UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Silvio E. Quincozes

DETECÇÃO DE INTRUSÕES ATRAVÉS DA SELEÇÃO DINÂMICA DE CLASSIFICADOR BASEADA EM REDES DE CONSELHOS

Santa Maria, RS
2018

Silvio E. Quincozes

DETECÇÃO DE INTRUSÕES ATRAVÉS DA SELEÇÃO DINÂMICA DE CLASSIFICADOR BASEADA EM REDES DE CONSELHOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**

Orientador: Prof. Dr. Carlos Raniery P. dos Santos

Co-orientador: Prof. Dr. Raul Ceretta Nunes

Santa Maria, RS

2018

E. Quincozes, Silvio

Detecção de Intrusões através da Seleção Dinâmica de Classificador baseada em Redes de Conselhos / por Silvio E. Quincozes. – 2018.
81 f.: il.; 30 cm.

Orientador: Carlos Raniery P. dos Santos
Co-orientador: Raul Ceretta Nunes
Dissertação (Mestrado) - Universidade Federal de Santa Maria, Centro de Tecnologia, Pós-Graduação em Ciência da Computação , RS, 2018.

1. Sistema de Detecção de Intrusões. 2. Análise de Fontes Heterogêneas de Dados. 3. Classificação de Dados. 4. Seleção Dinâmica de Classificador. 5. Rede de Conselhos. I. Raniery P. dos Santos, Carlos. II. Ceretta Nunes, Raul. III. Título.

E-mail: sequincozes@inf.ufsm.br

Silvio E. Quincozes

DETECÇÃO DE INTRUSÕES ATRAVÉS DA SELEÇÃO DINÂMICA DE CLASSIFICADOR BASEADA EM REDES DE CONSELHOS

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação (PGCC) da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do título de **Mestre em Ciência da Computação**

Aprovado em 01 de 02 de 2018:

Carlos Raniery P. dos Santos, Dr.
(Presidente/Orientador)

Rogério Turchetti, Dr. (UFSM)

Weverton Luis da Costa Cordeiro

Weverton Cordeiro, Dr. (UFRGS)

Santa Maria, RS

2018

DEDICATÓRIA

Dedico este trabalho a todas aquelas pessoas que acreditam em si mesmas. São poucos aqueles que buscam a autossuperação a fim de alcançar o progresso mental contínuo, longe de suas zonas de conforto.

AGRADECIMENTOS

Agradeço, primeiramente, a Deus por ser meu principal guia em toda a jornada que originou esta conquista. Em seguida, agradeço aos familiares que estiveram presentes, prestando seu apoio de todas as formas a mim, quando este foi necessário. Agradeço pela compreensão durante os momentos em que me ausentei de atividades familiares para dar andamento nos trabalhos aqui apresentados. Expresso, também, meus agradecimentos aos meus orientadores pelos sábios aconselhamentos e também pela compreensão com o fato de que precisei, muitas vezes, me dividir entre os estudos e trabalho. E, por fim, gostaria de agradecer aos meus colegas de trabalho, principalmente, ao Fernando Giaretta e Luiz Fogliato, os quais foram meus lideres nas duas empresas que trabalhei durante a minha jornada como mestrando.

"O valor de uma coisa depende da maneira como a abordamos mentalmente e não da coisa em si"

(JIGORO KANO)

RESUMO

DETECÇÃO DE INTRUSÕES ATRAVÉS DA SELEÇÃO DINÂMICA DE CLASSIFICADOR BASEADA EM REDES DE CONSELHOS

AUTOR: SILVIO E. QUINCOZES
ORIENTADOR: CARLOS RANIERY P. DOS SANTOS
CO-ORIENTADOR: RAUL CERETTA NUNES

Sistemas Detectores de Intrusão são comumente usados para analisar as informações disponíveis em redes de computadores e sistemas computacionais. Por meio do uso de técnicas, tais como a classificação de dados, é possível a identificação de atividades maliciosas. Todavia, o uso de tal técnica apresenta um desafio que consiste na escolha do classificador ideal frente à múltiplas possibilidades de ataques. Trabalhos existentes tentam mitigar esse problema com o emprego de múltiplos classificadores, contudo, essa abordagem pode apresentar classificadores com resultados conflitantes, onde é necessária a seleção do qual classificador deve-se confiar. Ademais, existem casos onde a fonte analisada por um detector não traz informações suficientes para uma decisão precisa, possibilitando que até mesmo o melhor classificador seja incapaz de alcançar uma acurácia satisfatória. O objetivo deste trabalho consiste na proposta de uma arquitetura de detecção de intrusões através da seleção dinâmica de classificadores, onde é explorada a consulta a conselheiros de forma a correlacionar múltiplas e heterogêneas fontes de dados. Os resultados preliminares revelam que a arquitetura é promissora, resolvendo conflitos e aumentando a acurácia na detecção de intrusões.

Palavras-chave: Sistema de Detecção de Intrusões. Análise de Fontes Heterogêneas de Dados. Classificação de Dados. seleção Dinâmica de Classificador. Rede de Conselhos.

ABSTRACT

INTRUSION DETECTION THROUGH THE DYNAMIC CLASSIFIER SELECTION BASED ON COUNSELORS NETWORK

AUTHOR: SILVIO E. QUINCOZES
ADVISOR: CARLOS RANIERY P. DOS SANTOS
COADVISOR: RAUL CERETTA NUNES

Intrusion Detection Systems are common used for information analysis, that are collected from computer networks and computer systems. Through the use of techniques such as data classification, it is possible to identify malicious activities. However, the use of such technique presents a challenge that consists of choosing the ideal classifier against multiple possibilities of attacks. Existing efforts try to mitigate this problem with the use of multiple classifiers, however, this approach often introduces conflicts in decision making. In addition, there are cases where a source analyzed by a detector does not provide sufficient information for a precise decision. The objective of this work is the creation of an intrusion detection architecture through the dynamic selection of classifiers in council networks, where it is explored the consultation of counselors who analyzes multiple and heterogeneous data sources. Preliminary results show that the architecture is promising, resolving conflicts and increasing security in intrusion detection.

Keywords: Intrusion Detetion System. Heterogeneous Data Source Analysis. Data Classification. Dynamic Classifier Selection. Council Network.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

IDS	*Intrusion Detection Sistem*
UFSM	Universidade Federal de Santa Maria
JSON	*Javascript Object Notation*
DoS	*Denial-of-Service*
UDP	*User Datagram Protocol*
ICMP	*Internet Control Message Protocol*
PoD	*Ping of Death*
TCP SYN	*Transfer Control Protocol Synchronize*
LAND	*Local Area Network Denial*
U2R	*User-To-Root*
SAINT	*Security Administrator Integrated Network Tools*
KNN	*K-nearest Neighbours*
DBSCAN	*Density-Based Spatial Clustering of Applications with Noise*
SATAN	*Secutiry Administrator Tool for Analyzing Networks*

SUMÁRIO

1 INTRODUÇÃO

Sistemas Detectores de Intrusões (IDS) têm por finalidade identificar ações oriundas de atacantes que possam ameaçar redes de computadores ou sistemas computacionais. Para tanto, os seus motores de detecção costumam aplicar métodos que possibilitam a análise de dados, a fim de identificar características que compõe perfis típicos de ataques. Uma das categorias de algoritmos com esse propósito consiste na classificação de dados, a qual visa a construção de modelos, baseados em conjuntos de assinaturas, a fim de classificar as amostras desconhecidas de acordo com a similaridade de suas características (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996a; FARID; HARBI; RAHMAN, 2010; ELEKAR; WAGHMARE; PRIYADARSHI, 2015).

A classificação de dados pode ser implementada por diferentes algoritmos classificadores, onde cada um deles possui seus próprios critérios e particularidades para a classificação de amostras. Uma das consequências dessa diversidade de implementações consiste no problema da escolha do algoritmo ideal para a classificação de cada tipo de dados. No contexto da detecção de intrusões, um dos grandes desafios no uso de técnicas de classificação consiste na escolha do classificador mais promissor para cada um dos múltiplos tipos de ataques.

A construção de um classificador monolítico[1] o qual seria capaz de cobrir todas as variedades de dados não é viável (NGAI; XIU; CHAU, 2009; ELEKAR; WAGHMARE; PRIYADARSHI, 2015). Portanto, na literatura, diversos classificadores existentes vêm sendo utilizados isoladamente. No entanto, estudos apontam que o classificador que apresenta melhor desempenho na identificação de algumas classes de ataques (e.g, *teardrop*), muitas vezes, se apresenta inapropriado para outras (e.g., *back ddos*). Por conseguinte, a confiabilidade do resultado final é comprometida (AZIZ et al., 2013; BRITTO; SABOURIN; OLIVEIRA, 2014).

Atualmente, existem algumas soluções baseadas em múltiplos classificadores, visando solucionar o problema da escolha do classificador ideal (SABOURIN et al., 1993). Uma abordagem bastante utilizada consiste na combinação de classificadores, de forma a ampliar a acurácia dos resultados (KEVRIC; JUKIC; SUBASI, 2017). Todavia, apesar das melhorias apresentadas, esse tipo de abordagem possui limitações, tais como a escolha da melhor combinação de classificadores (TORRES-PEREIRA et al., 2014).

Uma interessante alternativa encontrada para a mitigação desse tipo de problema con-

[1] Classficador monolítico seria uma implementação onde todos os algoritmos classificadores seriam unificados em um único algoritmo final.

siste na construção de mecanismos que permitam a escolha do melhor classificador para cada cenário, de forma dinâmica. Nesse contexto, diferentes métodos foram propostos na literatura (KURZYNSKI et al., 2010; AZIZ et al., 2013). Contudo, as abordagens existentes apresentam limitações em cenários onde múltiplos classificadores, avaliados com alta acurácia, apresentam resultados divergentes para as amostras desconhecidas (MI; SIMA, 2016). Além disso, esse tipo de abordagem não garante a escolha, de maneira confiável, de classificadores em cenários onde a fonte de dados analisada não possui todas as informações necessárias para o devido treinamento dos classificadores (KEVRIC; JUKIC; SUBASI, 2017). Uma das possíveis soluções para esses casos consiste na correlação de dados entre múltiplas e heterogêneas fontes (BRITTO; SABOURIN; OLIVEIRA, 2014).

1.1 PROBLEMA DE PESQUISA ESTUDADO

Atacantes exploram novas possibilidades de ataques constantemente, tornando necessário que os sistemas detectores de intrusões tenham a robustez e flexibilidade necessária para identificar padrões de ataques ao observar múltiplas e heterogêneas fontes de dados. Com isso, se faz necessário o estudo de novos métodos que possibilitem o aprendizado de detectores e troca de conhecimento entre eles.

A mineração de dados, especialmente, oferece um significativo suporte aos IDS. Entretanto, existem questões em aberto que necessitam ser abordadas para que os resultados finais das classificações alcancem uma maior confiabilidade. Assim, especialmente, o problema da classificação dinâmica de classificadores frente à múltiplas possibilidades de ataques requer boas soluções. Portanto, em um cenário ideal, os sensores dispostos em diferentes fontes de dados capturam as informações que permitem uma análise mais ampla e uma contribuição mútua através da troca de conhecimento entre detectores que analisam tais fontes.

1.2 OBJETIVO

Este trabalho tem por objetivo propor uma arquitetura para detecção de intrusões por meio da seleção dinâmica do algoritmo mais promissor para tipo de amostra a ser analisada, de acordo com a similaridade entre essas amostras e as assinaturas de perfis de ataques conhecidos. Desse modo, um conjunto de classificadores tem seu desempenho medido a partir de assinaturas conhecidas, identificando-se qual deles possui maior aptidão à classificação das amostras

desconhecidas similares.

A seleção dinâmica de classificadores proposta tem sua confiabilidade avaliada em tempo de execução. Portanto, em situações onde classificadores apresentam resultados conflitantes ou de baixa confiabilidade, os detectores podem recorrer aos conselhos originados a partir das análises de outros detectores.

A troca de conselhos acontece, tipicamente, entre detectores que observam diferentes conjuntos de dados. Tais conjuntos podem conter informações capturadas a partir de múltiplas e heterogêneas fontes. Assim, uma rede heterogênea de detectores conselheiros é composta, com propósito de cooperação mútua. Com isso, a arquitetura permite o aprendizado de novos perfis de ataques com base em sugestões provindas de análises de diferentes tipos de informações.

1.2.1 Objetivos Específicos

A partir do objetivo geral, apresentado anteriormente, as contribuições da presente proposta podem ser elencadas através dos seguintes objetivos específicos.

a) Propor um método de escolha dinâmica de classificadores;

b) Propor uma rede de detectores conselheiros, permitindo a troca cooperativa de conselhos entre detectores;

c) Propor uma arquitetura para a escolha dinâmica de classificadores tolerante a conflitos, por meio da redes de conselheiros proposta.

1.3 ORGANIZAÇÃO DO DOCUMENTO

O restante do documento está organizado como segue: No Capítulo 2, é apresentada a fundamentação teórica. Em seguida, apresenta-se a revisão da literatura, no Capítulo 3, seguida da arquitetura proposta neste trabalho, no Capítulo 4. O Capítulo 5 apresentada os detalhes de implementação da arquitetura. Em seguida, o Capítulo 6 discute as experimentações realizadas, bem como seus resultados. Por fim, as conclusões e trabalhos futuros são apresentadas no Capítulo 7.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo tem por finalidade estabelecer a discussão dos principais conhecimentos abordados no tema deste trabalho, abordando tanto as populares possibilidades de ataques existentes quanto as técnicas aplicadas a mitigação desses ataques.

2.1 ATAQUES E INTRUSÕES

Em cenários reais de redes de computadores e sistemas computacionais, existem múltiplas maneiras de um atacante atingir uma vítima. Neste Capítulo serão apresentadas algumas das principais ameaças. As diferentes classificações de ataques e ferramentas utilizadas para sua execução serão apresentadas a seguir. Cada um dos itens discutidos visa gerar um dano específico ao usuário ou sistema final. Portanto, a discussão a seguir será dividida em quatros classes principais de ataques, as quais são definidas com base na forma de execução e nos respectivos impactos causados.

2.1.1 Denial-of-Service (DoS)

Ataques de Negação de Serviço, do inglês, *Denial-Of-Service* (DoS) consistem em uma grande ameaça à disponibilidade dos serviços prestados pelas suas vítimas, portanto requerem atenção contínua da academia e indústria a fim de sua mitigação (CAMPOS; LIMA, 2012). Tais ataques atuam esgotando os recursos da vítima, de modo torna-la indisponível para requisições legítimas ou impossibilitar a entrega de um serviço em tempo hábil.

Uma situação de negação de serviço pode ser provocada de diversas maneiras, usualmente, visando o congestionamento de rede ou esgotamento de recursos computacionais. O congestionamento de rede da vítima pode, ainda, ocorrer por meio de ataques de inundação (*flooding*), ou amplificação de pacotes. Já a sobrecarga de recursos computacionais, pode ser uma consequência de ataques de exploração de protocolos ou envio de pacotes malformados. A Figura 2.1, ilustra a taxonomia de subclasses de DoS baseada no estudo (GUPTA; BADVE, 2017). Esses e outros tipos de ataques serão detalhados a seguir:

a) **Ataques Flooding**: envolvem “zumbis” – máquinas controladas por atacantes – que fazem o envio de um largo volume de tráfego para a vítima a fim de congestionar a sua rede, esgotando a largura de banda disponível com tráfego IP. As consequências deste ataque

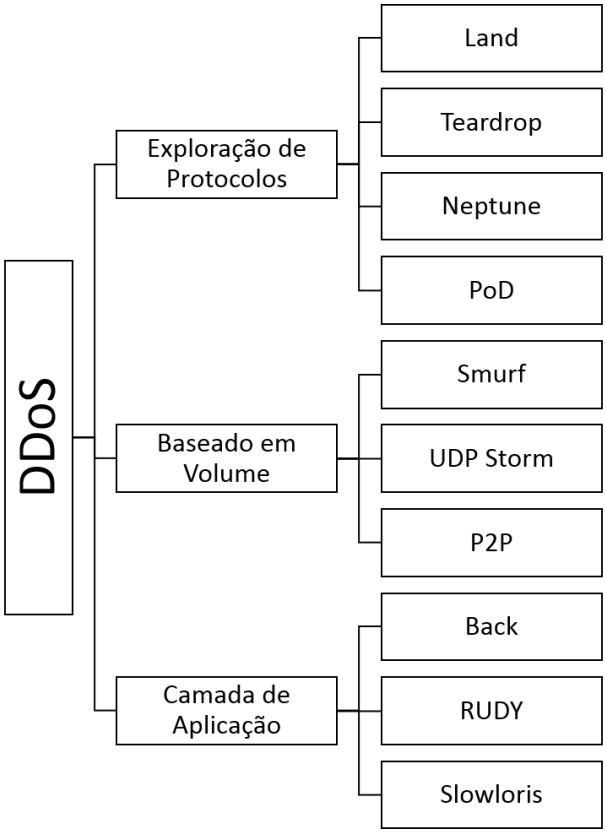


Figura 2.1: Taxonomia de ataques de Negação de Serviço baseda em (GUPTA; BADVE, 2017).

consistem em travamentos, demoras no sistema ou indisponibilidade no provimento de serviços. Tais ataques podem ser praticados através dos protocolos UDP (User Datagram Protocol) ou ICMP (Internet Control Message Protocol). Uma das variações deste tipo de ataque consiste no PoD (*Ping of Death*). Nesse ataque, são enviados inúmeras requisições malformadas, tipo ICMP Ping, ocasionando-se a negação do serviço no host atacado (SPECHT; LEE, 2004; SANTOS et al., 2016). Esse ataque não costuma funcionar em máquinas mais modernas. A situação de instabilidade se dá porque há estouro de *buffer* na máquina receptora, quando um pacote maior do que o tamanho máximo permitido de 65.535 bytes é recebido (AZEVEDO, 2012; DDOS ATTACK DEFINITIONS – DDOS-PEDIA, 2018);

b) **Ataques de amplificação**: são aqueles ataques que envolvem o envio de mensagens para um endereço IP de *broadcast* ou para muitos destinos, modificando o endereço IP de origem das requisições para o endereço IP da vítima primária. A consequência consiste no direcionamento das respostas para o alvo do ataque. Quando um endereço de broadcast

é especificado como endereço de destino, os roteadores replicam o pacote e enviam-no para todos os endereços IP dentro do seu intervalo. Dessa forma o tráfego é amplificado, reduzindo a largura de banda disponível da vítima, ou atingindo o seu esgotamento (SPECHT; LEE, 2004; OLIVEIRA ROCHA, 2013). Um exemplo de ataque pertencente a essa subclasse consiste no Smurf, o qual envolve o envio de pedidos de *eco ICMP* (ping) de tráfego para o endereço de broadcast de roteadores e outros dispositivos de rede em grandes redes de computadores com um endereço de origem falsificado, onde é inserido o endereço do alvo. Como o dispositivo que recebe a solicitação de eco ICMP original a transmite a todos os outros dispositivos conectados, cada um desses dispositivos envia uma resposta para o endereço da vítima, gerando uma alta taxa de tráfego ICMP, que pode causar a negação de serviço ou instabilidade na a rede de destino (AZEVEDO, 2012; SPECHT; LEE, 2004);

c) **Ataques de exploração de protocolos**: esses ataques se caracterizam por explorar alguma característica específica ou falha de implementação de algum protocolo instalado no sistema da vítima. Os principais ataques por exploração de protocolos fazem o uso indevido de pacotes *TCP SYN* (*Transfer Control Protocol Synchronize*) ou de pacotes *TCP PUSH+ACK*. O ataque conhecido como **Neptune** é um exemplo desta categoria. Ele opera através do envio de diversas requisições do tipo TCP-SYN, com objetivo de inundar o host vítima. Dessa forma, requisições válidas não conseguem obter uma resposta em tempo hábil (AZEVEDO, 2012; SPECHT; LEE, 2004). Outro exemplo consiste no ataque *Back*, onde são enviadas várias requisições a um servidor web Apache com diversas barras no endereço, por um atacante, tornando o tempo de resposta lento devido à dificuldade do processo de localização do documento pelo servidor. Em alguns ataques o número de barras pode chegar a 100 (AZEVEDO, 2012);

d) **Ataques de Pacotes Malformados** consistem na instrução de zumbis a enviarem pacotes adulterados para confundir a vítima, resultando no seu travamento. Uma das formas de executar este ataque consiste no envio de pacotes contendo o mesmo IP como origem e destino, como é o caso do ataque *Local Area Network Denial* (Land), em que o atacante envia pacotes TCP SYN falsificados, onde os IPs e portas origem e destino são idênticos. Quando a máquina de destino tenta responder, ela entra em *loop*, enviando respostas, repetidamente, para si mesmo. Isso pode, eventualmente, fazer com que a máquina da vítima falhe (DDOS ATTACK DEFINITIONS – DDOSPEDIA, 2018). Outro

exemplo desta categoria consiste no ataque *Teardrop*, o qual consiste na adulteração do campo "offset de fragmento", que indica a posição inicial, ou deslocamento, dos dados contidos em um pacote fragmentado relativo aos dados no pacote original. O objetivo dessa adulteração consiste em fazer com que a soma do deslocamento e tamanho de um pacote fragmentado apresentem divergência em relação ao próximo pacote fragmentado. Esta ação resulta na sobreposição de pacotes, o que impossibilita a vítima de remontá-los, ocasionando uma situação de negação de serviço (AZEVEDO, 2012).

