UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

Hipercaixas Delimitadoras na Detecção de Intrusos em Redes de Computadores

João Daher Neto

Itajubá, Agosto de 2014

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

João Daher Neto

Hipercaixas Delimitadoras na Detecção de Intrusos em Redes de Computadores

Dissertação submetida ao Programa de Pós-Graduação em Ciência e Tecnologia da Computação como parte dos requisitos para obtenção do Título de Mestre em Ciência e Tecnologia da Computação

Área de Concentração: Matemática Computacional

Orientador: Prof. Dr. Carlos Henrique Valério de Moraes
Co-Orientador: Prof. Dr. Rodrigo Maximiano Antunes de Almeida

Agosto de 2014
Itajubá / MG

Ficha catalográfica elaborada pela Biblioteca Mauá –
Bibliotecária Margareth Ribeiro- CRB_6/1700

V697u

Daher Neto, João
Hipercaixas delimitadoras na detecção de intrusos em redes de computadores / João Daher Neto. -- Itajubá, (MG) : [s.n.], 2014.
64 p. : il.

Orientador: Prof. Dr. Carlos Henrique Valério de Moraes.
Coorientador: Prof. Dr. Rodrigo M. Antunes de Almeida.
Dissertação (Mestrado) – Universidade Federal de Itajubá.

1. Volume delimitador. 2. Detecção de intrusos. 3. Classifica_ dor semi-supervisonado. I. Moraes, Carlos Henrique Valério de, orient. II. Almeida, Rodrigo Maximiniano Antunes de, coorient. III. Universidade Federal de Itajubá. IV. Título.

UNIVERSIDADE FEDERAL DE ITAJUBÁ
PROGRAMA DE PÓS GRADUAÇÃO EM
CIÊNCIA E TECNOLOGIA DA COMPUTAÇÃO

João Daher Neto

Hipercaixas Delimitadoras na Detecção de Intrusos em Redes de Computadores

Dissertação aprovada por banca examinadora em 18 de agosto de 2014, conferindo ao autor o título de ***Mestre em Ciências e Tecnologia da Computação.***

Banca Examinadora:
Prof. Dr. Carlos Henrique Valério de Moraes (Orientador)
Prof. Dr. Rodrigo Maximiano Antunes de Almeida (Co-orientador)
Prof. Dr. Luiz Eduardo da Silva (UNIFAL)
Prof. Dr. Maurílio Pereira Coutinho (UNIFEI)

Itajubá / MG

Agradecimentos

A Deus, por me oferecer tantas oportunidades de crescimento pessoal e profissional, além de me agraciar com a saúde e inteligência necessários para seguir os caminhos que me propus.

A meus pais, Rosangela e Francisco, e irmãos, Bruna e Thiago, por serem o alicerce e o exemplo de conquista, superando todas e quaisquer adversidades que vida eventualmente impõe.

A minha namorada, Mariana, e amigos, pelo companheirismo e incentivo durante todo o processo de estudos, especialmente nos momentos em que essa conquista parecia tão distante.

Aos meus orientor e co-orientador, Carlos Henrique e Rodrigo Maximiano, e meus professores, por disporem de seu tempo e conhecimento para esclarecer dúvidas e propor ideias, sempre com muita paciência e discernimento, além de servirem de inspiração para prosseguir com a carreira acadêmica.

A inteligência vale mais que a força bruta.

Merlin - A Espada Era a Lei (Disney, 1963)

Resumo

Este trabalho apresenta um sistema de classificação supervisionado através da junção de uma técnica de colisão de objetos, AABB, estratégias de quebra de caixas e inferência numérica. Adaptado para n-dimensões e com regras estatísticas de decisão, os experimentos mostraram bons resultados na classificação de conjuntos de dados conhecidos, melhores que algumas técnicas existentes em literatura.

O classificador desenvolvido foi uma abordagem inovadora na área de segurança de redes, sendo capaz de analisar características de pacotes da rede e identificar diferentes tipos de intrusos de forma satisfatória.

Abstract

This work presents a supervised classification system created from the aggregation of a object collision technique, AABB, box-breaking and numeric inference strategies. It was adapted to work with n-dimensions and to use statistical decision rules; the tests showed great results when classifying well-known datasets, eve better than some techniques found in the literature

The classifier was developed by using a novel approach on the network security field, it has been able to analyse packet's features and identify different sorts of intruders satisfactorily.

Sumário

Lista de Figuras

Lista de Tabelas

Glossário

AABB	*Axis Aligned Bounding Boxes*
ACL	*Access Control List*
AV	*Antivírus*
BV	*Bounding Volume*
DDoS	*Dynamic Denial of Service*
DNS	*Domain Name System*
EPM	*Erro Padrão da Média*
FN	*Falso Negativo*
FP	*Falso Positivo*
HIDS	*Host-based Intrusion Detection System*
IDE	*Integrated Development Environment*
IDS	*Intrusion Detection System*
IP	*Internet Protocol*
IPS	*Intrusion Protection System*
KBS	*Knowledge-Based System*
NIDS	*Network-based Intrusion Detection System*
NSA	*National Security Agency*
OBB	*Oriented Bounding Boxes*
R2L	*Remote to Local*
SBV	*Sphere Bounding Volume*
SERPRO	*Serviço Federal de Processamento de Dados*
SVM	*Support Vector Machine*
U2R	*User to Root*
VN	*Verdadeiro Negativo*
VP	*Verdadeiro Positivo*

1 Introdução

Os usuários de computadores, em especial aqueles conectados à Internet, têm se deparado cada vez mais com ameaças (KASPERSKY, 2014) que afetam a integridade, confidencialidade e disponibilidade de seus dados e serviços. Tanto grandes empresas quanto usuários comuns são alvos de ataques cibernéticos, motivados por curiosidade, dinheiro ou pela defesa de alguma causa política ou social.

O crescimento de ataques a computadores teve crescimento exponencial nos últimos anos: de acordo com AV (2014), de 2013 para 2014, o número de *malwares* identificados aumentou 44 vezes, totalizando mais de 220 milhões de ameaças registadas até o início do ano de 2014.

As notícias de invasão de *sites* e sistemas computacionais são cada vez mais comuns, e os prejuízos causados nas empresas vítimas dos ataques chegam a bilhões. Por exemplo, a invasão à Playstation Network[1] (CRUNCH, 2011; NETWORK, 2011) que, além de deixar os dados de milhões de usuários vulneráveis, fez com que as ações da empresa caíssem drasticamente (aproximadamente 3%) devido à diminuição da confiabilidade dos serviços oferecidos.

Em 2011, cibercriminosos tiveram sucesso ao invadir e extrair informações de um banco de dados do Exército Brasileiro[2] (GLOBO, 2011). Foram divulgadas publicamente informações contendo usuário e senha de mais de 300 pessoas registradas no sistema do Exército.

A onda de ataques ao governo foi frustrada em 2011, ocasião em que um grupo tentou invadir alguns sites do Governo Brasileiro[3] (RECORD, 2011), não obtendo sucesso graças às medidas de segurança tomadas pelo SERPRO[4], empresa responsável pela prestação de serviços públicos relacionados à Tecnologia da Informação.

[1] http://us.playstation.com/
[2] http:/www.eb.mil.br/
[3] http:/www.presidencia.gov.br/
[4] https://www.serpro.gov.br/

Mais recentemente, estimulados pela atenção que o Brasil tem recebido devido à Copa do Mundo, o sistema de e-mails do Itamaraty[5] teve diversos documentos - muitos confidenciais - vazados após uma invasão ao sistema (VEJA, 2014). A divulgação de mensagens confidenciais pode comprometer a relação exterior do Brasil, como já aconteceu com diversos países após o vazamento de informações relacionadas à espionagem feita pela NSA[6] (CBSNEWS, 2014) em diversos outros países, baseando majoritariamente na exploração de falhas de segurança nos sistemas conectados à Internet.

A necessidade de uma técnica versátil de detecção de ameaças surge à medida em que a maioria dos *firewalls* e antivírus convencionais têm dificuldade de acompanhar a alta taxa de criação e sofisticação de novos *malwares* e formas de ataque. Como declarado recentemente pelo vice-presidente da Symantec[7], Bryan Dye: "[o antivírus] está morto" (tradução livre) (YADRON, 2014), ressaltando a grande dificuldade em combater *malwares* através das técnicas atuais de criação de vacinas.

Além da necessidade de adaptação, outra questão inerente à área de detecção de intrusos é a dificuldade em analisar e extrair informações de *Big Data*. O conceito de *Big Data* determina uma quantidade enorme de dados, pública ou privada, em constante crescimento, incapaz de ser processada por métodos convencionais de banco de dados (ECONOMIST, 2010). Na área de segurança computacional, os pacotes que trafegam na rede constituem um volume enorme de informação que deve ser analisada de forma eficiente, a fim de não comprometer o desempenho da rede.

1.1 Revisão Bibliográfica

Com o intuito de se desenvolver um classificador, há 2 abordagens muito comuns no que se refere à estrutura do sistema: sistemas baseados em conhecimento (KBS) e sistemas baseados em regras.

Os classificadores com base em conhecimento utilizam um conjunto de informações representadas a partir do conhecimento existente fora do mundo computacional, denominado base de conhecimento. O *software* MYCIN (SHORTLIFFE, 1984), por exemplo, é do tipo KBS (denominado sistema especialista), de aplicação médica, capaz de identificar bactérias a partir de determinadas infecções e, por fim, recomendar medicamentos profiláticos. Este sistema foi construído a partir de diversas definições de bactérias, mecanismos

[5]http:/www.itamaraty.gov.br/
[6]http://www.nsa.gov/
[7]http://www.symantec.com/

de ação e sintomas, além de um extenso cadastro de medicamentos e seus efeitos.

No trabalho realizado por UCHOA (2009), um detector de intrusos é desenvolvido com o conceito de sistema baseado em conhecimento, utilizando uma abordagem bio-inspirada: algoritmo imunológico. Baseado no sistema imune humano, o algoritmo simula células B, T e macrófagos para detectar padrões de ameaças, sendo que cada célula possui uma especialidade de combater um tipo específico de antígeno (*i.e.* intruso), assim como acontece no sistema imune real.

Após a criação do sistema imune, cada célula é responsável por tentar reconhecer a ameaça utilizando seus próprios conhecimentos (adquiridos durante o processo de treinamento). A inferência do tipo de ameaça é feita após a análise de todas as respostas obtidas, como uma espécie de votação.

Uma das vantagens da técnica de UCHOA (2009) é a configuração do sistema baseada na *Danger Theory*, que consiste na adaptação coletiva das células em ambiente muito ou pouco reativos (*e.g.* durante um período em que o sistema estiver sendo constantemente atacado, o detector pode se tornar mais rigoroso na classificação). A desvantagem deste sistema, bem como de outros baseados em algoritmos evolutivos, é a vasta sensibilidade paramétrica (11 parâmetros contínuos): a diversidade de configurações possíveis pode dificultar a obtenção de um detector de intrusos ideal, exigindo inúmeros testes sem jamais poder afirmar ter-se obtido um configuração ótima.

Por outro lado, sistemas baseados em regras utilizam expressões lógicas para classificar os dados. As regras mais simples de se adotar são clássicas, que trabalham em um sistema binário: sim ou não. O fato de serem extremamente pontuais lhes confere, na maioria das vezes, um baixo custo computacional[8], que pode variar de acordo com a quantidade de regras.

Tem se tornado cada vez mais frequente o uso de sistemas baseados em regras não-clássicas. Técnicas *fuzzy* são umas das abordagens mais comuns para regras não-clássicas. Estes sistemas fazem uso da probabilidade de um determinado dado ser classificado como diversas classes diferentes (pertinência). Dessa forma, os sistemas deixam de trabalhar de forma binária (0 ou 1) e adotam a flexibilidade probabilística (*de* 0 a 1).

O trabalho de Sanz et al. (2014) cria um classificador baseado em regras linguísticas *fuzzy* com o objetivo de determinar um diagnóstico médico. Em seu trabalho, o classificador é treinado utilizando informações sobre o estado do paciente, em sua maioria,

[8]Regras clássicas podem ser convertidas em expressões lógicas *if-then-else*, executadas em tempo inferior que cálculo de funções de alta complexidade.

descrições de sintomas e atividades da pessoa avaliada. O sistema infere o nível de risco em que o paciente se encontra, determinando uma probabilidade do mesmo apresentar alguma doença cardiovascular dentre as diversas que o sistema é capaz de identificar após seu treinamento.

A eficiência de sistemas baseados em regras já pode ser comprovada em diversos trabalhos. O trabalho de Bilar (2006) procura explorar a similaridade dos *malwares* e ataques, denominada assinatura. As regras são criadas após a análise dinâmica e estrutural de arquivos executáveis, obtida basicamente após a execução do *malware* e constante monitoramento de outros arquivos, serviços e registros alterados, bem como de chamadas de sistema e *dump* de memória[9].

O sistema analisa o comportamento dos *malwares* de acordo com os tipos de chamadas de sistema que são realizadas durante a invasão, identificando padrões entre os *malwares* de mesma categoria (*e.g. trojans, worms, botnets*).

O IDS (Sistema de Detecção de Intrusos) proposto por Mbikayi (2012), por sua vez, é capaz de identificar conexões anômalas (*e.g. ataque DoS*) analisando os pacotes que trafegam na rede. Além da geração de um sistema de regras, neste trabalho, o autor busca simplificar o IDS final com o auxílio de algoritmo genético, outra técnica já consolidada na área de otimização.

De acordo com o trabalho de Leavitt (2013), sistemas baseados em regras possuem a versatilidade desejada, sendo capaz de identificar características e padrões, simplificando a análise de informações, por isso têm sido adotados cada vez com maior frequência para o desenvolvimento de classificadores.

1.2 Organização do Trabalho

A fim de explanar o sistema desenvolvido, este trabalho está organizado da seguinte forma:

Capítulo 1: contextualiza a área de segurança de redes e aborda a necessidade de novas técnicas de combate a ataques. Também exibe um comparativo de outros trabalhos de mesma aplicação, ressaltando as diferenças em relação a este trabalho;

Capítulo 2: são apresentados alguns conceitos sobre segurança computacional, vulnera-

[9] *Dump* de memória é a extração de dados contidos em uma região específica da memória RAM do sistema.

bilidade de sistemas e estratégias de detecção de intrusos.

Capítulo 3: trata das técnicas e algoritmos utilizados para a criação do sistema, bem como suas variações;

Capítulo 4: aborda o processo de criação do IDS proposto, em suas 3 etapas: representação de dados para ter compatibilidade com o sistema numérico utilizado, modelagem do sistema utilizando a técnica de colisão de objetos modificada e a criação do sistema de regras a partir do modelo obtido;

Capítulo 5: são apresentadas as bases de dados utilizadas para a criação e validação do sistema, as métricas adotadas para verificar a eficiência e confiabilidade do classificador e, por fim, os resultados obtidos acompanhados da análise de desempenho.

Capítulo 6: conclui o trabalho analisando os resultados obtidos e propondo possíveis melhorias para trabalhos futuros.

2 Segurança Computacional

À medida em que o tráfego e armazenamento de informações na Internet tornaram-se atividades que exigiam segurança, foram criadas ferramentas para garantir a integridade, confidencialidade e disponibilidade dos dados e serviços.

