THIAGO ROSCIA CERDEIRO DE LIMA

MACHINE LEARNING EM SISTEMAS DE DETECÇÃO DE INTRUSÃO EM REDE BASEADOS EM ANOMALIAS

CURITIBA

2015

THIAGO ROSCIA CERDEIRO DE LIMA

MACHINE LEARNING EM SISTEMAS DE DETECÇÃO DE INTRUSÃO EM REDE BASEADOS EM ANOMALIAS

Monografia apresentada para obtenção do Grau de Bacharel em Ciência da Computação pela Universidade Federal do Paraná.
Orientadores:
Prof. Dr. Luis Allan Kunzle
Prof. Dr. Luiz Eduardo S. Oliveira

CURITIBA

2015

SUMÁRIO

RESUMO

Este trabalho de graduação tem como intuito apresentar um estudo acerca da usabilidade e viabilidade de algoritmos de *data mining* e *machine learning* no contexto de sistemas de detecção de intrusão em rede. Sua utilização teria grande impacto na segurança tanto de servidores quanto de clientes, pois promete evitar ataques ao sistema detectando anomalias na rede. Neste artigo são apresentadas técnicas de *machine learning*; a arquitetura geral de um sistema baseado em anomalia; o processo de extração e processamento de dados; e, por fim, demonstra os impactos de uma base de dados representativa, de tipos de ataques desconhecidos e da seleção de atributos. Os estudos sugerem que a imprevisibilidade de novos tipos de ataques ainda é um empecílio, mesmo para sistemas com capacidade de aprender ao ser alimentado com novos dados.

Palavras-chave: NIDS, network security, intrusion detection, machine learning, data mining, pattern recognition, segurança, aprendizado de máquina, mineiração de dados

CAPÍTULO 1

INTRODUÇÃO

Ao longo da última década, a realidade da comunicação em rede entre computadores pessoais e servidores evoluiu para uma grande parcela da população, deixando de servir apenas para compartilhamento de websites. Tornou-se um completo ambiente utilizado por cidadãos para armazenamento de arquivos pessoais na núvem; por bancos para efetuar transações monetárias; e por governos para trocas de informações preciosas. Assim como é possível se conectar de uma máquina pessoal a um servidor *web*, é possível fazer o contrário. Qualquer máquina conectada a uma rede está teoricamente sujeita ao acesso de fora. Roteadores e sistemas operacionais atuais estão equipados com ferramentas básicas para previnir acesso não autorizado ao computador. Ainda assim, há pessoas dedicadas a burlar tais proteções. Alguns desses invasores conseguem copiar arquivos, instalar *malware* e até ganhar controle sobre o sistema sem serem notados pelo dono.

A fim de cobrir esse ponto cego, foram criados sistemas de detecção de intrusão, ou IDS (*intrusion detection systems*). Seu objetivo é reconhecer comportamentos incomuns dentro de uma rede. No meio acadêmico, muitos artigos foram publicados explorando algoritmos de *machine learning* nesta aplicação.

Existem diversos tipos de ataques e diferentes protocolos de comunicação, mas o problema de se detectar uma intrusão pode ser simplificado em um problema de classificação binária do comportamento do sistema, onde os dois possíveis resultados seriam "normal"ou "intrusivo". Existem duas "escolas de pensamento"bem definidas no meio acadêmico acerca do assunto:

1. *Signature-based* (baseado em assinatura) – Esse é o método padrão e atualmente o único utilizado comercialmente. Ele possui baixa adaptabilidade a novas ameaças.

2. *Anomaly-based* (baseado em anomalia) – Ainda uma área de estudo, possui alta taxa de Falso Positivo, mas boa adaptabilidade e será o foco deste trabalho.

CAPÍTULO 2

SIGNATURE-BASED NIDS

Apesar de não ser o foco deste artigo, vamos descrever brevemente os sistemas baseados em assinatura, a fim de comparação com os métodos nos quais vamos nos aprofundar.

Sistemas baseados em assinatura têm sido os mais bem sucedidos para detecção de intrusão até hoje. A ideia é comparar o tráfego da rede com padrões de comportamento conhecidos durante certos ataques. O maior problema é a ineficiência em detectar ataques novos.

Os primeiros sistemas utilizavam apenas o método de reconhecimento de padrão. Essa técnica possui um banco de dados preenchido com assinaturas de cada ameaça conhecida. Um evento malicioso é detectado se o atual estado da rede se iguala a uma assinatura. Como ele tenta comparar todas as assinaturas e a quantidade de dados transmitidos simultaneamente cresce a cada ano, o custo computacional tronou-se muito alto.

Uma segunda técnica é o método de *implication rules*, ou regras de inferência. Ela fornece um conjunto de regras que descrevem eventos conhecidos que podem inferir o acontecimento de uma intrusão. De qualquer modo, IDS de assinatura geralmente necessita de um humano capacitado, que cria um novo conjunto de regras toda vez que um novo tipo de ataque surge, para adquirir modelos de tráfego.

Daí surge a motivação para se utilizar técnicas de *data mining* em sistemas de assinatura. Elas proveem um modo de aumentar a automação no momento de construção e de ajuste do modelo. Eles podem adaptar os modelos à medida que acessam tráfego de rede contendo novas ameaças ou detectando diferentes versões de um ataque conhecido. Ainda assim, é impossível identificar comportamentos maliciosos completamente desconhecidos.

CAPÍTULO 3

ANOMALY-BASED NIDS

A técnica de detecção de anomalia analisa o atual comportamento da rede para checar se corresponde ou não a um comportamento normal. O maior benefício desse método é a habilidade de identificar novos ataques com sucesso. A desvantagem é a alta taxa de Falso Positivo. Ao invés de requerer um novo modelo a ser adaptado, sistemas baseados em anomalia necessitam de dados de tráfego de rede livres de ataques. *Data mining* também tem sido utilizado em sistemas baseados em anomalia, junto a outros métodos estatísticos e de *machine learning* (aprendizado de máquina).