2.1.2 Remote-To-Local (R2L)

Ataques da classe remoto para local (*remote-to-local* – R2L) correspondem a situações onde o atacante estabelece conexão remota com a máquina da vítima através da exploração de vulnerabilidades existentes, sem a necessidade de posse de uma conta de usuário válida. Portanto, ataques R2L representam um tipo de ameaça onde é possível o acesso não autorizado a conteúdo de terceiros (ZIN et al., 2015). Alguns ataques desta classe, bem como algumas das ferramentas conhecidas que viabilizam tal ataque são listados a seguir:

a) **Netcat**: esta ferramenta de ataque R2L exige a infecção da vítima com código malicioso a fim de permitir o acesso do atacante através da porta número 53. O programa Netcat funciona como um *backdoor* – recurso utilizado por diversos malwares para garantir acesso remoto ao sistema ou à rede infectada sem nenhum nome de usuário e senha (HOQUE et al., 2014);

b) **Ntfsdos**: o atacante ganha acesso ao console de máquinas WinNT através da execução desta ferramenta maliciosa. O programa monta os drivers de discos da máquina, permitindo o acesso a arquivos confidenciais por parte do atacante (HOQUE et al., 2014);

c) **Phf**: o ataque Phf consiste na exploração de uma vulnerabilidade existente nas primeiras versões do Apache Web. Através desta vulnerabilidade, o atacante é capaz de executar comandos na máquina da vítima (ZIN et al., 2015);

d) **Spy**: o ataque Espião, do inglês, Spy, tem por finalidade a exploração de vulnerabilidades em máquinas vítimas a fim de obter o acesso local a esta máquina. Para este fim, o atacante envia pacotes por meio da rede, através do protocolo TCP e serviço telnet (ZIN et al., 2015);

e) **Imap** O ataque Imap é semelhante ao ataque Spy, discutido anteriormente, onde o atacante tem por finalidade o acesso não autorizado à máquina da vítima. Para este fim, o atacante envia pacotes TCP através do serviço imap4 (ZIN et al., 2015);

f) **Warezmaster**: tem por finalidade a exploração de uma falha de sistema associada ao protocolo de transferência de arquivos (FTP). Este ataque ocorre quando um servidor FTP dá a permissão indevida de escrita à usuários não autorizados. Durante a exeucução do ataque, o usuário atacante cria um diretório oculto e faz o upload de "*wares*" – cópias de softwares ilegais – para o servidor (ZIN et al., 2015);

g) **Warezclient**: este ataque se dá como sequência do ataque *Warezmaster*, onde os *wares* já estão hospedados no servidor FTP atacado. Assim, os usuários podem fazer o download desses arquivos de forma a aparentar uma atividade normal (ZIN et al., 2015);

h) **Ftp-Write**: esse padrão de ataques consiste na exploração de falhas de configurações em serviços de FTP que não necessitam de usuário e senhas. Uma vez que tal vulnerabilidade for detectada, o atacante tem a possibilidade de ganhar permissão de escrita de arquivos no servidor e até mesmo controle remoto da máquina atacada.

2.1.3 User-To-Root (U2R)

Em ataques classificados como "Usuário para root", do inglês, *User-to-root* (U2R), parte-se da premissa de que o atacante já possui acesso a uma conta de usuário normal no sistema e é capaz de explorar vulnerabilidades para obter acesso de *root* ao sistema. Os ataques U2R podem ser originados a partir de várias formas, tais como captura de senha por software malicioso, ataques de dicionário/força bruta ou ainda ataques de engenharia social, onde o usuário é induzido a divulgar as suas credenciais de forma implícita ao atacante (REVATHI; MALATHI, 2014). A seguir, alguns dos ataques desta categoria serão listados:

a) **Buffer Overflow**: em tradução livre do inglês, o Estouro de *buffer*, consiste na cópia de muitos dados em *buffer* estático sem verificar se realmente existe espaço em disponível para o conteúdo copiado. Assim, uma vez que o espaço requerido ultrapassa tais limites disponíveis, ocorre o chamado estouro de *buffer*. Este é um dos principais ataques U2R (REVATHI; MALATHI, 2014);

b) **Loadmodule**: o ataque loadmodule tem por finalidade o acesso a um shell de root. Este ataque explora vulnerabilidades no sistema operacional SunOS 4.1 que utiliza o sistema de janelas Xnews. Devido a um erro de programação no módulo *loadmodule*, é possível a obtenção indevida de acessos administrativos (FARIAS, 2011);

c) **Ps**: os ataques Ps conduzem a uma condição de corrida explorável nas ações de um único programa, ou dois ou mais programas em execução simultaneamente. O invasor executa códigos arbitrários para acessar privilégios de root (REVATHI; MALATHI, 2014);

d) **Xterm**: o ataque Xterm explora um estouro de *buffer* na biblioteca Xaw distribuída no Redhat Linux e permite que um invasor execute instruções arbitrárias com privilégio de root (FARIAS, 2011);

e) **Rootkit**: ataques do tipo rootkits visam o comprometimento do kernel na máqina vítima a fim de possibilitar o acesso irrestrito a todos recursos do sistema. Assim, o atacante passa a ter controle sobre o alvo do ataque (RILEY; JIANG; XU, 2008);

f) **Perl** essa categoria de ataques tem por finalidade a exploração de falhas na linguagem de scripts *Pearl*. O *suidpeal* – que consiste em um módulo da linguagem Perl – contém um erro de programação que permite a qualquer usuário do sistema o acesso de administrador (FARIAS, 2011) (JIRAPUMMIN; WATTANAPONGSAKORN; KANTHAMANON, 2002).

2.1.4 Probing

Esta classe de ataques tem por finalidade estabelecer o reconhecimento (Probing) dos futuros alvos, de forma a identificar as vítimas vulneráveis para a execução de ataques com dano direto à vítima (JIRAPUMMIN; WATTANAPONGSAKORN; KANTHAMANON, 2002) (PALIWAL; GUPTA, 2012). A seguir, algumas das ferramentas que possibilitam ataques probing serão brevemente discutidas:

a) **IPSweep**: tem por finalidade o reconhecimento de alvos vulneráveis, arquiteturas de redes de computadores, através da execução de varreduras. Esse é um tipo de ataque que não gera efeitos imediatos contra o alvo, entretanto, ele é capaz de gerar informações para futuros ataques (FARIAS, 2011) (JIRAPUMMIN; WATTANAPONGSAKORN; KANTHAMANON, 2002);

b) **Mscan**: esta categoria de ataques, que também visa o reconhecimento das possíveis vítimas, executa consultas em servidores de DNS a fim de localizar vulnerabilidades específicas nas máquinas com os endereços IP descobertos a partir desses servidores (FARIAS, 2011);

c) **Nmap**: esta ferramenta possibilita a varredura e reconhecimento de máquinas e serviços, que pode ser utilizada para a finalidade de identificação de vulnerabilidades. O principal objetivo, usualmente, é a descoberta de portas abertas na máquina da vítima. Dessa forma, é possível a identificação dos serviços que estão atualmente ativos, além de versões dos softwares presentes. Com isso, o atacante pode aplicar ataques específicos às vulnerabilidades existentes (FARIAS, 2011) (PALIWAL; GUPTA, 2012);

d) **Saint**: a ferramentas de rede integradas ao administrador de segurança, do inglês, *Security Administrator Integrated Network Tools* (SAINT) foi desenvolvida para administradores de rede, sistemas e profissionais da segurança da informação. Entretanto, nas mãos de atacantes, essa ferramenta possibilita a geração de serviços ativos em máquinas remotas. Assim como as possibilidades de *probbing* descritas anteriormente, esta é uma importante forma de geração de informação (FARIAS, 2011) (PALIWAL; GUPTA, 2012);

e) **Satan**: de maneira semelhante à ferramenta discutida anteriormente, a ferramenta de análise de redes para administradores de segurança, do inglês, *Secutiry Administrator Tool for Analyzing Networks* (Satan), também permite a descoberta de serviço e vulnerabilidades que podem vir a tornar-se futuros alvos (FARIAS, 2011) (PALIWAL; GUPTA, 2012).

2.2 DETECÇÃO DE INTRUSÕES

A detecção de intrusões tem como finalidade identificar a presença de ameaças que visam atingir redes de computadores ou sistemas computacionais. Para este fim, os sistemas detectores de intrusões (*Intrusion Detection System* - IDS), devem ser implementados de modo a contemplar as capacidades de ler dados de algum tipo de fonte e processa-los.

Esta Seção, se concentra nas técnicas computacionais utilizadas para o processamento e análise de dados que visam distinguir as condições de ataques das condições normais. Dentre as técnicas existentes, fazem parte do escopo deste trabalho aquelas classificadas como mineração de dados. Tais técnicas tem por finalidade a identificação de padrões que representam

informações úteis e não triviais, a fim de viabilizar a descoberta de conhecimento em banco de dados (CAMPOS; LIMA, 2012). Assim, no contexto de grandes volumes de dados, busca-se extrair conhecimentos de alto nível a partir de dados de baixo nível (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b).

Em geral, a detecção de intrusões através de técnicas de mineração de dados pode ocorrer de duas formas: supervisionada e não supervisionada (TAN et al., 2006). A seguir, ambas serão discutidas.

2.2.1 Detecção Supervisionada

A detecção de intrusões por meio de aprendizado supervisionado é aplicável para cenários onde já existem assinaturas representativas das classes em que os dados pertencerão. A classificação de dados é uma técnica dessa categoria, pois um conjunto de dados com classes conhecidas é utilizado para treinamento dos classificadores. Somente após a construção de um modelo de classificação é que dados desconhecidos podem ser classificados (FAYYAD; PIATETSKY-SHAPIRO; SMYTH, 1996b; TAN et al., 2006; SPECHT; LEE, 2004). Alguns dos populares classificadores existentes na literatura são apresentados a seguir:

a) **K-nearest Neighbours (KNN):** Consiste em um classificador que se baseia nas classes de treinamento para predizer a classe de teste mais próxima para cada amostra. Para tanto, a distância entre pontos de treinamento e teste devem ser calculados para definir qual o "vizinho mais próximo" dentre as amostras pertencentes ao conjunto de treinamento. Por padrão, esse cálculo se baseia na distância euclidiana, podendo-se optar por outros métodos semelhantes (WITTEN et al., 2016);

b) **J48:** Consiste em uma implementação em Java do algoritmo de árvore de decisão C4.5 (CAMPOS; LIMA, 2012). Este classificador constrói árvores a partir de um conjunto de dados de treinamento, utilizando o conceito de entropia (LIMA et al., 2012) para predizer as classes dos objetos a serem classificados (WITTEN et al., 2016);

c) **REPTree:** Consiste em um algoritmo de aprendizagem de árvore de decisão rápida que é baseado no princípio de computar o ganho de informação com a utilização do método de entropia, a fim de minimizar o erro resultante da variância durante o processo de classificação (WITTEN et al., 2016; WITTEN; MINING, 2005);

d) **Naive Bayes:** Fornece uma abordagem simples, com semântica clara, para representar, usar e aprender o conhecimento probabilístico. O método é projetado para uso em tarefas de indução supervisionadas e funciona com base no teorema bayesiano (JOHN; LANGLEY, 1995);

e) **NB Tree:** Segmenta os dados usando uma árvore de decisão univariada, tornando a segmentação mais fácil de entender. Cada folha é um classificador Naive-Bayes, descrito anteriormente (KOHAVI, 1996);

f) **Random Tree:** Constrói uma árvore que considera K atributos escolhidos aleatoriamente em cada nó. Não executa nenhuma poda – diferentemente do REPTree. Possui também a opção de permitir a estimativa de probabilidades de classe (WITTEN et al., 2016).

2.2.2 Detecção Não-Supervisionada

A detecção de intrusões supervisionada consiste na identificação de anomalias comportamentais. Tais anomalias podem ser identificadas através da criação de perfis de ataque e tráfego normal, ou ainda, baseando-se em limiares aceitos como normais ou anômalos (BROWN; STALLINGS, 2017).

A técnica de clusterização, por exemplo, tem por finalidade executar a segregação das amostras em diferentes grupos, ou *clusters*. Tipicamente, cada um desses agrupamentos, tende a concentrar amostras de um determinado perfil, como de tipos específicos de ataques. Assim, é possível a detecção de amostras que representam intrusões. A seguir serão listados e discutidos alguns dos algoritmos de *clustering* existentes.

a) **K-means**: este método objetiva particionar o conjunto total de amostras observáveis em um número de grupos previamente especificado. Tal número é chamado de *k*. Durante o particionamento, cada amostra passa a pertencer ao *cluster* com *centróide* - o qual é definido pela amostra média mais próxima (MACQUEEN et al., 1967). Isso resulta em uma divisão do espaço de dados em um Diagrama de Voronoi (AURENHAMMER, 1991);

b) **DBSCAN**: o algoritmo de clusterização espacial baseada em densidade de aplicações com ruído, do inglês, *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN), é um algoritmo de agrupamento de dados proposto por (ESTER et al., 1996). É um

algoritmo de agrupamento baseado em densidade porque encontra uma série de *clusters* a partir da distribuição de densidade estimada dos nós correspondentes;

c) **Farthest First**: este algoritmo é um variante do algoritmo K-means, que coloca cada centro de cluster por sua vez no ponto mais distante dos centros de cluster existentes. Este ponto deve estar dentro da área de dados (SHARMA; BAJPAI; LITORIYA, 2012).

2.3 DISCUSSÃO

Conforme relatado neste Capítulo, existem inúmeras formas de danos que podem ser ocasionados às vítimas por meio de ações maliciosas de indivíduos atacantes. Em cenários reais, essa situação tem se mostrado preocupante, onde novas ameaças surgem constantemente. Tais ameaças visam atingir os todo tipo de recurso, seja ele computacional ou de rede, de forma a prejudicar o uso e oferta de serviços.

Por outro lado, existem também inúmeras técnicas que podem ser utilizadas, aprimoradas e combinadas a fim de mitigar as ameaças conhecidas, através de análises supervisionadas, e também aquelas que ainda não se tem conhecimento, através de análises não supervisionadas.

Portanto, o uso do melhor conjunto de técnicas para cada cenário é um desafio que motiva esforços de pesquisas por parte da academia e também da indústria, devido a criticidade dos impactos apresentados por meio dessas ameaças.

3 TRABALHOS RELACIONADOS

Neste capítulo serão discutidos os trabalhos relacionados. Primeiramente, será discutido o estado da arte em arquiteturas que analisam únicas fontes, as quais incluem o uso de seleção dinâmica de classificadores. Em seguida, estudos que exploram arquiteturas colaborativas serão discutidos.

3.1 ANÁLISE DE FONTES HOMOGÊNEAS

Uma das técnicas comumente utilizadas na análise de fontes homogêneas para a detecção de intrusões consiste na classificação de dados. Por meio da classificação é possível a predição de ameaças com base em assinaturas conhecidas. Entretanto, um dos principais desafios na utilização desta técnica consiste na escolha do método de classificação mais apropriado frente à múltiplas possibilidades de ameaças. A implementação de um classificador monolítico capaz de abranger todas as variações de dados não é viável (BRITTO; SABOURIN; OLIVEIRA, 2014). Além disso, estudos demonstram que sistemas que fazem o uso de múltiplos classificadores tendem a apresentar resultados superiores (AZIZ et al., 2013; KEVRIC; JUKIC; SUBASI, 2017). Portanto, esta Seção discute soluções propostas, a fim de definir o estado da arte na seleção dinâmica de classificadores em fontes de dados homogêneos.

Em (SABOURIN et al., 1993), uma das abordagens pioneiras na seleção dinâmica de classificadores é introduzida. Tal abordagem consiste em um método de ranqueamento de classificadores baseado na corretude de seus resultados, a qual é estimada a partir de parte da base de treinamento. Uma versão simplificada desse trabalho é proposta em (WOODS et al., 1997), onde os classificadores são ranqueados com base em sua acurácia na classificação das amostras vizinhas conhecidas. Entretanto, em ambas as abordagens, existe um problema em comum: o classificador selecionado pode apresentar bom desempenho para amostras vizinhas e ainda falhar para a amostra desconhecida. Todavia, tal técnica não considera o uso de múltiplas fontes, nem mesmo um critério para solução de conflitos eventualmente apresentados por classificadores de mesma posição no ranking.

No trabalho de (KURZYNSKI et al., 2010), os classificadores são avaliados de duas formas. A primeira delas define a competência de cada classificador com base no suporte que o mesmo dá para a classe correta de cada amostra de validação. A segunda, por sua vez, se dá de

acordo com a probabilidade de classificação correta de uma amostra desconhecida, calculada a partir do resultado médio de suporte dado para uma dada classe. Portanto, através dessas metodologias, pode-se estimar o melhor classificador para a classificação de uma determinada classe. Contudo, em cenários reais, ataques de múltiplas classes podem ocorrer, tornando esse processo oneroso.

Em (MI; SIMA, 2016) é proposto um método de seleção de classificadores baseado em clustering e média ponderada. Com isso, é possível medir o desempenho de cada classificador para cada cluster formado a partir de uma base de amostras conhecidas. Então, o classificador com melhor desempenho médio e o classificador com melhor desempenho para o cluster vizinho mais próximo da amostra desconhecida são selecionados para classificá-la. Dentre os dois classificadores selecionados, o que apresentar maior confiabilidade, através de uma matriz de confusão de resultados, deve tomar a decisão final. Todavia, nesse cenário, conflitos podem ocorrer em ambas as seleções. Ainda, um terceiro conflito pode ocorrer na comparação de confiabilidade.

O trabalho de (BARBARA et al., 2001) propõe um Método de Análise e Mineração de Dados de Auditoria (ADAM), onde a anomalia a detecção não supervisionada é seguida pela detecção supervisionada. ADAM usa uma combinação de regras de associação e um método de classificação para detectar ataques. Primeiramente, o modelo de detecção de anomalia que usa a mineração de regras de associação localiza conexões de tráfego suspeitas e passa as conexões ao modelo de detecção que é baseado em classificação. Em seguida, esse modelo classifica as conexões suspeitas como normais (alarme falso do modelo de detecção de anomalia), conexões de ataques conhecidos e conexões de ataque desconhecidas. No método ADAM, as conexões que não podem ser confiadamente classificadas como normais ou ataques conhecidos são classificadas como ataques desconhecidos. No entanto, o número de alarmes falsos está propenso a ser alto, devido à decisão que é tomada em casos de baixa confiabilidade.

O trabalho de (REHAK et al., 2009) propõe um framework distribuído para otimizar comportamentos de sistemas detectores de intrusões homogêneos. Os detectores são testados isoladamente por meio de injeção local de eventos vindos de uma base de ataques conhecida a fim de determinar sua especialidade. Além disso, cada nó detector pode comparar feedbacks de outros nós a fim de tomar conhecimento da respectiva especialidade. Desse modo, a acurácia tende a ser acrescida. Todavia, cenários onde múltiplos IDS especialistas apresentam resultados conflitantes não são tratados.

