a esse trabalho, outra tese de mestrado [51] com defesa prevista para maio de 2005 com mesmo orientador, estava realizando os mesmos testes sobre os mesmos ataques referenciando a mesma base de dados de treinamento e de teste utilizando lógica neuro-difusa (Neuro-Fuzzy). Para alguns ataques os vetores de características usados foram os mesmos, mas para outros o mesmo conjunto de vetores não se mostrou eficiente tendo que ser ajustado conforme a necessidade.

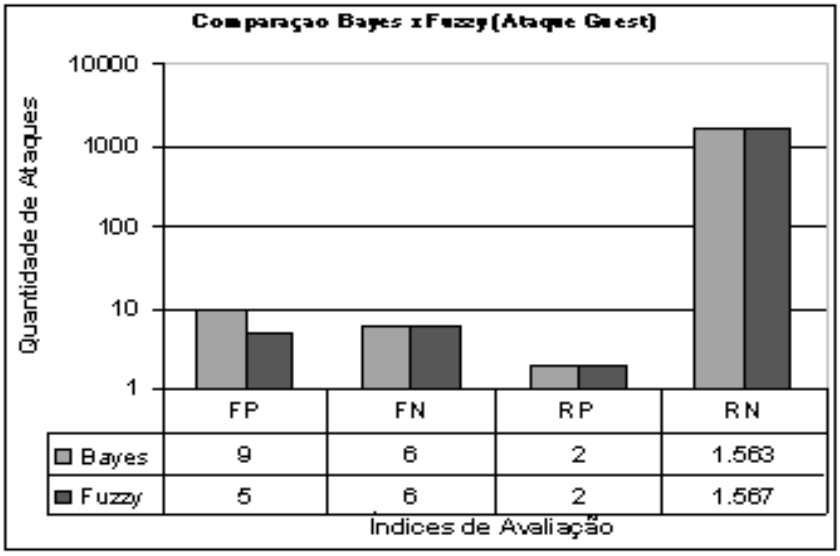


Figura 7.2: Comparação entre o classificador Bayesiana e Neuro-Fuzzy para o ataque Guest.

Como pode ser observado na Figura 7.2 do ataque Guest o classificador Bayesiano assinalou mais falsos positivos que o Neuro-Fuzzy, 9 contra 5 respectivamente, mostrando que com poucos dados de treinamento as curvas de pertinências do Neuro-Fuzzy são mais sensíveis do que as funções de densidade probabilísticas do Bayes. Nos demais índices os dois foram compatíveis.

No ataque Neptune o classificador Bayesiana se mostrou superior ao Neuro-Fuzzy, como podemos observar na Figura 7.3 através dos índices Falso Positivo (FP) onde do total de 26 ataques o Bayes obteve 26 acertos (100%) contra 23 (88,46%) do Neuro-Fuzzy e Real Positivo (RP) onde o Bayes se confundiu em 10 casos, 0,1% em relação à quantidade de Real Negativo (RN), e o Neuro-Fuzzy em 541 casos, representando 5,6% sobre a mesma referência.

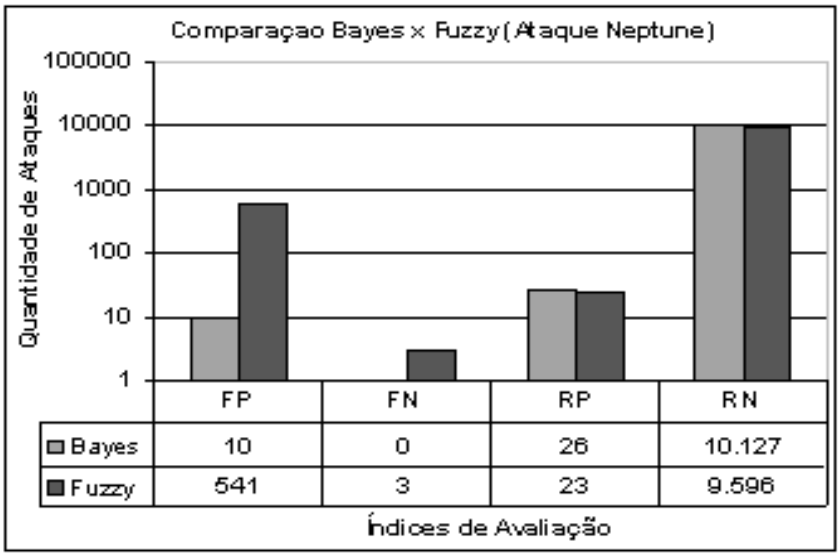


Figura 7.3: Comparação entre o classificador Bayesiana e Neuro-Fuzzy para o ataque Neptune.

Em relação ao ataque Portsweep os dois modelos obtiveram resultados muito semelhantes (Figura 7.4). Os problemas que afetaram o classificador Bayesiano também afetaram o Neuro-Fuzzy, que para chegar a esse resultado precisou dividir os valores dos vetores de características em 4 faixas e realizar a análise em cada uma dessas 4 faixas.

Olhando os resultados dos dois classificadores podemos concluir que cada um possui suas particularidades, o Bayes mais simples e o Neuro-Fuzzy mais robusto, mas chegaram

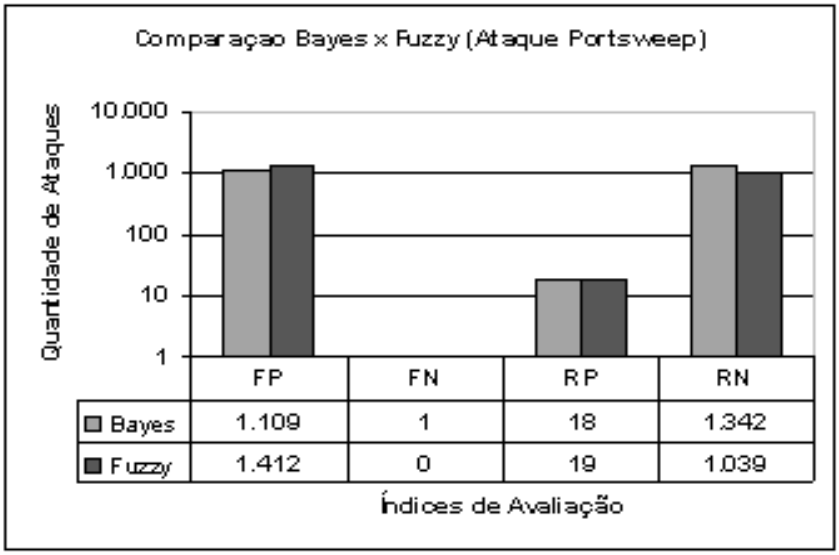


Figura 7.4: Comparação entre o classificador Bayesiana e Neuro-Fuzzy para o ataque Portsweep.

praticamente aos mesmos resultados. Então concluímos que os dois métodos são muito interessantes e podem ser aplicados na finalidade em questão.

Pelo exposto neste trabalho o classificador Bayesiano se mostrou um excelente método para ser utilizado na detecção da intrusão. A sua utilização comercial tem sido percebida em alguns softwares de anti-spam. Existem outros trabalhos de pesquisa sendo realizados para utilizá-lo como uma plataforma genérica de tomada de decisão como o Hugin [52]. O ponto fraco desse modelo, como em todos os modelos que utilizam treinamento supervisionado, é o alto grau de dependência do especialista para a determinação dos vetores de características para cada tipo de ataque novo que surge. A utilização de PCA (Principal Component Analysis) e clusterização têm sido estudadas para resolver esse problema.