O *firewall* é o conceito mais primitivo de segurança na rede. Um *firewall* é um sistema posicionado na rede e tem função análoga à da portaria de um condomínio residencial: decidir quem entra e quem sai. O *firewall* busca proteger a rede interna analisando cada pacote que tenta entrar na rede, verificando em uma lista de permissões (ACL) se um determinado pacote tem autorização para trafegar em um determinado endereço IP, porta ou serviço do computador.

Atualmente os *firewalls* se tornaram ferramentas sofisticadas, muitas vezes ganhando hardware dedicado para executarem suas rotinas especialmente em redes de tráfego intenso.

Embora a maioria das ameaças chegue através da rede, há a necessidade de proteger a máquina localmente, especialmente no caso de uma ameaça ter conseguido invadir o sistema. Para esse propósito, foram criados os antivírus. Um sistema antivírus é um software especializado em reconhecer, prevenir e remover *malwares*[1] de um modo geral, especialmente vírus.

Um antivírus atua localmente em uma máquina, constantemente buscando por ameaças nos arquivos e serviços do sistema. O modo mais comum de detecção de *malwares* é a assinatura: um trecho de código ou *hash*[2] que identifica de forma única o *malware*.

Entretanto, o uso de assinaturas não é capaz de detectar novas ameaças, denominadas ameaças *zero-day*, uma vez que a disponibilização de uma assinatura depende da velocidade da equipe de desenvolvimento em encontrar a ameaça. Para este propósito, são

[1] *Malware* é qualquer tipo de código de propósito malicioso: obter acesso indevido, extrair informações sem autorização ou interromper o funcionamento normal do computador.

[2] Valor simplificado obtido através de uma função que diferencia objetos.

utilizadas heurísticas, que são basicamente variações das assinaturas conhecidas. Embora menos precisas, estas técnicas fornecem um nível de segurança extra ao sistema.

O grande diferencial e potencial de um antivírus é sua capacidade de reagir às ameaças, removendo o *malware* do sistema, mesmo que este esteja embutido em um arquivo do sistema. Esta remoção só é possível caso haja disponível uma vacina para aquela ameaça, sendo esta normalmente disponibilizada pela equipe de desenvolvimento do *software* através de atualizações.

Um IDS (Sistema Detector de Intrusos), por sua vez, avalia o contexto e até o comportamento (como será abordado na Seção 2.2). Este tipo de sistema é capaz de detectar ataques ao sistema ou a uma rede inteira. Alguns sistemas IDS têm a capacidade reagir aos ataques detectados, os quais são denominados IPS (Sistema de Proteção de Intrusos) (PEREIRA, 2006).

Recentemente é possível notar a evolução dos sistemas antivírus, passando a detectar não só arquivos e serviços, mas tráfego na rede como SPAMs e até ataques. Percebe-se a tendência de se tornarem sistemas mais completos, como um IPS, a fim de atender à necessidade de proteger contra dados vindo diretamente da rede e não apenas aqueles armazenados localmente.

2.1 Ataques Cibernéticos

A comunidade cibercriminosa busca o reconhecimento em suas ações: quanto mais importante o alvo do ataque, maior o reconhecimento dos autores. A fim de realizar tais ataques, os invasores buscam explorar vulnerabilidades nos sistemas computacionais. Estas falhas podem estar presentes por diversos motivos:

- *Bugs* no software utilizado, como no recente caso da falha de segurança Heartbleed (GUJRATHI, 2014), que tomou repercussão mundial por se tratar de uma vulnerabilidade em um protocolo adotado pela massiva maioria dos servidores;
- Falhas na infraestrutura, como, por exemplo, a má configuração do *firewall* de uma empresa que acaba liberando acesso indefinido à rede interna;
- Mal comportamento dos usuários, como no caso do uso de senhas fracas.

Embora os usuários comuns sejam alvos de ataques, os servidores são mais suscetíveis por estarem disponíveis na internet a maior parte do tempo. Os *black hackers* - ou

crackers, como são chamados em alguns meios - possuem diversas ferramentas e técnicas para realizar um ataque a um sistema computacional. Como mostrado em (UCHOA, 2009), os termos e ataques mais comuns são:

Footprinting: consiste em coletar informações sobre um alvo específico. É possível realizar o *footprint* de forma manual, através da leitura de informações na própria página ou em sites de busca. É comum tentar-se descobrir o administrador do sistema a fim de invadir a conta para obter acesso irrestrito ao sistema. Também há ferramentas que automatizam o *footprinting*: o ZenMap[3] implementa o comando *nmap* (Network Mapper), tornando-o capaz de identificar características sobre o sistema alvo.

Como mostrado na Figura 1, foi possível identificar que o alvo (o *host*) está utilizando o sistema operacional Micrososft Windows XP com a atualização SP2. Sabendo que já foi disponibilizada a atualização mais recente, SP3, o invasor pode buscar explorar as falhas ainda presentes no sistema desatualizado.

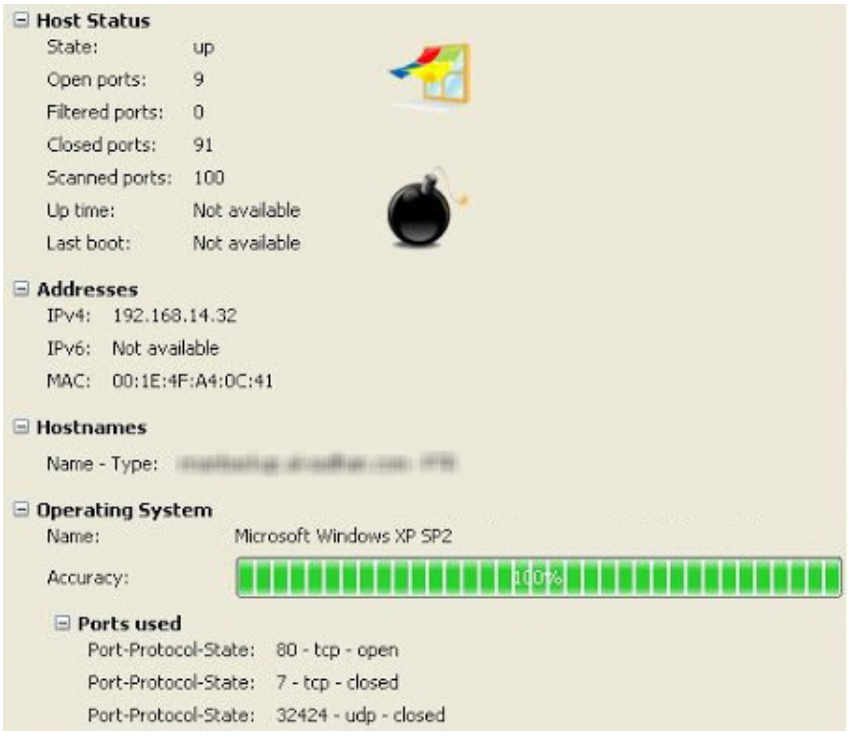

Figura 1: Footprinting através do NMap

Probe: um *scanner* é capaz de identificar portas abertas em uma determinada máquina na rede. Algumas ferramentas mais sofisticadas, como o ZenMap e o OpenVAS[4],

[3]http://nmap.org/zenmap/
[4]http://www.openvas.org/

identificam portas com potencial vulnerabilidade que podem ser exploradas pelo invasor.

Sniffer: em uma rede física, especialmente as sem fio, embora os pacotes estejam acessíveis a todas as máquinas, cada máquina se interessa apenas por aqueles que lhe foram destinados. A técnica de *sniffer* consiste em forçar a própria máquina a ler todos os pacotes que trafegam na rede. Este modo de execução da placa de rede se chama modo promíscuo. A leitura de pacotes da rede que possuem outros destinos permite que o invasor descubra informações sobre outros usuários da rede, obtidas através de conversas ou senhas interceptadas.

Spoofing: a técnica de *spoofing* é semelhante à de falsa identidade que acontece fora do mundo cibernético. O objetivo é sempre burlar a autenticidade se passando pela vítima no envio de mensagens ou obtenção de acesso. É comum o invasor procurar impedir o envio de pacotes de um usuário e enviar pacotes forjados ao destino em nome da vítima, consumando o ataque denominado *man-in-the-middle*.

Os pacotes também podem ter o IP forjado, ludibriando o destinatário e fazendo-o responder ao pedido do invasor, o que caracteriza um ataque de *IP Spoofing*. Também é possível forjar pacotes DNS, fazendo com que o usuário acesse uma página de internet falsa.

Denial of Service - DoS: uma das formas mais simples e mais comuns de ataque. A negação de serviço consiste prover inúmeros acessos simultâneos a uma mesma máquina, sobrecarregando o servidor e impedindo sua execução.

Se o invasor tentar as múltiplas conexões a partir de sua máquina, o *firewall* do alvo é capaz de identificar o ataque, bloqueando seu acesso antes do ataque ser concluído. A fim de obter os inúmeros acessos de forma não centralizada, os invasores "zumbificam" outras máquinas e as obrigam a acessar o mesmo servidor de forma simultânea, como ilustrado na Figura 2. Esta variação da negação de serviço é denominada DDoS.

Os ataques de DoS têm ganhado mais atenção ultimamente à medida em que grandes empresas, como Amazon[5] e UOL[6], foram alvos desse tipo de ataque, impedindo o funcionamento normal de seus serviços aos clientes legítimos.

Código Malicioso: *malwares* são programas, ou trechos de código alterados, com fins ilícitos. Possuem objetivos e funções diversas, mas sempre buscam se instalar na

[5]http://www.amazon.com/
[6]http://www.uol.com.br/

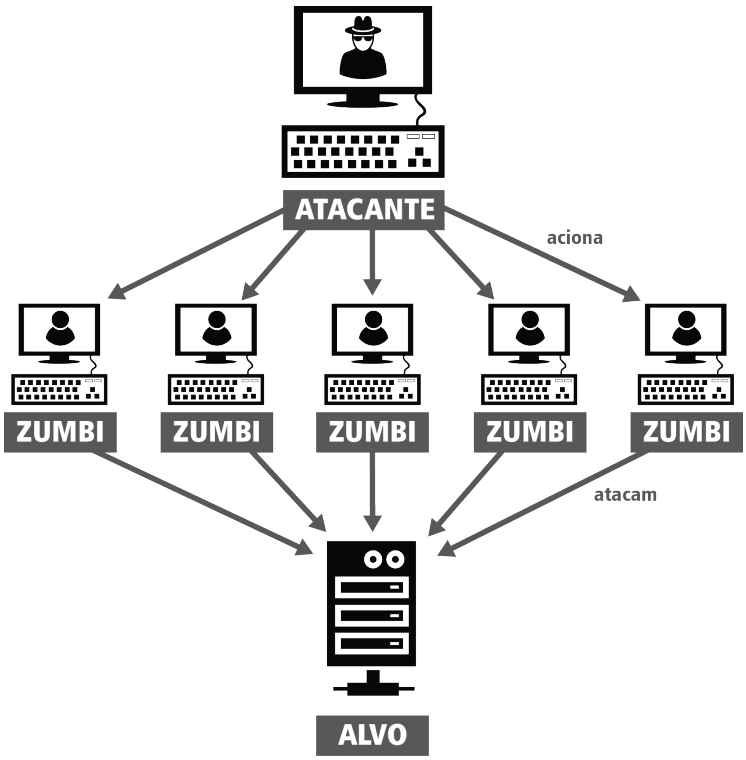

Figura 2: Estrutura do Ataque DDoS

máquina da vítima para realizar suas tarefas. Ainda não há uma taxonomia bem definida para os *malwares*, e alguns deles sequer possuem funções bem definidas, dificultando sua classificação. Os códigos maliciosos que merecem destaque são:

***Vírus*:** são programas que executam tarefas específicas determinadas pelos seus criadores, desde apagar arquivos do computador até forçar a reinicialização do sistema impossibilitando o uso da máquina. O que difere os vírus dos demais tipos de *malwares* é sua capacidade de se multiplicar no sistema, através da infecção de arquivos. Muitos vírus procuram infectar arquivos importantes do sistema operacional, fato que dificulta sua remoção sem a vacina adequada.

***Cavalo de Tróia*:** os *trojans* são alterações de programas conhecidos (daí a origem do nome, referência à estratégia de guerra de Tróia). Os propósitos são variados, mas em sua maioria buscam o roubo de informações através do monitoramento da atividade do teclado (*keylogging*) ou da tela do usuários (*screenlogging*).

Alguns tipos de cavalo-de-troia podem acessar a imagem da *webcam* do usuário, a fim de prover informações para crimes como divulgação de imagem pessoal sem autorização ou até assalto a residência.

***Backdoor*:** este tipo de *malware* busca prover acesso remoto à máquina. Uma vez

instalado no computador da vítima, ele é capaz de verificar e criar vulnerabilidades internas a fim de permitir que o *hacker* acesse de forma irrestrita o sistema.

Exploit: *exploits* são programas desenvolvidos para explorar vulnerabilidades específicas. A falha mais comum é o *buffer overflow*, presente na maioria dos *softwares*. A exploração por *buffer overflow* consiste em alterar os dados presentes na memória do computador; o programa - com falha - ocasionalmente executa o código malicioso inserido pelo invasor.

Ataque de Senhas: o objetivo deste ataque é descobrir a senha do usuário e ter seus privilégios no sistema. A quebra da senha, como é comumente denominada, pode ser realizada de forma manual através da investigação de possíveis palavras-chave relacionadas ao usuário (*e.g.* data de nascimento); ou pode ser feita de forma automatizada, utilizando força bruta (*i.e.* tentar todas as combinações possíveis) ou ataque de dicionário (*i.e.* lista de palavras possíveis para servir de base para as tentativas de quebra).

Phishing: é uma das técnicas de Engenharia Social, que visa explorar a confiança ou ingenuidade das pessoas para obter informações sigilosas (MITNICK K. L.; SIMON, 2006).

A técnica de *phishing*, especificamente, faz uso de mensagens falsas enviadas através de mensagens de e-mail ou anúncios, induzindo os usuários a acessarem *websites* falsos ou fornecerem seus dados sigilosos. As páginas falsas possuem a mesma aparência e funcionalidade da original, enganando usuários leigos.

SPAM: são mensagens de e-mail não solicitadas enviadas para diversos usuários. O alto fluxo de mensagens SPAM representa, além de um incômodo aos usuários, uma sobrecarga nos servidores de e-mail do mundo. Em 2014, os SPAMs representam mais de 65% de todo o tráfego de mensagens eletrônicas de toda a Internet (SHCHERBAKOVA, 2014).

A maioria das mensagens não solicitadas são consideradas um vetor para propagar outros tipos de ameaças, como *malwares* e *phishing*. Entretanto, há outras práticas de SPAM, como:

- marketing de produtos com ofertas - falsas ou não - para clientes que não solicitaram o envio de tais mensagens;

- propagação de notícias falsas que visam disseminar ódio, revolta ou preocupação (denominadas *hoaxes*);
- falsa promessa de benefícios para aqueles que compartilharem a mensagem (denominadas correntes), muito comum em redes sociais.

Intrusão (R2L): uma intrusão, ou R2L (Remoto para Local) consiste em obter acesso ilícito a um sistema ou serviço. O objetivo de uma intrusão cabe ao invasor; uma vez que o cibercriminoso tem acesso ao sistema, ele pode extrair informações, alterar ou apagar dados.

Como o objetivo do *cracker* mais comum é o reconhecimento pelo ataque bem sucedido, em intrusões a *websites* é comum a alteração do conteúdo disponibilizado *online*, como uma espécie de *graffiti*. Esta técnica é denominada *defacement*.