As abordagens híbridas paralelas utilizam um modelo de detecção de anomalia, de maneira não supervisionada, e um modelo de detecção de perfis de ataques, de maneira supervisionada. O trabalho (DEPREN et al., 2005) sugere um sistema inteligente de detecção de intrusão híbrida que consiste em um modelo de detecção de anomalia, um modelo de detecção de perfis de ataques e um sistema de suporte à decisão. Assim, os autores implementaram o modelo de detecção de anomalia com um mapa de auto-organização e o modelo de detecção de uso indevido com uma árvore de decisão. Cada modelo foi treinado de forma independente e, em seguida, o sistema de suporte à decisão simplesmente combina os resultados de classificação de ambos os modelos. Entretanto, fontes heterogêneas não são exploradas.

Em (KIM; LEE; KIM, 2014) é proposto um novo método de detecção de intrusão híbrida que integra hierarquicamente um modelo de detecção supervisionado e um modelo de detecção de anomalia (não supervisionado) em uma estrutura de decomposição. Primeiramente, um modelo de detecção de supervisionado é construído com base no algoritmo de árvore de decisão C4.5 e, em seguida, os dados de treinamento normais são decompostos em subconjuntos menores usando o modelo. Assim, vários modelos máquina de vetor de suporte (SVM) de uma classe são criados para os subconjuntos decompostos. Como resultado, cada modelo de detecção de anomalia não usa apenas indiretamente a informação de ataque conhecida, mas também constrói os perfis do comportamento normal. Entretanto, tal método não explora a correlação de fontes heterogêneas.

O trabalho de (FUNG; ZHU, 2016) propõe a colaboração de detectores de intrusões, os quais são implantados em diferentes subredes. Desse modo, cada detector é capaz de executar suas próprias análises e também consultar resultados de outros detectores que possuam diferentes especialidades ou bases de conhecimento. Entretanto, todos os IDS da rede analisam fontes homogêneas e os resultados são redundantes do ponto de vista global.

Portanto, os trabalhos apresentados nesta seção apresentam importantes evoluçõs em relação ao uso de um único classificador. Entretanto, a resolução de conflitos permanece desafiadora. Ademais, a possibilidade da ocorrência de falsos positivos ou negativos é proporcional à relevância da fonte de dados analisada. Assim, a análise de uma única fonte não é suficiente para a detecção de todos os tipos de ataques existentes.

3.2 ANÁLISE DE FONTES HETEROGÊNEAS

Atualmente, um massivo volume de informações é gerado por diferentes fontes. Muitas vezes, tais dados são ricos em informações que podem ser utilizadas para detecção de situações anômalas, tais como intrusões. Todavia, muitos detectores analisam somente dados coletados a partir de uma única fonte. A interconexão e colaboração entre sistemas que fazem análise de múltiplas e heterogêneas fontes ainda é pouco explorada (ZUECH et al., 2015). Esta Seção tem por finalidade discutir o estado da arte da cooperação entre detectores de intrusões, especialmente, onde múltiplas e heterogêneas fontes são observadas.

Em (HWANG; LIU; CHEN, 2004), uma arquitetura para correlação de alertas heterogêneos é proposta, de modo que um Sistema de Detecção de Anomalias (ADS) é implementado a fim de analisar eventos e estatísticas de tráfego, para que anomalias recorrentes tenham suas assinaturas registradas. Com base nas assinaturas geradas pelo ADS, o detector Snort (ROESCH et al., 1999) é reconfigurado, possibilitando a detecção de padrões previamente desconhecidos. Entretanto, as assinaturas geradas estão propensas a serem falsos positivos, comprometendo a confiabilidade do detector. Essa limitação poderia ser mitigada por meio da consulta um detector especialista, por exemplo.

Em (GANAME et al., 2008), eventos *syslog* são correlacionados com alertas gerados pelo Snort (ROESCH et al., 1999). Tal correlação se dá baseada no contexto em que eventos e alertas são gerados, o qual é definido por meio de uma janela de tempo. Assim, registros com sequências predefinidas, vindas de ambas as fontes, permitem a detecção de ataques mais complexos, que possuem múltiplas etapas. Todavia, segundo (ZUECH et al., 2015), a estrutura original da arquitetura é propensa a negação de serviço e tem problemas de desempenho em sua versão distribuída.

O trabalho de (MARCHAL et al., 2014) propõe uma arquitetura para detecção de intrusões através da correlação de múltiplas e heterogêneas fontes (HTTP, DNS, IP flow, etc.) com base em seus endereços de IPs. Soluções existentes de Big Data são aplicadas a fim de prover escalabilidade. Todavia, a tomada final de decisão com base nos dados correlacionados depende da definição de regras. Assim, a eficiência da arquitetura depende do nível de conhecimento de um administrador especialista.

Em (CHEN; LAI; YOUNG, 2016) são analisados logs nos formatos Netflow e Syslog, a fim de identificar padrões sequenciais para um mesmo endereço de IP. Essa abordagem tem

por finalidade detectar ataques que são aplicados por meio de múltiplas etapas, onde ações maliciosas podem ser identificadas em diferentes momentos. Um exemplo desse tipo de análise se dá através da correlação de logs de erros de autenticação com logs de falha ao tentar acesso ao usuário root do sistema operacional. Todavia, o administrador de rede deve fazer o monitoramento e tomar medidas de prevenção. Portanto, esse tipo de abordagem apresenta a mesma limitação discutida anteriormente: a forte dependência da ação humana para tomada de decisões.

3.3 DISCUSSÃO

A seleção dinâmica de classificadores resolve, parcialmente, o problema da escolha do melhor classificador para cada contexto. Entretanto, ainda existem limitações que necessitam ser melhores abordadas, tais como a resolução de conflitos entre classificadores. Além disso, existe a carência de trabalhos que permitam a colaboração de detectores que analisam diferentes tipos de fontes de dados. Esse tipo de colaboração pode alavancar a confiabilidade dos resultados dos detectores. Desse modo, a adoção de conceitos de arquiteturas colaborativas aplicada à seleção dinâmica de classificadores é vista como uma solução para aumentar a confiabilidade no resultado final do detector, resolvendo os conflitos existentes. Todavia, tal união de conceitos ainda não foi explorada na literatura.

A Tabela 3.1 sumariza a análise dos trabalhos discutidos anteriormente. Na coluna **Coorrelação**, foram destacados os trabalhos que possuem algum tipo de coorelação de fontes heterogêneas de dados. Além disso, a metodologia de cada um dos trabalhos é mencionada na coluna **Metodologia**.

Tabela 3.1: Sumarizaçao dos trabalhos relacionados apresentados anteriormente.

Referência	**Coorelação**	**Metodogoliga**
(SABOURIN et al., 1993)	Nenhuma	Ranqueamento de Classificadores
(WOODS et al., 1997)	Nenhuma	Ranqueamento de Classificadores
(BARBARA et al., 2001)	Nenhuma	Geração de Assinatuas e Classificação
(HWANG; LIU; CHEN, 2004)	Sim	Reconfiguração de IDS Baseado em ADS
(DEPREN et al., 2005)	Nenhuma	Árvores de Decisão
(GANAME et al., 2008)	Sim	Correlação de Eventos com IDS Snort
(REHAK et al., 2009)	Nenhuma	Descentralização de Detecção
(KURZYNSKI et al., 2010)	Nenhuma	Seleção de Classificador por Probabilidade
(KIM; LEE; KIM, 2014)	Nenhuma	Máquina de Vetor de Suporte (SVM)
(MARCHAL et al., 2014)	Sim	Baseada na Definição de Regras
(MI; SIMA, 2016)	Nenhuma	Seleção de Classificador via *Clustering*
(CHEN; LAI; YOUNG, 2016)	Sim	Análise de Eventos Sequênciais
(FUNG; ZHU, 2016)	Nenhuma	Reconfiguração de IDS Baseado em ADS

4 ARQUITETURA PROPOSTA

Atualmente, a detecção de intrusões é uma área que deve evoluir constantemente a fim de acompanhar as inovações em ataques praticados por usuários maliciosos. Nesse contexto, conforme abordado no Capítulo 3, existem esforços de pesquisa que buscam refinar o processo de classificação e amostras desconhecidas. Trabalhos nessa linha visam selecionar dinamicamente o classificador mais promissor para a análise de cada tipo de perfil de ameaças. Entretanto, por mais sofisticado que sejam os algoritmos de classificação, o resultado final sempre irá depender da qualidade dos dados em que se está analisando.

Existe, portanto, uma outra linha de pesquisa que concentra-se na análise de dados provindos de múltiplas e heterogêneas fontes de dados. Esse tipo de abordagem amplia a quantidade de características em que se é possível a análise em prol da detecção de intrusões. Todavia, as soluções existentes na literatura, que são capazes de efetuar a análise de múltiplas fontes de dados, carecem de um método de decisão onde seja possível identificar quais das fontes analisadas apresenta maior confiabilidade. Consequentemente, a taxa de alarmes falsos tende a aumentar, diminuindo a acurácia dos resultados.

O objetivo da arquitetura proposta neste Capítulo consiste em estabelecer um método que aumente a precisão dos resultados através da seleção dinâmica de classificadores e da troca de conselhos entre detectores que analisam fontes de dados heterogêneos. Com isso, resultados provindos da análise de diferentes fontes são confrontados com base na acurácia do algoritmo classificador com melhor desempenho para a respectiva fonte. Por fim, detectores que apresentam confiabilidades superiores podem prestar conselhos aos detectores que analisam fontes mais pobres em informações sobre intrusões. Cabe ainda ressaltar que as questões relacionadas ao desempenho computacional e escalabilidade não fazem parte do escopo deste trabalho, os quais serão tratados em trabalhos futuros.

4.1 VISÃO GERAL

A arquitetura proposta tem como propósito permitir a extração e análise de *features* a partir de qualquer tipo de fonte de dados, através de técnicas supervisionadas (classificação) e não-supervisionadas (*clustering*) de mineração de dados. Essas *features* consistem em atributos, como contagens de números de pacotes enviados por meio de um *host*, ou contagem de eventos

de um determinado tipo. A Figura 4.1 ilustra a visão geral da arquitetura.

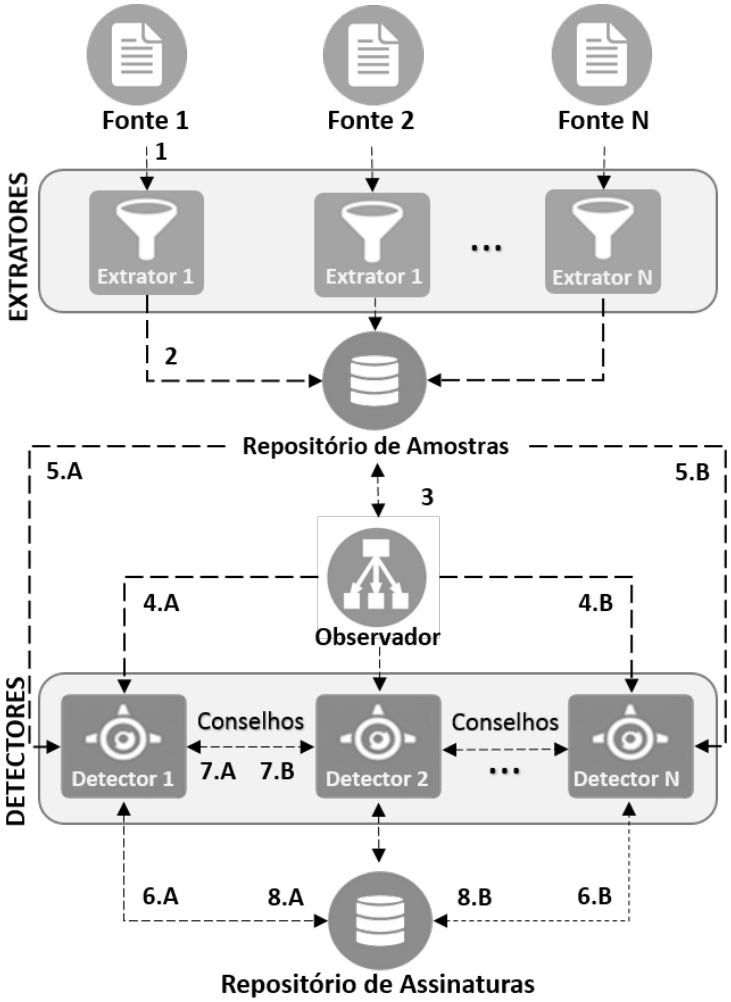


Figura 4.1: Visão geral da arquitetura proposta.

Inicialmente, os componentes **extratores** fazem a captura dos dados brutos a partir da fonte em que são designados a analisar (1). Uma vez capturados, tais dados passam por um processo de transformação, onde informações podem ser agregadas ou sumarizadas de modo a gerar um conjunto de *features*. Cada uma dessas *features* representa um tipo de informação, que foi transformada e está pronta para ser consumida pelos **detectores**. Portanto, tais informações transformadas são persistidas, temporariamente, no **repositório de amostras desconhecidas** (2).

Os **motores de detecção**, também referidos como **detectores**, são notificados, por meio do componente **observador**, sempre que novos dados são persistidos no **repositório de amostras desconhecidas** (4). Para este fim, esse componente faz consultas periódicas a tal repositório (3), viabilizando a sincronização de amostras coletadas pelos **extratores** com os **detectores**

que as consomem. Uma vez notificados, os **detectores** recuperam as novas amostras (5) e iniciam o processo análise dessas.

A análise executada pelos **detectores** consiste, basicamente, na aplicação técnicas e mineração de dados, por meio de algoritmos classificadores e de *clustering*. Nesse contexto, um conjunto de classificadores é avaliado através da classificação das assinaturas conhecidas mais similares à amostra desconhecida, as quais são recuperadas do **repositório de assinaturas** (6). Tal similaridade é medida através de um algoritmo de *clustering*, portanto, amostras pertencentes a um mesmo *cluster* são consideradas similares. Assim, é possível a seleção dinâmica do classificador mais promissor para a cada perfil de amostras.

Por fim, a principal contribuição da arquitetura proposta se dá na introdução da possibilidade de troca de conselhos (7), onde, os **detectores** cooperam entre si, através da consulta aos resultados correlacionados, oriundos das análises de outras fontes. Desse modo, detectores conselheiros podem sugerir resultados que aumentam a confiabilidade e solucionam conflitos que ocorrem em **detectores** solicitantes. Tais aconselhamentos podem originar novas assinaturas, que as quais são armazenadas no **Repositório de Assinaturas** (8), permitindo o constante aprendizado e evolução da capacidade de detecção dos **detectores**.

A seguir, na Seção 4.2, os demais detalhes de cada um dos componentes e repositórios da arquitetura serão detalhados.

4.2 COMPONENTE EXTRATOR

O componente **extrator** tem por finalidade a extração de informações a partir de fontes de dados brutos a fim de executar a transformação dos mesmos, de modo a gerar *features* representativas que permitam o estabelecimento de perfis de amostras. Uma *feature* pode ser representada de maneira descritiva, como o protocolo utilizado em uma conexão, ou quantitativa, como o número de bytes transferidos nesta conexão. Uma amostra, por sua vez, consiste em um conjunto de *features*.

Uma vez que o componente **extrator** conclui o processo de transformação, o mesmo deve persistir as amostras geradas no **repositório de amostras desconhecidas**. Neste contexto, existem diferentes fontes de dados ricos em informações sobre o estado das redes de computadores e sistemas computacionais. Assim, é importante que a heterogeneidade de informações seja suportada para que a capacidade de detecção seja potencializada. Portanto, as informações são armazenadas no formato *JavaScript Object Notation* (JSON), o qual unifica informações

provindas de diferentes tipos de fontes que são capturadas em diferentes formatos.

A Figura 4.2 ilustra o processo de extração de informações a partir de fontes heterogêneas, seguido pela transformação e persistência.

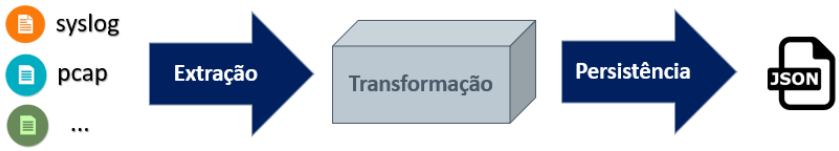


Figura 4.2: Componente Extrator de Informações.

Tipicamente, os perfis de ataques são observados nas fontes de dados que são afetadas pela ação dos atacantes. Por esse motivo, a definição de quais fontes que devem ser analisadas deve levar em conta o propósito de cada detecção. Ataques que esgotam recursos de rede, por exemplo, podem ser observados através da captura e análise de pacotes, a partir do formato *network capture* (*pcap*). Existem ataques, entretanto, que são perceptíveis através da análise de *logs*, como aqueles recuperados através do padrão *syslog*. De modo geral, a arquitetura proposta suporta a análise de qualquer tipo de informação desde que a extração de informações seja devidamente implementada.

Portanto, o componente **extrator** é o principal responsável pelo suporte à heterogeneidade de fontes analisadas. Uma vez que a transformação de dados é concluída, estes são persistidos no **repositório de amostras desconhecidas**. Tal repositório será discutido a seguir, na Seção 4.3.

4.3 REPOSITÓRIO DE AMOSTRAS DESCONHECIDAS

Esse repositório funciona como balanceador de carga, onde as amostras desconhecidas, coletada pelos **extratores**, são armazenadas, temporariamente, e aguardam a análise por meio dos respectivos **motores de detecção**. Sendo assim, a escalabilidade da solução proposta depende do modo em que as informações são armazenadas e recuperadas, pois em cenários onde grandes volumes de dados são processados, gargalos no armazenamento podem comprometer a arquitetura.

Conforme mencionado anteriormente, na Seção 4.2, as informações são armazenadas nos repositórios de persistência no formato JSON. A utilização deste formato possibilita que todas as amostras sejam persistidas e fornecidas com interoperabilidade para os demais com-

ponentes da arquitetura, responsáveis pela análise de dados. Ademais, múltiplas subdivisões podem existir, de forma a segmentar a persistência de acordo com cada perfil amostral ou fonte de dados que originaram tais amostras.

4.4 REPOSITÓRIO DE ASSINATURAS:

Na prática, assinaturas e amostras possuem a mesma estrutura, exceto pelo fato de que uma assinatura deve possuir um campo adicional, o qual demarca a qual classificação pertence a amostra assinada. Portanto, toda amostra pode ser transformada em uma assinatura, após ser processada e devidamente classificada. Todavia, o propósito de utilização de uma assinatura se difere do propósito de consumo de amostras. Uma vez que as assinaturas já possuem classes definidas, as mesmas são usadas para o treinamento e validação de classificadores que visam a classificação de amostras desconhecidas similares.

Portanto, inicialmente, este repositório deve ser pré-carregado a fim de permitir o treinamento de classificadores e também a avaliação dos mesmos. Essa carga inicial pode ser feita através de uma base de assinaturas existente na literatura, tal como o *dataset* KDD CUP 99' (CUP, 1999). Além disso, novas classificações de amostras desconhecidas podem alimentar a base de assinaturas em tempo de execução, de forma a permitir que novos perfis de ataques descobertos sejam mapeados e sirvam como modelo para a detecção de ameaças similares.

4.5 MOTORES DE DETECÇÃO

Uma vez que múltiplas e heterogêneas fontes de informações são suportadas na extração de *features*, tais informações devem ser analisadas, isoladamente, de acordo com seu propósito, a fim de possibilitar a detecção de intrusões. A arquitetura proposta suporta a execução de múltiplos **motores de detecção** em simultâneo, os quais devem ser ajustados de acordo com o propósito de suas análises. Alguns dos ajustes básicos consistem na especificação das fontes e seleção de um conjunto de *features* a serem analisadas pelo **detector**. Portanto, a recuperação de informações, a partir do **repositório de amostras desconhecidas** e do **repositório de assinaturas**, se dá através de *referências* a segmentos específicos do repositório.

Assim, os **motores de detecção** acessam tais amostras desconhecidas e, após a análise, armazenam o resultado final como assinatura, podendo-se emitir, neste instante, um alerta para o administrador do sistema. Basicamente, existem três etapas fundamentais: Filtragem de *Fea-*

tures, Clusterização de Assinaturas e Seleção de Classificadores. Adicionalmente, uma quarta etapa é introduzida: consulta a uma rede de **detectores conselheiros**.

A consulta à **rede de conselhos** tem como finalidade possibilitar a obtenção de conselhos provindos de **detectores conselheiros** que analisam fontes heterogêneas à fonte analisada pelo motor que efetuar a solicitação. Um conselho pode ser solicitado de forma a resolver conflitos entre classificadores especialistas, ou ainda, aumentar a confiabilidade das análises que não possuem resultados suficientemente confiáveis.