As diferentes técnicas podem ser agrupadas em três categorias, de acordo com o processamento envolvido: estatístico, knowledge-based (baseado em conhecimento) ou algoritmos de aprendizado de máquina.

3.1 Técnicas Estatísticas

Um modelo de probabilidade de um determinado comportamento é criado a partir da atividade capturada do tráfego de rede, tal como taxa de tráfego, número de pacotes e quantidade de endereços de IP distintos.

Ele não requer conhecimento prévio pois está capacitado a aprender o comportamento normal a partir de observações sem ataques. Outro benefício das técnicas estatísticas é a possibilidade de identificar, com precisão, atividades maliciosas que ocorrem ao longo de períodos de tempo mais extensos. Por outro lado, nem todos os possíveis comportamentos conseguem ser modelados e o equilíbrio entre taxas de Falso Positivo e Falso Negativo dependem fortemente na configuração correta dos parâmetros.

3.2 Técnicas Baseadas em Conhecimento

Também chamados de *expert systems*, sistemas baseados em conhecimento são implementados criando-se um conjunto de regras de classificação para categorizar os dados. O modelo é geralmente criado por um humano experiente no campo da aplicação. Esse tipo de sistema não aponta novas atividades inofensivas como sendo maliciosas, garantindo, assim, um número reduzido de Falso Positivo. Por essa razão, o conjunto de regras precisa ser específico o suficiente e, apesar de ser possível atingir um certo nível de automação usando uma máquina de estados finitos, requer grande conhecimento sobre o comportamento da rede e tempo significativo para ser desenvolvido.

3.3 Técnicas de Machine Learning

Métodos de aprendizado de máquina exigem um conjunto de dados categorizados para treinar o modelo do comportamento esperado. A característica principal dessa técnica é a habilidade de adaptar suas regras de classificação à medida que novos dados são recebidos. Isso garante pouca necessidade por intervenção humana, uma vez funcionando. O lado ruim do aprendizado de máquina é o alto custo computacional. Os algoritmos mais populares usados em A-NIDS são:

- Rede Bayesiana – cria uma rede de relações de probabilidade entre características.
- Modelos de Markov – compara a probabilidade dos dados observados com um *threshold* definido.
- Redes neurais – cria uma rede de *perceptrons* com habilidade de se adaptar.
- Lógica Fuzzy – enxerga características como variáveis *fuzzy* e classifica com base em valores contínuos.
- Algoritmos Genéticos – deriva regras de classificação e seleciona as características mais discriminantes.

- Support Vector Machines – encontra um hiperplano que eficientemente separa ambas as opções de classificação.
- Árvores de Decisão – cria uma árvore onde cada folha representa uma classe e arestas correspondem a diferentes valores de atributos.
- Clustering – algoritmo não-supervisionado que agrupa dados com base em suas similaridades.

3.4 Combinação de Classificadores

Como cada método possui seus pontos fortes e pontos fracos, uma ideia bastante estudada no meio acadêmico é a combinação de classificadores. Tais combinações podem se dar de diversas maneiras, como por exemplo: Classificadores Híbridos, Composição de Classificadores [1] e Classificadores em Cascata.

3.4.1 Classificadores Híbridos

Um classificador híbrido consiste de dois componentes. O primeiro pré-processa o dado de entrada e envia ao segundo classificador, que chega ao resultado final. O primeiro componente de classificadores híbridos pode ser usado tanto como uma técnica de clustering para achar as classes que o segundo componente utilizará para categorizar os dados; quanto como um otimizador de performance para o segundo modelo, o que corresponderia ao método *integrado* [1].

3.4.2 Classificadores em Cascata

Pode ser considerado uma extensão dos classificadores híbridos, podendo consistir de inúmeros classificadores, onde o n-ésimo classificador utiliza como entrada os dados rejeitados pelo classificador anterior, ou seja, que não atingiram um grau de certeza satisfatório. Alguns estudos sugerem otimização de *thresholds* para aumentar precisão [10].

[1] Tradução livre do termo em inglês Ensemble Classifiers.

3.4.3 Composição de Classificadores

Uma composição de classificadores pode ser obtida usando-se técnicas fracas de aprendizado (geralmente mais rápidas). Cada classificador é treinado usando um subconjunto distinto dos dados. A base de testes é, então, processada por todos eles e, finalmente, categorizados pela maioria.

CAPÍTULO 4

SISTEMAS A-NIDS FUNCIONAIS

O estudo feito em [6] apresenta sistemas A-NIDS divididos em duas categorias: comerciais e de pesquisa. Sistemas comerciais geralmente trabalham com um núcleo baseado em assinatura, enquanto sistemas de pesquisa desenvolvem protótipos e buscam metodologias inovadoras.

4.1 Arquitetura

Enquanto alguns detalhes de implementação podem variar, o padrão da esquematização básica de um A-NIDS é como apresentado em [3]:

- Aquisição de dados de tráfego – coleta informação sobre os quadros da rede para processamento futuro.
- Gerador de características do tráfego – extrai características do tráfego capturado. Tais características podem ser classificadas como "baixo nível"(obtidas diretamente do dado bruto), "alto nível"(deduzidas de um processamento subsequente), "pacote"(coletadas de cabeçalhos de pacotes), "fluxo"(contendo informação das conexões) e "payload"(obtidos da carga do pacote).
- Detector de incidente – identifica atividades intrusivas. Pode ou não conter um segundo núcleo baseado em assinatura..
- Gerador de modelo de tráfego – contém informação base usada para guiar o detector de incidente.
- Gerenciamento de resposta – inicia manobras para superar uma intrusão em potencial. É inicializado pelo detector de incidente.

4.2 Comercialmente

Um dos primeiros projetos de detecção de anomalia a ser amplamente conhecido foi o SPADE, um plugin para o Snort que analisava transferência de pacotes procurando comportamentos fora do comum. Alternativamente, Stealthwatch utilizava detecção de anomalia baseada em fluxo.