7.2 Trabalhos Futuros

A conclusão é que muito trabalho ainda pode ser feito utilizando-se esse método na detecção da intrusão como, por exemplo: o mapeamento de outros ataques, a melhoria do treinamento, testar outros algoritmos de inferências mais eficientes, otimizar a escolha

dos atributos das classes e fazer correlação de eventos de rede com os de host.

Como mencionado anteriormente a fragilidade dos sistemas inteligentes está no treinamento, principalmente quando este é supervisionado. Uma das saídas para esse problema é a utilização de redes Bayesianas em conjunto com clusterização Fuzzy [53], permitindo que o espaço amostral do conjunto de treinamento seja dividido em subconjuntos (cluster) que irão definir as melhores classes a serem utilizadas no processo.

FRIEDMAN, GEIGER e GILDSZMIDT [54] propuseram um algoritmo para melhorar a precisão das redes Bayesianas e pode ser aplicado ao Naïve Bayes. A proposta transforma o modelo de rede estrela do Naïve Bayes em uma rede com conexões somente entre os atributos, não se estendendo ao nó pai. Esse modelo cria uma interdependência entre os atributos dando maior sensibilidade ao classificador.

Outra proposta é a utilização de redes Bayesianas dinâmicas (DBN – Dynamic Bayesian Network) apresentada por [55] onde a ocorrência de eventos no tempo pode ser capturada utilizando-se uma mistura dos modelos de Markov (HMM – Hidden Markov Model) e sistemas lineares dinâmicos (LDS - Linear Dynamical Systems).

A técnica de Análise de Componentes Principais ou PCA (Principal Component Analysis) [56] tem sido alvo de estudo para a otimização da escolha dos vetores de características para sistemas inteligentes.

Referências Bibliográficas

[1] NCSC. *Trusted Computer System Evaluation Criteria*, red book ed. Department of Defence, 1985.

[2] NCSC. *Trusted Computer System Evaluation Criteria*, orange book ed. Department of Defence, 1985.

[3] MAKHERJEE, B., HEBERLEIN, L. T., E LEVITT, K. N. Network intrusion detection. *IEEE Network 8* (1994), 26–41.

[4] VOYDOCK, V., E KENT, S. Security in high-level network protocols. *IEEE Commum. Mag. 23*, 7 (july de 1985), 12–24.

[5] BACE, R. G. *Intrusion Detection*, 1 ed. Macmillan Technical Publishing, Indianapolis, 2000.

[6] DENNING, D. E. An intrusion detection model. *IEEE Transaction on Software Eng 13* (1987), 222–232.

[7] SUNDARAM, A. An introduction to intrusion detection. "http://www.acm.org/crossroads/xrds2-4/intrus.html", 1996. ACM Crossroads.

[8] FRANK, J. Artificial intelligence and intrusion detection: Current and future directions. In *Proceedings of the 17th National Computer Security Conference* (Baltimore, MD, 1994).

[9] JACKSON, P. *Introduction to expert systems*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[10] GARVEY, T. D., E LUNT, T. F. Model-based intrusion detection. In *Proceedings of the 14:th National Computer Security Conference* (Baltimore, MD, USA, outubro de 1991), NIST, National Institute of Standards and Technology/National Computer Security Center, pp. 372–385.

[11] ILGUN, K. USTAT: A real-time intrusion detection system for UNIX. In *Proceedings of the 1993 IEEE Symposium on Research in Security and Privacy* (Oakland, CA, 1993), pp. 16–28.

[12] PORRAS, P., E KEMMERER, R. Penetration state transition analysis – Arule-based intrusion detection approach. In *Proceedings of the Eighth Annual Computer Security Applications Conference* (San Antonio, TX, nov de 1992), IEEE, IEEE Computer Society Press, pp. 220–229.

[13] KUMAR, S., E SPAFFORD, E. H. A Pattern Matching Model for Misuse Intrusion Detection. In *Proceedings of the 17th National Computer Security Conference* (1994), pp. 11–21.

[14] TENG, H. S., CHEN, K., E LU, S. C.-Y. Security audit trail analysis using inductively generated predictive rules. In *Proceedings of the 6th Conference on Artificial Intelligence Applications* (março de 1990), IEEE, IEEE Service Center, Piscataway, NJ, pp. 24–29.

[15] DEBAR, H., BECKER, M., E SIBONI, D. A neural network component for an intrusion detection system. In *Proceedings of the 1992 IEEE Computer Society Symposium on Research in Security and Privacy* (Oakland, CA, maio de 1992), pp. 240–250.

[16] MÉ, L. Gassata, a genetic algorithm as an alternative tool for security audit trail analysis. In *Proceedings of th 1th international workshop on the recent advances in intrusion detection held in Louvain-la-Neuve* (Belgiun, 1998), pp. 14–16.

[17] LUO, J. Integrating fuzzy logic with data mining methods for intrusion detection. Tese de Mestrado, Mississippi State University, August de 1999.

[18] MAIA, R., SOARES, A. A., E LEÃO, J. Utilização da lógica difusa na detecção da intrusão. *Congresso de Ciências da Computação e Sistemas da Informação da Região Sul* (2004).

[19] CROSBIE, M., E SPAFFORD, E. H. Defending a computer system using autonomous agents. In *Proc. 18th NIST-NCSC National Information Systems Security Conference* (1995), pp. 549–558.

[20] LEE, W., STOLFO, S. J., E MOK, K. W. Mining audit data to build intrusion detection models. In *Proceedings of 4th internacional conference on knowledge discovery and data mining held in New York* (New York, August de 1998), edited by Rakesh Agrawal e Poul Stolorz, Ed., NY:AAAI Press, pp. 66–72.

[21] SEBYALA, A. A., OLUKEMI, T., E SACKS, L. Active platform security through intrusion detection using naive bayesian network for anomaly detection. In *Proceedings of the London Communications Symposium 2002* (jul de 2002).

[22] KRUEGEL, C., MUTZ, D., ROBERTSON, W., E VALEUR, F. Bayesian event classification for intrusion detection. In *ACSAC '03: Proceedings of the 19th Annual Computer Security Applications Conference* (2003), IEEE Computer Society, p. 14.

[23] PUTTINI, R. S., MARRAKCHI, Z., E MÉ, L. A Bayesian Classification Model for Real-Time Intrusion Detection. In *AIP Conf. Proc. 659: Bayesian Inference and Maximum Entropy Methods in Science and Engineering* (março de 2003), pp. 150–162.

[24] VALDES, A., E SKINNER, K. Adaptive, model-based monitoring for cyber attack detection. In *RAID '00: Proceedings of the Third International Workshop on Recent Advances in Intrusion Detection* (2000), Springer-Verlag, pp. 80–92.

[25] DANIEL BARBARÁ, N. W., E JAJODIA, S. Detecting novel network intrusions using bayes estimators. In *the 1st SIAM International Conference on Data Mining* (April de 2001).