A seguir, tais etapas serão apresentadas.

4.5.1 Filtragem de Features

As diferentes fontes de dados que podem ser exploradas pelos coletores de amostras produzem uma série de informações que podem ou não ser significativas para o propósito de análise, o qual é definido de acordo com as configurações do respectivo **motor de detecção**. Portanto, os motores de detecção devem ser capazes de executar filtros a fim de consumir os subconjuntos de *features* que melhor representam os perfis das ameaças alvos da detecção.

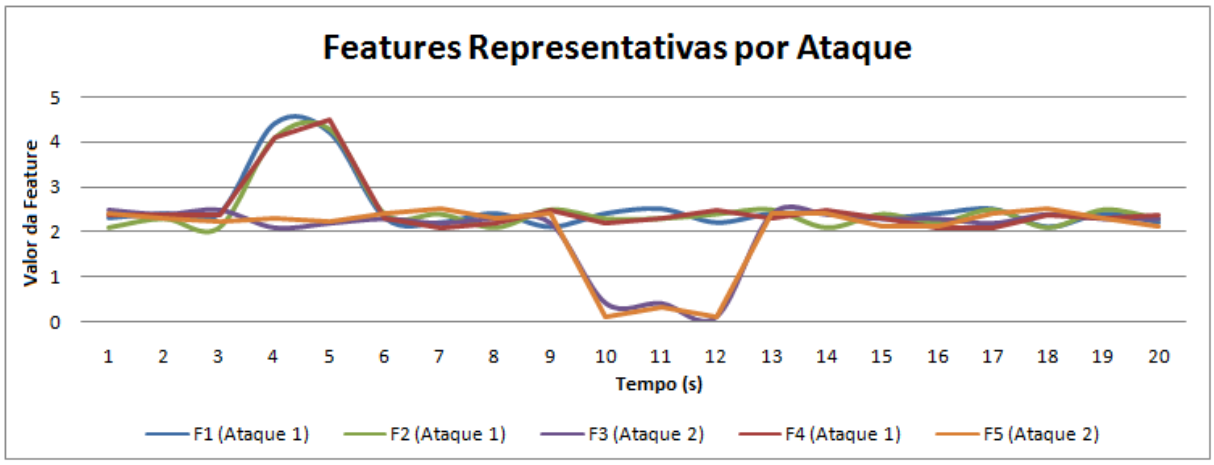


Figura 4.3: Ilustração hipotética de *features* relevantes para ataques específicos.

A Figura 4.3 ilustra uma situação hipotética, onde as *features F1*, *F2* e *F4* sofrem alterações em um cenário demarcado entre os instantes 3 a 6, na barra horizontal inferior da figura. Nesta anomalia, as demais *features* permanecem com os seus comportamentos padrões, sem apresentar distorções. Em um segundo momento, as *features F3* e *F5* apresentam valores anômalos. Tais deformidades comportamentais, em cenários reais, podem representar tipos específicos de ataques. Assim, é possível definir quais *features* devem ser levadas em consideração

para cada tipo de análise.

Portanto, dado que em cenários reais existem inúmeros perfis de ameaças, a seleção das *features* representativas para cada perfil é um aspecto importante a ser considerado. Tal importância se justifica porque a seleção de *features* apresenta impacto direto nos resultados da análise, uma vez que a análise de *features* irrelevantes pode confundir ou prejudicar tais resultados. Além disso, a seleção de *features* evita o uso de recursos computacionais de forma desnecessária e contribui para a redução do tráfego de informações na rede.

A definição do conjunto de *features* ideal para cada perfil de ataque consiste em um desafio para os detectores de intrusões. Portanto, a utilização da arquitetura proposta permite que o os motores de detecção sejam configurados de forma personalizada para cada objetivo de análise. Não faz parte do escopo deste trabalho a análise de relevância de *features*. Todavia, recomenda-se o uso de métodos baseados em entropia[2] para esta finalidade, tal como é feito em (OLUSOLA; OLADELE; ABOSEDE, 2010). Uma vez devidamente definido, o filtro de *features* deve ser aplicado tanto para as assinaturas quanto para as amostras desconhecidas.

4.5.2 Clusterização de Assinaturas

O processo de clusterização tem por finalidade efetuar o agrupamento de assinaturas de acordo com a similaridade de seus perfis. Cada um desses agrupamentos tende a reunir tipos de ataques de classes iguais ou semelhantes. Assim, para que aconteça a clusterização, as assinaturas de ataques das classes que estão de acordo com o propósito do motor de detecção são recuperadas do componente de persistência. Portanto, um algoritmo de *clustering* deve ser aplicado a fim de efetua a geração de agrupamentos de assinaturas de acordo com a similaridade de suas *features*. Tal agrupamento pode ocorrer por meio do algoritmo K-Means, por exemplo.

É importante observar que o modelo de *clusters* passa por transformações de acordo com a evolução da base de assinaturas. Desse modo, uma amostra que é assinalada a um *cluster*, pode ser assinalada a outro agrupamento no futuro. Isso ocorre porque a base de assinaturas utilizadas para treinamento e avaliação é constantemente atualizada de forma a permitir o aprimoramento dos motores de detecção.

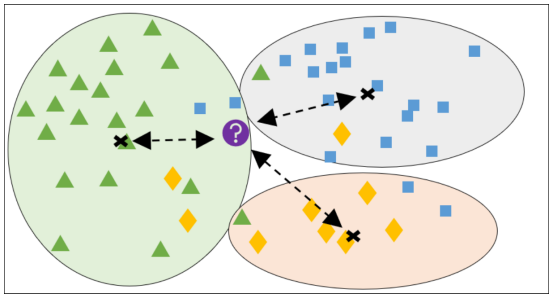
Dado que as análises de amostras desconhecidas gerarão assinaturas, as quais são persistidas no repositório de assinaturas, novas amostras passarão a ser centroides dos respectivos

[2] Entropia é uma técnica utilizada para determinar o quão informativo uma *feature* de entrada específica é sobre o a classe de saída para um subconjunto de amostras (OLUSOLA; OLADELE; ABOSEDE, 2010).

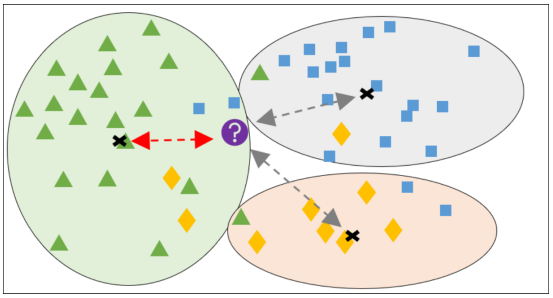
clusters. Com isso, os motores de detecção são especializados conforme o seu uso. Essa especialização pode acarretar na escolha dinâmica de um novo classificador nas situações em que as novas assinaturas interferem na medição da acurácia dos classificadores. A seleção dinâmica de classificadores será abordada a seguir.

4.5.3 Seleção Dinâmica de Classificador

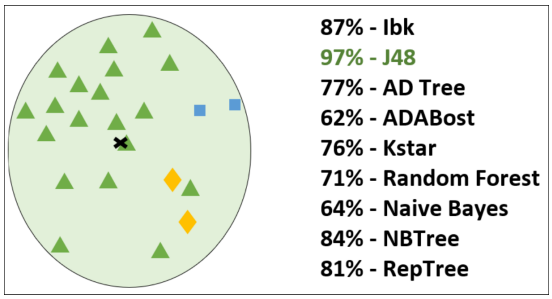
Uma vez que as amostras desconhecidas são devidamente filtradas e recuperadas, o motor de detecção deve selecionar um de seus classificadores para efetuar a classificação e, portanto, a detecção de intrusões. Dentre os classificadores implementados em um motor de detecção, deve-se escolher aquele que apresenta melhores resultados para o *cluster* de amostras com centroide mais próximo da amostra desconhecida. Entende-se por centroide o ponto que representa o centro de um agrupamento de amostras. Para tanto, é calculada a distância euclidiana entre a amostra desconhecida e os centroides de cada um dos *clusters* gerados durante o processo de Clusterização, o qual foi descrito anteriormente. As Figura 4.4(a) e 4.4(b) ilustram esse procedimento, onde o centroide mais próximo define a qual *cluster* a amostra pertence.



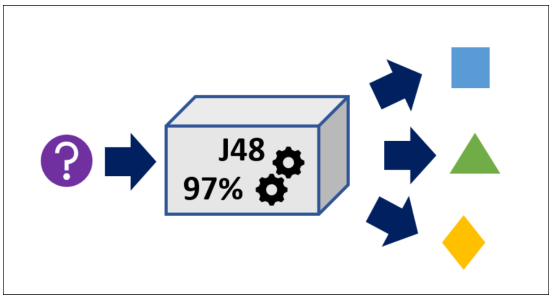
(a) Clusterização de assinaturas e mensuração da distância euclidiana dos centroides em relação à amostra desconhecida.



(b) Identificação do cluster com centroide mais próximo da amostra desconhecida.



(c) Avaliação dos classificadores para as assinaturas contidas no cluster selecionado e seleção do melhor deles.



(d) Classificação da amostra desconhecida por meio do melhor classificador, selecionado anteriormente.

Figura 4.4: Seleção dinâmica do classificador mais promissor para detecção de amostras desconhecidas.

Tendo conhecimento do *cluster* ao qual a amostra desconhecida é pertencente, gera-se uma suspeita de que tal amostra pertence a um determinado perfil, o qual é correspondente a este *cluster*. De modo a validar essa suspeita, deve-se selecionar o classificador com maior acurácia durante a avaliação das assinaturas pertencentes a tal *cluster*, a fim de efetuar a classificação dessa amostra desconhecida. Esse processo se chama Seleção Dinâmica de Classificador. Dizemos que a escolha é dinâmica porque depende de um processo de avaliação que pode gerar resultados dinâmicos de acordo com a base de treinamento e avaliação, que, por sua vez, é constantemente incrementada.

Neste momento, parte das assinaturas deste agrupamento são utilizadas para o treinamento dos classificadores. As demais assinaturas, que não fazem parte do subconjunto de treinamento, são utilizadas para a avaliação desses classificadores. Assim, as assinaturas que fazem parte do conjunto de avaliação são transformadas em amostras, removendo-se a informação de sua classe de origem, e enviadas para a classificação por cada um dos classificadores configurados.

Desse modo, é possível a mensuração do desempenho de cada um dos classificadores frente à cada perfil de assinaturas. Por fim, com base nessa mensuração, é possível a seleção do classificador mais promissor para a análise do perfil amostral associado à amostra desconhecida. A Figura 4.4(c) representa esse processo, onde, hipoteticamente, o classificador J48 apresenta melhor resultado e portanto é o escolhido para efetuar a classificação da amostra desconhecida, como ilustra a Figura 4.4(d).

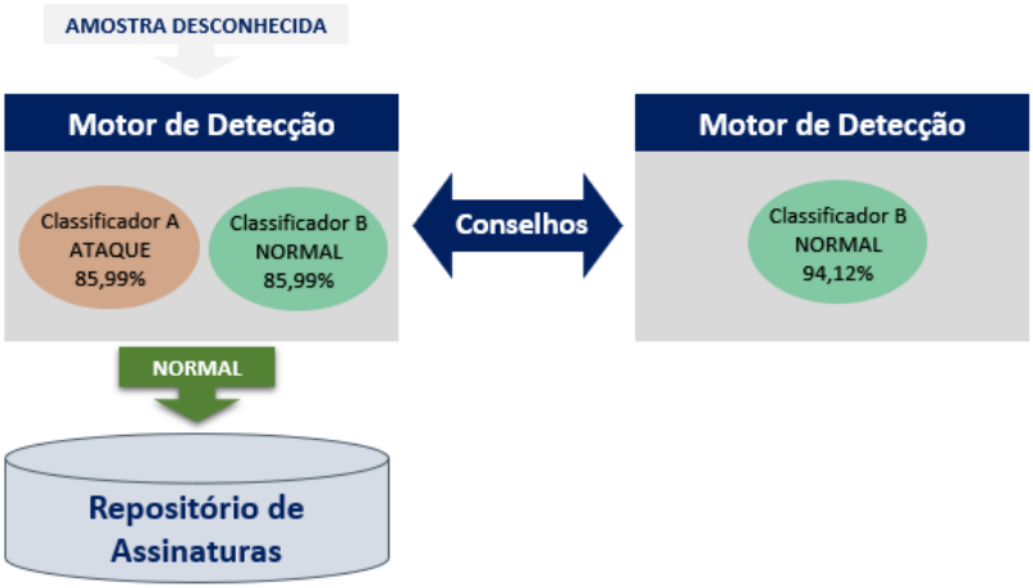
Com isso, quando novas amostras desconhecidas são coletadas, é possível a seleção dinâmica do classificador mais promissor para sua análise, de acordo com sua respectiva similaridade com os perfis conhecidos.

Um dos problemas pertinentes na literatura consiste na escolha dinâmica entre classificadores que apresentam acurácia muito próxima ou igual. Nesse caso, ambos os classificadores que apresentaram melhor desempenho são escolhidos para a avaliação da amostra desconhecida. Entretanto, se houver divergência nos resultados das classificações destes, é necessária a tomada de decisão de qual desses classificadores deve-se confiar o resultado final.

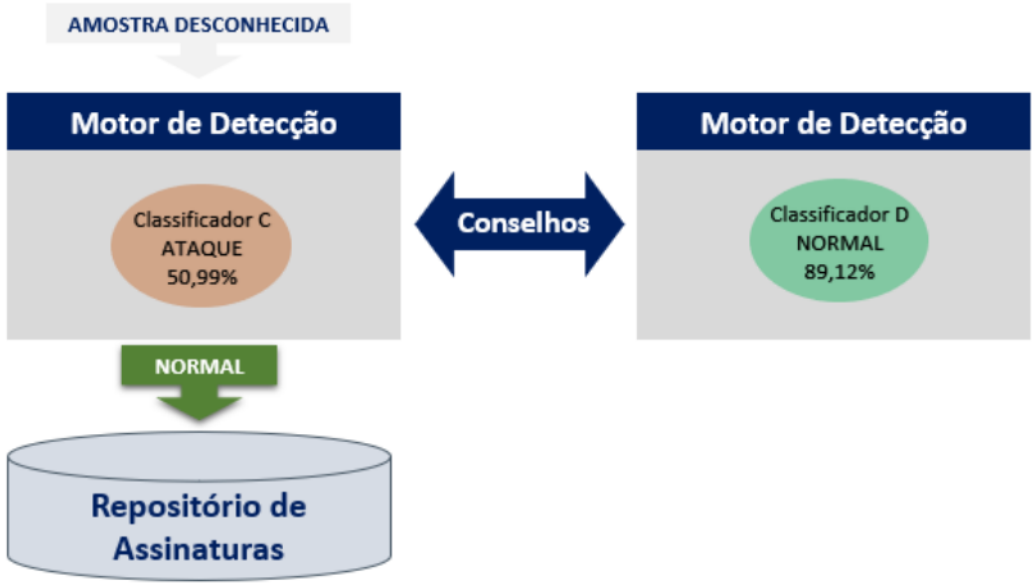
Este trabalho mitiga este problema através da proposta de um sistema de consultas a conselheiros. Um conselheiro consiste em um outro motor de detecção, o qual analisa fontes diferentes, mas ainda com o propósito de detecção dos mesmos tipos de ataques. A consulta à rede de conselhos também é uma forma de aumentar a confiabilidade em uma decisão naqueles

casos em que a acurácia do melhor classificador ainda é considerada baixa, com base em um limiar. Essa nova abordagem é apresentada na Seção 4.6.

4.6 REDES DE CONSELHOS



(a) Solicitação de conselho ocasionada por conflito entre classificadores de bom desempenho, selecionados dinamicamente.



(b) Solicitação de conselho ocasionada baixa confiabilidade no resultado do classificador selecionado dinamicamente.

Figura 4.5: Cenários que requerem troca de conselhos.

Uma vez que a arquitetura permite a análise de múltiplas e heterogêneas fontes de dados, a colaboração entre motores de detecção almeja aumentar a acurácia na detecção de ataques que

são invisíveis ou confusos por um **detector** que analisa uma única fonte. Assim, a arquitetura proposta introduz o conceito de **rede de detectores conselheiros**. Tal rede de conselhos tem por finalidade a troca mútua de conhecimento entre diferentes motores de detecção. Com isso, a decisão final é empoderada e conflitos são resolvidos. As Figuras 4.5 (a) e (b) ilustram duas situações onde um motor de detecção conselheiro deve ser requisitado para, respectivamente, resolver problemas de conflitos entre classificadores e solucionar o problema da insuficiência de informações provindas da fonte analisada pelo detector solicitante.

4.6.1 Definição de Conselheiros

Conselhos consistem em requisições feitas a partir de um motor de detecção para um ou mais motores que têm mesmo propósito, ou seja, que têm a finalidade de detectar as mesmas classes de ataques. A diferença básica entre um detector que faz a requisição e o conselheiro, que atende a esta, consiste nas suas configurações de entrada. Tais configurações variam no que diz respeito à fonte de dados e conjunto de features analisadas a partir dessa fonte. Com isso, todo detector que solicita conselhos, pode também prestar conselhos aos demais detectores. Portanto, conselheiro é a denominação que se dá para um motor de detecção qualquer, no momento em que ele é requisitado por outro motor para responder a uma consulta baseada em suas próprias análises.

Dado que a troca de conselhos acontece entre detectores que analisam fontes de dados distintas e heterogêneas, muitas vezes, não existem *features* em comum que possam servir de parâmetro para correlação entre as análises de diferentes detectores. Portanto, a abordagem de redes de conselhos proposta se baseia na *timestamp* (marca temporal, em inglês). Assim, detectores trocam conselhos com base nas análises que acontecem em simultâneo nos diferentes motores. Portanto, as requisições devem conter sempre a *timestamp* da análise em que se quer consultar o resultado.

Supondo que um detector que analisa tráfego de rede buscando por ataques de negação de serviço, do tipo *flooding* (inundação), apresenta conflitos entre seus classificadores durante a classificação de uma determinada amostra. Tal detector pode consultar um outro que está configurado para analisar os *logs* a nível de aplicação. Neste cenário, o detector que analisa *logs* de aplicações é o conselheiro, que por sua vez, responde com os resultados das análises no momento relacionado à *timestamp* enviada pelo detector requisitante. Em um outro momento, os papéis podem se inverter, onde o detector que faz a análise de *logs* de aplicações poderá

requisitar conselhos ao detector que faz a análise de tráfego de rede, constituindo-se assim uma rede de troca mútua de conselhos.

Junto a resposta a uma requisição, o detector que exerce função de conselheiro deve informar a acurácia obtida durante a validação do classificador responsável pelo resultado da análise que originou tal conselho. Desse modo, o detector que solicita o conselho é capaz de avaliar, de forma automática, a necessidade de uma nova consulta a um outro conselheiro. No pior caso, quando não existem outros conselheiros e a acurácia do classificador solicitante é duvidosa, isto é, fora dos intervalos definidos como aceitáveis, um alerta deve ser emitido ao administrador do sistema. Ambas as possibilidades discutidas anteriormente podem ser configuradas para cada detector, através da definição de limiares e seus respectivos gatilhos a fim de efetuar o disparo de ações quando estes forem atingidos. As configurações de detectores serão discutidas na Seção 4.6.2.

O impacto da abordagem de troca de conselhos vai além do aumento da confiabilidade na decisão de uma análise ou da solução de conflitos no momento em que eles acontecem. As decisões tomadas com base em conselhos originam novas assinaturas, que até o momento da consulta ao conselheiro eram amostras desconhecidas. Assim, em futuras análises de amostras com perfis semelhantes à estas, o detector que solicitou o conselho estará habilitado a agir imediatamente, sem a necessidade de uma nova consulta. Por esse motivo, pode-se afirmar que a arquitetura proposta permite o aprendizado por parte dos motores de detecção. Com isso, novos padrões de ataques poderão ser mapeados e poderão ser detectados sempre que novas assinaturas inseridas no repositório de assinaturas, situado no componente de persistência, como discutido anteriormente.

4.6.2 Parametros de Detectores

A abordagem de conselhos traz uma série de possibilidades que mitigam a necessidade da ação humana em casos onde um detector não é suficiente por si só para gerar resultados conclusivos em relação a análise de determinadas amostras desconhecidas. Por esse motivo, é necessário que alguns parâmetros sejam configurados, evitando que ações indesejadas sejam tomadas, como o aceite de conselhos com baixa confiabilidade, por exemplo. A seguir, as possibilidades de configurações serão discutidas.