Sistemas recentes usam uma arquitetura distribuída utilizando sensores e um console central para gerenciar o processo de detecção. Esse é o caso do DeepSight, que usa um método estatístico. A maioria dos sistemas comerciais utilizam um módulo de detecção baseado em assinatura aliado com um núcleo baseado em anomalia. Todos esses sistemas se enquadram na categoria de classificadores híbridos.

CAPÍTULO 5

DADOS

Os passos mais relevantes para o modelamento de dados de um A-NIDS são os seguintes:

1. Criação da base – identifica dados categorizados (normal ou anômalo) representativos para treino e teste. Categorização de tráfego de rede pode ser uma tarefa difícil e longa, que geralmente envolve um especialista.

2. Construção de características – cria características com maior discriminabilidade. Tais características podem ser construídas por um humano ou por algoritmos de aprendizado de máquina.

3. Redução – também chamado de "seleção de caterísticas", diminui a dimensionalidade da base de dados descartando características irrelevantes ou redundantes [15]. Usado para atenuar a "Maldição da Dimensionalidade"[1].

5.1 Origem dos Dados

A escolha de informações da rede é amplamente afetada pelos requisitos de detecção ao se projetar o sistema. É possível ter sistemas de detecção específicos, utilizando um conjunto de características limitado [9]. Para um sistema mais genérico, o ideal seria utilizar detectores separados utilizando conjuntos de características distintos, um para cada especificidade. Dependendo da origem desses dados, têm-se algumas vantagens e desvantagens. Técnicas para análise de conteúdo ainda não estão tão concretizadas quanto as que extraem características de cabeçalhos, assim como analisar conteúdo do lado do cliente ainda é um campo não tão estudado quanto o lado do servidor numa perspectiva de A-NIDS. Técnicas do lado do cliente almejam detectar ameaças em aplicações web, como *drive-by downloads*, *cross-site scripting* e trechos maliciosos de JavaScript.

[1] *Curse of Dimensionality*: Ao se usar muitas características, supostamente se obtém baixa precisão.

5.1.1 Cabeçalho de pacote

O conjunto de característica mais simples contém atributos básicos extraídos dos cabeçalhos. Características obtidas através de cabeçalhos de pacotes têm a qualidade de serem de rápida extração, sem exigir muito processamento ou memória, e evitarem preocupações legais acerca de análise de dados da rede. Essas características podem ser usadas para apontar pacotes individuais que são anômalos quando comaprados ao modelo de treino; ou como um processo de filtragem para que apenas pacotes incomuns sejam usados por algoritmos de detecção posteriores. Entretanto, pacotes individuais não podem ser usados para identificar padrões incomuns durante um grande período. Existem ataques que contêm cabeçalhos normais quando analisados individualmente, enquanto sua repetição durante um certo tempo pode ser considerada anômala. Um exemplo seria o ataque de negação de serviço, popularmente conhecido como *DoS*.

5.1.2 Corpo de pacote

Quando ataques são destinados a aplicações, os bytes maliciosos estão dentro do corpo do pacote e, portanto, as técnicas baseadas em cabeçalho não podem ser usadas. Isso representa um defeito considerável, principalmente porque diversos ataques da atualidade não são direcionados à rede em si, mas a aplicações conectadas a ela.

NIDS devem utilizar características baseadas em conteúdo, extraídas do corpo dos pacotes, para detectar tais tipos de ataques, uma vez que cabeçalhos podem aparentar completamente normais. Análise de conteúdo é computacionalmente mais cara do que análise de cabeçalho porque requer uma inspeção mais profunda do pacote. Ela lida com uma variedade de tipo de conteúdo (pdf, jpg, HTML), compressão, e métodos que encobrem dados. Entretanto, o benefício da análise do corpo é ter acesso a todos os bytes transferidos entre os dispositivos na rede, permitindo a construção de um rico conjunto de características baseadas em conteúdo para detecção de anomalia.

Como análise de conteúdo possui uma alta complexidade, diversos métodos focam em pequenos subconjuntos de conteúdo, como requisições HTTP ou apenas o JavaScript de um conteúdo baixado. Métodos baseados em anomalia não tentam comparar assinaturas

de malware conhecido, mas podem aplicar heurísticas, como Casamento de Padrões para detectar a presença de código shell.

5.2 Conjuntos de Características

5.2.1 Conexões múltiplas

Grande parte dos NIDS analisados em [4] usam dados de rede referentes a fluxo ou sessão. Características são construídas a partir do fluxo. O método mais popular é o de cabeçalho de pacotes utilizando características derivadas de múltiplas conexões (MCD). Esses atributos são geralmente derivados usando média, desvio padrão, e porcentagem de fluxos, cobrindo múltiplas sessões. NIDS baseados em anomalia que utilizam essas caracterśiticas são capazes de discernir entre atividade normal da rede e tráfego incomum como *DoS* e *scanning*.

Características MCD são geralmente extraídas a partir de conexões dentro de um intervalo de tempo. A maioria das Características de MCD são baseadas em volume, como o a quantidade de conexões a um endereço de IP e porta em um espaço de tempo. Portanto, atributos derivados de múltiplas conexões podem ser facilmente usados para detectar volumes excessivos de tráfego relacionados a *DoS* e *probing*. Como pacotes anômalos individuais não suprem o valor baseado em volume, eles podem ser ignorados.

5.2.2 Conexão individual

Características derivadas de conexões individuais (SCD) são utilizadas para detectar comportamento anômalo dentro de uma úncia sessão. Elas podem apontar um protocolo inesperado, tamanhos incomuns de dados, *timestamp* não condizente, ou sequências incomuns de *flags* de TCP. Portanto, características SCD permitem detecção de uso anômalo da rede por *backdoors*, túnel HTTP e afins.

As características SCD fornecem contexto que pode ser usado para encontrar anomalias não contextuais. Por exemplo, se a temporização de pacotes dentro de uma porta monitorada não se encaixa com um perfil esperado, um alerta pode ser disparado, uma

vez que pode se tratar de um protocolo de tunelamento.