[26] AMOR, N. B., BENFERHAT, S., E ELOUEDI, Z. Naive bayes vs decision trees in intrusion detection systems. In *SAC '04: Proceedings of the 2004 ACM symposium on Applied computing* (2004), ACM Press, pp. 420–424.

[27] PEARL, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[28] BORGELT, C., E KRUSE, R. *Graphical Models - Methods for Data Analysis and Mining*. J. Wiley & Sons, Chichester, United Kingdom, 2002.

[29] GOOD, I. J. *The Estimation of Probabilities. An Essay on Modern Bayesian Methods*. MIT Press, Cambridge, MA, 1965.

[30] DUDA, R. O., E HART, P. E. *Pattern Classification and Scene Analysis*. John Wiley & Sons, 1973.

[31] LANGLEY, P., IBA, W., E THOMPSON, K. An analysis of bayesian classifiers. In *National Conference on Artificial Intelligence* (1992), pp. 223–228.

[32] LANGLEY, P., E SAGE, S. Induction of selective bayesian classifiers. pp. 399–406.

[33] DOUGHERTY, J., KOHAVI, R., E SAHAMI, M. Supervised and unsupervised discretization of continuous features. In *ICML* (1995), pp. 194–202.

[34] FRIEDMAN, N., E GOLDSZMIDT, M. Building classifiers using bayesian networks. In *AAAI/IAAI, Vol. 2* (1996), pp. 1277–1284.

[35] BORGELT, C., E GEBHARDT, J. A naive Bayes style possibilistic classifier. In *Proc. 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT'99, Aachen, Germany)* (Aachen, Germany, 1999), Verlag Mainz.

[36] BLAKE, C., E MERZ, C. UCI repository of machine learning databases, 1998.

[37] LIPPMANN, R., FRIED, D., GRAF, I., HAINES, J., KENDALL, K., MCCLUNG, D., WEBER, D., WEBSTER, S., WYSCHOGROD, D., CUNNINGHAM, R., E ZISSMAN, M. Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *Proceedings of the DARPA Information Survivabi-*

lity Conference and Exposition (Los Alamitos, CA, 2000), IEEE Computer Society Press.

[38] LIPPMANN, R., HAINES, J. W., FRIED, D. J., KORBA, J., E DAS, K. Analysis and results of the 1999 darpa off-line intrusion detection evaluation. In *Recent Advances in Intrusion Detection* (2000), pp. 162–182.

[39] Darpa intrusion detection evaluation. "http://www.ll.mit.edu/IST/ideval/index.html", 2001.

[40] MCHUGH, J. The 1998 lincoln laboratory ids evaluation: A critique. In *Recent Advances in Intrusion Detection* (2000), pp. 145–161.

[41] DIHUA, L., HONGZHI, W., E XIUMEI, W. Data mining for intrusion detection. *The 2002 Conference Technology and Management* (2002), 41–50.

[42] BLOEDORN, E., CHRISTIANSEN, A. D., HILL, W., SKORUPKA, C., TALBOT, L. M., E TIVEL, J. Data mining for network intrusion detection: How to get started, August de 2001. Technical Paper.

[43] POSTEL, J. Dod standard transmission control protocol, January de 1980.

[44] RAMADAS, M. Tcptrace manual. Internetworking Research Group, August de 2003.

[45] KENDALL, K. A database of computer attacks for the evaluation of intrusion detection systems. S.m. thesis, Department of Electrical Engineering and Computer Science, MIT, June de 1999.

[46] GARFINKEL, S., E SPAFFORD, G. *Practical UNIX & Internet Security*, 2 ed ed. O'Reilly & Associates, California, 1996.

[47] CERT. Cert advisory ca-96.21, September de 1996.

[48] CAI, W., E LI, L. Anomaly detection using tcp header information.

[49] STANIFORD, S., HOAGLAND, J. A., E MCALERNEY, J. M. Practical automated detection of stealthy portscans. *J. Comput. Secur. 10*, 1-2 (2002), 105–136.

[50] NORTHCUTT, S. *Network Intrusion Detection - An Analyst's Handbook*, 1th ed. New Riders, 1999.

[51] SOARES, A. A. Detecção da intrusão utilizando lógica fuzzy. Tese de mestrado a defender, Universidade Federal do Rio de Janeiro, 2005.

[52] Hugin expert. "http://www.hugin.com/", 2004.

[53] BORGELT, C., TIMM, H., E KRUSE, R. Probabilistic networks and fuzzy clustering as generalizations of naive bayes classifiers. In *Computational Intelligence in Theory and Practice*, B. Reusch e K.-H. Temme, Eds., Advances in Soft Computing. Physica-Verlag, Heidelberg, Germany, 2001, pp. 121–138.

[54] FRIEDMAN, N., GEIGER, D., E GOLDSZMIDT, M. Bayesian network classifiers. *Machine Learning 29*, 2-3 (1997), 131–163.

[55] MURPHY, K. P. An introduction to graphical models. Unpublished, May de 2001.

[56] JOLIFFE, I. *Principal Component Analysis*, 2nd edition ed. Springer Verlag, 2002.

Apêndice A

Anexo I

O NBIS foi desenvolvido utilizando um conjunto de programas auxiliares em Perl e o software indutor de classificação Bayesiana (BCI - Bayesian Classifier Induction) desenvolvidos por Cristian Borgelt do Departamento de Engenharia de Linguagem e Processamento do Conhecimento da Universidade de Magdeburg na Alemanha [35] e [28].

A.1 BCI

O BCI foi todo desenvolvido em linguagem "C" e seu código fonte disponibilizado. Ele é composto por 3 utilitários principais. O *dom*, o *bci* e o *bcx*. O *dom* é usado para gerar o arquivo de configuração do domínio das classes usado na faze de treinamento. O *bci* é o programa de treinamento e usa o arquivo de domínio como base junto com o arquivo de dados de treinamento. O *bcx* é o classificador propriamente dito, ele é responsável por calcular os valores das probabilidades usadas na classificação Naïve Bayesiana e gerar o arquivo de saída com o resultado da classificação.

A.1.1 Programa dom

O programa dom lê o arquivo de entrada de dados de treinamento e gera na saída o arquivo de configuração do domínio com o tipo de cada vetor de característica e seus valo-

res. Quando o vetor de característica é uma variável aleatória numérica a saída apresenta somente o maior (max) e menos valor (min). A Tabela A.1 apresenta a lista de opções do programa. Um exemplo de utilização do comando é:

$$sh > dom\ -a\ arq\_entrada\_dados\ arq\_conf\_domnio$$

usage: dom [options] [-d|-h hdrfile] tabfile domfile
determine attribute domains
version 1.9 (2003.08.16) (c) 1995-2003 Christian Borgelt
-s sort domains alphabetically (default: order of appearance)
-S sort domains numerically/alphabetically
-a automatic type determination (default: all symbolic)
-i do not print intervals for numeric attributes
-l# output line length (default: no limit)
-b/f/r# blank characters, field and record separators(default: "\t\r", "\t", "\n")
-u# unknown value characters (default: "?")
-n number of tuple occurrences in last field
-d use default header (field names = field numbers)
-h read table header (field names) from hdrfile
hdrfile file containing table header (field names)
tabfile table file to read (field names in first record)
domfile file to write domain descriptions to

Tabela A.1: Lista de opções do programa dom.