- **Acurácia Local Mínima Aceitável (ALMC):** este parâmetro define qual é o mínimo de acurácia que um classificador deve apresentar durante a fase de validação para que sua

análise seja considerada suficientemente aceitável para a tomada de decisões para aquele perfil de amostras que fazem parte do agrupamento avaliado. Quando o ALMC não é alcançado por nenhum classificador, um conselho deverá ser solicitado no momento em que uma amostra desconhecida similar ao perfil do a grupamento for analisada.

- **Acurácia Mínima Aceitável do Conselheiro (AMAC):** este parâmetro define qual é a acurácia mínima para que um conselho seja considerado como decisão final para uma análise. Uma vez que a acurácia alcançada pelo melhor classificador do detector conselheiro for inferior à AMAC definida no detector autor da requisição, uma entre as seguintes ações deve ser tomada: (i) um novo conselheiro deve ser consultado se disponível, (ii) um alerta será emitido para que um administrador tome a decisão final ou (iii) o classificador com maior acurácia tomará a decisão final, seja este classificador pertencente ao detector local ou ao detector conselheiro. Tais ações podem ser configuradas em simultâneos, em ordem de prioridade a ser definida no momento da configuração.

- **Acurácia Mínima Conflitante (AMC):** este parâmetro define qual é a acurácia mínima que um classificador deve ter sobre os demais classificadores para que seus resultados não sejam confrontados com os resultados dos demais. Uma vez que existem múltiplos classificadores com a acurácia dentro do limiar definido, todos eles devem executar a classificação da amostra desconhecida e, em caso de resultados controversos, um conselho deve ser solicitado.

- **Habilitar Múltipla Escolha:** quando este parâmetro for definido como verdadeiro, o limiar **AMC** deve ser considerado. Assim, todos os classificadores com acurácia dentro do limiar estabelecido serão selecionados para a classificação da amostra desconhecida. A decisão final será tomada com base na votação ponderada entre os classificadores selecionados, onde o peso do voto é proporcional à acurácia individual de cada classificador. Caso desabilitada, o os conflitos descritos no item anterior podem ocorrer e devem ser resolvidos conforme a recomendação dada.

- **Número de *Clusters***: este parâmetro, também conhecido como K, define o número de agrupamentos de assinaturas que serão gerados durante a execução do algoritmo de *clustering*. Em outras palavras, esse número representa a quantidade de perfis que serão definidos por este algoritmo. Assim, o ideal é que a definição de K seja baseada no número de perfis de assinaturas utilizadas na fase de treinamento e avaliação de classificadores,

as quais devem também fazer parte do propósito de detecção do motor de detecção. Tal propósito pode ser amplo ou específico, de acordo com as necessidades de detecção. Um detector especializado em ataques de negação de serviço, por exemplo, deve ter o número de subclasses deste ataque, acrescidos de uma unidade, como parâmetro para a definição de *K*. O motivo pelo qual uma unidade deve ser acrescida é para que seja reservado um perfil para amostras do tipo normal, que não representa nenhuma subclasse de ataque.

- **Conjunto de Classificadores**: dentre os classificadores implementados, disponíveis para a classificação das amostras desconhecidas, o administrador tem a possibilidade de optar por quais desses que serão aplicados para cada detector. A razão para esta escolha pode ser, por exemplo, a customização de recursos computacionais ou de tempo. Com a especificação dos classificadores que farão parte da escolha dinâmica, é possível evitar a avaliação e treinamento de classificadores que apresentam, historicamente, resultados inferiores aos demais, por exemplo.

- **Algoritmo de Clustering**: a fim de executar o agrupamento de amostras baseando-se em sua similaridade, uma técnica de mineração de dados chamada clusterização (*clustering*) deve ser aplicada. Dentre os algoritmos existentes, deve-se selecionar aquele que melhor se adequa as necessidades do contexto em que o detector e encontra. Exemplos de algoritmos são: *Canopy, Cobweb, FarthestFirst e K-means* (SHARMA; JOSHI, 2015).

- **Métrica de Avaliação de Classificadores**: a fim de avaliar o desempenho dos classificadores, existem quatro variáveis básicas que podem ser analisadas: o número de Verdadeiro Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN). Sugere-se que a métrica utilizada para a avaliação dos classificadores em cada *cluster* de assinaturas e, consequentemente, a seleção do melhor classificador para a classificação de uma amostra desconhecida, seja baseada na acurácia alcançada no momento da validação, pois leva em conta os acertos, independentemente se foi um verdadeiro positivo ou verdadeiro negativo, conforme a Equação 4.1. Entretanto, existem outras métricas, tais como a sensibilidade, também chamada de taxa de verdadeiros positivos, ilustrada na Equação 4.2 e especificidade, também conhecida como taxa de verdadeiros negativos, ilustrada na Equação 4.3 (FAWCETT, 2006).

$$\frac{VP + VN}{VP + VN + FN + FP}$$

Equação 4.1: cálculo da acurácia.

$$\frac{VP}{VP + FN}$$

Equação 4.2: cálculo da sensibilidade.

$$\frac{VN}{VN + FP}$$

Equação 4.3: cálculo da especificidade.

5 IMPLEMENTAÇÃO

A fim de validar de forma prática a arquitetura proposta, um protótipo foi implementado utilizando-se a linguagem de programação Java e a ferramenta de armazenamento de dados online, em tempo real, *Firebase Realtime Database*[3]. Os detalhes do desenvolvimento de cada um dos componentes será apresentado a seguir, neste Capítulo.

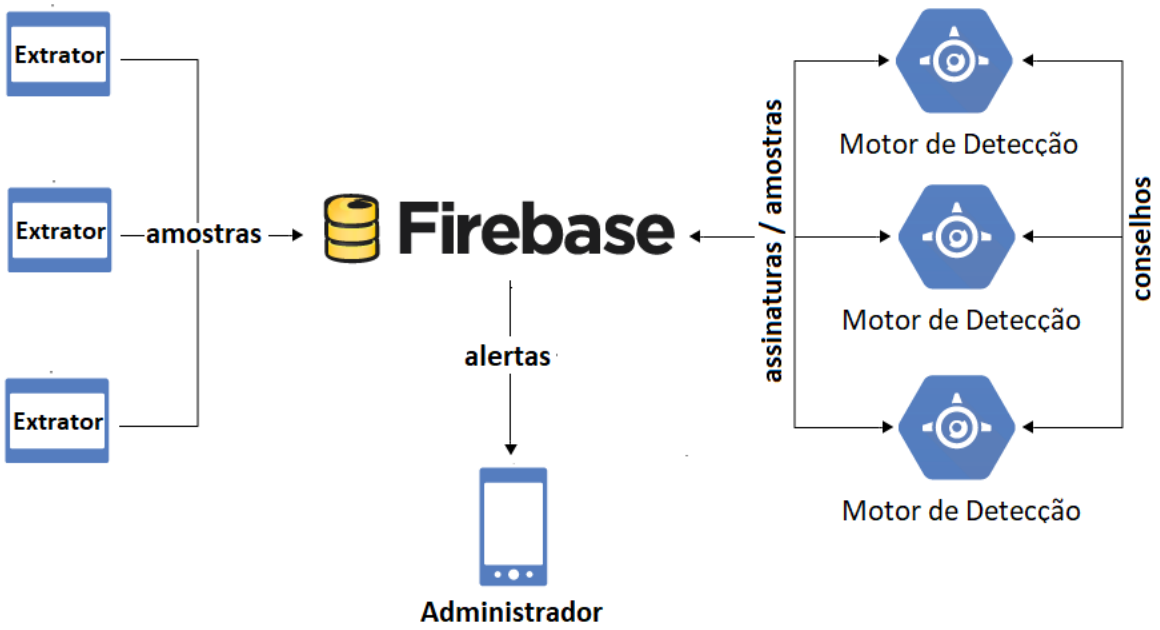


Figura 5.1: Visão geral da implementação da arquitetura proposta.

Conforme ilustra a Figura 5.1, a implementação descrita neste capítulo é composta de três extratores que coletam amostras e às persistem no Repositório de Amostras hospedado na ferramenta Firebase. Esta ferramenta fornece a possibilidade da implementação de *listeners* de eventos os quais são implementados nos motores de detecção e funcionam como observadores. Dessa forma, cada um dos motores de detecção possui a implementação que recebe os respectivos dados necessários para a sua análise interna.

5.1 EXTRATORES

Conforme descrito no Capítulo 4, a arquitetura proposta tem como objetivo habilitar a análise de múltiplas e heterogêneas fontes. Assim, na implementação apresentada neste capí-

[3] https://firebase.google.com/docs/database/?hl=pt-br

tulo, serão considerados os dados provindos de três extratores, cada um deles provendo dados de uma fonte distinta. Tais extratores serão discutidos a seguir:

5.1.1 Extratores de Tráfego de Rede

Dois dos extratores implementados tem por finalidade a captura de pacotes de rede. Um deles, o Extrator 1, visa a extração de *features* de conexões computadas a partir de uma janela de dois segundos, como taxas de pacotes com erro. O Extrator 2, por sua vez, analisa o mesmo tipo de fonte, mas com a finalidade de extrair *features* básicas de conexão, tais como o protocolo utilizado para a comunicação.

As Tabelas 5.1 e 5.2, listam as *features* coletadas pelo Extrator 1, onde, respectivamente, a primeira representa as informações baseadas em conexões que possuem o mesmo host da conexão atual, enquanto a segunda representa as taxas relativas às conexões para o mesmo serviço que a conexão atual. A Tabela 5.3, lista as *features* básicas de conexões TCP individuais, coletadas pelo Extrator 2. Tais conjuntos de *features*, assim como os demais apresentados neste capítulo, baseiam-se no conjunto de dados (*dataset*) da KDDCup99 (CUP, 1999).

Tabela 5.1: Features baseadas em conexões para o mesmo host.

Feature	*Descrição*
Contagem	Quantidade de conexões para o mesmo host.
Taxa de Erros SYN	Porcentagem de conexões que possuem erros “SYN”.
Taxa de Erros REJ	Porcentagem de conexões que possuem erros “REJ”.
Taxa de Mesmo Serviço	Porcentagem de conexões para o mesmo serviço.
Taxa de Diferente Serviço	Porcentagem de conexões para serviços diferentes.
Contagem Mesmo Serviço	Contagem de conexões para o mesmo serviço.

Tabela 5.2: *Features* baseadas em conexões para o mesmo serviço.

Feature	*Descrição*
Taxa de Erros SYN	Porcentagem de conexões que possuem erros “SYN”.
Taxa de Erros REJ	Porcentagem de conexões que possuem erros “REJ”.
Taxa de Diferente Host	Porcentagem de conexões para hosts diferentes.

Portanto, as implementações dos Extratores 1 e 2 foram baseadas na biblioteca jpcap[4], por exemplo. Essa biblioteca permite a captura de pacotes, no formato *pcap*, a partir das dife-

[4] http://jpcap.sourceforge.net/javadoc/index.html

Tabela 5.3: *Features* básicas de conexões TCP.

Feature	*Descrição*
Duração	Duração, em segundos, da conexão.
Protocolo	Tipo de protocolo, ex. TCP, UDP, etc.
Serviço	Serviço de rede de destino, ex: http, telnet, etc.
Bytes Recebidos	Número de bytes do destino para a origem.
Bytes Enviados	Número de bytes da origem para o destino.
Flag	Status da conexão (normal ou erro).
Land	Verdadeiro quando a conexão é de/para mesma porta/host.
Fragmentos *Wrong*	Número fragmentos demarcados como “wrong”.
Urgentes	Número de pacotes urgentes.

rentes interfaces de rede que a máquina executora tem acesso. Uma vez capturados, os pacotes passam por um processo de transformação, onde suas *features* são extraídas.

5.1.2 Extrator de Logs de Aplicação

Este extrator, por sua vez, tem a sua implementação voltada à captura e transformação de dados provindos da camada de aplicação. A partir desta fonte de coletas, portanto, é possível a extração de *features* relacionadas ao comportamento de usuários ao acessar um host. Com isso, determinados conjuntos de informações são capazes de gerar perfis que distinguem comportamentos de atacantes dos comportamentos de usuários legítimos. As Tabela 5.4 lista as *features* coletadas a partir do Extrator 3, as quais foram definidas, também, com base na KDDCup99.

A observação de atributos que envolvem a camada de aplicação é importante tanto para a detecção de padrões de ataques visíveis somente por meio da análise desse tipo de informação, quanto para fortalecer a confiabilidade na detecção de ataques que são monitorados a partir de múltiplas fontes. Por esse motivo, todas as amostras coletadas são gravadas junto à sua *timestamp*.

5.2 REPOSITÓRIO DE ASSINATURAS E AMOSTRAS DESCONHECIDAS

De forma a centralizar toda informação envolvida na detecção de intrusões, a arquitetura proposta requer um componente de armazenamento. Assim, um repositório foi implementado de modo a suportar dois propósitos: (i) a persistência de assinaturas, sejam elas provindas do

Tabela 5.4: *Features* conteúdo dentro de uma conexão, sugeridas pelo conhecimento de domínio.

Feature	*Descrição*
Indicadores “hot”	Contagem de indicadores “hot”.
Falhas de login	Número de falhas de login.
Sucesso no login	Login bem sucedido?
Eventos comprometidos	Contagem de marcações “compromised”.
Obteve acesso root	Acesso concedido ao shell como root?
Tentativas “su”	Houveram tentativas de comando “su”?
Contagem de acessos root	Número de acessos como root.
Criações de arquivos	Número de operações de criação de arquivos.
Contagem shell	Número de comandos shell.
Arquivos acessados	Número de operações de controle e acesso a arquivos.
Comandos de Saída	Número de comandos de saída em uma sessão FTP.
Pertencente à lista “hot”	O login pertence à lista “hot”?
Login convidado	O login foi feito como “guest”?

dataset de assinaturas inicial ou de resultados de análises de motores de detecção, e (ii) amostras desconhecidas, as quais são coletadas por meio dos extratores e precisam ser persistidas até que um motor de detecção execute seu processamento.

Portanto, este componente deve contemplar a interoperabilidade com os componentes extratores e também com os motores de detecção. Por esse motivo, a troca de dados entre os componentes da arquitetura foram unificados no formato JSON. Assim, independentemente da fonte das informações, os repositórios de assinaturas e amostras desconhecidas devem conter exclusivamente arquivos nesse formato. Desse modo, as amostras e assinaturas são transformadas em objetos JSON antes de chegar ao componente de persistência e suas respectivas *features*, transformadas em atributos desses objetos. Dado, que é necessária a diferenciação de amostras normais das amostras de ataques, vale destacar que as assinaturas podem ou não representar situações de intrusões. Em outras palavras, existirão também assinaturas “normais”.

Dados os requisitos exigidos para a implementação deste componente, utilizou-se a ferramenta de banco de dados não relacional *Firebase Realtime Database* como facilitadora. A seguir, tal ferramenta será apresentada, juntamente com a estrutura dos dados persistidos nesta.

5.2.1 Firebase Realtime

O *Firebase Realtime Database*[5] é um banco de dados não relacional hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Em outras palavras, todos os clientes compartilham uma instância do *Realtime Database* e recebem automaticamente atualizações com os dados mais recentes.

No contexto da implementação da presente proposta, todas as assinaturas de ataques e de perfis normais, bem como as amostras coletadas pelos extratores são persistidas na nuvem provida pelo *Firebase*. Tais informações são sincronizadas, em tempo real, com todos os componentes que implementam os métodos *listeners* de seus eventos. Estes, são acionados sempre que novos eventos de inserção, modificação ou exclusão são identificados no repositório centralizador. Portanto, a principal finalidade desta ferramenta consiste em manter atualizada a entidade que consome as informações de seu interesse. Na arquitetura proposta, os interessados são os motores de detecção. Assim, estes recebem atualizações em tempo real acerca dos dados coletados pelos extratores.

5.2.2 Persistência e Recuperação de Amostras

Baseando-se na ferramenta discutida anteriormente, os componentes extratores estão habilitados a persistirem toda a informação produzida, no formato JSON, no repositório de amostras. Os motores de detecção, por sua vez, estão habilitados a acessar tais dados. Dado que cada motor deve ser configurado para consumir apenas as informações que competem ao seu propósito, conforme suas configurações, as informações estão persistidas em diferentes ramificações do repositório de amostras. Cada uma dessas ramificações possui uma referência. É com base nessa referência que é definida a conexão de quais fontes serão analisadas por quais motores de detecção. A Figura 5.2 ilustra as conexões com os extratores e motores de detecção, os quais estão, respectivamente, persistindo e recuperando as amostras do repositório.

[5] https://firebase.google.com/docs/database

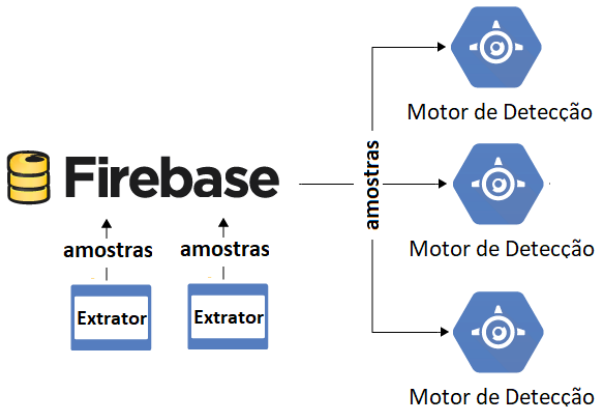


Figura 5.2: Processo de persistência e recuperação de amostras a partir do *Firebase Realtime Database*.

Algoritmo 1: MONITORAMENTO DE AMOSTRAS E DISPARO DE ANÁLISES.

Entrada: ramificacao; *//Referência a um subconjunto do repositório de amostras*
Saída: Gatilho para inicialização da detecção de intrusões

```
1 início
2 |  repositorio = Firebase.ouvir(ramificacao);
3 |  para cada amostra ∈ repositorio faça
4 |  |  executarAnalise(amostra)
5 |  fim
6 fim
```

O Algoritmo 1, ilustra o procedimento de monitoramento e recuperação de amostras, o qual se dá a partir de uma referência a um subconjunto do repositório de amostras. Tal subconjunto se refere ao tipo de perfil que é alvo da análise deste detector. Esse procedimento é repetido continuamente enquanto o motor de detecção que implementa este *listener* estiver em operação. Na linha 4, o procedimento de análise e detecção de intrusões é iniciado sempre que existir uma nova amostra no subconjunto delimitado pela respectiva referência (*ramificacao*). Os detalhes dessa análise serão abordados na Seção 5.3.

5.2.3 Persistência e Recuperação de Assinaturas

As assinaturas são artefatos fundamentais na detecção de intrusões supervisionada, conforme discutido no Capítulo 2. Portanto, para que os resultados das análises sejam concisos com a base de conhecimento da arquitetura, é necessária a sincronização entre o repositório de assinaturas e os motores de deteção.

Assim como no processo de sincronização de amostras, as assinaturas também contam com a ferramenta *Firebase Realtime Database* para que todos componentes detectores possuam

acesso aos dados sempre atualizados. A diferença, porém, está na origem da geração de assinaturas. Essas são geradas pelos próprios motores de detecção, a partir do processamento de amostras desconhecidas. Desse modo, motores de detecção compartilham suas bases de conhecimento, permitindo que as recorrências futuras de perfis já analisados sejam detectados com maior eficiência.

Dado que o repositório de assinaturas é atualizado constantemente, a sincronização deve levar em conta apenas aquele subconjunto de amostras que satisfaz os critérios definidos como propósito de sua detecção. Portanto, a recuperação de assinaturas se dá de acordo com a referência correspondente, conforme ilustra a Figura 5.3. O Algoritmo 2 descreve o procedimento de sincronização de assinaturas.

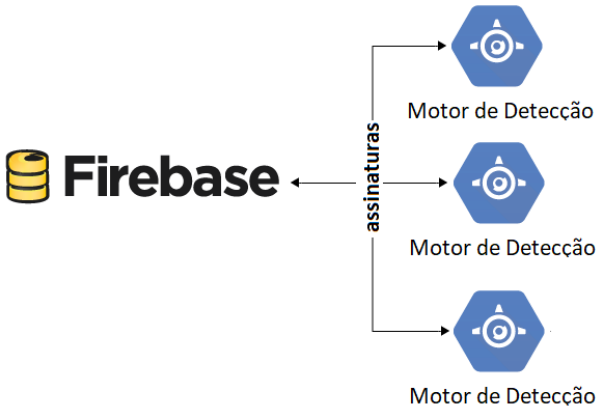


Figura 5.3: Processo de persistência e recuperação de assinaturas a partir do *Firebase Realtime Database*.