5.2.3 KDD Cup 99

O KDD Cup 99 se trata de uma base dados para uso livre. Ela possui 32 características quantitativas e 9 qualitativas, junto a tráfego normal e ataques de *probing*, negação de serviço, *user to root* e *remote to local*. Tais características foram construídas por um especialista e involvem informações de mais alto nível, como o número de tentativas de login sem sucesso, se acesso root ao terminal foi conseguido, e o número de operações de criação de arquivo.

Considerando que o KDD Cup 99 já tem mais de 15 anos, é provável que essas características não sejam mais úteis para detectar os tipos de ataques recentes na realidade do fluxo de dados na rede atualmente. Para se detectar ameaças atuais, é necessário construir novas características baseadas em conteúdo, uma vez que o conteúdo em si mudou.

5.3 Pré-processamento

Como anteriormente apontado, sistemas baseados em anomalia estão sujeitos a uma alta taxa de Falso Positivo. Apenas 1% sequer de Falso Positivo resulta em um número absurdo de falsos alertas. Deve-se ter em mente que servidores lidam com centenas de conexões simultâneas e inúmeros pacotes a cada centésimo de segundo.

O pré-processamento de dados é, então, exigido para atingir uma melhor performance na detecção de intrusão. O pré-processamento converte tráfego de rede em uma série de ocorrências, onde cada uma é representada por um vetor de características. As informações seguintes acerca dos dados e características foram baseadas nos estudos presentes em [4].

5.3.1 Derivação

Dfierentes tipos de características que podem ser usados em NIDS baseados em anomalia foram cobertos em [4]. Cada tipo de característica é derivada de vários métodos de pré-

processamento, incluindo organizar pacotes em fluxos, analisar conteúdo de aplicações em busca de campos de interesse ou percorrer cabeçalhos de cada pacote. A derivação do potencial conjunto de características é um passo essencial para o NIDS baseado em anomalia. Analogamente, pré-processamentos subsequentes podem ajudar ainda mais a aumentar a eficiência do NIDS.

Métodos de pré-processamento de mineiração de dados podem ser usados, incluindo tranformação, redução e discretização de dados, como os apresentados em [13]. Uma técnica de redução geralmente utilizada é a *Principal Component Analysis*. PCA tem se mostrado útil para redução da dimensionalidade dos dados, consequentemente reduzindo o custo computacional do sistema de detecção

Para eliminar características redundantes, diversos algoritmos automatizados para seleção de características também existem, resultando em redução de dados similar. Tais métodos de redução de dados fornecem um meio preciso para a conversão de um conjunto candidato para o conjunto final de características. É possível que métodos automáticos de redução possam melhorar ainda mais o NIDS, apesar de que alguns sistemas são construídos apenas se baseando no conhecimento de um especialista para construir conjuntos de características significativos.

5.3.2 Alternativas de processamento

Considerando métodos que almejam maior automação no momento do processamento dos dados e características, existem duas ferramentas aparentemente promissoras: N-grams e libAnomaly. N-grams pode ser usado com algoritmos de seleção automática de características, enquanto o libAnomaly se trata de uma biblioteca livre para modelos de pré-processamento de dados.

No N-grams, perfis são construídos para diferentes tipos de arquivos, calculando as distribuições de frequência de byte [18]. O NIDS compara, então, o tráfego observado com os perfis usando Distância Manhattan. N-gram é geralmente usado para analisar requisições ao servidor. É útil em detectar padrões anômalos, como código shell dentro de protocolos estruturados de aplicações, sem exigir muita supervisão.

O foco do libAnomaly, por sua vez, é construir um punhado de modelos representando a transferência normal de dados entre usuário e aplicação web. Requisições maliciosas provavelmente serão apontadas como anômalas quando comparadas com um ou mais modelos, pois geralmente se diferem de requisições normais de alguma forma.

Embora a intenção é que NIDS sejam completamente automáticos, sua criação e adaptação requerem um especialista. Mesmo o N-grams necessita de auxílio em algumas áreas da rede. LibAnomaly também requer experiência significativa para selecionar quais campos dentro da rede devem ser modelados. Por outro lado, detecção usando cabeçalhos de pacotes requerem menos conhecimento. Técnicas de mineiração de dados podem ser usadas para derivar características de cabeçalhos MCD para detectar algumas ameaças. O conjunto de características em potencial poderá, então, ser processado no estágio de seleção de característica. Essas técnicas automatizadas garantem que as características disponíveis mais discriminantes sejam escolhidas.

CAPÍTULO 6

ANÁLISE DE TRABALHOS RELACIONADOS

NIDS baseado em anomalia tem se mostrado uma área de estudo promissora, mas ainda necessita de bastante pesquisa. Analisando os dados de artigos publicados entre os anos 2000 e 2007 e as diferentes técnicas utilizadas [16], é possível extrair algumas inferências.

- A maioria dos artigos se baseia no uso de classificadores unitários, apesar de que o uso de classificadores híbridos tem aumentado durante os últimos anos. Classificadores compostos são usados por poucos.

- Os algoritmos mais estudados para detecção de instrusão foram SVM e KNN. Quanto a classificadores híbridos, o método *integrado* é o mais popular.

- A utilização de seleção de atributos têm crescido com o tempo. Enquanto em 2003 apenas um quarto dos estudos analisados por [16] utilizava essa técnica, em 2007 foi o caso de 9 dentre 11. Ou seja, apenas dois ([12], [7]) não utilizaram.

- Como há poucas bases de dados públicas para detecção de instrusão, a maioria dos estudos se baseia nas bases KDD Cup 99 e DARPA 1998.

Quanto aos dados, as técnicas pioneiras costumavam criar modelos considerando todo o corpo do pacote. Entretanto, eles separavam as instâncias com base no tamanho do pacote e portas de destino, diminuindo o contexto de cada modelo. Técnicas posteriores levam em consideração partes específicas do conteúdo do pacote. Aparentemente, anomalias mais sutis podem ser detectadas a uma taxa manor de Falso Positivo quando usando um contexto mais específico.