A.1.2 Programa bci

O programa bci lê o arquivo de entrada de dados de treinamento e o arquivo de configuração de domínio, gerado pelo programa dom, e gera na saída o arquivo de configuração de classe com as funções de densidade probabilística para cada vetor de característica e para cada classe definida. A Tabela A.2 apresenta a lista de opções do programa. Um exemplo de utilização do comando é:

$sh > bci\ -a\ arq\_conf\_domnio\ arq\_entrada\_dados\ arq\_sada\_\ conf\_classes$

usage: bci [options] domfile [-d|-h hdrfile] tabfile bcfile
naive and full Bayes classifier induction version 2.6 (2004.04.15)
-F induce a full Bayes classifier (default: naive Bayes)
-c# class field name (default: last field)
-w# balance class frequencies (weight tuples)
l: lower, b: boost, s: shift weights
-s# simplify classifier (naive Bayes only)
a: by adding, r: by removing attributes
-L# Laplace correction (default: 0)
-t distribute tuple weight for unknown values
-m use maximum likelihood estimate for the variance
-p print relative frequencies (in percent)
-l# output line length (default: no limit)
-b/f/r# blank characters, field and record separators (default: "\t\r", "\t", "\n")
-u# unknown value characters (default: "?")
-n number of tuple occurrences in last field
domfile file containing domain descriptions
-d use default table header (field names = field numbers)
-h read table header (field names) from hdrfile
hdrfile file containing table header (field names)
tabfile table file to read (field names in first record)
bcfile file to write Bayes classifier to

Tabela A.2: Lista de opções do programa bci.

A.1.3 Programa bcx

O programa bcx lê o arquivo de entrada de dados de teste e o arquivo de configuração de classes, gerado pelo programa bci, e gera na saída o arquivo de classificação. A Tabela A.3 apresenta a lista de opções do programa. Um exemplo de utilização do comando é:

$sh > bcx\ -a\ arq\_conf\_classes\ arq\_entrada\_dados\ arq\_sada\_classificao$

usage: bcx [options] bcfile [-d|-h hdrfile] tabfile [outfile]
naive and full Bayes classifier execution
version 2.8 (2004.04.15) (c) 1998-2004 Christian Borgelt
-c# classification field name (default: bc)
-p# confidence/probability field name (default: no confidence output)
-L# Laplace correction (default: as specified in classifier)
-v/V (do not) distribute tuple weight for unknown values
-m/M (do not) use maximum likelihood estimate for the variance
-a align fields (default: do not align)
-w do not write field names to the output file
-b/f/r# blank characters, field and record separators (default: "\t\r", "\t", "\n")
-u# unknown value characters (default: "?")
-n number of tuple occurrences in last field
bcfile file containing classifier description
-d use default table header (field names = field numbers)
-h read table header (field names) from hdrfile
hdrfile file containing table header (field names)
tabfile table file to read (field names in first record)
outfile file to write output table to (optional)

Tabela A.3: Lista de opções do programa bcx.