Algoritmo 2: ALGORITMO DE RECUPERAÇÃO DE ASSINATURAS.

Entrada: ramificacao; *//Caminho referência ao repositório de assinaturas*
Saída: novasAssinaturas[]; *//Assinaturas sincronizadas*

```
1 início
2 |  repositorio = Firebase.ouvir(ramificacao);
3 |  para cada assinatura ∈ repositorio faça
4 |  |  novasAssinaturas[] ← assinatura;
5 |  fim
6 |  retorna novasAssinaturas[];
7 fim
```

5.3 IMPLEMENTAÇÃO DE MOTORES DE DETECÇÃO

Os detectores podem ser configurados para filtrar as features da maneira que mais convém para cada propósito de análise. Entretanto, para fins didáticos, nesta implementação, cada

detector analisa o conjunto de *features* completo provindo de cada um dos extratores descritos anteriormente:

a) Detector 1: tem por finalidade a análise de *features* oriundas de conexões TCP;

b) Detector 2, visa a análise de *features* provindas de estatísticas de tráfego de rede;

c) Detector 3, por sua vez, processa dados extraídos a partir de logs de aplicações.

5.3.1 Cálculo de similaridade de Amostra Desconhecida e Assinaturas

Independentemente do propósito do detector, a execução de sua análise tem início no momento em que uma nova amostra desconhecida é recuperada a partir do componente de persistência, conforme ilustrado no Algoritmo 1. Assim, uma vez que tal recuperação acontece, o motor de detecção deve segmentar as assinaturas através da execução do algoritmo de *Clustering* a fim de identificar o agrupamento mais similar à amostra desconhecida. Esse processo assume que a base de assinaturas já tenha sido sincronizada, por meio do Algoritmo 2. Isso significa que a sincronização não deve ocorrer no momento da análise, a fim de garantir a conclusão dos resultados no menor tempo possível.

Então, as assinaturas atualizadas são agrupadas por meio do algoritmo de clusterização *Simple Kmeans* (KANUNGO et al., 2000). Conforme discutido no Capítulo 2, existem outros algoritmos que podem ser utilizados. A exploração do melhor algoritmo de clustering para esse papel não compôs o escopo deste trabalho. Nesse contexto, a biblioteca java Weka 3.8.0[6] foi utilizada. O número de clusters foi definido como seis (K = 6), de acordo com as características do dataset que foi utilizado para a validação. Em seguida, as distância Euclidiana (DEZA; DEZA, 2009) entre a amostra desconhecida e cada um dos centróides dos agrupamentos formados são medidas. Com isso, o assume-se que tal amostra possui maior similaridade com as assinaturas pertencentes ao cluster com centróide mais próximo à mesma.

5.3.2 Avaliação de Classificadores

A fim de prever, de forma dinâmica, o classificador mais promissor para a análise de uma amostra desconhecida[7], um conjunto de classificadores é treinado com uma fração das

[6] https://www.cs.waikato.ac.nz/ml/index.html

[7] assume-se como amostra desconhecida toda aquela amostra que foi extraída e ainda não foi rotulada por um classificador

assinaturas similares à mesma e, em seguida, outra fração é utilizada para a avaliação de cada um dos classificadores. Com isso, é possível estabelecer um grau de confiabilidade no resultado de cada um dos classificadores avaliados.

Primeiramente, foi definido um conjunto de classificadores para a implementação. Assim como na clusterização, os algoritmos de classificação também foram implementados por meio da biblioteca java Weka 3.8.0. Dentre os classificadores existentes, foram selecionados alguns dos mais populares na literatura: IBk, J48, AD Tree, ADABost M1, KStar, Random Forest, Naive Bayes, NB Tree e Rep Tree.

Uma vez implementados, os classificadores devem ser avaliados. Dado que está sendo considerado um cenário onde múltiplos ataques podem acontecer, e por consequência, existem múltiplos perfis de amostras e assinaturas, a avaliação ocorre separadamente para cada um dos agrupamentos gerados anteriormente. Portanto, é possível que o classificador ideal para determinados agrupamentos tenha baixo desempenho em agrupamentos distintos. Partindo-se da premissa de que a amostra desconhecida foi associada ao cluster mais similar, isso não deve ser um problema. Como critério de seleção do melhor classificador, foi definido o cálculo da acurácia do mesmo frente à classificação das assinaturas utilizadas para a avaliação, as quais são oriundas do mesmo cluster ao qual foi identificado maior similaridade com a amostra desconhecida.

5.3.3 Classificação de Amostras Desconhecidas

Finalizado o processo de avaliação de classificadores, aquele com maior acurácia durante tal avaliação deve executar a classificação da amostra desconhecida. No melhor dos cenários, um classificador possui acurácia alta o suficiente para tomar a decisão final acerca do resultado a ser emitido pelo motor de detecção. Nesse caso, o processo de detecção de intrusões é finalizado. Esse, porém, não é o único cenário possível. Existem ainda mais dois cenários, onde a classificação por meio do classificador mais promissor ainda não é auto-suficiente para gerar um resultado confiável. Portanto, medidas adicionais, tais como a consulta a conselheiros, devem ser tomadas.

O primeiro cenário onde o melhor classificador não deve ser ter a confiabilidade considerada satisfatória, consiste naquele que até mesmo o melhor classificador possui uma acurácia inferior a um limite pré-estabelecido. O reflexo dessa falta de conhecimento pode ser observado durante a avaliação de classificadores. Partindo-se da premissa de que as assinaturas utilizadas

para treinamento não contém *features* representativas para esse perfil de ataque, nenhum dos classificadores será capaz de distingui-las de assinaturas de comportamento normal. Portanto, a acurácia da classificação será baixa para qualquer um dos classificadores avaliados. Assumimos, nesta implementação, como acurácia aceitável o limiar de no mínimo 85%. Ao invés da acurácia, outras regras poderiam ser consideradas, conforme explicado anteriormente, no Capítulo 4.

Supondo que durante a detecção de um ataque de força bruta, muitas tentativas consecutivas e mal-sucedidas de login podem ser observadas por meio da análise de logs de aplicação. Um detector que analisa *features* conexões de rede, tais como números de bytes transferidos, provavelmente, não teria as informações necessárias que o habilitassem à detectar esse tipo de ataque. Portanto, um outro detector deve aconselhá-lo sobre qual resultado deve ser emitido.

Existe, entretanto, um segundo cenário onde apesar de existirem *features* representativas para os perfis de ataques analisados, dois ou mais classificadores apresentam bons resultados durante a avaliação. O simples fato de múltiplos classificadores possuírem bons resultados, apenas fortalece a confiabilidade de que a avaliação conta uma base de assinatura com boa representatividade dos perfis. O que realmente se torna um problema, são os momentos em que dois classificadores confiáveis apresentam resultados controversos.

Esse tipo de situação gera um conflito interno ao motor de detecção, o qual impossibilita a tomada de decisão com um nível de confiabilidade. Portanto, se faz necessária a definição de parâmetros que delimitem um cenário cenário conflitante, tal como a definição de uma margem de erro. Assim, nesta implementação, definimos que somente classificadores com acurácia até 2% inferior ao classificador com maior acurácia sejam considerados conflitantes, em caso de resultados controversos.

Em quaisquer dos casos acima mencionados, onde a seleção dinâmica de classificadores se mostra conflitante ou insuficiente para a análise de uma amostra, a solução proposta consiste na solicitação de conselhos para outros motores de detecção com maior especialidade. Os detalhes da implementação da rede de conselheiros será discutido na Seção 5.4.

Por fim, independentemente se o resultado final foi originado pela análise própria ou provinda de um conselheiro, o resultado final da classificação, junto às *features* da amostra desconhecida e também a *timestamp* da classificação, são armazenados em forma de uma nova assinatura no repositório de assinaturas.

5.4 IMPLEMENTAÇÃO DA REDE DE CONSELHEIROS

Esta Seção apresenta a implementação que tem por finalidade a construção de uma rede de motores de detecção interligados, os quais são denominados conselheiros, a fim de permitir a cooperação entre os mesmos.

Portanto, todo motor de detecção pode ser um conselheiro, basta que este responda às requisições provindas dos outros motores de detecção, com os resultados de suas análises. Em outro momento, o mesmo motor detector que prestou conselhos poderá requisitar conselhos aos demais motores.

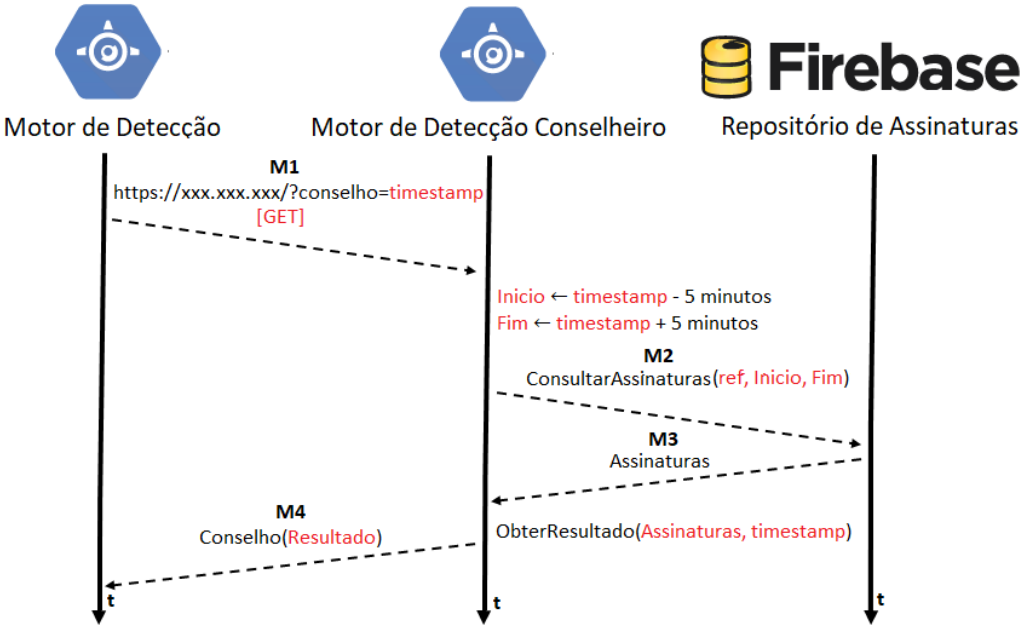


Figura 5.4: Processo de persistência e recuperação de amostras a partir do *Firebase Realtime Database*.

Conforme mencionado anteriormente, um conselho consiste na consulta ao resultado da análise de outro detector, o qual deve possuir diferentes entradas e/ou configurações. Portanto, a fim de permitir a conexão entre múltiplas instâncias de motores de detecção, todo detector deve implementar um serviço Web (*web service*) que ofereça métodos para a requisição e resposta à solicitações. A Figura 5.4, ilustra o procedimento de troca de conselhos entre dois motores de detecção. Os passos ilustrados nessa figura são descritos a seguir:

a) Na mensagem M1, o motor de detecção que necessita de conselhos faz uma requisição ao motor de detecção conselheiro. Essa requisição se dá através do HTTP, por meio

do método GET. Como parâmetro, o motor de detecção solicitante informa o *timestamp* que representa o momento em que a amostra desconhecida que está sendo analisada foi coletada pelos respectivos extratores;

b) O motor de detecção conselheiro, ao receber uma requisição de conselho, calcula os valores de *Início* e *Fim* com o propósito de estabelecer filtros que delimitam a recuperação de assinaturas. Com isso, somente as assinaturas dentro desse intervalo são recuperadas do Repositório de Assinaturas. Nesta implementação, foram consideradas as assinaturas geradas nos 5 minutos que precedem o *timestamp* da requisição e também as assinaturas geradas dentro dos 5 minutos que sucedem o *timestamp* da requisição;

c) Na mensagem *M2*, o motor de detecção conselheiro faz a consulta ao Repositório de Assinaturas especificado na referência (*ref*). Essa referência é informada no momento da configuração inicial do motor de detecção, conforme discutido anteriormente;

d) Uma vez recuperadas as assinaturas que satisfazem aos critérios de *timestamp* definidos, através da mensagem *M3*, o método *ObterResultado* define a classificação da assinatura de melhor correspondência como resultado final. Tal método tem como parâmetros de entrada o conjunto de assinaturas recuperadas. Portanto, entende-se como melhor correspondência aquela assinatura que tem menor diferença temporal entre sua *timestamp* e a *timestamp* da requisição;

e) O resultado final é enviado ao motor de detecção solicitante (na mensagem *M4*) como resposta à sua requisição (referente à mensagem *M1*).

5.5 MONITORAMENTO E RESULTADOS DE DETECÇÕES

De maneira semelhante à consulta que um motor de detecção executa antes de prestar um conselho, o administrador do sistema também está habilitado a fazer consultas baseadas em períodos específicos de tempo. Tais consultas podem se dar através da consulta aos mesmos *Web Services* consultados pelos motores requisitantes ou ainda diretamente ao banco de dados presente no Repositório de Assinaturas, de acordo com a referência desejada. A Figura 5.5 ilustra o Algoritmo, na linguagem Java, que cria um *listener* para recuperação de resultados, diretamente do repositório hospedado no *Firebase Realtime Database*, com base em sua *timestamp*, definindo-se um intervalo (*Início* e *Fim*).

```
ref.child("assinaturas").child("resultado").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(DataSnapshot resultadoAnalise) {
        Integer timestamp = resultadoAnalise.getValue(Integer.class);
        Query query = assinaturas.orderByChild("timestamp").startAt(Inicio).endAt(Fim);
        query.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(DataSnapshot dataSnapshot) {
                // Assinaturas ordenadas por timestamp
                DataSnapshot firstChild = dataSnapshot.getChildren().iterator().next();
                System.out.println("O primeiro resultado é:" + firstChild.getKey());
            }

            @Override
            public void onCancelled(DatabaseError databaseError) {
                // ...
            }
        });
    }

    @Override
    public void onCancelled(DatabaseError databaseError) {
        // ...
    }
});
```

Figura 5.5: Recuperação de assinaturas baseando-se em sua *timestamp*.

Além desse tipo de recuperação de informações, o *Firebase Realtime Database* permite que dispositivos administradores implementem seus *listeners* de eventos. Assim, quando definida uma referência a um repositório de assinaturas, as notificações sobre inserções de assinaturas de intrusões são sincronizadas em tempo real com o dispositivo monitor. Com isso, é possível a emissão de alarmes ou execução de tarefas preventivas, por exemplo. Entretanto, tais medidas não fazem parte do escopo deste trabalho. A Figura 5.6 ilustra o Algoritmo, na linguagem Java, que cria um *listener* para recuperação de novos resultados em tempo real, através de uma chamada assíncrona.

```
ref.addChildEventListener(new ChildEventListener() {
    @Override
    public void onChildAdded(DataSnapshot dataSnapshot, String prevChildKey) {
        Assinatura novaAssinatura= dataSnapshot.getValue(Assinatura.class);
        System.out.println("Resultado: " + novaAssinatura.resultado);
        }

    @Override
    public void onChildChanged(DataSnapshot dataSnapshot, String prevChildKey) {}

    @Override
    public void onChildRemoved(DataSnapshot dataSnapshot) {}

    @Override
    public void onChildMoved(DataSnapshot dataSnapshot, String prevChildKey) {}

    @Override
    public void onCancelled(DatabaseError databaseError) {}
});
```

Figura 5.6: Recuperação de assinaturas em tempo real.

6 EXPERIMENTAÇÃO

Com intuito de avaliar o desempenho da solução proposta, este capítulo descreve a experimentação prática. Todos os experimentos foram conduzidos em laboratório, com base em um conjunto de dados providos pelo *MIT Lincoln Labs*. Tal conjunto de dados contém amostras reais contendo uma variedades de intrusões simuladas em um ambiente de uma rede militar.

6.1 NSL-KDD DATA SET

A base de dados KDD Cup 99 (CUP, 1999), consiste em uma das mais populares compilações de amostras de tráfego real, contendo intrusões simuladas, utilizadas para testes de IDSs. Cada amostra contida nessa base representa dados acerca de uma conexão, os quais estão discriminados nas Tabelas 5.1, 5.2, 5.3 e 5.4. Cada uma das amostras possui 41 *features*, totalizando cerca de 100 bytes por amostra. O tamanho da amostra independe da quantidade de bytes trocados durante uma conexão, pois a amostra não contém a carga de dados trocados.

Toda amostra possui uma legenda que identifica se a mesma se refere a uma amostra de comportamento normal ou se esta caracteriza um perfil de algum tipo de ataque. Existem 22 tipos de ataques na KDD, sendo eles divididos em quatro classes principais: U2R (*User to Root*), R2L (*Remote to Local*), *Probing* e ataques de Negação de Serviço (*Denial-of-Service* – DoS). Este capítulo aborda, especialmente, as subclasses de ataques DoS. Portanto, durante os experimentos, serão analisados os seis tipos de ataques pertencentes às quatro subclasses de DoS, conforme apresentado no Capítulo 2. Todavia, a arquitetura proposta é capaz de efetuar a análise de quaisquer tipos de ataques, inclusive os demais discutidos anteriormente.

Em 2009, algumas melhorias foram feitas em cima da base KDD Cup 99, criando-se assim um novo *dataset*, chamado de NSL-KDD (TAVALLAEE et al., 2009). Os principais problemas resolvidos foram: remoção de registros redundantes do *dataset* de treinamento; remoção de amostras duplicadas do *dataset* de testes; o número de registros selecionados se baseia em seus níveis de dificuldade, a fim de possibilitar uma avaliação mais precisa de diferentes técnicas; os conjuntos de amostras para treino e testes possuem tamanhos razoáveis, evitando a necessidade da seleção de subconjuntos de forma aleatoriamente e portanto garantindo a consistência na comparação de diferentes trabalhos da literatura. Dessa forma, os *datasets* NSL-KDD de treino e teste são utilizados, respectivamente, para treinar e testar os classificadores nos experimentos

deste trabalho.

6.2 PARÂMETROS E CONFIGURAÇÕES INICIAIS

De forma a configurar detectores de diferentes tipos de fontes, as 41 features de conexões foram divididas de acordo com sua origem. Portanto, a experimentação se deu com base em uma rede de três detectores, cada um deles analisando features de um dos seguintes tipos:

a) **Detector 1:** Informações coletadas a partir de conexões TCP individuais;

b) **Detector 2:** Informações agregadas de múltiplas conexões, as quais são coletadas com base em uma janela de dois segundos antes de cada conexão e comparadas com a respectiva conexão;

c) **Detector 3:** Informações sugeridas pelo domínio de aplicação, baseada na contabilização de logs e eventos.

Conforme relatado no Capítulo 5, todos os detectores foram implementados na linguagem de programação Java, contemplando nove classificadores: IBk, J48, AD Tree, ADABost M1, KStar, Random Forest, Naive Bayes, NB Tree e Rep Tree. Para o processo de *clustering*, o número de *clusters* a serem formados foi definido como seis. Esse número corresponde ao número de ataques DoS presentes no *dataset* utilizado para experimentação. O Algoritmo responsável pela geração de clusters foi definido como o *Kmeans*, chamado de *SimpleKmeans*, na implementação da biblioteca utilizada. Outros algoritmos de clustering mais sofisticados podem ser utilizados, entretanto, nao faz parte do escopo deste trabalho a analise de qual algoritmo se da mais apropriada para cada tipo de ataque.

Além disso, acurácia mínima a ser alcançada por um classificador, durante a sua avaliação, para que seu resultado seja considerado como resultado final, foi definida como 85%. Esse mesmo valor foi definido como critério de aceite de conselhos provindos de outros motores de detecção. A acurácia mínima para que um classificador seja escolhido, exclusivamente, como o melhor classificador foi definida como 2% de superioridade em relação às acurácias dos demais classificadores. Tais valores foram selecionados devido ao fato de que os mesmos apresentaram-se eficientes em experimentos preliminares.