Dois artigos recentes trouxeram implementações promissoras a respeito de sistemas de detecção de anomalia. Um deles ([8]) apresenta um algoritmo inteligente e o outro ([11]) mostra o ganho de performance ao se programar o sistema diretamente em uma placa FPGA.

6.1 Processando grandes quantidades de dados

A simulação apresentada em [8] usou a base do KDD Cup 99. O algoritmo funciona pré-processando os dados de treino e teste e gerando uma solução inicial aleatória. A solução é atualizada a cada iteração do algoritmo. Enquanto um limite não é satisfeito, são utilizados Support Vector Machine (SVM) e Simulated Annealing (SA) para selecionar o melhor conjunto de características. Por conseguinte, Árvore de Decisão e SA são usados para aumentar a precisão do teste e construir regras de decisão. No final, a melhor precisão e as melhores características e regras de decisão são anunciadas.

Nos testes descritos no artigo, após a seleção de atributos, foram utilizados 23 durante o processo de classificação. O algoritmo conseguiu atingir uma precisão de 99,6%, o que representa um grande ganho quando comparado a outros resultados até o momento.

6.1.1 Support Vector Machine

SVM trabalha mapeando a base de treino em um espaço vetorial de maior dimensionalidade para aumentar a discriminação. Ele se baseia em minimizar o risco estrutural. Os dados de enrada são mapeados por uma *função núcleo*, a qual é geralmente escolhida entre *linear*, *polinomial*, *sigmoide* ou *radial basis*. A última é preferida no uso em SVM por usa capacidade de lidar com dados de alta dimensionalidade. SA é aplicado para escolher os melhores valores para os parâmetros usados na função *radial basis*.

6.1.2 Árvore de Decisão

DT (*Decision Tree*) é um método de classificação baseado em um algoritmo guloso, utilizando uma estratégia de divisão e conquista. Ele cria uma árvore de decisão contando com um nó raiz, que é usado para selecionar o atributo com a informação de maior valor; nós internos, que correspondem às características; arestas, que são os possíveis valores de cada característica; e folhas, que contêm a classe a ser atribuída ao caso de teste.

6.2 Aumentado a velocidade de processamento

Enquanto o algoritmo anterior pretende otimizar apenas a precisão, o estudo apresentado em [11] foca em na aceleração do processo de classificação. Aqui também foi utilizado SVM para implementar o A-NIDS, mas dessa vez diretamente em um FPGA, fazendo uso de seu potencial de processamento paralelo.

Foram propostas três arquiteturas diferentes.

1. *Heterogeneous Baseline Classifier* – O objetivo é explorar os requerimentos de precisão dos atributos. Ele tenta maximizar paralelização enquanto mantém a relação entre blocos de processamneto digital e blocos lógicos.

2. *Fit Cascade Chain* – Aproveita o potencial de aritmética customizada oferecido pelo FPGA para criar uma cascata com duas precisões aritméticas diferentes. Primeiro um classificador de baixo custo e alta vazão com baixa precisão é usado. Somente os dados que não puderem ser classificados são alimentados ao segundo classificador. Este possui alta precisão e, logo, maior custo, mas com menos requisições de vazão.

3. *Reconfigurable Cascade Chain* – Foi projetado para problemas com grandes quantidades de dados, que podem exceder as limitações de hardware. Essa arquitetura propõe uma reconfiguração do FPGA depois do estágio de treino e expande o espaço de design possível. Isso permite a utilização de mais blocos durante as classificação.

A base utilizada para testes foi a *MINIST Dataset*. Comparando com testes prévios em GPU, o design *Heterogeneous Baseline Classifier* processou os dados 7 vezes mais rápido. Os melhores resultados foram obtidos usando *Reconfigurable Cascade Chain*, com um ganho de 20 vezes, isto é, levando apenas 5% do tempo que a GPU consumiu.

CAPÍTULO 7

TESTES

Três baterias de testes foram conduzidas ao longo da produção deste trabalho de graduação a fim de ver na prática o impacto dos algoritmos de aprendizado de máquina quando aplicados em bases de dados com informações de rede. Em um dos testes, a base utilizada foi a supracitada KDD Cup 99. Para os outros, utilizou-se uma série de dados gerados pelo mestrando da PUC-PR Eduardo Viegas, orientado pelo Prof. Dr. Altair Santin. Em ambos os casos, os algoritmos foram executados através do Weka.

7.1 Weka

O *Waikato Environment for Knowledge Analysis* (Weka) é uma suíte de aplicações para aprendizagem de máquina e mineiração de dados. O software foi desenvolvido em Java na *University of Waikato* e lincenciado sob a *GNU General Public License*. Os destaques dessa ferramenta incluem a vasta quantidade de algoritmos implementados, pré-processamento de dados, seleção de atributos e visualização de dados. O Weka funciona como uma interface, permitindo que novos algoritmos em Java sejam produzidos e facilmente executados de forma gráfica e com resultados interativos [2].

7.2 Avaliação de Performance

Gráficos ROC (*Receiver Operating Characterístics*) são comumente usados como um método para visualizar a performance de classificadores e analisar as taxas de acertos e alarmes falsos [5]. Um gráfico ROC baseia-se em apenas duas classes: positivo ou negativo. Em termos de detecção de intrusão, uma conexão pode ser considerada inofensiva (positivo) ou prejudicial (negativa). Para avaliar a precisão de um classificador, duas outras categorias podem ser usadas: verdadeiro e falso. Quando cruzadas, obtém-se quatro

possíveis resultados, como mostra a tabela 7.1.