A.2 Programas Auxiliares

Para cada tipo de ataque criou-se um programa em Perl que acessava a base de dados e separava os vetores de características tanto para o treinamento como para a classificação. Esses programas também são responsáveis por gerar os relatórios de análise de resultados. Abaixo listamos o programa para o ataque Neptune que serviu de base para a construção do outros dois programas.

```
#!/usr/bin/perl

######################################################################

# Definição dos Pacotes Necessários ao Programa

######################################################################

use DBI;

######################################################################

# Abrindo e Conectando ao Banco

######################################################################

my $dbh = DBI->connect( 'DBI:mysql:SFIS:localhost',

'',

'',

) || die "Database connection not made: $DBI::errstr";

open ( TREINAMENTO, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/

DS/NeptuneDSetCompleto.dat");

open ( TPOS, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/

DS/TP_NeptuneDSet.dat");

open ( FPOS, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/

DS/FP_NeptuneDSet.dat");

open ( FNEG, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/

DS/FN_NeptuneDSet.dat");

open ( TNEG, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/

DS/TN_NeptuneDSet.dat");

open ( RESUMO, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/
```

```
DS/ResumoNeptuneDSet.dat");

system ( "clear");

print "############################################################ \n";

print "# \n";

print "# NBIS \n";

print "# Naive Bayesian Inference System \n";

print "# \n";

print "# Modulo Detector de Ataques Neptune (Data Set) \n";

print "# Versão 1.0.1 \n";

print "# \n";

print "############################################################ \n";

print "\n";

#####################################################################

# Limpando os Micro Fluxos

#####################################################################

$DeletaFluxosSQL = qq {Delete From Fluxos};

$DeletaFluxos = $dbh->prepare( $DeletaFluxosSQL );

$DeletaFluxos->execute();

#####################################################################

# Pega o Menor e o Maior Tempo

#####################################################################

$MinTime = 0;

$MaxTime = 0;
```

```
$AuxTime = 0;

$DeltaTime = 60;

$FP = 0;

$FN = 0;

$TP = 0;

$TN = 0;

$TotalMin = 0;

$TSessions = 0;

$FNAtaque = 0;

my $sql = qq {select Min(Start_Time) as MinTime, Max(Start_Time) as MaxTime
From DSBruto };

$sth = $dbh->prepare( $sql );

$sth->execute();

$sth->bind_columns( undef, \$MinTime, \$MaxTime );

$sth->fetch();

$AuxTime = $MinTime + $DeltaTime;

while( $MinTime < $MaxTime )

{

print "############################################################ \n";

print "Inicio da Analise para o Intervalo de Tempo [$MinTime,$AuxTime] \n";

print "\n";

#####################################################################

# Inserindo os MicroFluxos no Intervalo de Tempo Analisado
```

```
#####################################################################

$CriaFluxosSQL = qq { Insert Into Fluxos Select SessionID, Session, host_a,

host_b, port_a, port_b, Day, Month, Year, Hour, Minute, Second, Start_Time,

total_packets_a2b, total_packets_b2a, resets_sent_a2b, resets_sent_b2a,

ack_pkts_sent_a2b, ack_pkts_sent_b2a, pure_acks_sent_a2b,

pure_acks_sent_b2a, sack_pkts_sent_a2b, sack_pkts_sent_b2a,

dsack_pkts_sent_a2b, dsack_pkts_sent_b2a, max_sack_blks_ack_a2b,

max_sack_blks_ack_b2a, unique_bytes_sent_a2b, unique_bytes_sent_b2a,

actual_data_pkts_a2b, actual_data_pkts_b2a, actual_data_bytes_a2b,

actual_data_bytes_b2a, rexmt_data_pkts_a2b, rexmt_data_pkts_b2a,

rexmt_data_bytes_a2b, rexmt_data_bytes_b2a, zwnd_probe_pkts_a2b,

zwnd_probe_pkts_b2a, zwnd_probe_bytes_a2b, zwnd_probe_bytes_b2a,

outoforder_pkts_a2b, outoforder_pkts_b2a, pushed_data_pkts_a2b,

pushed_data_pkts_b2a, syn_pkts_sent_a2b, fin_pkts_sent_a2b,

syn_pkts_sent_b2a, fin_pkts_sent_b2a, req_1323_ws_ts_a2b,

req_1323_ws_ts_b2a, adv_wind_scale_a2b, adv_wind_scale_b2a,

req_sack_a2b, req_sack_b2a, sacks_sent_a2b, sacks_sent_b2a,

urgent_data_pkts_a2b, urgent_data_pkts_b2a, urgent_data_bytes_a2b,

urgent_data_bytes_b2a, mss_requested_a2b, mss_requested_b2a,

max_segm_size_a2b, max_segm_size_b2a, min_segm_size_a2b,

min_segm_size_b2a, avg_segm_size_a2b, avg_segm_size_b2a,

max_win_adv_a2b, max_win_adv_b2a, min_win_adv_a2b,

min_win_adv_b2a, zero_win_adv_a2b, zero_win_adv_b2a,
```

```
avg_win_adv_a2b, avg_win_adv_b2a, initial_window_bytes_a2b,
initial_window_bytes_b2a, initial_window_pkts_a2b, initial_window_pkts_b2a,
ttl_stream_length_a2b, ttl_stream_length_b2a, missed_data_a2b,
missed_data_b2a, truncated_data_a2b, truncated_data_b2a,
truncated_packets_a2b, truncated_packets_b2a, data_xmit_time_a2b,
data_xmit_time_b2a, idletime_max_a2b, idletime_max_b2a,
hardware_dups_a2b, hardware_dups_b2a, throughput_a2b, throughput_b2a,
Status, AttackName, AttackClass From DSBruto Where Start_Time >= $MinTime
And Start_Time < $AuxTime};
$CriaFluxos = $dbh->prepare( $CriaFluxosSQL );
$CriaFluxos->execute();
#####################################################################
# Conta o Numero de Sessoes
#####################################################################
$NSessions = 0;
$ChecaSessionsSQL = qq {select count(*) as NSessions From Fluxos Where
Start_Time >= $MinTime And Start_Time < $AuxTime };
$ChecaSessions = $dbh->prepare( $ChecaSessionsSQL );
$ChecaSessions->execute();
$ChecaSessions->bind_columns( undef, \$NSessions );
$ChecaSessions->fetch();
if ( $NSessions > 0 )
{
```

```
#####################################################################

# Verifica e existência de Portsweep no intervalo de tempo

#####################################################################

$TemAtaque1MinSQL = qq {select count(*) as TemAtaque1Min From Fluxos

Where Start_Time >= $MinTime And Start_Time < $AuxTime

And AttackName = "neptune"};

$TemAtaque1Min = $dbh->prepare( $TemAtaque1MinSQL );

$TemAtaque1Min->execute();

$TemAtaque1Min->bind_columns( undef, \$TemAtaque1Min );

$TemAtaque1Min->fetch();

#####################################################################

# Lista os SID Únicos Presentes em Fluxos

#####################################################################

$ListaSIDSQL = qq {select Distinct Host_a as SID, Host_b as DID

From Fluxos Where

Start_Time >= $MinTime And Start_Time < $AuxTime};

$ListaSID = $dbh->prepare( $ListaSIDSQL );

$ListaSID->execute();

$ListaSID->bind_columns( undef, \$SID, \$DID );

while ($ListaSID->fetch())

{

#####################################################################

# Calculo Trafego Global em cinco Minutos
```

```
#########################################################################

$TrafTotalSQL = qq {select count(*) As TxGlobal From Fluxos Where

Start_Time >= $MinTime And Start_Time < $AuxTime And

Host_a = \"$SID\"And Host_b = \"$DID\"};

$TrafTotal = $dbh->prepare( $TrafTotalSQL );

$TrafTotal->execute();

$TrafTotal->bind_columns( undef, \$TxGlobal );

$TrafTotal->fetch();

$ResetTotalSQL = qq { select sum(resets_sent_a2b + syn_pkts_sent_a2b +

fin_pkts_sent_a2b) as FSR_a, sum((resets_sent_a2b +

syn_pkts_sent_a2b +

fin_pkts_sent_a2b)/(total_packets_a2b + total_packets_b2a)) as FSR_a2b,

sum(resets_sent_b2a + syn_pkts_sent_b2a + fin_pkts_sent_b2a) as FSR_b,

sum((resets_sent_b2a + syn_pkts_sent_b2a + fin_pkts_sent_b2a)/

(total_packets_a2b + total_packets_b2a)) as FSR_b2a,

sum((total_packets_a2b + total_packets_b2a)) as BPP,

sum((ack_pkts_sent_a2b)) as ACK_a2b,

sum((ack_pkts_sent_b2a)) as ACK_b2a,

sum((ack_pkts_sent_a2b + ack_pkts_sent_b2a)) as ACK

From Fluxos Where Start_Time >= $MinTime And

Start_Time < $AuxTime And Host_a = \"$SID\"

And Host_b = \"$DID\"};

$ResetTotal = $dbh->prepare( $ResetTotalSQL );
```

```
$ResetTotal->execute();
$ResetTotal->bind_columns( undef, \$FSR_a, \$FSR_b, \$FSR_a2b, \$FSR_b2a,
\$BPP, \$ACK_a2b, \$ACK_b2a, \$ACK );
$ResetTotal->fetch();
$TemAtaqueSQL = qq {select count(*) as TemAtaque From Fluxos Where
Host_a = \"$SID\"And Host_b = \"$DID\"
And Start_Time >= $MinTime And
Start_Time < $AuxTime And AttackName = "neptune"};
$TemAtaque = $dbh->prepare( $TemAtaqueSQL );
$TemAtaque->execute();
$TemAtaque->bind_columns( undef, \$TemAtaque );
$TemAtaque->fetch();
$TotalLocalSQL = qq { select SessionID, Session, Hour, Minute, Second, Day,
Month, Year, Host_a, Port_a, Host_b, Port_b, Status, AttackName From Fluxos
Where Host_a = \"$SID\"And Host_b = \"$DID\"
And Start_Time >= $MinTime And Start_Time < $AuxTime};
$TotalLocal = $dbh->prepare( $TotalLocalSQL );
$TotalLocal->execute();
$TotalLocal->bind_columns( undef, \$SessionID, \$Session, \$Hour, \$Minute,