A Figura 3 mostra o conteúdo do *website* americano de regulação de direitos autorais alterado[7] pelo grupo de crackers *Anonymous*.

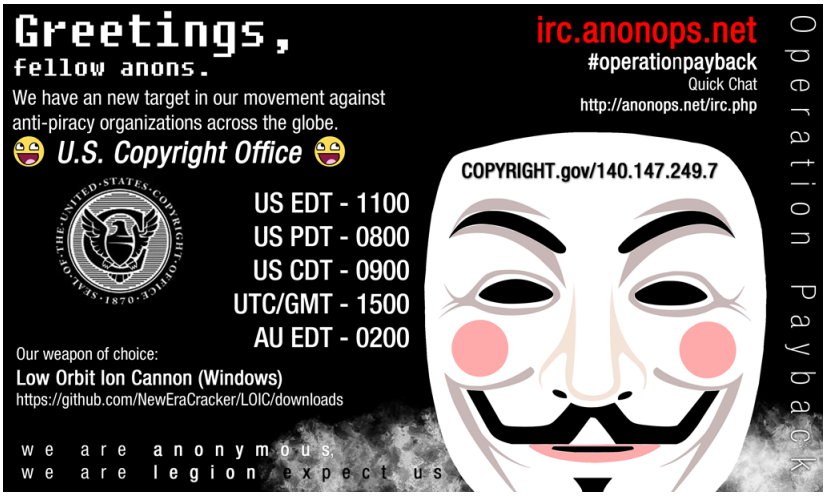

Figura 3: *Defacement* do *website* de Direitos Autorais Americano

Após empresas fonográficas conseguirem um mandado de fechamento de diversos sites de compartilhamento de conteúdo pirata, a comunidade *black hacker* se organizou com a Operação Payback, que visava ataques DDoS e *defacement* nos próprios sites das empresas fonográficas e de regulamento de direitos autorais, como a MPAA[8] e o US Copyright Group[9].

Escala de Privilégio (U2R): a escala de privilégios, ou U2R (Usuário para Administrador), esta intimamente associada à uma intrusão. Logo após o invasor obter

[7]Imagem extraída do *website* Anonymous: https://anonops.com/
[8]Motion Picture Association of America: http://www.mpaa.org/
[9]http://www.dunlapweaver.com/

acesso à maquina-alvo, é muito comum este acesso ter restrições de uso, por se tratar de uma conta de usuário. Sendo assim, o invasor busca obter acesso privilegiado, denominado *root* ou administrador, para ter permissão ilimitada no sistema invadido e realizar, então, seu objetivo, que pode ser qualquer um dos citados acima.

2.2 Sistemas de Detecção de Intrusos

Um sistema detector de intrusos (IDS) é um programa capaz de identificar ataques em uma rede ou sistema através da análise de pacotes de dados extraídos da rede. Não apenas as informações presentes no pacotes (como protocolo, destino e origem) são analisadas, mas também o seu contexto e comportamento (como quantidade de tentativas de login e frequência de destino). Sua eficiência está ligada diretamente à sua velocidade e precisão, fatores que permitem a reação do administrador ou de técnicas de prevenção (WEBER, 2000).

Atualmente há diversos IDSs à disposição no mercado, cada um destes possui um perfil de aplicação diferente (velocidade e segurança) e pode ser instalado em pontos diferentes da rede (interna ou externamente ao *firewall*) a fim de prover o desempenho desejado.

Dentre os NIDSs conhecidos, o que apresentou melhor desempenho, de acordo com os critérios de BIERMANN E.; CLOETE (2001), são aqueles baseados no motor de detecção Snort (ROESCH, 2013). A vantagem do Snort em relação aos concorrentes é sua caraterística *open-source*[10], que lhe permite a colaboração de uma comunidade com mais de 300.000 usuários em favor da constante atualização para as ameaças mais recentes. O IDS Sourcefire, que é baseado no Snort, foi considerado o melhor sistema de prevenção de intrusos de 2011, de acordo com Magazine (2011).

Outro IDS largamente utilizado é o desenvolvido pela Cisco. Embora não tenha uma comunidade aberta como o Snort, o motor de detecção da Cisco proporciona alta adaptabilidade e escalabilidade, se tornando um dos 6 sistemas mais utilizados atualmente (INSTITUTE, 2013).

É possível identificar um ataque através do monitoramento contínuo da rede ou do comportamento dos usuários, categorizando os IDSs em:

NIDS: são posicionados na rede, próximos ao sistema que desejam proteger, e monitoram

[10]Modelo de desenvolvimento de *software* de licença livre, que incentiva o desenvolvimento colaborativo através do auxílio de uma comunidade.

o tráfego de pacotes em busca de dados de comportamento duvidoso como, por exemplo, pacotes que tentam sobrecarregar o sistema, caracterizando um ataque DoS.

Os IDSs dessa categoria são mais comumente utilizados devido a sua capacidade de centralização: podem ser colocados em um roteador a fim de proteger toda a rede interna. Entretanto, não são capazes de detectar todos os tipos de ataques, principalmente aqueles encriptados (UCHOA, 2009);

HIDS: são instalados dentro do sistema, e verificam anomalias através da análise de dados locais, como arquivos e *logs*. Os sistemas Tripwire[11] e AIDE[12], por exemplo, provêm soluções de detecção de anomalias, monitorando o sistema e alertando sobre alterações em determinados arquivos configurados pelo administrador de segurança.

Como mostrado por KIM J.; BENTLEY (2007), este tipo de IDS é usado com menor frequência por exigirem uma configuração específica para a máquina em que irão atuar. Por outro lado, são capazes de detectar ataques locais e exploração de privilégios.

Em relação à estrutura de detecção, um IDS pode ser baseado em reconhecimento de anomalias ou de mal uso, como mostrado no comparativo de funcionalidades feito por (BIERMANN E.; CLOETE, 2001).

Anomalias: o IDS é pré-configurado com o padrão de comportamento dos usuários do sistema ou da rede em que irá atuar. Dessa forma, qualquer atividade diferente daquelas pré-determinadas será considerada um ataque em potencial. Por exemplo, em uma rede em que todos os usuários executam tarefas comuns de escritório (*i.e.* digitar textos, planilha, navegador de internet), caso haja um acesso *root*[13] ao sistema repentinamente, o IDS pode considerar esta atividade uma invasão, provavelmente escala de privilégio[14].

Entretanto, este tipo de sistema apresenta muitos falsos positivos, uma vez que o comportamento das pessoas é dinâmico, muda de forma natural com o tempo, e o IDS segue rigorosamente as rotinas pré-determinadas.

[11] http://www.tripwire.com/
[12] http://aide.sourceforge.net/
[13] Acesso privilegiado ao sistema.
[14] Um usuário que possui, por padrão, acesso limitado ao sistema, obtém acesso irrestrito ao sistema, podendo comprometer sua integridade.

Mau Uso: os sistemas de detecção baseados no mal uso realizam o reconhecimento de ataques através de uma lista de ameaças conhecidas, como uma espécie de antivírus.

Esta abordagem diminui o número de falsos positivos, entretanto tem baixa adaptabilidade (AICKELIN U.; GREENSMITH, 2004), ou seja, apresenta dificuldade em detectar ameaças desconhecidas.

De modo geral, um IDS é constituído por três componentes: coleta, detecção e resposta, como ilustrado na Figura 4.

Figura 4: Componentes de um IDS

O módulo de *coleta* é responsável por monitorar o tráfego na rede ou de uma máquina específica, feitas através de um *sensor* capaz de adquirir dados que venham a ser relevantes para a detecção. Todos os dados coletados são armazenados em um *log de eventos*, para efeitos de registro.

O módulo de *detecção*, por sua vez, consome os dados armazenados no *log de eventos*. Através de um algoritmo de detecção, a *unidade de análise* busca reconhecer padrões de ataques ou mal comportamento, baseando-se em uma política pré-determinada. Os algoritmos de detecção podem ser das formas mais variadas possíveis, as mais comuns são de detecção de assinaturas ou trechos de texto (COUTINHO, 2007).

Por fim, o componente de *resposta* implementa, através da *unidade de resposta*, algum tipo de alerta baseado na classificação fornecida pela unidade de detecção (*e.g.* ataque, comportamento anômalo, atividade normal). Ao contrário de um *firewall*, a resposta de um IDS não inclui nenhuma medida preventiva, limitando-se a um alerta para o administrador da rede.

3 Metodologia

Este trabalho utiliza técnicas de detecção de colisão para agrupar os dados de modo a reduzir os cálculos necessários para o treinamento do classificador supervisionado. Neste capítulo, estas técnicas são apresentadas em detalhes bem como as metodologias de avaliação de detectores de intrusos que foram utilizadas na avaliação do sistema desenvolvido.

3.1 Volume Delimitador

Quando se tem diversos objetos em um determinado espaço, surge a necessidade de identificar a colisão entre os mesmos, especialmente na área de jogos (BOURG, 2002) . Considera-se colisão entre 2 objetos A e B quando o objeto B encontra-se dentro do espaço limitado por A, conforme Figura 5.

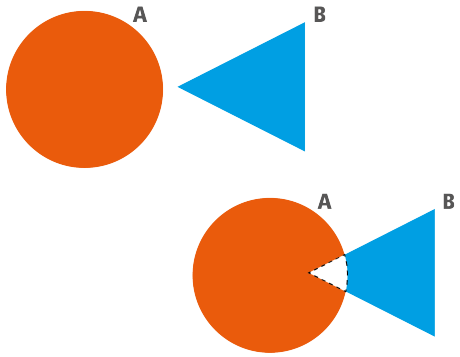

Figura 5: Colisão entre Objetos

Entretanto, analisar cada ponto que compõe as extremidades do objeto - como ocorre na técnica utilizada pelo Mesh Collider (ENGINE, 2009) - é um trabalho custoso computacionalmente, tornando-se um gargalo em aplicações de tempo-real como jogos e simulações. Por isso, há técnicas não exatas de detecção de colisões, mas que oferecem a velocidade desejada por determinadas aplicações.

Estas técnicas buscam simplificar o objeto da forma mais rápida possível, em detrimento da precisão da colisão. Como elas comumente apresentam falsas colisões, são

utilizadas na etapa inicial de um longo processo de detecção, como proposto por Sulaiman e Bade (2012).

O Volume Delimitador, ou BV, proposta por Gottschalk (2000), é uma técnica de agrupamento de objetos realizada através da criação de um volume ao redor da estrutura de cada objeto, absorvendo todos os pontos pertinentes a cada objeto. Estes volumes podem ser de qualquer natureza; a seguir serão detalhados os principais tipos de volumes delimitadores.

3.1.1 Esfera Delimitadora

A Esfera Delimitadora, ou SBV, cria esferas de menor raio possível ao redor de cada objeto no plano (GOTTSCHALK, 2000). A detecção de colisões no SBV é feita através da relação entre os raios e a distância entre os centros de cada par de esferas, como mostrada na Equação 3.1.

$$\begin{aligned} |r_a| + |r_b| \geq |d_{ab}| \quad &\rightarrow \quad \text{A e B não colidem} \\ |r_a| + |r_b| < |d_{ab}| \quad &\rightarrow \quad \text{A e B colidem} \end{aligned} \tag{3.1}$$

Onde:

r_a: raio da esfera A

r_b: raio da esfera B

d_{ab}: distância entre os centros das esferas A e B

Ao utilizar apenas uma soma dos raios das esferas, o tempo para detectar uma colisão diminui drasticamente. Entretanto, tal generalização do objeto em um formato de esfera apresenta uma alta possibilidade de falsas colisões (GOTTSCHALK, 2000). Uma falsa colisão ocorre quando a técnica indica a presença de uma colisão baseada no volume gerado, mas os objetos reais não se colidem, conforme a Figura 6.

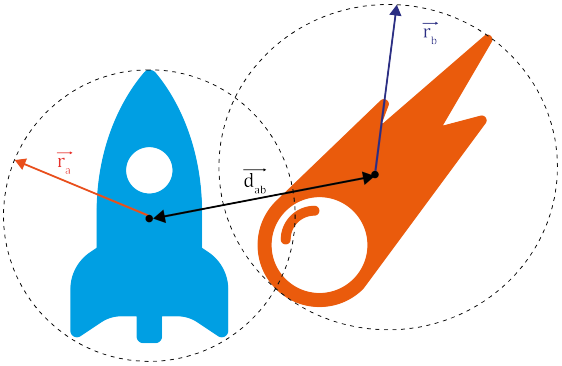

Figura 6: Falsa Colisão Detectada pelo SBV

3.1.2 Caixa Delimitadora Alinhada aos Eixos Globais

A Caixa Delimitadora Alinhada aos Eixos Globais, ou AABB (Axis Aligned Bounding Box), opta pela criação de caixas com lados sempre paralelos aos eixos globais ao redor de cada objeto, como pode ser visto na Figura 7. Nesse caso, a detecção de colisão é feita individualmente para cada eixo através da Equação 3.2, calculando da distância entre o centro e a borda da caixa, denominado alcance (reach).

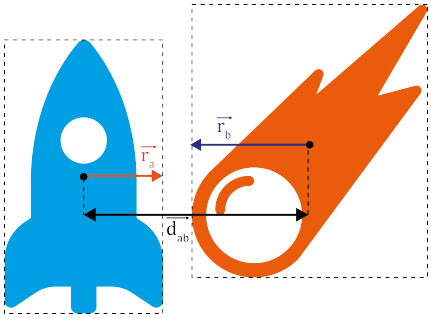

Figura 7: Detecção de Colisão pelo AABB

$$\begin{aligned} |r_a^i| + |r_b^i| \geq |d_{ab}^i| \quad &\rightarrow \quad \text{A e B não colidem} \\ |r_a^i| + |r_b^i| < |d_{ab}^i| \quad &\rightarrow \quad \text{A e B colidem} \end{aligned} \tag{3.2}$$

Onde:

$r_i = centro(i) - min(i)$

r_a^i: distância centro-face da caixa A no eixo i

r_b^i: distância centro-face da caixa B no eixo i

d_{ab}^i: distância centro-centro das caixas A e B no eixo i

O processo de detecção de colisão neste caso abusa do conveniente alinhamento das laterais da caixa com os eixos globais, entretanto é necessário analisar todos os eixos existentes a fim de fornecer o status de colisão das caixas. Este processo iterativo pode ser visto na Figura 8.

3.2 Classificação de Ameaças

Ao contrário da maioria dos sistemas de classificação, que definem apenas acertos e erros, em um sistema de detecção de intrusos (IDS) é interessante distinguir os resultados da classificação em 4 tipos, definidos por Macmillan (2004) na Tabela 1.