As definições apresentadas nos parágrafos anteriores são listadas na Tabela 6.1. Todos os motores de detecção foram configurados com as mesmas definições, exceto as fontes de dados.

Tabela 6.1: Definições consideradas nas experimentações para os três detectores.

Parâmetro	Definição
Acurácia Local Mínima Aceitável (ALMC)	85%
Acurácia Mínima Aceitável do Conselheiro (AMAC)	85%
Acurácia Mínima Conflitante (AMC)	2%
Habilitar Múltipla Escolha	Não
Número de Clusters	6
Número de Classificadores	9
Algoritmo de Clustering	Kmeans
Métrica de Avaliação de Classificadores	Acurácia

6.3 RESULTADOS E DISCUSSÕES

Uma vez definidos os dados de entrada, os mesmos foram analisados por meio dos motores de detecção configurados. A partir da análise de um subconjunto desses dados, contendo mil amostras de conexões e suas respectivas features, o processo de seleção formou seis *clusters* para cada detector, conforme definido durante as suas respectivas configurações.

O Detector 3, responsável por analisar estatísticas de logs de aplicações, ocasionou um típico caso de conflito na escolha do melhor classificador durante a classificação de uma amostra de ataque de negação de serviço *teardrop*. Em um de seus *clusters*, o qual agrupara 41 amostras similares, apresentou dois classificadores com mesmo desempenho. Ambos os classificadores IBk e NBTree apresentaram acurácias superiores aos demais, enquadrando-se nos critérios definidos como aceitáveis para a seleção. A Tabela 6.2, exibe os resultados da avaliação de classificadores, contendo as seguintes informações: nome do classificador (Classificador), acurácia alcançada na avaliação (Acurácia), Verdadeiros Positivos (VP), Falsos Negativos (FN), Falso Positivos (FP) e Verdadeiros Negativos (VN).

Dado que a seleção escolheu múltiplos classificadores, a análise deve ser executada por todos eles. Entretanto, nesse caso, houve divergência no resultado das classificações executadas pelos classificadores selecionados. O classificador IBk, que apresenta uma acurácia de 95,1295%, classificou a amostra desconhecida como intrusão, do tipo *teardrop*. Por outro lado, o classificador NBTree, com igual acurácia, classificou a mesma amostra como comportamento normal. Com base nas análises desses classificadores, o Detector 3 encontra-se em uma situação onde existe 50% de chances da emissão de um alarme falso positivo (com base no resultado do

Tabela 6.2: Conflito entre os classificadores IBk e NBTree, no contexto do Detector 3.

Classificador	Acurácia	VP	FN	FP	VN	Resultado
IBk	**95,12195%**	**28**	**2**	**0**	**11**	**Intrusão Detectada**
J48	85,36585%	24	6	0	11	-
NBTree	**95,12195%**	**28**	**2**	**0**	**11**	**Normal**
Naive Bayes	60,97561%	15	15	1	10	-
ADTree	85,36558%	24	6	0	11	-
ADABostM1	82,92683%	23	7	0	11	-
RepTree	48,78049%	9	21	0	11	-
KStar	92,68293%	27	3	0	11	-
Random Forest	75,60975%	20	10	0	11	-

classificador IBk, se este estiver incorreto) e 50% de chances de um verdadeiro negativo (com base no resultado do classificador NBTree, se este estiver incorreto).

De modo a resolver esse conflito, o Detector 3 envia uma solicitação de conselho ao Detector 1, o qual tem por responsabilidade a análise de conexões TCP individuais. Portanto, o Detector 1 fará o papel, nesse cenário, de Detector Conselheiro. Assim, ao analisar as suas respectivas *features* (ilustradas na Tabela 5.3) seleciona, sem conflitos, o classificador J48 como o melhor classificador para o perfil e fonte analisados. Portanto, o resultado enviado como resposta ao conselho solicitado se dá de forma concordante com o resultado do classificador J48, o qual confirma a detecção de intrusão do tipo *teardrop*, sugerida pelo classificador IBk, no Detector 3. A Tabela 6.3 ilustra a avaliação dos classificadores, no conselheiro (Detector 1).

Tabela 6.3: Avaliação de classificadores e o resultado do melhor classificador, no Detector 1, que analisa conexões TCP individuais.

Classifier	Accuraccy	VP	FN	VP	VN	Output
IBk	63,41463%	15	15	0	11	-
J48	**100%**	**30**	**0**	**0**	**11**	**Intrusão Detectada**
NBTree	63,41463%	15	15	0	11	-
Naive Bayes	80,48780%	30	0	8	3	-
ADTree	63,41463%	15	15	0	11	-
ADABostM1	63,41463%	15	15	0	11	-
RepTree	63,41463%	15	15	0	11	-
KStar	63,41463%	15	15	0	11	-
Random Forest	63,41463%	15	15	0	11	-

Portanto, o resultado final do Detector 3 se dá baseado no conselho vindo do Detector 1: a amostra desconhecida é classificada como ataque de negação de serviço, do tipo *teadrop*. Ademais, a base de assinaturas é atualizada com a nova amostra. Dada tal atualização, a partir desse momento, as novas ocorrências de mostras similares a esta serão automaticamente classificadas por meio do classificador IBk. Isso acontece porque a sua acurácia será superior (95,23810%) à acurácia do classificador NBTree (92,85714%). Tal fato confirma-se, quando uma nova amostra de ataque *teardrop* é analisada, na sequência.

A Tabela 6.4 demonstra a avaliação dos classificadores após a nova assinatura ter sido gerada, na detecção anterior. Neste cenário, nenhum conflito ocorre, pois não existe nenhum classificador com acurácia igual ou superior a 93,23810%, resultando na seleção direta do algoritmo IBk. Novos conflitos podem, ainda, surgir futuramente. Para que isso ocorra, o desempenho de pelo menos um dos demais classificadores devem ser afetado positivamente por meio da correta classificação de assinaturas, na fase de treinamento. Uma vez que um dos classificadores atingir uma diferença mínima de 2% em relação acurácia do algoritmo IBk, a possibilidade de conflitos passa existir novamente. Cabe ressaltar que isso não é algo negativo, pois graças a avaliação contínua dos classificadores, esse mecanismo torna a arquitetura resiliente à mudanças em padrões amostrais, garantindo que o classificador com melhor dempenho histórico seja selecionado.

Tabela 6.4: Classificaçao sem conflitos por meio do classificador especializado IBk, no contexto do Detector 3, após aprendizado baseado no conselho prestado pelo Detector 1.

Classificador	Acurácia	VP	FN	FP	VN	Resultado
IBk	**95,23810%**	**29**	**2**	**0**	**11**	**Intrusão Detectada**
J48	85,71429%	25	6	0	11	-
NBTree	92,85714%	**28**	3	0	11	-
Naive Bayes	61,90476%	16	15	1	10	-
ADTree	85,71429%	25	6	0	11	-
ADABostM1	83,33333%	24	7	0	11	-
RepTree	50,00000%	10	21	0	11	-
KStar	90,47619%	27	4	0	11	-
Random Forest	73,80952%	20	11	0	11	-

7 CONCLUSÕES

O uso de algoritmos classificadores aplicado à detecção de intrusões tem se mostrado uma alternativa interessante, pois permite que as amostras desconhecidas sejam classificadas de acordo com os perfis de assinaturas conhecidas. Entretanto, existe uma variedade de algoritmos que podem alcançar diferentes performances para cada perfil analisado. Uma vez que, em cenários reais de redes de computadores e sistemas computacionais, existe uma grande variedade de ameaças, o problema da seleção dinâmica de classificadores precisa ser abordado.

Nesse contexto, trabalhos existentes na literatura já mostram que o uso de múltiplos classificadores tende a alcançar resultados superiores aos alcançados com o uso de um único classificador. Por outro lado, existem trabalhos que exploram a escolha do único melhor classificador, o qual tem maior probabilidade de executar sua classificação corretamente, de acordo com o perfil da amostra desconhecida. Contudo, embora tais técnicas apresentam avanços em relação às técnicas anteriormente discutidas, ainda existem pontos que precisam de melhorias. A solução de conflitos entre classificadores especialistas e o incremento de confiabilidade na decisão quando a fonte analisada não possui informações suficientes consistem em desafios em abertos.

Portanto, nesta dissertação foi apresentada uma arquitetura que permite a escolha dinâmica entre classificadores que analizam múltiplas e heterogêneas fontes de dados. A seleção dinâmica de classificadores baseada em técnicas de clustering não é uma inovação da presente proposta, pois já existem trabalhos que fazem o uso dessa abordagem (MI; SIMA, 2016). Contudo, é sabido que nem sempre tal técnica resulta na escolha de um classificador que fará a correta classificação (KURZYNSKI et al., 2010). Por esse motivo, uma das principais contribuições da arquitetura proposta consiste na resiliência imposta por meio da análise de múltiplas fontes com um objetivo em comum, onde é possível medir a confiabilidade de cada um dos motores de deteção com base na avaliação de seus classifiadores.

Através da introdução do conceito de redes de conselhos, os dois problemas acima mencionados foram mitigados. Primeiramente, conflitos entre classificadores com altas taxas de detecções e acurácias aceitáveis são resolvidos por meio do uso da arquitetura proposta. Esse processo ocorre através da consulta à outros motores de detecção, os quais possuem diferentes configurações e analisam informações providas de diferentes fontes. Esses motores de detecção heterogêneos analisam dados que complementam a análise dos detectores que requisitam con-

selhos. Com isso, a segunda contribuição da presente proposta consiste no estabelecimento de um método de correlação temporal entre os motores que analisam múltiplos tipos de fontes.

De forma similar às técnicas apresentadas por (FUNG; ZHU, 2016; REHAK et al., 2009), múltiplos sistemas podem colaborar para uma análise mais robusta, em prol de efetuar a detecção de intrusões de forma eficiente. Contudo, diferentemente dessas abordagens, a arquitetura proposta possui maior independência de intervenções humanas para tomada de decisões. Ademais, o tal arquitetura possui suporte à múltiplas especialidades, onde diferentes detectores podem ser configurados de acordo com os objetivos de análise. Essa característica é alcançada por meio da escolha dinâmica do melhor classificador para cada padrão de ataque.

Os resultados revelam que a arquitetura é promissora, ao resolver conflitos na escolha do melhor classificador a partir de um conjunto de nove classificadores. Além disso, através da troca de conselhos, a acurácia do classificador conflitante que apresenta resultados concordantes ao conselho recebido é incrementada. Desse modo, em análises futuras do mesmo perfil, novos conflitos são evitados.

Em trabalhos futuros, planeja-se explorar cenários onde existem conflitos entre múltiplos motores de detecção conselheiros. Uma das possibilidades para esse tipo de abordagem consiste na implementação de um método que construa um ranking de confiabilidade, onde além da escolha dinâmica do melhor classificador, exista a escolha dinâmica do melhor detetor conselheiro.

REFERÊNCIAS

AURENHAMMER, F. Voronoi diagrams—a survey of a fundamental geometric data structure. **ACM Computing Surveys (CSUR)**, [S.l.], v.23, n.3, p.345–405, 1991.

AZEVEDO, R. P. de. Detecçao de ataques de negaçao de serviço em redes de computadores através da transformada wavelet 2d. , [S.l.], 2012.

AZIZ, A. S. A. et al. Multi-layer hybrid machine learning techniques for anomalies detection and classification approach. In: HYBRID INTELLIGENT SYSTEMS (HIS), 2013 13TH INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2013. p.215–220.

BARBARA, D. et al. ADAM: detecting intrusions by data mining. In: IN PROCEEDINGS OF THE IEEE WORKSHOP ON INFORMATION ASSURANCE AND SECURITY. **Anais...** [S.l.: s.n.], 2001.

BRITTO, A. S.; SABOURIN, R.; OLIVEIRA, L. E. Dynamic selection of classifiers—a comprehensive review. **Pattern Recognition**, [S.l.], v.47, n.11, p.3665–3680, 2014.

BROWN, L.; STALLINGS, W. **Segurança de Computadores**: princípios e práticas. [S.l.]: Elsevier Brasil, 2017.

CAMPOS, L. M. L. de; LIMA, A. S. Sistema para Detecção de Intrusão em Redes de Computadores com Uso de Técnica de Mineração de Dados. , [S.l.], 2012.

CHEN, C.-M.; LAI, G.-H.; YOUNG, P.-Y. Defense Joint Attacks Based on Stochastic Discrete Sequence Anomaly Detection. In: INFORMATION SECURITY (ASIAJCIS), 2016 11TH ASIA JOINT CONFERENCE ON. **Anais...** [S.l.: s.n.], 2016. p.74–79.

CUP, K. Dataset. **available at the following website http://kdd.ics.uci.edu/databases/kddcup99/kddcup99. html**, [S.l.], v.72, 1999.

DDOS Attack Definitions – DdoSPedia. Acessado: 10/01/2018, https://security.radware.com/ddos-knowledge-center/ddospedia/land-attack.

DEPREN, O. et al. An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks. **Expert systems with Applications**, [S.l.], v.29, n.4, p.713–722, 2005.

DEZA, M. M.; DEZA, E. Encyclopedia of distances. In: **Encyclopedia of Distances**. [S.l.]: Springer, 2009. p.1–583.

ELEKAR, K.; WAGHMARE, M.; PRIYADARSHI, A. Use of rule base data mining algorithm for intrusion detection. In: PERVASIVE COMPUTING (ICPC), 2015 INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 2015. p.1–5.

ESTER, M. et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD. **Anais...** [S.l.: s.n.], 1996. v.96, n.34, p.226–231.

FARIAS, G. Paiva Medeiros de. Detecção de intrusão em redes de computadores: uma abordagem usando extreme learning machines. , [S.l.], 2011.

FARID, D. M.; HARBI, N.; RAHMAN, M. Z. Combining naive bayes and decision tree for adaptive intrusion detection. **arXiv preprint arXiv:1005.4496**, [S.l.], 2010.

FAWCETT, T. An introduction to ROC analysis. **Pattern Recognition Letters**, [S.l.], v.27, n.8, p.861 – 874, 2006. ROC Analysis in Pattern Recognition.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. From data mining to knowledge discovery in databases. **AI magazine**, [S.l.], v.17, n.3, p.37, 1996.

FAYYAD, U.; PIATETSKY-SHAPIRO, G.; SMYTH, P. The KDD process for extracting useful knowledge from volumes of data. **Communications of the ACM**, [S.l.], v.39, n.11, p.27–34, 1996.

FUNG, C. J.; ZHU, Q. FACID: a trust-based collaborative decision framework for intrusion detection networks. **Ad Hoc Networks**, [S.l.], v.53, p.17–31, 2016.

GANAME, A. K. et al. A global security architecture for intrusion detection on computer networks. **computers & security**, [S.l.], v.27, n.1, p.30–47, 2008.

GUPTA, B.; BADVE, O. P. Taxonomy of DoS and DDoS attacks and desirable defense mechanism in a cloud computing environment. **Neural Computing and Applications**, [S.l.], v.28, n.12, p.3655–3682, 2017.

HOQUE, N. et al. Network attacks: taxonomy, tools and systems. **Journal of Network and Computer Applications**, [S.l.], v.40, p.307–324, 2014.

HWANG, K.; LIU, H.; CHEN, Y. Cooperative Anomaly and Intrusion Detection for Alert Correlation in Networked Computing Systems. **IEEE Transaction on Dependable and Secure Computing**, [S.l.], 2004.

JIRAPUMMIN, C.; WATTANAPONGSAKORN, N.; KANTHAMANON, P. Hybrid neural networks for intrusion detection system. In: ITC–CSCC. **Proceedings...** [S.l.: s.n.], 2002. p.928–931.

JOHN, G. H.; LANGLEY, P. Estimating continuous distributions in Bayesian classifiers. In: ELEVENTH CONFERENCE ON UNCERTAINTY IN ARTIFICIAL INTELLIGENCE. **Proceedings...** [S.l.: s.n.], 1995. p.338–345.

KANUNGO, T. et al. The analysis of a simple k-means clustering algorithm. In: COMPUTATIONAL GEOMETRY. **Proceedings...** [S.l.: s.n.], 2000. p.100–109.

KEVRIC, J.; JUKIC, S.; SUBASI, A. An effective combining classifier approach using tree algorithms for network intrusion detection. **Neural Computing and Applications**, [S.l.], v.28, n.1, p.1051–1058, 2017.

KIM, G.; LEE, S.; KIM, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. **Expert Systems with Applications**, [S.l.], v.41, n.4, p.1690–1700, 2014.

KOHAVI, R. Scaling Up the Accuracy of Naive-Bayes Classifiers: a decision-tree hybrid. In: KDD. **Anais...** [S.l.: s.n.], 1996. v.96, p.202–207.

KURZYNSKI, M. et al. On two measures of classifier competence for dynamic ensemble selection-experimental comparative analysis. In: COMMUNICATIONS AND INFORMATION TECHNOLOGIES (ISCIT), 2010 INTERNATIONAL SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2010. p.1108–1113.

LIMA, B. N. et al. Entropia: introdução à teoria matemática da (des) informação. , [S.l.], 2012.

MACQUEEN, J. et al. Some methods for classification and analysis of multivariate observations. In: BERKELEY SYMPOSIUM ON MATHEMATICAL STATISTICS AND PROBABILITY. **Proceedings...** [S.l.: s.n.], 1967. v.1, n.14, p.281–297.

MARCHAL, S. et al. A big data architecture for large scale security monitoring. In: BIG DATA (BIGDATA CONGRESS), 2014 IEEE INTERNATIONAL CONGRESS ON. **Anais...** [S.l.: s.n.], 2014. p.56–63.

MI, A.; SIMA, H. Classifier selection method based on clustering and weighted mean. **Journal of Intelligent & Fuzzy Systems**, [S.l.], v.31, n.4, p.2335–2340, 2016.

NGAI, E. W.; XIU, L.; CHAU, D. C. Application of data mining techniques in customer relationship management: a literature review and classification. **Expert systems with applications**, [S.l.], v.36, n.2, p.2592–2602, 2009.

OLIVEIRA ROCHA, R. C. de. **Detecção em Tempo-Real de Ataques de Negação de Serviço na Rede de Origem Usando um Classificador Baysiano Simples**. [S.l.]: Pontífica Universidade Católica de Minas Gerais, 2013.

OLUSOLA, A. A.; OLADELE, A. S.; ABOSEDE, D. O. Analysis of KDD'99 intrusion detection dataset for selection of relevance features. In: WORLD CONGRESS ON ENGINEERING AND COMPUTER SCIENCE. **Proceedings...** [S.l.: s.n.], 2010. v.1, p.20–22.

PALIWAL, S.; GUPTA, R. Denial-of-service, probing & remote to user (R2L) attack detection using genetic algorithm. **International Journal of Computer Applications**, [S.l.], v.60, n.19, p.57–62, 2012.

REHAK, M. et al. Adaptive multiagent system for network traffic monitoring. **IEEE Intelligent Systems**, [S.l.], v.24, n.3, 2009.

REVATHI, S.; MALATHI, A. Detecting user-to-root (U2R) attacks based on various machine learning techniques. **Int. J. Adv. Res. Comput. Commun. Eng**, [S.l.], v.3, n.4, p.6322–6324, 2014.

RILEY, R.; JIANG, X.; XU, D. Guest-transparent prevention of kernel rootkits with vmm-based memory shadowing. In: INTERNATIONAL WORKSHOP ON RECENT ADVANCES IN INTRUSION DETECTION. **Anais...** [S.l.: s.n.], 2008. p.1–20.

ROESCH, M. et al. Snort: lightweight intrusion detection for networks. In: LISA. **Anais...** [S.l.: s.n.], 1999. v.99, n.1, p.229–238.

SABOURIN, M. et al. Classifier combination for hand-printed digit recognition. In: DOCUMENT ANALYSIS AND RECOGNITION, 1993., PROCEEDINGS OF THE SECOND INTERNATIONAL CONFERENCE ON. **Anais...** [S.l.: s.n.], 1993. p.163–166.

SANTOS, M. E. C. dos et al. PUBLICAÇÃO DE DADOS GOVERNAMENTAIS ABERTOS CONECTADOS: um estudo de caso no âmbito da saúde pública brasileira. **WWW/INTERNET 2016**, [S.l.], p.36, 2016.

SHARMA, I.; JOSHI, K. Performance Evaluation with K-Mean and K-Mediod in Data Mining. , [S.l.], 2015.