Tabela 7.1: Categorias para análise de corretude

	Verdadeiro (T)	Falso (F)
Positivo	TP – Conexão inofensiva é classificada corretamente	FP – Conexão anômala é considerada normal
Negativo	TN – Conexão prejudicial é classificada corretamente	FN – Conexão normal é considerada anômala

Quando se sabe a verdadeira natureza dos casos de teste, é possível determinar a precisão, a taxa de Verdadeiro Positivo (TP_{rate}) e taxa de Falso Positivo (FP_{rate}). É possível inferir que uma das maiores preocupações ao se construir um A-NIDS é a minimização a taxa de Falso Positivo. Teoricamente se 100% das conexões forem consideradas anômalas e bloqueadas, teremos FP_{rate} igual a 0. Por um lado, o sistema estaria bloqueando conexões inofensivas, por outro, estaria com certeza evitando conexões ameaçadores. O cálculo das taxas se dá da seguinte maneira.

$$Precisão = \frac{TP + TN}{P + N}$$

$$TP_{rate} = \frac{TP}{P}$$

$$FP_{rate} = \frac{FP}{N}$$

Essas taxas estão contidas no espaço entre 0 e 1, inclusivo, e são utilizadas para plotar o ROC. A TP_{rate} é usado no eixo y e a FP_{rate}, no eixo x. Portanto, o gráfico ROC elicita a relação entre Verdadeiro Positivo e Falso Positivo.

A Figura 7.1 mostra uma curva ROC simples com 5 classificadores fictícios: A, B, C, D e E. Cada classificador retorna um único ponto no espaço ROC. Se o ponto estiver localizado na coordenada (0,0), como o classificador A na Figura 7.1, significa que nenhum caso foi considerado inofensivo. Classificadores que se encontram próximos ao eixo y podem ser considerados "conservadores", uma vez que necessitam de forte evidência para categorizar uma ocorrência como positiva. Em contrapartida, o ponto B, posicionado em (1,1), representa sistemas que não acusariam nenhuma conexão como anômala. Classificadores mais à direita do gráfico podem ser considerados "liberais"pois presumem

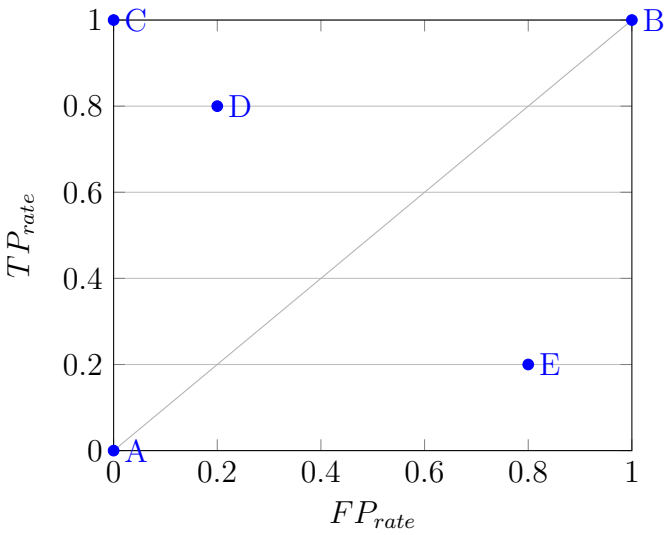

Figura 7.1: Exemplo da distribuição de 5 classificadores

como positivo mais facilmente.

Um classificador perfeito resultaria no ponto (0,1), onde C se encontra. Isso significaria possuir FP_{rate} de 0% e TP_{rate} de 100%, ou seja, todas as conexões intrusivas serem detectadas e nenhuma conexão normal ser bloqueada pelo sistema. Assim sendo, quanto maior o TP_{rate} e menor o FP_{rate}, mais preciso é o algoritmo. Se um classificador está localizado abaixo da linha diagonal ($y = x$), isto significa que sua performance é pior do que a escolha aleatória de classes. O triângulo inferior direito é geralmente encontrado vazio pois em casos de classificação binária com alta taxa de erro, é possível simplesmente inverter os resultados. Portanto, E (0.8,0.2) seria tão preciso quanto D(0.2,0.8)

7.3 Impacto de modelos representativos

O primeiro teste realizado levou em consideração apenas o SVM padrão fornecido pelo libSVM para o Weka. A finalidade era verificar o impacto da representabilidade da base de treino [17], analisando a performance do algoritmo quando treinado com bases de diferentes tamanhos. Para esse estudo, o KDD Cup 99 foi divido em 5 particionamentos, cada um com tamanhos distintos de bases de treino e de testes, usando a opção de *cross-validation* do Weka. O comando executado incluiu um pré-processamento de heurística de encolhimento, peso 1, *seed* de 3, erro de 0.001 e função núcleo *Radial Basis*.

Tabela 7.2: Comparação entre particionamentos de tamanhos diferentes

	Tamanho Treino	Tamanho Teste	Tempo	Precisão
1	10%	90%	16h13m	92.22%
2	20%	80%	24h02m	94.54%
3	30%	70%	27h32m	95.65%
4	40%	60%	32h11m	98.62%
5	50%	50%	38h45m	99.64%

A 7.2 mostra os as proporções de base de testes e treino em cada particionamento e seus respectivos resultados. É possível notar uma clara correlação entre o tamanho da base de treino e a porcentagem de acertos. O melhor resultado foi de 99.64% quando metade da base foi usada para gerar o modelo. Entretanto, em uma situação real não há dados disponíveis o suficiente para se representar metade das transmissões que ocorrem diariamente.

7.4 Impacto da imprevisibilidade

Uma abordagem recente direcionou os estudos desse trabalho para a análise da real capacidade de algoritmos de aprendizado de máquinas em se adaptar a novas realidades e detectar ameaças totalmente desconhecidas [14].