Classificações consideradas Verdadeiro Positivo ou Verdadeiro Negativo representam

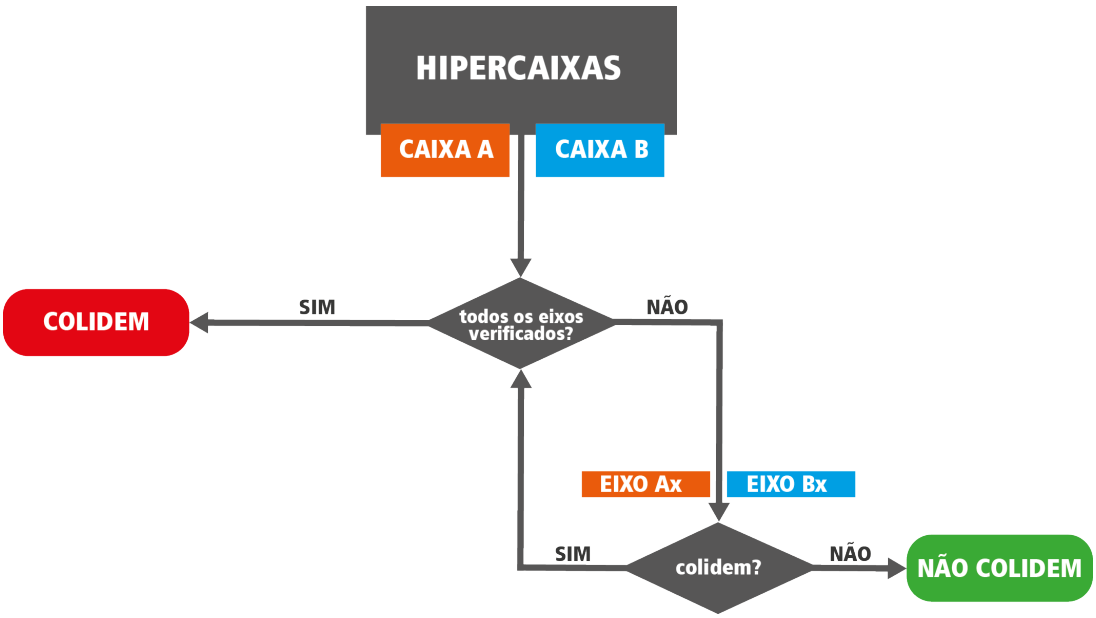

Figura 8: Fluxograma de Funcionamento do AABB

Tabela 1: Tabela de Predição

		Ameaça	
		Ausente	**Presente**
Predição	**Positiva**	Falso Positivo	Verdadeiro Positivo
	Negativa	Verdadeiro Negativo	Falso Negativo

a capacidade do sistema de identificar corretamente as informações, sejam elas ameaças ou normais, respectivamente.

Falsos Positivos são dados inofensivos erroneamente considerados ameaça. Um alto número de falsos positivos em um IDS não representa nenhuma dano no sistema, mas pode se tornar inconveniente ao bloquear informações normais. Falsos Negativos, por sua vez, são ameaças que são autorizadas pelo IDS, podendo causar dano ao sistema.

Sendo assim, um IDS pode ser configurado de forma a se adaptar ao ambiente em questão: pode ter seu foco na segurança na sistema minimizando os falsos negativos; ou pode também buscar minimizar os falsos positivos, garantindo velocidade ao sistema ao evitar o bloqueio de atividades legítimas.

4 Desenvolvimento

Optou-se pelo algoritmo de caixas alinhadas em detrimento das técnica de esferas delimitadoras por ser mais versátil e não apresentar tantas ocorrências de falsa colisão.

4.1 Modelagem

A primeira etapa do processo de criação do IDS consiste em treinar o classificador com amostras rotuladas[1] da base de dados. Em um ambiente real, é necessário realizar este treinamento uma única vez, o que pode ser feito de forma *offline*[2]. Sendo assim, o custo computacional desta etapa não é crucial para o bom desempenho do sistema.

4.1.1 Representação dos Pacotes

Os parâmetros podem ser contínuos - valores numéricos dentre um valor mínimo e máximo (*e.g. bytes received*), ou discretos - números ou textos que não representam uma sequência (*e.g. tcp, udp*). Sabendo disso, surge a necessidade de representar cada pacote como um ponto no plano cartesiano, no qual cada coordenada é definida por um parâmetro do pacote.

Uma vez que a técnica AABB manipula exclusivamente dados numéricos (GOTTSCHALK, 2000), os dados não-numéricos passam por um processo de expansão antes de serem adicionados ao hiperplano: representando cada palavra por um número inteiro único maior do que zero. Este processo é realizado através de uma função de mapeamento baseada em dicionário: cada palavra única recebe um índice inteiro para representá-la. Dessa forma, mesmo os parâmetros formados por palavras (*e.g. tcp, udp*) são capazes de representar uma coordenada do ponto.

A Figura 9 exemplifica uma situação em que 15 pacotes, possuindo 2 parâmetros

[1] Rotulação consiste em classificar de forma manual e precisa um determinado dado.
[2] Sem interação direta com rede de dados, como a captura de pacotes em tempo real.

cada, foram convertidos para pontos em um plano cartesiano. Estes pontos foram gerados aleatoriamente e fazem parte de um banco de dados criado para exemplificar a metodologia deste trabalho. Mais informações sobre o banco serão apresentadas na subseção 5.1.1.

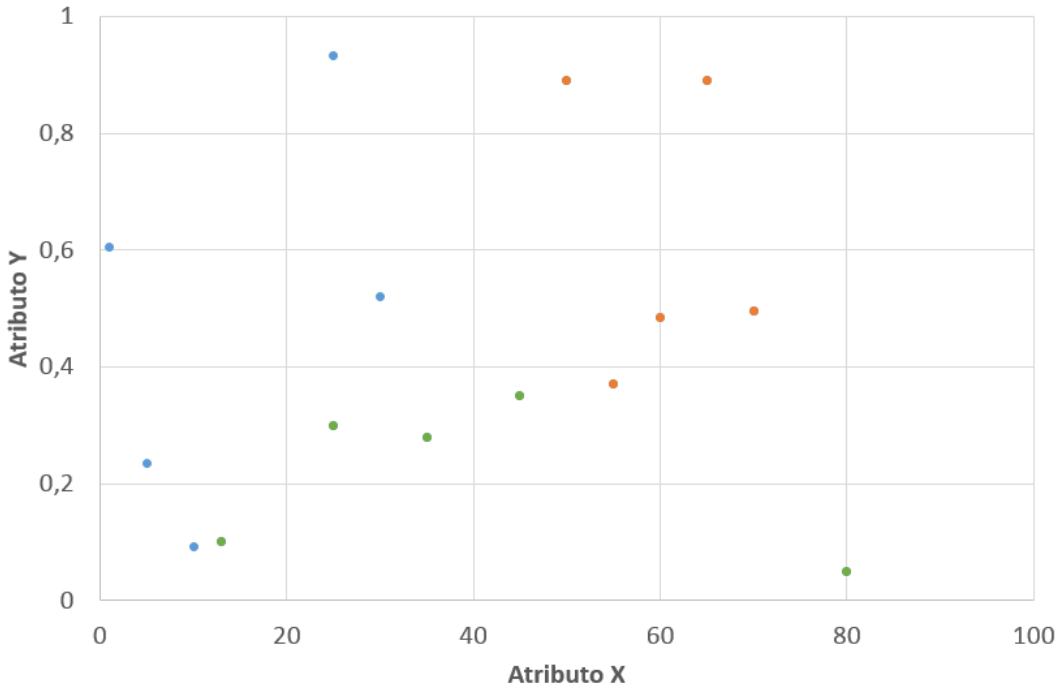

Figura 9: Exemplo de Representação de Pacotes em Plano Cartesiano

O tipo do pacote, por sua vez, mantém-se um dado discreto, pois não é utilizado como coordenada do ponto. O tipo de pacote irá definir, posteriormente, a classe do ponto, utilizada na criação de caixas a seguir.

4.1.2 Criação de Hipercaixas

Com o espaço cartesiano populado de pacotes (agora representados por pontos), o sistema agrupa todos os pontos de uma determinada classe em uma mesma estrutura, denominada hipercaixa. Como pode ser visto na Figura 10, os mesmos 15 pacotes da Figura 9 agora pertencem a três caixas, considerando que os dados pertencem a três classes, representadas nas cores verde, amarelo e azul.

Esta hipercaixa oferece diversas informações sobre o conjunto de pontos que a compõem, como média, centro geométrico, largura e desvio padrão de cada eixo especificamente, atributos estes que colaboram para o bom desempenho do sistema.

Nesse momento, a técnica AABB se encarrega de comparar as caixas e informar as colisões existentes, fornecendo as informações necessárias para as etapas seguintes: os

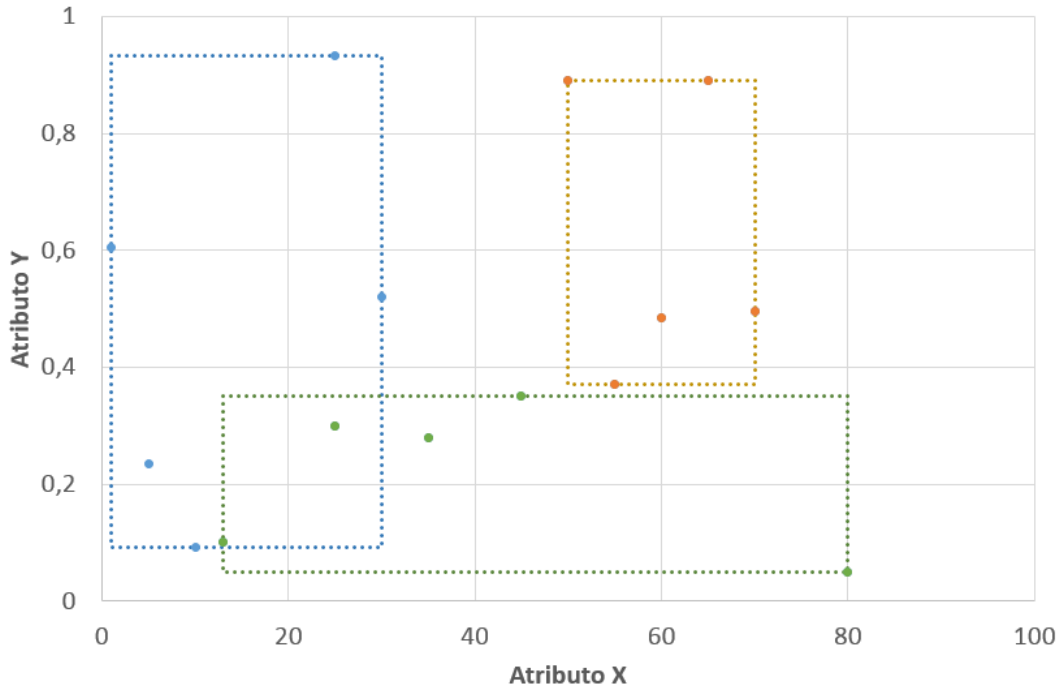

Figura 10: Exemplo de Criação de Hipercaixas

processo de seleção e particionamento.

4.1.3 Processo de Seleção

Em cada iteração, apenas uma caixa é particionada, portanto é necessário classificar as caixas que colidem com outras, criando um ranking, para servir de critério de seleção. O fator de ranqueamento de uma caixa é calculado através da esparsidade de seus eixos: uma relação entre largura e desvio padrão dos valores de cada eixo, conforme a Equação 4.1

$$s = \sigma * L \tag{4.1}$$

Onde:
σ: desvio padrão
L: comprimento

A caixa que será escolhida para ser particionada na interação será aquela que apresentar o maior fator de ranqueamento dentre todos seus eixos. Este processo de seleção proposto permite valorizar as caixas que têm os pontos mais distribuídos e/ou com os eixos de maior comprimento. Estas caixas apresentam as características ideais (GOTTSCHALK, 2000) para a divisão.

4.1.4 Particionamento

Uma vez selecionada, é necessário definir um valor em um dos eixos da caixa para servir de ponto de corte, a fim de gerar novas caixas. O objetivo da divisão das caixas é criar as menores sub caixas possíveis, fato que colabora para evitar as colisões existentes.

É muito comum realizar a quebra da caixa no ponto central do eixo mais longo (GOTTSCHALK, 2000), entretanto neste trabalho optou-se por uma técnica de quebra diferente, que faz uso do valor de dispersão, por apresentar resultados empíricos melhores.

Em um primeiro momento, retirar-se-á o ruído da caixa no eixo mais esparso, removendo apenas os pontos que se diferem muito do valor médio (centro de massa). Em sistemas com distribuição normal, qualquer dado que possua valor fora do intervalo de 3-sigma pode ser considerado como ruído ou como um ponto mal classificado. Deste modo os pontos ideais para o corte são aqueles que separam os pontos que podem ser considerados como ruído daqueles que estão no intervalo de 6-sigma (média aritmética +- 3 desvios padrão). Essa abordagem garante que 99,8% dos pontos mais próximos do valor médio serão mantidos (MOOD F. A. GRAYBILL, 1974), como visto na Figura 11.

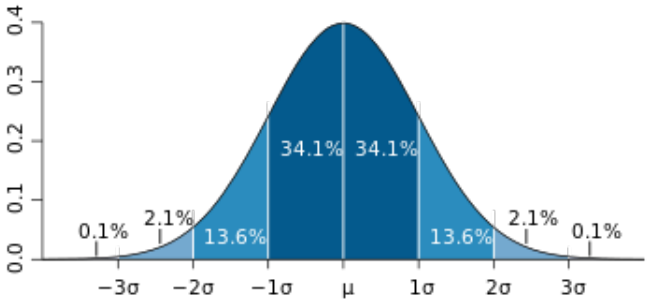

Figura 11: Gráfico de Distribuição Normal

O cálculo dos pontos de remoção de ruído pode fornecer 3 novas caixas, ao invés de 2 caixas como é comumente realizado com esta técnica. Esta característica colabora para a conversão mais acelerada do sistema, uma vez que remove grande parte do ruído existente em uma única interação.

Ocasionalmente, o valor de 6-sigma pode fornecer ambos os pontos fora das limitações da caixa. Nesse caso, assume-se que a caixa não possui ruído, efetuando o corte no centro de massa do eixo em questão, o que garante[3] a geração de pelo menos 2 novas caixas.

Em geral, só há necessidade de remover o ruído apenas uma vez para cada eixo de cada caixa, portanto há um controle dos eixos já analisados. Caso o eixo já tenha tido

[3]Válido para caixas com, no mínimo, 2 pontos de valores distintos no eixo analisado.

seu ruído removido, o ponto de corte é simplificado para o centro de massa do eixo em todas as iterações posteriores. O fluxograma da etapa de particionamento é mostrado na Figura 12.

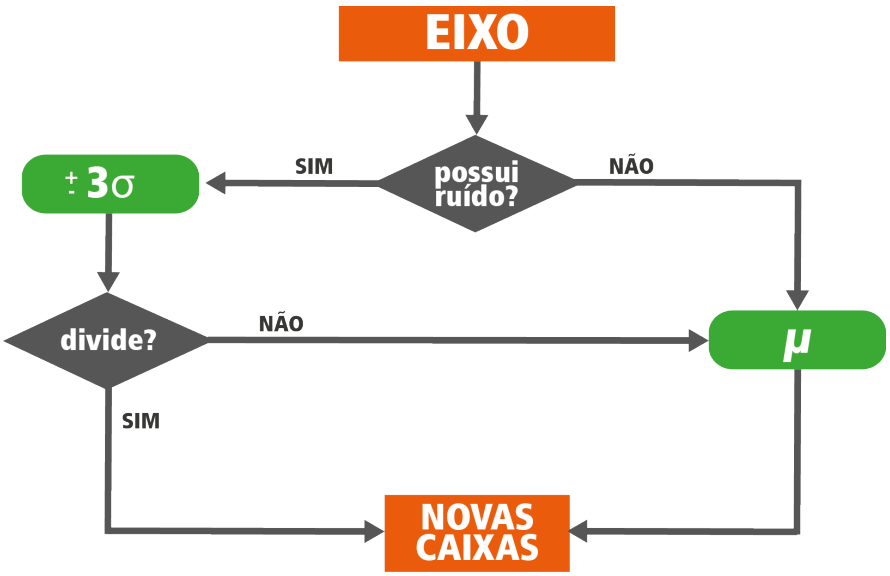

Figura 12: Etapa de Particonamento

4.2 Sistema de Regras

A última etapa do processo de criação do IDS é a geração de um sistema de regras. O objetivo é transformar as hipercaixas obtidas anteriormente em regras de classificação para novos pacotes.