\$Second, \$Day, \$Month, \$Year, \$Host_a, \$Port_a, \$Host_b, \$Port_b,
\$Status, \$AttackName );
$TotalLocal->fetch();
if ( $TxGlobal > 0 )
```

```
{
$TSFR = $FSR_a2b + $FSR_b2a;
$TXGBPP = $BPP/$TxGlobal;
$MBPP = sprintf('%d',$TXGBPP);
open ( OUTPUT, ">/root/BIS/Dados/BIS_Ataque_Analise/Neptune/TS/Neptune.dat");
print "$TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP ($Status)
$AttackName $SessionID $Day\/$Month\/$Year $Hour\:$Minute\:$Second
$Host_a\:$Port_a $Host_b\:$Port_b\n";
if ( $TemAtaque > 0 )
{
print TREINAMENTO "$TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP Ataque\n";
print OUTPUT "$TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP Ataque\n";
$RealAtaqueName = "neptune"
}
else
{
print TREINAMENTO "$TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP NAtaque\n";
print OUTPUT "$TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP NAtaque\n";
$RealAtaqueName = "Nao Ataque"
}
close OUTPUT;
system ( "bcx -M -pConf -h
/root/BIS/Dados/BIS_Ataque_Analise/Neptune/TS/Neptune.hdr
```

```
/root/BIS/Dados/BIS_Ataque_Analise/Neptune/TS/Neptune.nbc

/root/BIS/Dados/BIS_Ataque_Analise/Neptune/TS/Neptune.dat

/root/BIS/Dados/BIS_Ataque_Analise/Neptune/TS/Neptune.out > /dev/null");

open ( INPUT, "/root/BIS/Dados/BIS_Ataque_Analise/Neptune/TS/Neptune.out");

$Linha = <INPUT>;

while ( $Linha = <INPUT> )

{

chomp ($Linha);

( $Lixo1, $Lixo2, $Lixo3, $Lixo4, $Lixo5, $Lixo6, $Decisão,

$BC, $Conf) = split ( / /, $Linha);

$Conf = $Conf * 100;

$ListaReg = qq { select Host_a, Host_b, Port_a, Port_b, Session, SessionID,

Day, Month, Year, Hour, Minute, Second, Status, AttackName From Fluxos

Where Host_a = \"$SID\"

And Host_b = \"$DID\"And Start_Time >= $MinTime

And Start_Time < $AuxTime};

$ListaReg = $dbh->prepare( $ListaReg );

$ListaReg->execute();

$ListaReg->bind_columns ( undef, \$Host_a, \$Host_b, \$Port_a, \$Port_b,

\$Session, \$SessionID, \$Day, \$Month,

\$Year, \$Hour, \$Minute, \$Second,

\$Status, \$AttackName);

if ( $RealAtaqueName eq "neptune"and $BC eq "Ataque")
```

```
{
$FlagAtaque = 1;
print "= [Neptune System [ $MinTime - $AuxTime ] ] ========== \n";
print TPOS "= [Neptune System [ $MinTime - $AuxTime ] ] ====== \n";
print TPOS "Darpa: $RealAtaqueName \n";
print "Darpa: $RealAtaqueName \n";
print TPOS "Bayes: $BC com Confiança de $Conf%\n";
print "Bayes: $BC com Confiança de $Conf%\n";
print TPOS "VC = $TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP \n";
print "VC = $TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP \n";
print TPOS "$TemAtaque Conexões Neptune\n";
print "$TemAtaque Conexões Neptune\n";
while ($ListaReg->fetch())
{
print "$AttackName ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
print TPOS "$AttackName Ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
}
print "============================================== \n";
print TPOS "======================================== \n";
}
elsif ( $RealAtaqueName ne "neptune"and $BC eq "Ataque")
```

```
{
$FlagAtaque = 1;
print "= [Neptune System [ $MinTime - $AuxTime ] ] ========== \n";
print FPOS "= [Neptune System [ $MinTime - $AuxTime ] ] ======= \n";
print FPOS "Darpa: $RealAtaqueName \n";
print "Darpa: $RealAtaqueName \n";
print FPOS "Bayes: $BC com Confiança de $Conf%\n";
print "Bayes: $BC com Confiança de $Conf%\n";
print FPOS "VC = $TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP \n";
print "VC = $TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP \n";
print FPOS "$TemAtaque Conexões Neptune \n";
print "$TemAtaque Conexões Neptune \n";
while ($ListaReg->fetch())
{
print "$AttackName ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
print FPOS "$AttackName Ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
}
print "============================================= \n";
print FPOS "========================================= \n";
}
elsif ( $RealAtaqueName eq "neptune"and $BC eq "NAtaque")
```

```
{
print "= [Neptune System [ $MinTime - $AuxTime ] ] ========== \n";
print FNEG "= [Neptune System [ $MinTime - $AuxTime ] ] ========== \n";
print FNEG "Darpa: $RealAtaqueName \n";
print "Darpa: $RealAtaqueName \n";
print FNEG "Bayes: $BC com Confiança de $Conf%\n";
print "Bayes: $BC com Confiança de $Conf%\n";
print FNEG "VC = $TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP \n";
print "Bayes: $BC com Confiança de $Conf%\n";
print FNEG "$TemAtaque Conexões Neptune \n";
print "$TemAtaque Conexões Neptune \n";
while ($ListaReg->fetch())
{
print "$AttackName ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
print FNEG "$AttackName Ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
}
print "========================================== \n";
print FNEG "========================================== \n";
}
else
{
```

```
if ( $RealAtaqueName eq "Nao Ataque"and $BC eq "NAtaque")
{
print "= [Neptune System [ $MinTime - $AuxTime ] ] ========== \n";
print TNEG "= [Neptune System [ $MinTime - $AuxTime ] ] ========== \n";
print TNEG "Darpa: $RealAtaqueName \n";
print TNEG "Bayes: $BC com Confiança de $Conf%\n";
print TNEG "VC = $TxGlobal $FSR_a $FSR_b $ACK_a2b $BPP $MBPP \n";
print TNEG "$TemAtaque Conexões Neptune \n";
while ($ListaReg->fetch())
{
print "$AttackName ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
print TNEG "$AttackName Ataque na Sessão:$SessionID em $Day\/$Month\/$Year
do $Host_a:$Port_a para $Host_b:$Port_b \n";
}
print "================================================= \n";
print TNEG "================================================= \n";
}
}
}
}
}
##########################################################################
```

```
# Contabilização dos ataques no intervalo de tempo
#####################################################################
print "TESTE: TemAtaque1Min=$TemAtaque1Min FlagAtaque=$FlagAtaque
TSessions=$TSessions \n";
$TSessions = $TSessions + 1;
if ( $TemAtaque1Min > 0 and $FlagAtaque > 0 )
{
$TP = $TP + 1;
print "TP=$TP \n";
}
elsif ( $TemAtaque1Min < 1 and $FlagAtaque > 0 )
{
$FP = $FP + 1;
print "FP=$FP \n";
}
elsif ( $TemAtaque1Min > 0 and $FlagAtaque < 1 )
{
$FN = $FN + 1;
print "FN=$FN \n";
}
elsif ( $TemAtaque1Min < 1 and $FlagAtaque < 1 )
{
$TN = $TN + 1;
```

```
print "TN=$TN \n";
}
$FlagAtaque = 0;
$TemAtaque1Min = 0;
}
#####################################################################
# Limpando os Micro Fluxos
#####################################################################
$DeletaFluxosSQL = qq {Delete From Fluxos };
$DeletaFluxos = $dbh->prepare( $DeletaFluxosSQL );
$DeletaFluxos->execute();
#####################################################################
# Atualizando os Limites de Tempo
#####################################################################
$TotalMin = $TotalMin + 1;
$MinTime = $AuxTime;
$AuxTime = $MinTime + 60;
}
#####################################################################
# Resumo das Informações
#####################################################################
$TAtaques = $TP + $FN;
if ( $TP < 1 )
```

```
{
$PAcerto = 0;
$PTP = 0;
}
else
{
$PAcerto = ($TP * 100)/$TAtaques;
$PTP = $TP / $TSessions;
}
if ( $FP < 1 )
{
$PFP = 0;
}
else
{
$PFP = $FP / $TSessions;
}
if ( $FN < 1 )
{
$PFN = 0;
}
else
{
```

```
$PFN = $FN / $TSessions;

}

if ( $TN < 1 )

{

$PTN = 0;

}

else

{

$PTN = $TN / $TSessions;

}

if ( $TN < 1 and $TP < 1 )

{

$QA = 0;

}

else

{

$QA = (($TN + $TP)*100)/$TSessions;

}

if ( $FP < 1 and $FN < 1 )

{

$QE = 0;

}

else
```

```
{
$QE = (($FP + $FN)*100)/$TSessions;
}
if ( $PAcerto < 1 )
{
$QR = 0;
}
else
{
$QR = ($PAcerto*100)/$TSessions;
}
print RESUMO "##############################################\n";
print RESUMO "# BIS - Bayesian Inference System \n";
print RESUMO "# Modulo Contador da Qualidade dos Alarmes \n";
print RESUMO "# Versão 1.0.0 \n";
print RESUMO "# \n";
print RESUMO "##############################################\n";
print RESUMO "\n";
print RESUMO "############# Números Gerais #################\n";
print RESUMO "Numero Total de Intervalos = $TotalMin \n";
print RESUMO "Numero Total de Intervalos Analisados = $TSessions \n";
print RESUMO "Numero Total de Ataques = $TAtaques \n";
print RESUMO "Perc. de Acerto Referencial = $PAcerto \n";
```

```
print RESUMO "\n";

print RESUMO "######## Resumo do Processo #################\n";

print RESUMO "FP = $FP ($PFP) \n";

print RESUMO "FN = $FN ($PFN) \n";

print RESUMO "TP = $TP ($PTP) \n";

print RESUMO "TN = $TN ($PTN) \n";

print RESUMO "\n";

print RESUMO "########### Avaliação do Processo ###############\n";

print RESUMO "Qualidade de Acerto = $QA \n";

print RESUMO "Qualidade de Erro = $QE \n";

print RESUMO "Qualidade de Ref = $QR \n";

print RESUMO "\n";

print RESUMO "################################################\n";

close TREINAMENTO;

close TPOS;

close FPOS;

close FNEG;

exit;
```