Através de simulações de rede, foi gerado um conjunto de dados, contendo conexões normais e anômalas, dispostos em três grupos distintos, os quais chamaremos de *Inicial*, *Análogo* e *Desconhecido*. O primeiro é um limitado conjunto de conexões utilizando protocolos HTTP e SMTP. Este foi dividido em dois grupos: um para treino, com 258880 instâncias, e outro para testes, com 275204. O grupo *Análogo* possui dados de 148343 conexões e, como o nome sugere, possui ataques semelhantes ou derivados daqueles no grupo anterior. Já o terceiro engloba, também, ataques completamente desconhecidos provenientes de outros protocolos, como SNMP e SSH, totalizando 276548 ocorrências. Apenas o grupo *Inicial* foi utilizado para treino. O objetivo é observar o comportamento do sistema quando novos ataques são recebidos, sejam eles derivados dos conhecidos ou completamente novos.

Tanto o SVM quanto DT foram utilizados nesta análise. As tabelas 7.3 e 7.4 mostram

os resultados em cada etapa do processo, em termos de taxa de Verdadeiro Positivo, taxa de Falso Positivo, tempo para classificar todas as instâncias da respectiva base, precisão e porcentagem de instâncias classificadas corretamente. As taxas de acerto evidenciam uma grande perda de confiabilidade conforme o surgimento de novos ataques, sendo a classificação da base *Desconhecido* praticamente tão eficiente quanto uma adivinhação aleatória.

Tabela 7.3: Resultados de DT em cada base

	DT				
	TP_{rate}	FP_{rate}	Tempo	Acertos	Precisão
Inicial	0.982	0.018	6.92s	98.18%	0.982
Análogo	0.912	0.856	3.73s	91.23%	0.885
Desconhecido	0.503	0.497	6.95s	50.29%	0.751

Tabela 7.4: Resultados de SVM em cada base

	SVM				
	TP_{rate}	FP_{rate}	Tempo	Acertos	Precisão
Inicial	0.971	0.029	15h13min	98.43%	0.984
Análogo	0.902	0.845	8h12min	90.27%	0.891
Desconhecido	0.52	0.484	15h29min	50.78%	0.753

7.5 Impacto da seleção de atributos

Tendo em vista o alto custo computacional para processar as 51 características iniciais, foi ponderada a viabilidade de se utilizar uma seleção de atributos prévia para agilizar o processamento na tomada de decisões. Utilizando o PCA, a quantidade de características foi reduzida para 39. Novos testes foram executados, seguindo o mesmo método da análise anterior.

Tabela 7.5: Resultados de DT em cada base com seleção de atributos

	DT				
	TP_{rate}	FP_{rate}	Tempo	Acertos	Precisão
Inicial	1.0	0.0	5.75s	99.98%	1.0
Análogo	0.848	0.429	3.1s	84.82%	0.914
Desconhecido	0.499	0.501	5.78s	49.86%	0.414

Tabela 7.6: Resultados de SVM em cada base com seleção de atributos

	SVM				
	TP_{rate}	FP_{rate}	Tempo	Acertos	Precisão
Inicial	0.503	0.502	14h45min	50.3%	0.75
Análogo	0.068	0.068	7h56min	6.78%	0.937
Desconhecido	0.5	0.5	14h51min	50.0%	0.75

As tabelas 7.5 e 7.6 mostram que o resultado dos algoritmos nesse cenário teve nenhum ganho significativo. Enquanto os atributos selecionados podem ser mais descritivos para os ataques vistos na base *Inicial*, talvez não sejam interessantes para identificar os ataques correntes. A figura 7.2 evidencia uma diminuição de Falso Positivo na base *Análogo* quando usada a seleção de atributo, mas ainda sendo muito superior ao *Inicial*. O comportamento ao se classificar a base totalmente desconhecida mostrou-se tão bom quanto escolha aleatória.

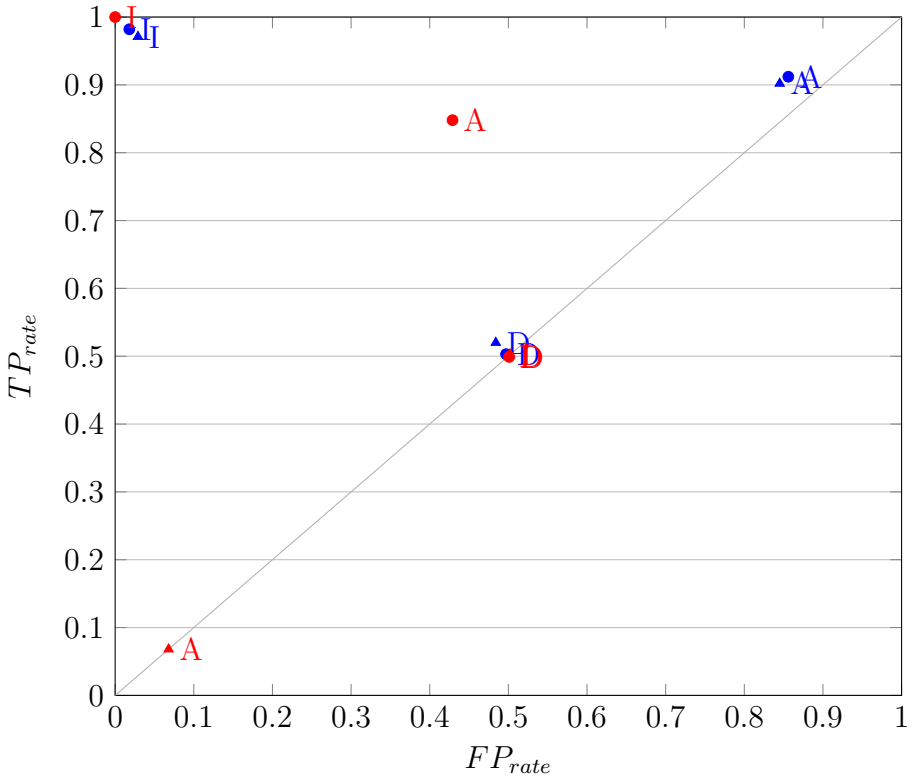

Figura 7.2: Comparativo dos algoritmos SVM (triângulo) e DT (círculo) em base sem e com seleção de atributo (azul e vermelho, respectivamente). I – *Inicial*; A – *Análogo*; D – *Desconhecido*

CAPÍTULO 8

CONCLUSÃO

Em suma, os algoritmos de aprendizado tentam automatizar o processo de reconhecer ameaças dentro da rede através da detecção de conexões anômalas, dando luz aos Sistemas de Detecção de Intrusão em Rede baseados em Anomalia. Neste trabalho, foram analisados os diferentes tipos de dados utilizados para a detecção de intrusões em rede e a performance de algoritmos sugeridos em trabalhos relacionados. Também foram estudados os impactos de uma base de treino suficientemente representativa da rede para predição de ataques desconhecidos pelo sistema.