A utilização de caixas para criar o sistema de regras possui a vantagem de simplificar os pontos utilizados para testes, ou seja, a quantidade de pontos não influencia na velocidade de detecção, uma vez que estes estão agrupados em caixas, que são as responsáveis pela detecção do padrão. Apenas a quantidade de caixas e parâmetros que impactam na velocidade de detecção, reforçando a necessidade de uma boa técnica de quebra de caixas que minimize as divisões e mantenha a separação entre caixas distintas.

O sistema de inferência adotado é numérico, e não lógico. Embora o sistema lógico seja mais rápido e preciso, o sistema numérico fornece a capacidade de generalização, que é muito importante para a detecção de intrusos, pois possibilita a classificação de ameaças sofisticadas que se diferem das convencionais.

Cada hipercaixa gera 1 regra composta por n condições, onde n é quantidade de características (parâmetros) da base. Para realizar a classificação é necessário que o

ponto seja testado contra todas as regras. O processo de classificação de um novo pacote de rede segue as seguintes etapas:

1. O novo pacote é representado como um ponto P conforme modelo adotado na primeira etapa(Subseção 4.1.1);
2. Calcula-se a distância relativa $d(P_A)$ [4] do ponto P até cada uma das faces de cada hipercaixa, analisando todos os n eixos;
3. Procura-se a hipercaixa que contenha o ponto P, ou na ausência desta, a hipercaixa mais próxima.

A distância $d(P_A)$ é calculada para todas as hipercaixas e pode ser obtida através da Equação 4.2. A equação proposta consiste em uma simplificação da equação de Pitágoras, provendo a velocidade desejadas para o sistema na classificação dos pontos.

$$d(P_A) = \sum_{i=0}^{n} \left(\begin{cases} 0 & \text{se } Min_i > P_i > Max_i \\ (|P_i - Centro_{Ai}| - \frac{L_{Ai}}{2})^2 & \text{para outros casos} \end{cases} \right) \quad (4.2)$$

Onde:

P_i: valor do ponto P no eixo i

$Centro_{Ai}$: valor do centro da caixa A no eixo i

L_{Ai}: comprimento da lateral da caixa A no eixo i

Na Figura 13 é apresentado graficamente a distância entre um determinado ponto P até a face mais próxima de uma caixa A qualquer.

Para pontos que se encontram dentro dos limites das faces de uma hipercaixa, em pelo menos um eixo, é possível ignorar o cálculo de distância normal de Pitágoras neste eixo e adotar apenas a distância entre o ponto e a face da caixa, como ilustrado na Figura 14.

Para se realizar a classificação, compara-se as distâncias relativas obtidas entre o ponto P e cada uma das faces das hipercaixas. Se houver um valor de $d(P_A)$ igual a zero, o ponto P encontra-se dentro da hipercaixa A, sendo possível afirmar que o pacote possui a mesma classe dos demais pontos desta. Entretanto, se $d(P_A)$ for diferente de zero para todas as faces das hipercaixas, o ponto P não pertence a nenhuma classe *a priori*. Neste caso, infere-se a classe do ponto por proximidade, assumindo a caixa de menor distância.

[4] Como é realizada a comparação das distâncias, não é necessário realizar o cálculo exato através do Teorema de Pitágoras.

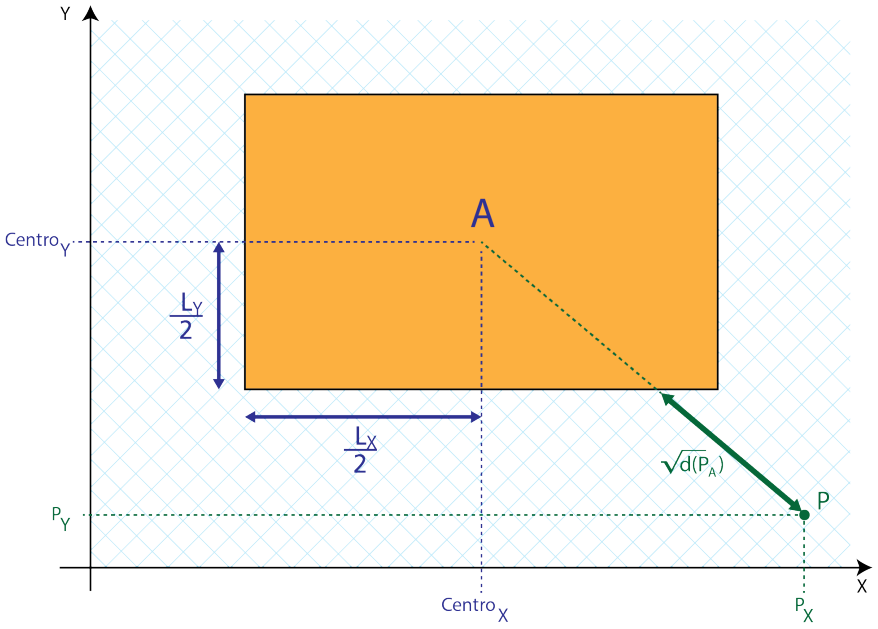

Figura 13: Exemplo de Medidas de Distância

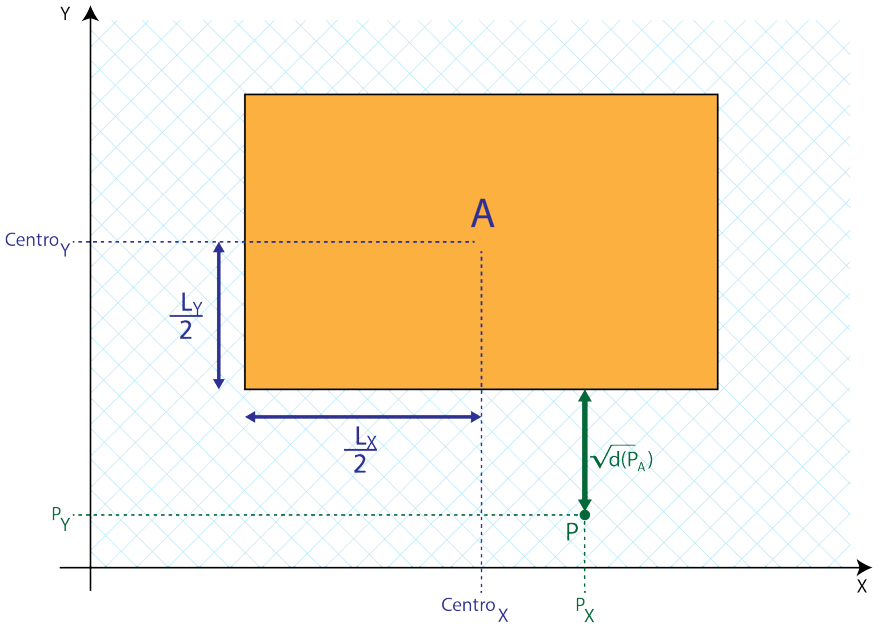

Figura 14: Distância Ponto-Face de Ponto Lateral Externo

4.3 Considerações Finais

A utilização do sistema de inferência numérico é um grande diferencial para o classificador, proporcionando grande capacidade de generalização ao sistema. Em um ambiente real, em que o sistema deve ser capaz de lidar com intrusos sofisticados, a capacidade de inferir a classe de um pacote que se difere daqueles utilizados para o treinamento do IDS é de extrema importância para garantir a adaptabilidade do sistema.

Embora o processo de classificação possa ser demorado para sistemas com um grande número de caixas, esta etapa é completamente passível de paralelização. A adoção de sistemas computacionais multi-tarefa pode tornar o custo computacional da classificação de pacotes irrisório, mesmo para grande quantidade de regras (hipercaixas).

5 Experimentos

5.1 Bases de Dados Utilizadas

A fim de validar o sistema desenvolvido como um classificador de propósito geral, foram utilizados diversas bases de dados rotulados, frequentemente adotadas em trabalhos da área. Para fins didáticos foi criada uma base exemplo para ser usada como demonstração visual do processo de treinamento.

5.1.1 Base Exemplo

A base de dados fictícia foi criada com 3 (três) classes contendo 5 (cinco) pontos cada. Cada ponto possui apenas 2 atributos, a fim de facilitar a ilustração em um plano cartesiano de 2 (duas) dimensões. Os valores de cada uma das instâncias dessa base é mostrada na Tabela 2

Tabela 2: Base de Dados Fictícia

Atributo 1	**Atributo 2**	**Classe**
10	0,091	A
25	0,93	A
10	0,23	A
1	0,60	A
30	0,52	A
50	0,89	B
55	0,37	B
60	0,48	B
65	0,89	B
70	0,49	B
25	0,3	C
35	0,28	C
80	0,05	C
13	0,1	C
45	0,35	C

5.1.2 Iris

A base de dados *Iris* (FISHER, 1988) é uma das mais utilizadas em classificadores, devido a sua baixa complexidade.

O objetivo é identificar a espécie de uma flor com base em 4 características: comprimento e largura da pétala, comprimento e largura da sépala. Há apenas 3 espécies de flores possíveis: Iris-Setosa, Iris-Virgínica e Iris-Versicolor.

5.1.3 Glass Identification

A base de dados *Glass* (GERMAN, 1987) propõe a classificação de vidros através da análise de 9 atributos: índice de refração, porcentagem de sódio, magnésio, alumínio, silício, potássio, cálcio, bário e ferro. O objetivo é definir o tipo de vidro dentre seis classes, de acordo com o propósito de uso, que vai desde fabricação de janelas até talheres.

Esta é uma das base de dados mais complexas, uma vez que não há separação definida entre as classes em nenhum dos atributos.

5.1.4 Wine

Esta base de dados (AEBERHARD, 1991) é focada na classificação de vinhos. São fornecidas 13 características químicas e físicas de uma amostra de vinho, todos valores contínuos. Deve-se determinar a qualidade do vinho analisado, dentre as 3 classes propostas.

5.1.5 Vowel

A base de dados *Vowel* (ROBINSON, 1989) busca classificar o som entre 10 tipos de vogais, segundo notação fonética internacional. As amostras foram extraídas da voz de 15 pessoas, contendo 11 atributos referentes ao sinal capturado.

5.1.6 Image Segmentation

A base de dados *Image Segmentation* (GROUP, 1990) possui diversas imagens aéreas de regiões. Cada pixel da imagem foi classificado manualmente de acordo com o material: pedra, céu, folhagem, cimento, janela, caminho e grama. O objetivo é determinar o tipo de região através de 19 características fornecidas a respeito do pixel e seu entorno.

5.1.7 Vehicle Silhouettes

A base de dados *Vehicle Silhouettes* (MOWFORTH; SHEPHERD, 1987) possui diversas imagens bidimensionais de quatro veículos: Opel[1], Saab[2], ônibus e vans. As imagens dos veículos foram extraídas de forma padronizada: mesma resolução e cores, sem ruídos. Destas imagens foram retiradas 18 características a respeito das dimensões e ângulos dos componentes externos dos veículos.

5.1.8 Cover Type

A base de dados *Cover Type* (BLACKARD, 1993) consiste em identificar o tipo de vegetação de cobre uma determinada região. São 6 tipos de vegetação conhecidas, e análise deve ser feita através de imagens 30x30m extraídas pelo Serviço Florestal dos Estados Unidos[3]. Cada secção da região possui 54 atributos, que vão desde tipo de solo até distância a focos de incêndios na região.

5.1.9 Car Evaluation

O objetivo da base de dados *Car Evaluation* (BOHANEC, 1997) é determinar o valor de um carro (4 classes) de acordo com suas características. São avaliados 6 atributos discretos sobre a composição do carro: quantidade de portas, segurança, tamanho do porta-malas, preço de custo e de manutenção, e lotação.

5.1.10 KDD99

A base de dados KDD99 (HETTICH; BAY, 1999), ou DARPA/KDD-99, inclui vários exemplos de ataques a redes de computadores, assim como pacotes de tráfego normal de rede, simulando um ambiente de rede militar. Cada pacote possui 41 características, incluindo o conteúdo do pacote e o protocolo utilizado.

Essa base foi desenvolvida pelo Programa e Avaliação de Detecção de Intrusos da DARPA, em 1998, possuindo aproximadamente cinco milhões de registros, dos mais diferentes tipos de ataques (YU Z.; TSAI, 2007).

[1]Fabricante automotiva alemã: http://www.opel.com/
[2]Fabricante automotiva suíça: http://www.saabcars.com/
[3]http://www.fs.fed.us/

5.2 Metodologia de Teste

Cada base de dados analisada é dividida em 2 partes: uma para treinamento e outra para testes. Os dados utilizados para o treinamento têm o atributo que identifica a classe de cada instância, a fim de permitir que o sistema AABB agrupe-os em caixas.

Os dados de teste, por sua vez, são apresentados ao sistema de regras (Seção 4.2) sem incluir a classe, para que, dessa forma, o sistema informe a qual classe a instância pertence. A seleção das instâncias que irão compor cada um dos grupos de treinamento e teste é feita de forma estocástica.

Diversos trabalhos na área de classificação apresentam os resultados obtidos a partir de uma seleção específica dos dados. Como os dados são selecionados aleatoriamente, o resultado pode variar para seleções diferentes de instâncias.

Os testes foram configurados de acordo com a metodologia *Ten Fold*, a mesma utilizada nos trabalhos de (CHIH WEI H.; CHIH, 2002) e (FUNG; MANGASARIAN, 2001), que consistem em dividir, de forma aleatória, o conjunto de dados em 10 partes iguais. Uma parte é selecionada para o teste do sistema, e outras nove partes são utilizadas para o treinamento (Figura 15).