Apêndice B

Apêndice II

Foi realizada uma modificação no programa BCI para gerar saídas dos resultados parciais da inferência Naïve Bayesiana. Apresentaremos a seguir a saída com os resultados para cada tipo de ataque. As seguintes designações devem ser observadas:

Classe0 => Não Ataque.

Classe1 => Ataque.

Classe Provável => Resultado do classificador Naïve Bayesiano.

Classe Definida => Marcação do DARPA

B.1 Ataque Guest

DARPA = 0 e Bayes = 0 => RN ou TN

Variável de Entrada cl1 (Vin): 1

Media (u) da cl1: 49.4949

Distancia p/ a Media (Vin - u): -48.4949

Variância (o2) cl1: 8.3454

Calc. Epsilon (exp-(Vin-u)2/s): 6.4172e-62

Calc. sqrt (sqrt 2PI*o2): 7.24125

Calc. Likelihood (Epsilon/sqrt) da cl1: 8.862e-63

Variável de Entrada cl0 (Vin): 1

Media (u) da cl0: 2.52918

Distancia p/ a Media (Vin - u): -1.52918

Variância (o2) cl0: 48.8221

Calc. Epsilon (exp-(Vin-u)2/s): 0.976336

Calc. sqrt (sqrt 2PI*o2): 17.5145

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.0557444

LLH1: 8.862e-63

LLH0: 0.0557444

Prob. a Prior1 (igual a Prob. Poster anterior): 0.371612

Prob. a Prior0 (igual a Prob. Poster anterior): 0.628388

Calc. da Prob. a Poster1 (llh1*prior1): 3.29322e-63

Calc. da Prob. a Poster0 (llh0*prior0): 0.0350291

Soma das Prob. a Poster (poster0+poster1): 0.0350291

Variável de Entrada cl1 (Vin): 0

Media (u) da cl1: 94.4434

Distancia p/ a Media (Vin - u): -94.4434

Variância (o2) cl1: 8.21882

Calc. Epsilon (exp-(Vin-u)2/s): 2.18176e-236

Calc. sqrt (sqrt 2PI*o2): 7.18612

Calc. Likelihood (Epsilon/sqrt) da cl1: 3.03607e-237

Variável de Entrada cl0 (Vin): 0

Media (u) da cl0: 0.111981

Distancia p/ a Media (Vin - u): -0.111981

Variância (o2) cl0: 1.31262

Calc. Epsilon (exp-(Vin-u)2/s): 0.995235

Calc. sqrt (sqrt 2PI*o2): 2.87183

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.34655

LLH1: 3.03607e-237

LLH0: 0.34655

Prob. a Prior1 (igual a Prob. Poster anterior): 3.29322e-63

Prob. a Prior0 (igual a Prob. Poster anterior): 0.0350291

Calc. da Prob. a Poster1 (llh1*prior1): 9.99845e-300

Calc. da Prob. a Poster0 (llh0*prior0): 0.0121394

Soma das Prob. a Poster (poster0+poster1): 0.0121394

Prob. a Poster da cl0: 0.0121394

Prob. a Poster da cl1: 9.99845e-300

Prob. a Poster Final (cl0 se cl0>cl1/cl1 se cl1>cl0): 0.0121394

Soma das Prob. a Poster (poster0+poster1): 0.0121394

Confiança: 1

Classe Provável: 0

Classe Definida: 0

B.2 Ataque Neptune

DARPA = 0 e Bayes = 0 => RN ou TN

Variável de Entrada cl1 (Vin): 23

Media (u) da cl1: 1.12037

Distancia p/ a Media (Vin - u): 21.8796

Variância (o2) cl1: 0.105881

Calc. Epsilon (exp-(Vin-u)2/s): 0

Calc. sqrt (sqrt 2PI*o2): 0.815641

Calc. Likelihood (Epsilon/sqrt) da cl1: 0

Variável de Entrada cl0 (Vin): 23

Media (u) da cl0: 61.1082

Distancia p/ a Media (Vin - u): -38.1082

Variância (o2) cl0: 1.08983e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.999334

Calc. sqrt (sqrt 2PI*o2): 2616.79

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.000381893

LLH1: 0

LLH0: 0.000381893

Prob. a Prior1 (igual a Prob. Poster anterior): 0.00169551

Prob. a Prior0 (igual a Prob. Poster anterior): 0.998304

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 0.000381245

Soma das Prob. a Poster (poster0+poster1): 0.000381245

Variável de Entrada cl1 (Vin): 46

Media (u) da cl1: 4018.22

Distancia p/ a Media (Vin - u): -3972.22

Variância (o2) cl1: 5.73788e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.252854

Calc. sqrt (sqrt 2PI*o2): 6004.35

Calc. Likelihood (Epsilon/sqrt) da cl1: 4.21118e-05

Variável de Entrada cl0 (Vin): 46

Media (u) da cl0: 103.485

Distancia p/ a Media (Vin - u): -57.485

Variância (o2) cl0: 1.39199e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.998814