O primeiro teste, usando apenas SVM, mostrou a necessidade de se gerar uma base de treino grande o suficiente, a fim de se evitar amostras pouco representativas. Os resultados do segundo teste sugerem que, possivelmente, os A-NIDS não tenham tanta escalabilidade com o surgimento de ameaças completamente desconhecidas e que, portanto, seriam necessárias bases específicas para aplicações particulares ao invés de sistemas genéricos. O último teste tinha a intenção de melhorar os resultados anteriores através da seleção de atributos, mas o seu ganho de performance não foi significativo.

O campo ainda está aberto para novos estudos e análises quanto a viabilidade do uso de machine learning em tempo real para a detecção de novas ameaças. Existem outros pontos não abordados nesse trabalho que também têm espaço para serem otimizados, tais quais a atual falta de métricas unificadas; o processamento do alto, rápido e contínuo fluxo de dados na rede; análise de dados cifrados e a taxa aceitável de Falso Positivo. Por fim, pode-se considerar que o maior desafio nos testes de detecção de intrusão em rede reside em adquirir uma base de treino e de testes que represente precisamente o atual comportamento da rede.

BIBLIOGRAFIA

[1] M. Ali Aydin, A. Halim Zaim, e K. Gökhan Ceylan. A hybrid intrusion detection system design for computer network security. *Comput. Electr. Eng.*, 35(3):517–526, maio de 2009.

[2] Remco R. Bouckaert, Eibe Frank, Mark A. Hall, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, e Ian H. Witten. Weka—experiences with a java open-source project. *J. Mach. Learn. Res.*, 11:2533–2541, dezembro de 2010.

[3] Carlos A. Catania e Carlos García Garino. Automatic network intrusion detection: Current techniques and open issues. *Comput. Electr. Eng.*, 38(5):1062–1072, setembro de 2012.

[4] Jonathan J. Davis e Andrew J. Clark. Data preprocessing for anomaly based network intrusion detection: A review. volume 30, páginas 353–375, Oxford, UK, UK, setembro de 2011. Elsevier Advanced Technology Publications.

[5] Tom Fawcett. Roc graphs: Notes and practical considerations for researchers. Relatório técnico, Hawlett Packard, 2004.

[6] P. García-Teodoro, J. Díaz-Verdejo, G. Maciá-Fernández, e E. Vázquez. Anomaly-based network intrusion detection: Techniques, systems and challenges. *Computers & Security*, 28(1-2):18 – 28, 2009.

[7] Yang Li e Li Guo. An active learning based tcm-knn algorithm for supervised network intrusion detection. *Computers & Security*, 26(7-8):459 – 467, 2007.

[8] Shih-Wei Lin, Kuo-Ching Ying, Chou-Yuan Lee, e Zne-Jung Lee. An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection. volume 12, páginas 3285 – 3290, 2012.

[9] Jun Ma, Guanzhong Dai, e Jing Zhou. Anomalous payload detection system using analysis of frequent sequential pattern. *Information Assurance and Security, 2009. IAS '09. Fifth International Conference on*, volume 1, páginas 75–78, Aug de 2009.

[10] L. Oliveira, A.S. Britto, e R. Sabourin. Optimizing class-related thresholds with particle swarm optimization. *Neural Networks, 2005. IJCNN '05. Proceedings. 2005 IEEE International Joint Conference on*, volume 3, páginas 1511–1516 vol. 3, July de 2005.

[11] M. Papadonikolakis e C.-S. Bouganis. Novel Cascade FPGA Accelerator for Support Vector Machines Classification. volume 23, páginas 1040–1052, July de 2012.

[12] Sandhya Peddabachigari, Ajith Abraham, Crina Grosan, e Johnson Thomas. Modeling intrusion detection system using hybrid intelligent systems. volume 30, páginas 114 – 132, 2007. Network and Information Security: A Computational Intelligence Approach Network and Information Security: A Computational Intelligence Approach.

[13] Marcela Xavier Ribeiro, Agma J. M. Traina, e Caetano Traina, Jr. A new algorithm for data discretization and feature selection. *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, páginas 953–954, New York, NY, USA, 2008. ACM.

[14] R. Sommer e V. Paxson. Outside the closed world: On using machine learning for network intrusion detection. *Security and Privacy (SP), 2010 IEEE Symposium on*, páginas 305–316, May de 2010.

[15] Jingping Song, Zhiliang Zhu, Peter Scully, e Chris Price. Selecting features for anomaly intrusion detection: A novel method using fuzzy c means and decision tree classification. Guojun Wang, Indrakshi Ray, Dengguo Feng, e Muttukrishnan Rajarajan, editors, *Cyberspace Safety and Security*, volume 8300 of *Lecture Notes in Computer Science*, páginas 299–307. Springer International Publishing, 2013.

[16] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, e Wei-Yang Lin. Intrusion detection by machine learning: A review. *Expert Systems with Applications*, 36(10):11994 – 12000, 2009.

[17] A.G. Yaman e T. Can. The effect of representative training dataset selection on the classification performance of the promoter sequences. *Health Informatics and Bioinformatics (HIBIT), 2011 6th International Symposium on*, páginas 55–58, May de 2011.

[18] Li Zhuowei, A. Das, e S. Nandi. Utilizing statistical characteristics of n-grams for intrusion detection. *Cyberworlds, 2003. Proceedings. 2003 International Conference on*, páginas 486–493, Dec de 2003.