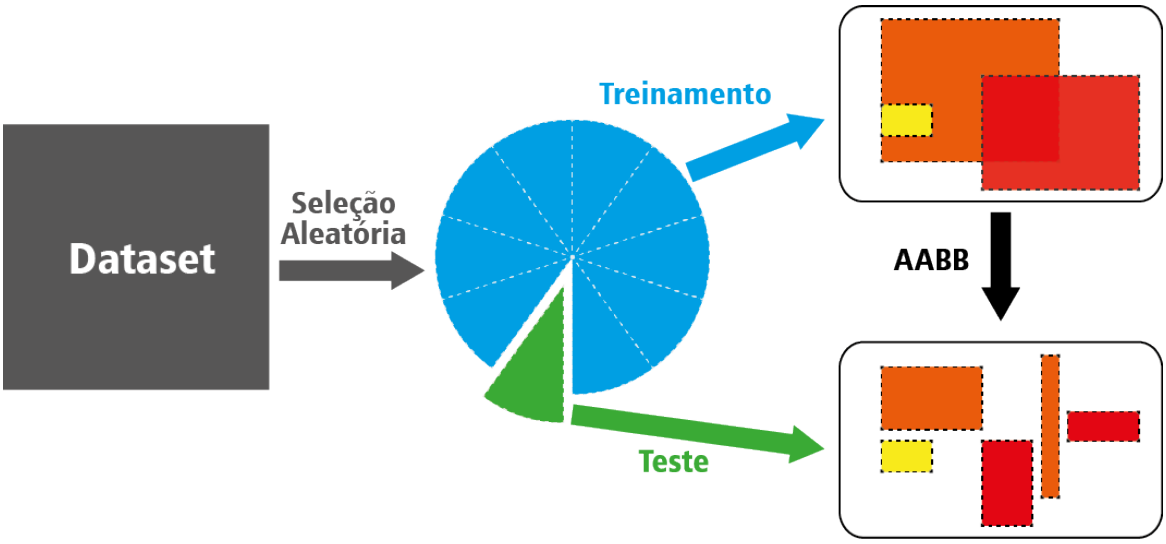

Figura 15: Divisão de Grupos de Treinamento e Teste

Posteriormente, escolhe-se outra parte para servir de teste, e as demais são utilizadas no treinamento. Este processo é realizado sucessivamente até que todas as 10 partes tenham sido utilizadas uma vez para o teste, como mostrado na Figura 16

Uma vez que as instâncias de cada uma das partes são selecionadas aleatoriamente, o processo *Ten Fold* é repetido até que se obtenha uma diferença mínima entre os resultados

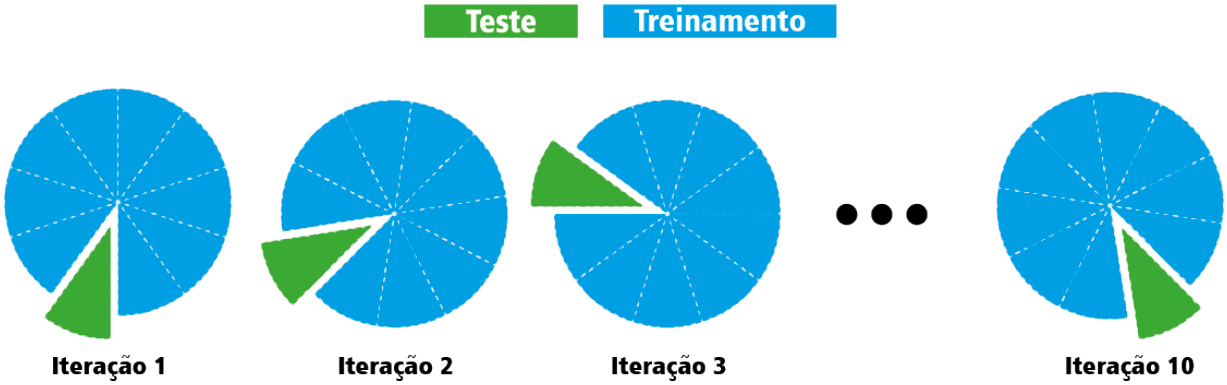

Figura 16: Metodologia Ten Fold

obtidos. Este critério de parada garante valor estatístico aos resultados, tornando-os praticamente independentes da amostra utilizada para teste e treinamento. O cálculo da diferença mínima entre os resultados do sistema é o Erro Padrão da Média (EVERITT, 2010), mostrado na Equação 5.1

$$EPM = \frac{s_n(a)}{\sqrt{n}} \tag{5.1}$$

Onde:

$s_n(a)$: desvio padrão das taxas de acerto (Subseção 5.2.1) até a iteração n

n: quantidade de repetições até a iteração n

5.2.1 Medidas Utilizadas para Avaliação

O desempenho da predição do sistema pode ser avaliado através de diversas métricas, como visto em (FAWCETT, 2013). A principal delas é a acurácia (Equação 5.2). O cálculo da acurácia de um classificador é obtido através da relação entre a quantidade de instâncias corretamente classificadas e o total de instância testadas. Além da acurácia, é comum o uso de 4 outras métricas ao se avaliar a possibilidade da aplicação do sistema classificador como um IDS.

Acurácia : capacidade de classificar corretamente uma instância

$$a = \frac{VP + VN}{n} \tag{5.2}$$

Onde:

n: total de instância testadas

Sensibilidade : capacidade de reconhecer uma ameaça quando uma se apresentar.

$$s = \frac{VP}{VP + FN} \tag{5.3}$$

Especificidade : capacidade de reconhecer um pacote normal quando um se apresentar.

$$e = \frac{VN}{VN + VP} \tag{5.4}$$

Valor Preditivo Positivo : possibilidade de reconhecer uma ameaça.

$$vpp = \frac{VP}{VP + FP} \tag{5.5}$$

Valor Preditivo Negativo : possibilidade de reconhecer um pacote normal.

$$vpn = \frac{VN}{VN + FN} \tag{5.6}$$

5.3 Resultados Obtidos

5.3.1 Base de Exemplo

Na Figura 17 é apresentada a representação gráfica do banco exemplo, com os pontos dessa base representados em um plano cartesiano de 2 coordenadas. Os pontos foram agrupados em suas caixas iniciais, delimitadas pelas linhas tracejadas, conforme a metodologia AABB.

Durante o treinamento do classificador supervisionado, foi detectada uma colisão entre as caixas, visualizada pela sobreposição das linhas tracejadas. Após a quebra da caixa azul (superior esquerda), são obtidas 2 novas caixas de mesma classe, como pode ser visto na Figura 18.

Após a primeira quebra o sistema verifica que não há mais colisões entre as caixas, encerrando a etapa de treinamento. Deste modo, para esta base de dados fictícia, o classificador resultou em 4 hipercaixas, que podem ser representadas pelas 8 comparações mostradas na Tabela 3.

Sabendo que o treinamento se inicia com uma quantidade de caixas no mínimo igual à quantidade de classes, qualquer quantidade de caixas acima deste número representa um esforço do algoritmo em realizar quebras para evitar colisões.

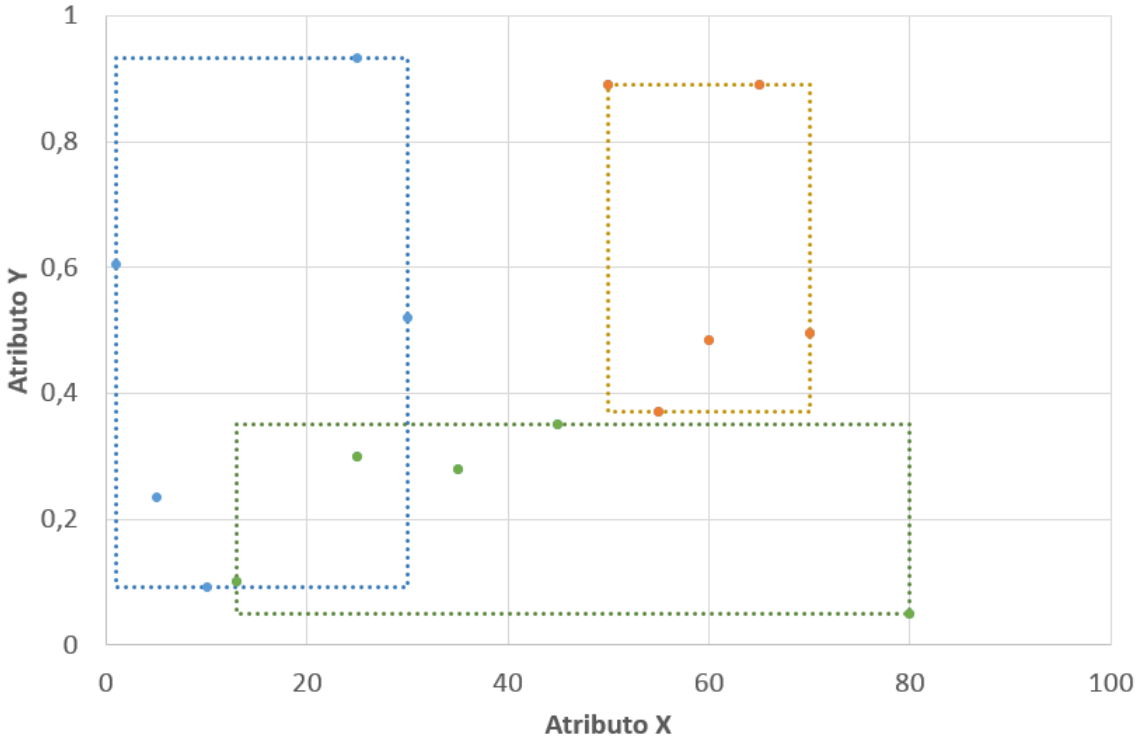

Figura 17: Hipercaixas Iniciais em um *dataset* exemplo

Tabela 3: Hipercaixas da Base de Dados Fictícia

Atributo 1		Atributo 2		Classe
Mínimo	Máximo	Mínimo	Máximo	
1	30	0,52	0,93	A
5	10	0,09	0,23	A
50	70	0,37	0,89	B
13	80	0,05	0,35	C

5.3.2 Bases de Dados de Propósito Geral

Após o treinamento do classificador supervisionado, foram analisadas a quantidade de hipercaixas obtidas e a quantidade de comparações necessárias para a classificação de cada instância. Na Tabela 4 é mostrado um comparativo das características do *dataset* (detalhadas na Seção 5.1) com a dimensão do classificador obtido.

Os resultados de acurácia obtidos foram comparados com os de 2 trabalhos de aplicação similar, os quais obtiveram resultados excelentes na literatura da área.

(FUNG; MANGASARIAN, 2001) : classificador criado através de SVM e testado com diversos bancos de dados, tendo resultados superiores nas bases *Iris* e *Wine*;

(HSU; LIN, 2002) : classificador criado através de SVM e testado com diversos bancos

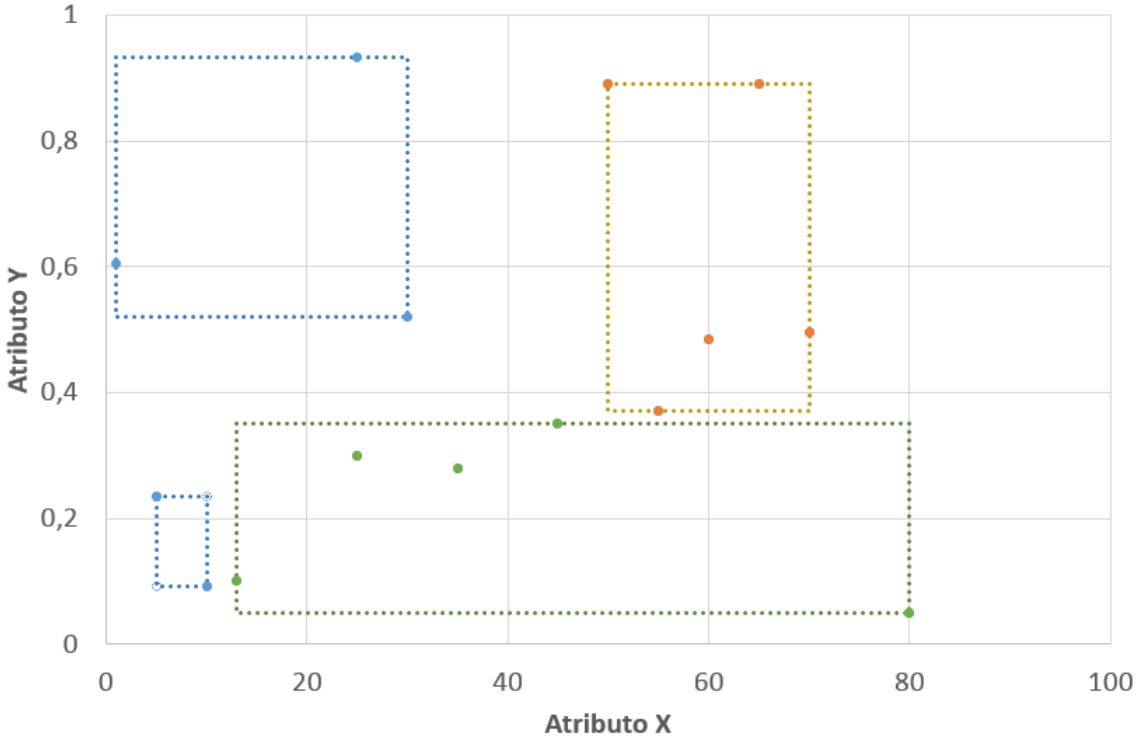

Figura 18: Hipercaixas Finais em um *dataset* exemplo

Tabela 4: Quantidade de Hipercaixas e Comparações obtidas

Dataset	**Classes**	**Atributos**	**Caixas**	**Comparações**
Iris	3	4	23	92
Glass	6	10	12	120
Wine	3	13	22	286
Vowel	11	10	123	1230
Segmentation	7	19	30	570
Vehicle	4	18	211	3798

de dados, tendo resultados superiores nas bases *Glass Identification*, *Vowel*, *Image Segmentation* e *Vehicle Sillhouetes*.

Tabela 5: Resultados Comparativos com Literatura

Dataset	**IDS AABB**	**Literatura**
Iris	$95,0\% \pm 0,6\%$	$98,7\%$
Glass	$98,5\% \pm 0,4\%$	$72,9\%$
Wine	$80,0\% \pm 1,7\%$	$100,0\%$
Vowel	$96,0\% \pm 0,6\%$	$98,8\%$
Segmentation	$62,5\% \pm 0,7\%$	$97,0\%$
Vehicle	$60,6\% \pm 0,9\%$	$82,2\%$

É importante ressaltar que os resultados presentes na literatura representam os va-

lores máximo de acurácia, obtidos através de seleções específicas das instâncias para o treinamento e teste, o que não representa o comportamento regular do sistema.

Entretanto, houve alguns casos em que o sistema desenvolvido não foi capaz de gerar um modelo. As bases de dados *Car Evaluation* e *Cover Type* possuem dados que não são separáveis facilmente em nenhum dos atributos, fator que dificulta a geração de planos separadores e, consequentemente, remoção da colisão das hipercaixas.

5.3.3 Base KDD99

Excepcionalmente para a base de dados KDD99, a metodologia *Ten Fold* não foi utilizada devido à quantidade enorme de dados presentes, aproximadamente 4,8 milhões de instâncias.

Para esta base foi utilizada a mesma metodologia aplicada na competição de detecção de intrusos KDD Cup (SIGKDD, 1999). O treinamento é realizado com uma amostra pré-determinada de 10% dos dados, e o teste do classificador é realizado com 100% dos dados.

Após o treinamento do classificador, foram analisadas a quantidade de hipercaixas obtidas, bem como o número de comparações necessário para inferir a classe de uma instância qualquer. Esses resultados são comparados com as características da base de dados na Tabela 6.

Tabela 6: Quantidade de Hipercaixas e Comparações obtidas com KDD99

Dataset	**Classes**	**Atributos**	**Caixas**	**Comparações**
KDD99	23	41	597	24477

Na Tabela 7, são mostrados os resultados de verdadeiros positivo e negativo, e falsos positivo e negativo: quatro métricas comumente utilizadas na avaliação de resultados binários de sistemas de detecção.

Tabela 7: Taxas de Predição

Métrica	**Taxa**
Verdadeiro Positivo	$80,122\%$
Verdadeiro Negativo	$19,853\%$
Falso Positivo	$0,006\%$
Falso Negativo	$0,019\%$

Também foram calculadas diversas métricas de análise de resultados dependentes daquelas descritas na tabela acima. Tais métricas, cujo significado individual é abordado na

Seção 3.2, são mostradas na Tabela 8.