Calc. sqrt (sqrt 2PI*o2): 2957.39

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.000337735

LLH1: 4.21118e-05

LLH0: 0.000337735

Prob. a Prior1 (igual a Prob. Poster anterior): 0

Prob. a Prior0 (igual a Prob. Poster anterior): 0.000381245

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 1.2876e-07

Soma das Prob. a Poster (poster0+poster1): 1.2876e-07

Variável de Entrada cl1 (Vin): 22

Media (u) da cl1: 0.0925926

Distancia p/ a Media (Vin - u): 21.9074

Variância (o2) cl1: 3.69513

Calc. Epsilon (exp-(Vin-u)2/s): 6.2556e-29

Calc. sqrt (sqrt 2PI*o2): 4.81842

Calc. Likelihood (Epsilon/sqrt) da cl1: 1.29827e-29

Variável de Entrada cl0 (Vin): 22

Media (u) da cl0: 53.0078

Distancia p/ a Media (Vin - u): -31.0078

Variância (o2) cl0: 441635

Calc. Epsilon (exp-(Vin-u)2/s): 0.998912

Calc. sqrt (sqrt 2PI*o2): 1665.8

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.000599661

LLH1: 1.29827e-29

LLH0: 0.000599661

Prob. a Prior1 (igual a Prob. Poster anterior): 0

Prob. a Prior0 (igual a Prob. Poster anterior): 1.2876e-07

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 7.72123e-11

Soma das Prob. a Poster (poster0+poster1): 7.72123e-11

Variável de Entrada cl1 (Vin): 0.17

Media (u) da cl1: 3298.91

Distancia p/ a Media (Vin - u): -3298.74

Variância (o2) cl1: 3.37976e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.199921

Calc. sqrt (sqrt 2PI*o2): 4608.22

Calc. Likelihood (Epsilon/sqrt) da cl1: 4.33837e-05

Variável de Entrada cl0 (Vin): 0.17

Media (u) da cl0: 0.705143

Distancia p/ a Media (Vin - u): -0.535143

Variância (o2) cl0: 181.526

Calc. Epsilon (exp-(Vin-u)2/s): 0.999212

Calc. sqrt (sqrt 2PI*o2): 33.7722

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.0295868

LLH1: 4.33837e-05

LLH0: 0.0295868

Prob. a Prior1 (igual a Prob. Poster anterior): 0

Prob. a Prior0 (igual a Prob. Poster anterior): 7.72123e-11

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 2.28447e-12

Soma das Prob. a Poster (poster0+poster1): 2.28447e-12

Variável de Entrada cl1 (Vin): 4

Media (u) da cl1: 3544.55

Distancia p/ a Media (Vin - u): -3540.55

Variância (o2) cl1: 3.26797e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.14691

Calc. sqrt (sqrt 2PI*o2): 4531.36

Calc. Likelihood (Epsilon/sqrt) da cl1: 3.24208e-05

Variável de Entrada cl0 (Vin): 4

Media (u) da cl0: 6.44402

Distancia p/ a Media (Vin - u): -2.44402

Variância (o2) cl0: 826.722

Calc. Epsilon (exp-(Vin-u)2/s): 0.996394

Calc. sqrt (sqrt 2PI*o2): 72.0725

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.0138249

LLH1: 3.24208e-05

LLH0: 0.0138249

Prob. a Prior1 (igual a Prob. Poster anterior): 0

Prob. a Prior0 (igual a Prob. Poster anterior): 2.28447e-12

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 3.15825e-14

Soma das Prob. a Poster (poster0+poster1): 3.15825e-14

Variável de Entrada cl1 (Vin): 2

Media (u) da cl1: 3533.9

Distancia p/ a Media (Vin - u): -3531.9

Variância (o2) cl1: 3.25894e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.14751

Calc. sqrt (sqrt 2PI*o2): 4525.1

Calc. Likelihood (Epsilon/sqrt) da cl1: 3.25981e-05

Variável de Entrada cl0 (Vin): 2

Media (u) da cl0: 3.42846

Distancia p/ a Media (Vin - u): -1.42846

Variância (o2) cl0: 660.706

Calc. Epsilon (exp-(Vin-u)2/s): 0.998457

Calc. sqrt (sqrt 2PI*o2): 64.4309

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.0154966

LLH1: 3.25981e-05

LLH0: 0.0154966

Prob. a Prior1 (igual a Prob. Poster anterior): 0

Prob. a Prior0 (igual a Prob. Poster anterior): 3.15825e-14

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 4.89419e-16

Soma das Prob. a Poster (poster0+poster1): 4.89419e-16

Prob. a Poster da cl0: 4.89419e-16

Prob. a Poster da cl1: 0

Prob. a Poster Final (cl0 se cl0>cl1/cl1 se cl1>cl0): 4.89419e-16

Soma das Prob. a Poster (poster0+poster1): 4.89419e-16

Confiança: 1

Classe Provável: 0

Classe Definida: 0

B.3 Ataque Portsweep

DARPA = 0 e Bayes = 0 => RN ou TN

Variável de Entrada cl1 (Vin): 32

Media (u) da cl1: 1.91917

Distancia p/ a Media (Vin - u): 30.0808

Variância (o2) cl1: 0.0743002

Calc. Epsilon (exp-(Vin-u)2/s): 0

Calc. sqrt (sqrt 2PI*o2): 0.683258

Calc. Likelihood (Epsilon/sqrt) da cl1: 0

Variável de Entrada cl0 (Vin): 32

Media (u) da cl0: 56.2387

Distancia p/ a Media (Vin - u): -24.2387

Variância (o2) cl0: 1.15124e+06

Calc. Epsilon (exp-(Vin-u)2/s): 0.999745

Calc. sqrt (sqrt 2PI*o2): 2689.51

Calc. Likelihood (Epsilon/sqrt) da cl0: 0.00037172

LLH1: 0

LLH0: 0.00037172

Prob. a Prior1 (igual a Prob. Poster anterior): 0.00502424

Prob. a Prior0 (igual a Prob. Poster anterior): 0.994976

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 0.000369853

Soma das Prob. a Poster (poster0+poster1): 0.000369853

Variável de Entrada cl1 (Vin): 1

Media (u) da cl1: 14.3638

Distancia p/ a Media (Vin - u): -13.3638

Variância (o2) cl1: 1964.12

Calc. Epsilon (exp-(Vin-u)2/s): 0.955555

Calc. sqrt (sqrt 2PI*o2): 111.09

Calc. Likelihood (Epsilon/sqrt) da cl1: 0.00860165

Variável de Entrada cl0 (Vin): 1

Media (u) da cl0: 1

Distancia p/ a Media (Vin - u): 0

Variância (o2) cl0: 0

Calc. Epsilon (exp-(Vin-u)2/s): 1

Calc. sqrt (sqrt 2PI*o2): 1.77245e-06

Calc. Likelihood (Epsilon/sqrt) da cl0: 564190

LLH1: 0.00860165

LLH0: 564190

Prob. a Prior1 (igual a Prob. Poster anterior): 0

Prob. a Prior0 (igual a Prob. Poster anterior): 0.000369853

Calc. da Prob. a Poster1 (llh1*prior1): 0

Calc. da Prob. a Poster0 (llh0*prior0): 208.667

Soma das Prob. a Poster (poster0+poster1): 208.667

Prob. a Poster da cl0: 208.667

Prob. a Poster da cl1: 0

Prob. a Poster Final (cl0 se cl0>cl1/cl1 se cl1>cl0): 208.667

Soma das Prob. a Poster (poster0+poster1): 208.667

Confiança: 1

Classe Provável: 0

Classe Definida: 0