Tabela 8: Métricas de Resultados da Base KDD99

Métrica	**Taxa**
Acurácia	99, 970%
Sensibilidade	99, 973%
Especificidade	99, 959%
Valor Preditivo Positivo	99, 990%
Valor Preditivo Negativo	99, 982%

O resultado da acurácia obtida é considerado muito alto para a aplicação de detecção de intrusos, bem próximo e até superior a outros trabalhos na área, como o IDS criado com SOM[4] de (UCHOA, 2009), acurácia de 96,5%, e o IDS de (KAYACIK H. G.; ZINCIR-HEYWOOD, 2007), com umas das maiores taxas de acerto da literatura: 99,5%.

Durante a competição oficial (SIGKDD, 1999) de classificação de intrusos, foi analisado o desempenho em classificar os pacotes em 5 categorias segundo a taxonomia de ataques proposta.

Na Tabela 9 é possível verificar o desempenho superior da técnica adotada neste trabalho em relação à equipe vencedora da competição.

Tabela 9: Comparação de Acurácia com a KDD Cup

Classe	**IDS AABB**	**KDD99 Cup**
Normal	99, 97%	99, 45%
DoS	99, 99%	83, 32%
U2R	100, 00%	97, 12%
R2L	100, 00%	13, 16%
Probe	96, 58%	8, 40%

5.4 Desempenho

A aplicação, desenvolvida na linguagem C# com .NET Framework 4.5.1, teve seus experimentos realizados em uma máquina Intel Core i7 3.2GHz (8 núcleos), na plataforma Microsoft Windows 8.1 64 bits.

O cálculo de tempo de execução e estatísticas do processo (*e.g.* número de caixas, métricas de avaliação de desempenho) de modelagem e teste do IDS foi realizado por ins-

[4]Mapa Auto Organizável, um tipo de rede neural de aprendizado não-supervisionado.

trumentação manual[5]. O consumo de memória e processamento é obtido estatisticamente por amostragem, cujas rotinas são executadas pelo *Profiler*[6] nativo do IDE Visual Studio[7] 2013.

Na Figura 19, é exibido temporalmente o consumo de processamento durante a execução dos testes com as bases *Iris*, *Glass* e *Wine*, sendo possível notar um padrão de comportamento com, no máximo, 40% de uso de CPU, mesmo com rotinas paralelizadas com 16 linhas de execução (*threads*).

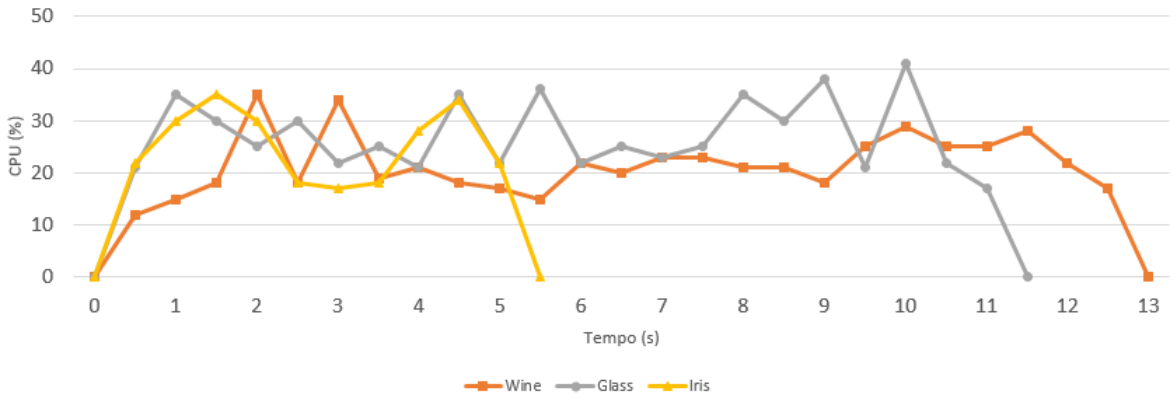

Figura 19: Consumo de CPU

O consumo de memória é diretamente proporcional à dimensão dos dados (número de instâncias e quantidade de parâmetros). Na Tabela 10 são mostrados o consumo de memória RAM máxima durante a execução do treinamento e teste do classificador com as bases *Iris*, *Glass*, *Wine* e *KDD99*, além do tempo de execução durante o processo de modelagem.

Tabela 10: Consumo de Memória e Tempo de Execução

Dataset	**Consumo de Memória (MB)**	**Tempo de Execução (s)**
Iris	109	3,2
Glass	115	3,2
Wine	173	4
KDD99	2915	86

[5]Adição, por parte do programador, de rotinas no código que efetua cálculos desejados a respeito da execução do programa.

[6]Ferramenta que possui rotinas de análise de código.

[7]http://www.visualstudio.com/

5.5 Considerações Finais

Em alguns casos, como as bases *Iris* e *Vowel*, o sistema desenvolvido apresentou taxas de acerto ligeiramente menores que outros classificadores. Entretanto, os resultados de acurácia são valores garantidos (médias) para os conjuntos de dados, independente de quais devem ser selecionados para treinamento.

Os resultados obtidos com bases de dados não relacionadas à detecção de intrusos tem como objetivo validar o sistema como um classificador e apontar em quais conjuntos de dados o algoritmo se mostra capaz de lidar com mais eficiência.

6 Conclusão

Este trabalho apresentou uma modificação na técnica de colisão de objetos com caixas alinhadas a fim de se adaptar a bases de dados com grandes quantidades de parâmetros numéricos. Além disso, foi desenvolvido um sistema de regras não-clássico que faz uso de hipercaixas delimitadoras a fim de aplicar na detecção de intrusos em redes de computadores.

O algoritmo de caixas alinhadas (AABB) foi escolhido devido ao baixo custo computacional envolvido para a detecção de colisões de volumes, em detrimento do algoritmo de caixas orientadas (OBB) que, apesar de conceitualmente fornecerem maior precisão, envolvem diversos cálculos geométricos para a criação de hipercaixas, fator que comprometeria o desempenho com abordagem *big data*.

Na maioria das bases utilizadas para avaliação técnica, o classificador obteve excelentes resultados, sem ter dificuldade em gerar o modelo. Uma das razões para o bom desempenho é a característica inerente em algumas bases de dados, as quais podem ter suas classes facilmente separadas com poucos hiperplanos, como no caso da *Iris* (ASUNCION A; NEWMAN, 2007).

O sistema desenvolvido apresentou melhores resultados com bases de dados contínuos. Estas bases costumam apresentar mais dificuldade para classificadores, como acontece com o algoritmo de conjuntos aproximados (PAWLAK, 1997), que é obrigado a discretizar os dados numéricos, ocasionando perda de informação.

As regras adotadas no trabalho são não-clássicas, pois utilizam distâncias euclidianas entre o ponto (um caso) até as hipercaixas (classes). A adoção desse sistema de inferência numérico apresenta diversas vantagens, principalmente pela capacidade de generalização inexistentes em outras técnicas baseada em regras.

O desempenho do sistema de forma geral atendeu às expectativas. O baixo consumo de memória e processamento, aliados à velocidade de detecção colaboram para a aplicação em um ambiente real. A possibilidade de paralelização das rotinas de modelagem e,

principalmente, de classificação auxiliam na escalabilidade do sistema para rede de tráfego intenso.

Os resultados obtidos durante os testes do classificador com a base de dados KDD99 comprovou a eficácia do classificador na classificação de intrusos em redes de computadores. Com taxas de acertos altíssimas, o sistema desenvolvido com a técnica proposta neste trabalho apresentou-se promissora não só como um classificador de propósito geral, mas como um sistema detector de intrusos.

6.1 Trabalhos Futuros

Para futuros trabalhos, espera-se avaliar a fundo o desempenho da técnica OBB na aplicação de detecção de intrusos. A etapa de geração de regras pode ser otimizada a fim de utilizar as hipercaixas com diversas orientações de forma rápida para não comprometer o desempenho do classificador em um ambiente real.

O cálculo do grau de complexidade também é um importante fator para validar a escalabilidade da técnica utilizada. Por fim, ainda é possível adicionar aprendizado ao sistema, para torná-lo ainda mais capaz de lidar com novos ataques e ameaças sofisticadas que venham a ser criadas após a modelagem.

Almeja-se, também, adaptar o sistema de detecção para integrá-lo como módulo de análise no motor do *software* Snort, a fim de possibilitar sua aplicação em um ambiente real. A criação de um *honeypot*[1] pode servir para simular uma rede sendo atacada por ameaças comuns nos dias de hoje.

[1]Rede de computadores conectados à Internet com diversas vulnerabilidades criadas propositalmente para atrair ataques.

Referências

AEBERHARD, S. *Wine recognition data*. [S.l.], 1991.

AICKELIN U.; GREENSMITH, J. T. J. Immune system approaches to intrusion detection. *Arti?cial Immune Systems, Third International Conference*, 2004.

ASUNCION A; NEWMAN, D. J. *UCI Machine Learning Repository*. [S.l.], 2007.

AV. *Malware Statistics over the Years*. [S.l.], 2014. Disponível em: <http: //www.av-test.org/en/statistics/malware/>.

BIERMANN E.; CLOETE, E. V. L. M. A comparison of intrusion detection systems. *Computers and Security*, v. 20, n. 8, p. 676–683, 2001. ISSN 0167-4048.

BILAR, D. Statistical structures: Fingerprinting malware for classification and analysis. In: WELLESLEY COLLEGE, MA. *Black Hat Briefings*. [S.l.], 2006. p. 50.

BLACKARD, J. A. *Forest Covertype data*. [S.l.], 1993.

BOHANEC, M. *Car Evaluation Database*. [S.l.], 1997.

BOURG, D. M. *Physics for Game Developers*. [S.l.]: O'Reilly, 2002.

CBSNEWS. Nsa surveillance exposed. *CBS*, 2014.

CHIH WEI H.; CHIH, J. L. A comparison of methods for multiclass support vector machines. *IEEE Transactions on Neural Networks*, v. 2, n. 13, 2002.

COUTINHO, M. P. *Detecção De Ataques Em Infra-Estruturas Críticas De Sistemas Elétricos De Potência Usando Técnicas Inteligentes*. Tese (Doutorado) — Universidade Federal De Itajubá, 2007.

CRUNCH, R. E. T. *Hack Attack: Sony Confirms PlayStation Network Outage Caused By 'External Intrusion'*. 2011. Disponível em: <http://techcrunch.com/2011/04/23/ hack-attack-sony-confirms-playstation-network-outage-caused-by-external-intrusion/>.

ECONOMIST, T. Data, data everywhere. *The Economist*, Fevereiro 2010.

ENGINE, U. G. *Mesh Collider*. [S.l.], 2009.

EVERITT, B. S. *The Cambridge Dictionary of Statistics*. 4. ed. [S.l.]: Cambridge University Press, 2010. 478 p.

FAWCETT, T. Roc graphs: Notes and practical considerations for data mining researchers. *Intelligent Enterprise Technologies Laboratory, HP Laboratories*, 2013.

FISHER, R. *Iris Plants Database*. [S.l.], 1988.

FUNG, G. M.; MANGASARIAN, O. L. *Multicategory proximal support vector machine classifiers*. [S.l.], 2001.

GERMAN, B. *Glass Identification Database*. [S.l.], 1987.

GLOBO, O. *Hacker acessa dados do Exército brasileiro e divulga informações de quase mil funcionários*. 2011.

GOTTSCHALK, S. *Collision Queries using Oriented Bounding Boxes*. Tese (Doutorado) — Department of Computer Science, UNC, 2000.

GROUP, V. *Image Segmentation data*. [S.l.], 1990.

GUJRATHI, S. Heartbleed bug: Anopenssl heartbeat vulnerability. *International Journal of Computer Science and Engine ter Science and Engineering*, v. 2, n. 5, p. 61–64, 2014. E-ISSN: 2347-2693.

HETTICH, S.; BAY, S. D. *The UCI KDD Archive*. [S.l.], 1999.

HSU, C.-W.; LIN, C.-J. *A Comparison of Methods for Multiclass Support Vector Machines*. 2002.

INSTITUTE, S. T. *SANS*. 2013.

KASPERSKY. *Kaspersky Security Bulletin 2013*. [S.l.], 2014.

KAYACIK H. G.; ZINCIR-HEYWOOD, A. N. H. M. I. A hierarchical som-based intrusion detection system. *Engineering Applications of Arti?cial Intelligence*, v. 20, n. 4, p. 439–451, 2007. ISSN 0952-1976.

KIM J.; BENTLEY, P. J. A. U. G. J. T. G. T. J. Immune system approaches to intrusion detection - a review. *Natural Computing*, v. 6, n. 4, p. 413–466, 2007.

LEAVITT, N. A technology that comes highly recommended. *Computer*, v. 46, n. 3, p. 14–17, March 2013. ISSN 0018-9162.

MACMILLAN, C. D. C. N. A. *Detection Theory: A User's Guide*. [S.l.]: Taylor & Francis, 2004.

MAGAZINE, S. Best ids/ips. *SC Magazine*, 2011. Disponível em: <http://www.scmagazine.com/best-idsips/article/196010/>.

MBIKAYI, H. K. An evolution strategy approach toward rule-set generation for network intrusion detection systems (ids). *CoRR*, abs/1212.0170, 2012.

MITNICK K. L.; SIMON, W. L. *A Arte de Invadir: As verdadeiras histórias por trás das ações de hackers, intrusos e criminosos eletrônicos*. [S.l.]: Prentice Hall, 2006.

MOOD F. A. GRAYBILL, D. C. B. A. M. *Introduction to the Theory of Statistics*. [S.l.]: McGraw-Hill, 1974. 577 p.

MOWFORTH, D.; SHEPHERD, B. *Vehicle Silhouettes data*. [S.l.], 1987.

NETWORK, M. B. E. *The Sony PlayStation 3 hack deciphered: what consumer-electronics designers can learn from the failure to protect a billion-dollar product ecosystem*. 2011.

PAWLAK, Z. Rough set approach to knowledge-based decision support. *European journal of operational research*, 1997.

PEREIRA, J. P. Comparison of firewall, intrusion prevention and antivirus technologies. *Juniper Networks, Inc*, v. 1, p. 1–8, 2006.

RECORD, R. *Grupo de hackers tenta invadir sites do Governo Federal Brasileiro*. 2011. Disponível em: <http://rederecord.r7.com/video/grupo-de-hackers-tenta-invadir-sites-do-governo-federal-brasileiro-4e027e19b51a4b961b36afa6/>.

ROBINSON, D. D. M. N. T. *Vowel Recognition (Deterding data)*. [S.l.], 1989.

ROESCH, M. *Snort*. 2013. Sourcefire, Inc. Disponível em: <http://www.snort.org/>.

SANZ, J. A. et al. Medical diagnosis of cardiovascular diseases using an interval-valued fuzzy rule-based classification system. *Applied Soft Computing*, v. 20, n. 0, p. 103–111, 2014. ISSN 1568-4946. Hybrid intelligent methods for health technologies.

SHCHERBAKOVA, T. *Spam Report May*. [S.l.], 2014.

SHORTLIFFE, B. G. B. E. H. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. [S.l.]: Addison Wesley, 1984.

SIGKDD. *KDD Cup 1999: Computer network intrusion detection*. [S.l.], 1999. Disponível em: <http://www.sigkdd.org/kdd-cup-1999-computer-network-intrusion-detection>.

SULAIMAN, H. A.; BADE, A. *Bounding Volume Hierarchies for Collision Detection, Computer Graphics*. [S.l.: s.n.], 2012.

UCHOA, J. Q. *Algoritmos Imunoinspirados aplicados em segurança computacional: utilização de algoritmos inspirados no sistema imune para detecção de intrusos em redes de computadores*. Tese (Doutorado) — Universidade Federal de Minas Gerais, Maio 2009.

VEJA, R. *Invasão ao Itamaraty permitiu que hackers tivessem acesso a documentos da Copa*. 2014.

WEBER, R. F. Segurança na internet. *Anais da XIX JAI - Jornada de Atualização em Informática*, p. 43–82, 2000. Curitiba: PUCPR.

YADRON, W. D. Symantec develops new attack on cyberhacking. *Wall Street Journal*, 2014. Disponível em: <http://online.wsj.com/news/article_email/SB10001424052702303417104579542140235850578-lMyQjAxMTA0MDAwNTEwNDUyWj>.

YU Z.; TSAI, J. J. P. W. T. J. An automatically tuning intrusion detection system. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, v. 7, n. 2, p. 373–384, 2007.

APÊNDICE A – Regras da Base Iris

A seguir, serão mostradas todas as regras extraídas a partir das hipercaixas obtidas após o treinamento do classificador com a base de dados *Iris*.

Atributos: comprimento da sépala; largura da sépala; comprimento da pétala; largura da pétala

(4,3:5,8); (2,3:4,4); (1:1,9); (0,1:0,6); **Iris-setosa**
(6,7:7,9); (2,5:3,8); (5,1:6,9); (1,8:2,5); **Iris-virginica**
(4,9:5,9); (2:3); (3,3:4,5); (1:1,5); **Iris-versicolor**
(6,1:6,5); (2,6:3,4); (5,3:6); (1,4:2,5); **Iris-virginica**
(6:6,4); (2,2:2,9); (4:4,5); (1:1,5); **Iris-versicolor**
(6:6,4); (3:3,4); (4,5:4,7); (1,4:1,6); **Iris-versicolor**
(6,6:7); (3:3,2); (4,4:5); (1,4:1,7); **Iris-versicolor**
(5,6:6); (2,2:3); (4,8:5,1); (1,5:2,4); **Iris-virginica**
(6,5:6,8); (2,8:2,9); (4,6:4,8); (1,3:1,5); **Iris-versicolor**
(6,1:6,5); (3:3,2); (4,9:5,2); (1,8:2); **Iris-virginica**
(6,3:6,3); (2,8:2,8); (5,1:5,1); (1,5:1,5); **Iris-virginica**
(6,3:6,3); (2,5:2,7); (4,9:5); (1,8:1,9); **Iris-virginica**
(6,1:6,1); (2,8:2,9); (4,7:4,7); (1,2:1,4); **Iris-versicolor**
(6:6,3); (2,5:2,7); (4,9:5,1); (1,5:1,6); **Iris-versicolor**

APÊNDICE B – Regras da Base KDD99

A seguir, serão mostradas 80 das 579 regras extraídas a partir das hipercaixas obtidas após o treinamento do classificador com a base de dados *KDD99*.

Atributos: duration; protocol_type; service; flag; src_bytes; dst_bytes; land; wrong_fragment; urgent; hot; num_failed_logins; logged_in; num_compromised; root_shell; su_attempted; num_root; num_file_creations; num_shells; num_access_files; num_outbound_cmds; is_host_login; is_guest_login; count; srv_count; serror_rate; srv_serror_rate; rerror_rate; srv_rerror_rate; same_srv_rate; diff_srv_rate; srv_diff_host_rate; dst_host_count; dst_host_srv_count; dst_host_same_srv dst_host_diff_srv_rate; dst_host_same_src_port_rate; dst_host_srv_diff_host_rate; dst_host_serror_rate; dst_host_srv_serror_rate; dst_host_rerror_rate; dst_host_srv_rerror_rate;

0:0); (1:1); (11:11); (0:0); (28:28); (0:28); (0:0); (1:3); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:200); (1:100); (0:0,99); (0:0); (0:0,05); (0:0); (0,01:1); (0:0,1); (0:0); (73:255); (1:185); (0:0,73); (0,01:1); (0:0,73); (0:0); (0:0,09); (0:0); (0:0,91); (0:0); **teardrop**

(2:2); (0:0); (2:2); (6:6); (693375640:693375640); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (57:57); (3:3); (0,79:0,79); (0,67:0,67); (0,21:0,21); (0,33:0,33); (0,05:0,05); (0,39:0,39); (0:0); (255:255); (3:3); (0,01:0,01); (0,09:0,09); (0,22:0,22); (0:0); (0,18:0,18); (0,67:0,67); (0,05:0,05); (0,33:0,33); **portsweep**

(9:10); (0:0); (13:13); (0:0); (0:0); (5149533:5155468); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:3); (1:3); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (3:17); (3:17); (1:1); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); **warezmaster**

(5032:5068); (0:0); (10:13); (0:5); (5131424:5135678); (0:0); (0:0); (0:0); (0:0); (0:7); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:2); (1:19); (0:1); (0:1); (0:0); (0:0); (0,5:1); (0:1); (0:1); (1:255); (1:83); (0:1); (0:0,29); (0,32:1); (0:0,25); (0:0,29); (0:1); (0:0,14); (0:0,02); **warezclient**

(2:2); (0:0); (13:13); (0:0); (0:0); (1159100:1159100); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (18:18); (18:18); (1:1); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); **warezmaster**

(2426:2426); (0:0); (13:13); (7:7); (2500058:2500058); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (1:1); (1:1); (1:1); (0:0); (0:0); (3:3); (43:43); (1:1); (0:0); (1:1); (0,09:0,09); (0:0); (0:0); (0,33:0,33); (0,02:0,02); **warezclient**

(0:1); (0:0); (13:13); (0:0); (0:0); (467968:988002); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:2); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:1); (2:3); (2:3); (1:1); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); **multihop**

(0:53); (0:0); (0:5); (0:3); (133:812); (635115:1570327); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:8); (1:15); (0:0,5); (0:0,07); (0:0); (0:0); (0,5:1); (0:1); (0:0,13); (1:255); (1:255); (0,02:1); (0:0,06); (0:1); (0:0,25); (0:0,01); (0:0); (0:0,15); (0:0,52); **normal**

(6:5046); (0:0); (0:13); (0:0); (0:1794); (1653021:5134218); (0:0); (0:0); (0:0); (0:0); (0:3); (0:1); (0:18); (0:1); (0:2); (0:9); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (25:255); (2:255); (0,01:1); (0:0,12); (0:0,04); (0:0,4); (0:0,75); (0:0,86); (0:0,77); (0:0); **normal**

(0:18848); (0:0); (0:60); (0:6); (158:715240); (124015:600787); (0:0); (0:0); (0:3); (0:2); (0:2); (0:1); (0:884); (0:1); (0:2); (0:993); (0:10); (0:0); (0:8); (0:0); (0:0); (0:0); (1:32); (1:60); (0:0,07); (0:0,07); (0:1); (0:1); (0,5:1); (0:1); (0:0,67); (1:255); (1:255); (0,01:1);

(0:0,83); (0:1); (0:0,5); (0:0,76); (0:0,62); (0:0,77); (0:0,82); **normal**

(0:0); (0:0); (25:25); (1:1); (1492:1492); (649186:649186); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (1:1); (1:1); (0:0); (0:0); (1:1); (0:0); (0:0); (11:11); (11:11); (1:1); (0:0); (0,09:0,09); (0:0); (0,55:0,55); (0,55:0,55); (0:0); (0:0); **imap**

(0:5); (0:0); (1:13); (0:1); (112813:2194619); (0:364); (0:0); (0:0); (0:0); (0:0); (0:0); (0:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:3); (1:25); (0:0,5); (0:0,5); (0:0); (0:0); (0,5:1); (0:1); (0:1); (10:255); (5:200); (0,02:0,95); (0,01:0,46); (0:0,75); (0:0,2); (0:0,94); (0:0,07); (0:0,05); (0:0,02); **normal**

(290:290); (0:0); (5:5); (0:0); (415:415); (70529:70529); (0:0); (0:0); (0:0); (3:3); (0:0); (1:1); (4:4); (0:0); (0:0); (4:4); (4:4); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (1:1); (1:1); (1:1); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); **buffer_overflow**

(0:17833); (0:0); (0:5); (0:1); (143:7971); (51100:121702); (0:0); (0:0); (0:0); (0:1); (0:1); (1:1); (0:275); (0:1); (0:2); (0:306); (0:15); (0:0); (0:3); (0:0); (0:0); (0:0); (1:55); (1:60); (0:1); (0:1); (0:0); (0:0,5); (0,5:1); (0:1); (0:1); (1:255); (3:255); (0,02:1); (0:0,57); (0:1); (0:0,67); (0:0,93); (0:0,82); (0:0,54); (0:0,57); **normal**

(0:11867); (0:0); (0:60); (0:1); (137:61548); (34814:50859); (0:0); (0:0); (0:0); (0:2); (0:0); (1:1); (0:238); (0:1); (0:2); (0:268); (0:0); (0:0); (0:2); (0:0); (0:0); (0:0); (1:42); (1:61); (0:1); (0:1); (0:0,25); (0:0,5); (1:1); (0:0); (0:1); (1:255); (2:255); (0,02:1); (0:0,2); (0:1); (0:1); (0:0,33); (0:0,87); (0:0,98); (0:0,96); **normal**

(134:134); (0:0); (24:24); (0:0); (100:100); (39445:39445); (0:0); (0:0); (2:2); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (1:1); (0:0); (0:0); (1:1); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (2:2); (1:1); (0,5:0,5); (1:1); (0,5:0,5); (0:0); (0:0); (0:0); (0:0); (0:0); **ftp_write**

(25:50); (0:0); (60:60); (0:0); (64952:100788); (32476:32708); (0:0); (0:0); (0:0); (0:0);

(0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (38:255); (15:25); (0,06:0,66); (0,03:0,08); (0:0,03); (0:0); (0:0,03); (0:0,04); (0:0,03); (0:0,04); **normal**

(0:65); (0:0); (1:13); (0:1); (22532:106536); (0:1437); (0:0); (0:0); (0:0); (0:0); (0:0); (0:1); (0:0); (0:0); (0:0); (0:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:24); (1:24); (0:0,12); (0:0,09); (0:0); (0:0); (0,05:1); (0:0,75); (0:1); (2:255); (1:246); (0,01:1); (0:0,7); (0:1); (0:0,08); (0:0,74); (0:0,05); (0:0,67); (0:0,04); **normal**

(198:718); (0:0); (5:5); (0:0); (562:1412); (9139:25260); (0:0); (0:0); (0:0); (3:15); (0:0); (1:1); (22:38); (1:1); (0:0); (39:54); (4:4); (1:2); (0:2); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (1:1); (1:1); (1:1); (0:0); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); **multihop**

(0:244); (0:0); (0:13); (0:3); (11388:21103); (0:1874); (0:0); (0:0); (0:0); (0:0); (0:0); (0:1); (0:0); (0:0); (0:0); (0:0); (0:1); (0:0); (0:1); (0:0); (0:0); (0:0); (1:23); (1:24); (0:1); (0:0,5); (0:0); (0:0,5); (0,04:1); (0:1); (0:1); (1:255); (1:255); (0,01:1); (0:0,89); (0:1); (0:0,17); (0:0,87); (0:0,06); (0:0,76); (0:0,05); **normal**

(0:14); (0:0); (0:0); (7:7); (13140:38568); (1460:8315); (0:0); (0:0); (0:0); (0:2); (0:0); (1:1); (0:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:6); (1:6); (0:0); (0:0); (0,17:1); (0,17:1); (1:1); (0:0); (0:0,4); (2:255); (2:255); (1:1); (0:0); (0:0,5); (0:0); (0:0,01); (0:0,01); (0,02:1); (0,02:1); **back**

(0:14); (0:0); (0:0); (0:7); (40636:54540); (0:8315); (0:0); (0:0); (0:0); (0:3); (0:0); (1:1); (0:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:19); (1:21); (0:0,5); (0:0,5); (0:1); (0:1); (0,67:1); (0:0,67); (0:1); (1:255); (1:255); (1:1); (0:0); (0:1); (0:0); (0:0,04); (0:0,04); (0:1); (0:1); **back**

(0:187); (0:0); (0:12); (0:5); (0:372); (21087:34720); (0:0); (0:0); (0:0); (0:3); (0:0); (0:1); (0:3); (0:1); (0:0); (0:3); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:47); (1:60); (0:1); (0:1); (0:0,81); (0:0,81); (1:1); (0:0); (0:1); (1:255); (1:255); (0:1); (0:0,87); (0:1); (0:1); (0:1); (0:0,1); (0:0,96); (0:0,96); **normal**

(0:14918); (0:0); (0:5); (0:0); (374:3676); (21181:34430); (0:0); (0:0); (0:0); (0:22); (0:0); (1:1); (0:6); (0:0); (0:0); (0:4); (0:20); (0:0); (0:0); (0:0); (0:0); (0:0); (1:15); (1:20); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0,5); (1:255); (3:255); (0,07:1); (0:0,42); (0:1); (0:0,67); (0:0,66); (0:0,11); (0:0,1); (0:0,25); **normal**

(708:708); (0:0); (5:5); (0:0); (1727:1727); (24080:24080); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (6:6); (0:0); (0:0); (7:7); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:1); (1:1); (0:0); (0:0); (0:0); (0:0); (1:1); (0:0); (0:0); (255:255); (3:3); (0,01:0,01); (0,02:0,02); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); **rootkit**

(40121:42448); (0:0); (0:46); (7:7); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:2); (1:3); (0:0); (0:0); (1:1); (1:1); (1:1); (0:0); (0:0,67); (159:255); (1:86); (0,01:0,52); (0,14:0,47); (0,25:0,94); (0:0,02); (0:0); (0:0); (0,25:0,94); (0,01:1); **portsweep**

(25420:36783); (0:0); (11:37); (7:8); (0:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:2); (1:2); (0:0); (0:0); (1:1); (1:1); (1:1); (0:0); (0:0); (76:255); (1:2); (0:0,01); (0,11:0,82); (0,05:1); (0:0); (0:0); (0:0); (0,05:1); (1:1); **portsweep**

(15515:58329); (1:1); (3:10); (0:0); (42:147); (44:105); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (1:10); (1:11); (0:0); (0:0); (0:0); (0:0); (0,2:1); (0:1); (0:0,18); (29:255); (1:224); (0:0,88); (0,01:0,88); (0,04:1); (0:0); (0:0); (0:0); (0:0); (0:0); **normal**

(20983:22921); (0:0); (11:11); (7:7); (1:1); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (0:0); (2:2); (2:2); (0:0); (0:0); (1:1); (1:1); (1:1); (0:0); (0:0); (255:255); (2:2); (0,01:0,01); (0,62:0,73); (1:1); (0:0); (0:0); (0:0); (1:1); (1:1); **portsweep**

(0:19487); (0:0); (0:60); (0:7); (0:33996); (10358:21075); (0:0); (0:0); (0:0); (0:2); (0:0); (0:1); (0:5); (0:1); (0:0); (0:3); (0:28); (0:0); (0:4); (0:0); (0:0); (0:0); (1:65); (1:83); (0:1);

(0:1); (0:1); (0:1); (0,5:1); (0:1); (0:1); (1:255); (1:255); (0:1); (0:0,29); (0:1); (0:1); (0:1); (0:0,92); (0:0,95); (0:1); **normal**