SHARMA, N.; BAJPAI, A.; LITORIYA, M. R. Comparison the various clustering algorithms of weka tools. **facilities**, [S.l.], v.4, n.7, 2012.

SPECHT, S. M.; LEE, R. B. Distributed Denial of Service: taxonomies of attacks, tools, and countermeasures. In: ISCA PDCS. **Anais...** [S.l.: s.n.], 2004. p.543–550.

TAN, P.-N. et al. **Introduction to data mining**. [S.l.]: Pearson Education India, 2006.

TAVALLAEE, M. et al. A detailed analysis of the KDD CUP 99 data set. In: COMPUTATIONAL INTELLIGENCE FOR SECURITY AND DEFENSE APPLICATIONS, 2009. CISDA 2009. IEEE SYMPOSIUM ON. **Anais...** [S.l.: s.n.], 2009. p.1–6.

TORRES-PEREIRA, E. et al. Hybrid Parallel Cascade Classifier Training for Object Detection. In: IBEROAMERICAN CONGRESS ON PATTERN RECOGNITION. **Anais...** [S.l.: s.n.], 2014. p.810–817.

WITTEN, I. H. et al. **Data Mining**: practical machine learning tools and techniques. [S.l.]: Morgan Kaufmann, 2016.

WITTEN, I. H.; MINING, E. F. D. **Practical machine learning tools and techniques Morgan Kaufmann**. [S.l.]: San Francisco, 2005.

WOODS, K. et al. Combination of multiple classifiers using local accuracy estimates. **IEEE transactions on pattern analysis and machine intelligence**, [S.l.], v.19, n.4, p.405–410, 1997.

ZIN, T. et al. **Genetic and Evolutionary Computing**: proceedings of the ninth international conference on genetic and evolutionary computing, august 26-28, 2015, yangon, myanmar -.

[S.l.]: Springer International Publishing, 2015. n.v. 1. (Advances in Intelligent Systems and Computing).

ZUECH, R. et al. Intrusion detection and big heterogeneous data: a survey. **Journal of Big Data**, [S.l.], v.2, n.1, p.3, 2015.

8 ARTIGO SUBMETIDO AO ISCC

A Counselors-Based Intrusion Detection Architecture.

A Counselors-Based Intrusion Detection Architecture

Silvio E. Quincozes
Department of Applied Computing
Universidade Federal de Santa Maria
Santa Maria, Brazil 30332–0250
Email: sequincozes@inf.ufsm.br

Raul Ceretta Nunes
Department of Applied Computing
Universidade Federal de Santa Maria
Santa Maria, Brazil 30332–0250
Email: ceretta@inf.ufsm.br

Carlos Raniery P. dos Santos
Department of Applied Computing
Universidade Federal de Santa Maria
Santa Maria, Brazil 30332–0250
Email: csantos@inf.ufsm.br

***Abstract*—Intrusion Detection Systems (IDSes) are a fundamental component of defensive solutions to detect anomalies on computer systems and networks. IDSes usually rely on data classification algorithms to detect malicious activities during the analysis process. However, the development of such systems presents challenging possibilities of research. For example, when several classification algorithms are employed, conflicts on the obtained results can lead to unreliable actions. Further, analyzing just one source of information is not enough for a proper decision. This work aims to solve those problems by introducing a novel architecture based on the dynamic selection of classifiers. Those classifiers are grouped in council networks, responsible for analyzing multiple and heterogeneous data sources. The results demonstrate the feasibility of the proposed architecture in improving the accuracy of the intrusion detection process.**

I. Introduction

Nowadays there is a wide variety of possibilities for attackers to perform intrusive or malicious activities. In order to reduce the damages caused by those attacks, it is fundamental to implement and use Intrusion Detection Systems (IDSes). The purpose of an IDS is to monitor an environment and identify when a suspicious activity is happening [1]. When an attack is properly identified, the appropriate countermeasures can be taken in order to mitigate, or eliminate, the damages in the target system.

In order to distinguish between suspicious and normal activities, disparate data analysis techniques could be employed (*e.g.*, data mining). In the plethora of alternatives, classification techniques are the most used in the literature for detecting anomalies. This method allows the prediction of unknown samples based on a training step, where known signatures are used to prepare the classifiers [2].

Selecting the most appropriate classifier algorithm is a challenging task. There are diverse algorithms available in the literature, but there's no "one size fits all" classifier, i.e., a classifier that performs better in identifying one class of attack may be inefficient for other classes. Further, the development of one monolithic classifier that covers all the possible attacks is not feasible. This leads to a scenario where the result calculated by an IDS may be unreliable [3].

In order to solve this issue, some research efforts propose the use of methods to dynamically select the most appropriate classifier for each attack pattern [4], [5]. Besides the advantages of such approach, they are limited in scenarios where multiple expert classifiers present divergent results. For example, the chosen classifier can still provide wrong results depending on the data available for analysis.

By considering the limitations of existing solutions, this work proposes an architecture that combines dynamic classifier selection with the analysis of multiple and heterogeneous data sources. In this way, a counselors' network – a novel concept introduced in this paper – enables each detector node to solve conflicts by relying on the results from expert classifiers, thus leading to an increased reliability of the final decision.

The remainder of the paper is organized as follows: in Section II the related efforts available in the literature are presented. In Section III, the proposed architecture is introduced and discussed. In Section IV, the proposed solution is evaluated and the obtained results are discussed. Finally, in Section V the conclusions and future works are presented.

II. Related Works

The literature shows that IDSs that use multiple classifiers usually present superior results [6][7]. Further, collaboration among multiple systems analyzing different data sources is also an interesting approach to be used in IDSs. In this context, this Section presents the advances in the field of dynamic selection of classifiers and IDS collaborative architectures.

A. Multiple Classifier Selection

Sabourin [8] introduces one of the first solutions for the dynamic selection of classifiers. The solution is based on a classifier ranking method, which uses the results estimated from a training database. A simplified version of this work is proposed in [5], where the classifiers are ranked based on their accuracy in the classification of the known neighboring samples. However, in both approaches there is a common problem: they are prone to fail for unknown samples. Additionally, these proposals do not present any decision criteria when there is a conflict among the classifiers.

In [4], the authors present a methodology for estimating the best classifier of a given class. In the proposed solution, the competence of each classifier is evaluated according with it performance on specifics classes. Thus, the probability of

correct classification of an unknown sample is defined from results average given for each class evaluated. However, in real scenarios there are many attack types and classes, and this approach becomes costly to apply.

A classifier selection method based on clustering and weighted average is proposed in [7]. This method consists in clustering know samples and measuring the performance of each classifier for all the formed clusters. Then, the unknown sample is associated to the nearest centroid cluster. The classifier with best average performance to nearest cluster and the classifier with the best performance for the nearest neighboring cluster are selected to classify it. Finally, from these selected classifiers, the one with the highest reliability is chosen. Reliability is defined by means of a confusion matrix from known results. However, conflicts can occur either in the selection of the both classifiers or even in the final decision from selected classifiers.

The proposals presented in this section present improvements when compared to a single classifier. Even so, the choice of the best classifier remains an open issue, specially when considering conflict among the classifiers. In addition, these works does not consider collaboration from multiples data sources.

B. Collaborative Architectures

Although the collaboration between specialists systems to analyze multiple and heterogeneous sources is barely investigated in the literature [9], there are some works that propose approaches to consolidate data from different sources for the detection of intrusions.

In [10], syslog events are correlated with alerts generated by Snort [11]. This correlation is based on the context in which events and alerts are generated – defined by a time window. Therefore, records with predefined sequences coming from both sources allow the detection of more complex attacks. However, according to [9], the original architecture structure is prone to Denial-of-Service (DoS) and has performance issues when being executed in a distributed fashion.

In [12], a distributed framework is presented to optimize results from heterogeneous IDSes. These IDSes are evaluated separately by event injection, where labeled events are used to check it specialty. Additionally, each IDS node can compare feedbacks with other to verify each node specialty. Nevertheless, conflicting scenarios are not addressed. Also, multiple source correlation is not considered.

A multiple heterogeneous source architecture for intrusion detection is presented in [13]. Network flows, such as HTTP and DNS, are correlated by using their IP addresses. However, a system administrator is required to define rules and make the final decision when multiple correlated alarms are presented.

In [14], an architecture for processing heterogeneous alerts is proposed. This architecture has an anomaly detector that works with Snort IDS. The main purpose of this proposal is to reconfigure Snort to support new signatures. However, those signatures can often be false alarms since they are generated based on some rules created by a human.

An architecture for detecting multiple-step attacks is proposed in [15]. The authors analyze logs in netflow and syslog formats in order to identify sequential patterns for the same IP address. This approach aims to detect malicious actions that can be identified at different times. An example of this type of analysis is the correlation of authentication error logs with failure logs in attempting to access the operating system's root user. However, the network administrator must monitor and take preventive measures. Therefore, this type of approach presents the same limitation discussed previously: the strong dependence of human action for decision making.

A collaborative network of IDS is proposed by [16]. In this proposal, detectors are deployed in different parts of the network. In this way, each detector is able to perform its own analysis and also consult other detectors that have different specialties or signature knowledge. All IDS in network analyze homogeneous sources, therefore, the results are redundant from a global point of view. Despite the limitations, we consider this work as an inspiration for the creation of intrusion detection networks. However, we note that this network could be improved if multiple and heterogeneous sources are analyzed.

C. Sumary

Some efforts available in the literature present issues when multiple sources are analyzed, thus leading to the need of employing a human for the decision making. On the other hand, architectures that have greater independence from human action usually analyze just one single data source.

As discussed, the existing architectures in literature, usually, depends of human interaction for decision making. As will be presented in next section, our proposal mitigates the need for human involvement in decision making, making this an automated procedure. Our architecture learns and gains maturity as new signatures are added to the database of known samples. Additionally, in our proposal integrates multiple data sources, enabling a stronger final decision.

TABLE I
RELATED WORKS COMPARISON

Ref	Souce	Integration	Self-learning	Human Dependency
[14]	Single	No	Yes	Highw
[12]	Single	Yes	Yes	Low
[16]	Single	Yes	Yes	Low
[10]	Multiple	Yes	No	High
[15]	Multiple	Yes	No	High
[13]	Multiple	No	No	High
Our	Multiple	Yes	Yes	Low

In Table I, our proposal is compared to others literature solutions in terms of the potential to run without human intervention, the possibility of integration with multiple detectors, the ability to self-learn and the possibility of dealing with heterogeneity of data from different sources.

III. Proposed Architecture

This section presents the proposed IDS counselors-based architecture. Figure 1 illustrates the architecture overview and its components are explained in the following sections.

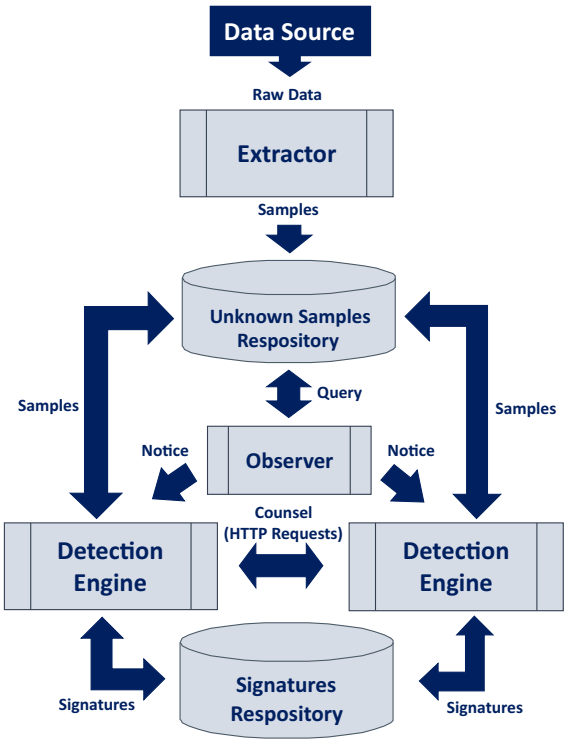


Fig. 1. Architecture Overview.

A. Extractor

The first step in analyzing data is to collect it. Thus, the extractor component is responsible for reading data from heterogeneous sources and extracting relevant features, so it must implement the required methods according to source that is being monitored, such as Application logs or capture network packets.

All data collected by the Extractor is transfered to the Unknown Samples Respository, that stores them and provides access to detection engines. In real scenarios, where large volumes of data must be processed, bottlenecks in storage can compromise system performance. In this context, we employed tools that provide good performance for retrieving the stored data.

B. Observer

An interesting component of the proposed architecture is the Observer. This component aims to constantly monitor the unknown samples repository in order to notify the detectors about new data for analysis. Thus, any unnecessary processing overhead is removed from the detection component, which should focus only on data analysis. Therefore, whenever the extractor collects and inserts new records in the repository of unknown samples, the observer should check which detectors are consumers of that content and notify them. To this end, the types of sources and analysis purposes of each of the detectors must be considered.

C. Detection Engines

The Detection Engines are the most important components of the architecture. They contains all the algorithms used to detecting intrusions, such as data mining techniques. These algorithms are applied on data features retrieved from unknown samples repository.

Additionally, they are responsible for constitute a counselors network. In this network, detectors can exchange advices with others that consumes features from different sources. This exchange occur by web service requests.

The process performed by the Detection Engines is organized in four steps: Data consumption, clustering and classifier evaluation, classifier selection and counselors exchanging.

In the beginning, both attacks and normal signatures are recovered from a Signature Repository. These steps are detailed in the follow.

1) Data Consumption: Data consumption occurs in two different moments. First, the detector needs be trained with known and labeled samples in order to learn its features. In the second moment, a detector can perform the intrusion detection. In this context, even that the Observer notices when a new sample is collected, this sample needs first to be consumed in order to be analyzed.

For learning purpose, a detector must download both attack and normal signatures from the signatures repository. Signatures are composed by a set of features. Then, a detector can filter for those that is relevant for it analysis. This is a important procedure because prevents the existence of irrelevant features that can interfere negatively in the results [17].

During the intrusion detection process, notifications from Observer trigger new analysis tasks. At this detection stage, the engine will only update the signatures if the Observer notifies it about changes in the central signatures repository.

2) Clustering and Classifier Evaluation: The clustering process allows all features of known samples to be grouped according to their similarities. Then, it is possible to validate the performance of each classifier for each cluster and defines the best classifier for each of them, as shown in Figure 2.

By default, our proposed solution evaluates the classifiers based on it accuracy. However, others metrics such as detection rate or false alarms rate can also be used.

3) Classifier Selection: This step consists on the dynamic selection of the most promising classifier for a specific sample classification. To this end, the Euclidean distance of each unknown sample to each centroid of the clusters generated in the previous step is measured. We rely on this metric because it is common to be used by clustering algorithms. Then, the nearest centroid defines which cluster the sample belongs to. In this way, it is possible to verify which is the classifier with more accuracy for the selected cluster. Then, the selected classifier will take a final decision.

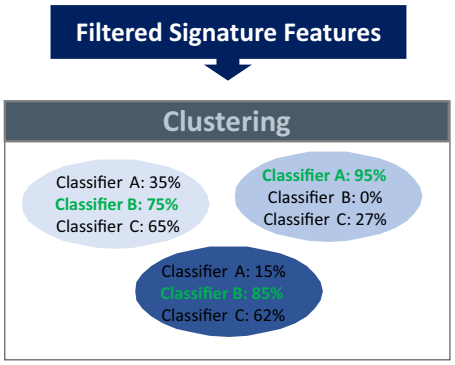


Fig. 2. Clustering and Classifier Evaluation step

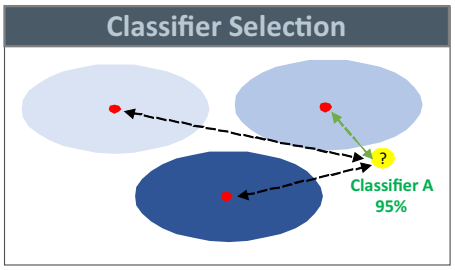


Fig. 3. Dynamic Classifier Selection

Therefore, as shown in Figure 3, there are no challenges in simply using the best cluster classifier to classify the unknown sample. On the other hand, there are two situations where it is not so interesting. A first one is when there are conflicts between different classifiers with good performance. In this case, it's fundamental to define which one is presenting true and reliable results. Another challenge is choosing a reliable classifier when none of the classifiers used can achieve good detection. In that case, any decision can be reckless.

Fortunately, our proposal addresses these issues. To deal with insufficient data or with conflicts between classifiers, we introduced a network of heterogeneous counselors detectors. Further detail will be explained below.

4) Counselors Network : The main purpose of this network is to enable the mutual knowledge exchanging between detectors with different points of view of the same threat situation. So, this network is composed of detectors that analyzes multiples and heterogeneous fonts.

Heterogeneous Detectors Counselors (HDC) pairs must analyze data from multiple and heterogeneous sources, but still with the same detecting finality (e.g. DoS attacks). In this scenario, host detectors and network detectors can mutually ask for advices. For example, Figure 4 illustrates one detector requesting an advice to a HDC pair. Based on the accuracy from the HDC pairs, a detector can choose which one has a higher level of confidence for a particular purpose. In the same way, a detector should responds to requests from other HDCs. Figure 5 illustrates a detector playing the HDC hole, responding to advice's request from other HDC pairs. The

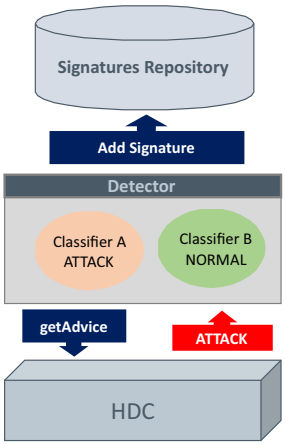


Fig. 4. Detector requesting an advice to HDC.

advice exchanging between detectors occurs through Web Service requests.

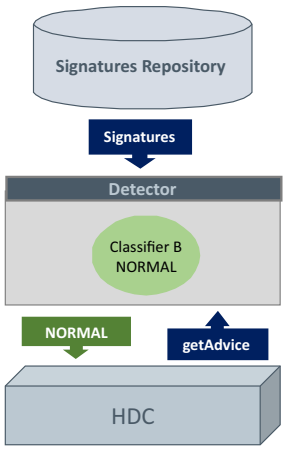


Fig. 5. Detector as HDC responding a advice request from another HDC.

Whenever a detector receives a valid advice – with acceptable accuracy – , it must generate a new signature and persist it on the repository. Thus, enabling new patterns to be learned and avoiding further requests for advices. This process can be observed in *Add Signature* message, in Figure 4.

IV. Evaluation

In order to validate the proposed architecture, an experimental environment was developed. Therefore, to evaluate the architecture performance, we implemented and configured its components according to parameters that will be described in this section.

A. Setup

We used a Java implementation, with data mining algorithms powered by Weka library, version 3.8.0. Then, we chose a set of classifiers, taking as a criterion the choice of those most used in the literature. In addition, we have also

chosen SimpleKMeans algorithm, with K=6, was chosen for clustering because of the characteristics of our dataset.

For classification purpose, we used a set of nine classifiers: KStar, IBk, J48, RandomForest, REPTree, NBTree, ADTree, NaiveBayes and AdaBoostM1. As a criterion for choosing the best classifier, we rely on its local accuracy within each of the training signature clusters, which are formed by the clustering algorithm. Thus, when the detectors is running, the most appropriate classification algorithm is chosen based on its local class accuracy.

In order to assign an unknown sample to the most similar cluster, we rely on the distance between these samples to each cluster centroid. The nearest cluster is choose. For this purpose, we use the Euclidean distance function, which is the default function used by SimpleKMeans.

Finally, for the request of councils, we assume the following parameters: the minimum acceptable accuracy for a classifier to be chosen is 80%; A conflict occurs whenever two classifiers are evaluated with less than 5% difference between their accuracies, and one presents different results from other to an unknown sample. As counselors, we consider the detectors that analyze different types of sources, but that have the same analytical purpose – employed in detection of same intrusion classes. To maintain consistency between advice from different sources, we defined a two-seconds time window from timestamps sent in the request. This definition is based on KDD Cup 99 sample dataset [18].

B. Data Source

In order to simulate a scenario with multiple attacks, we used a dataset that is widely used in the literature, called NSL-KDD [19]. This dataset consists in an improved version of the popular KDD Cup 99.

From the 41 existing features in this dataset we chose those that allow the data sources simulation. For this purpose, it was divided according to their origin. Then, three detectors of heterogeneous data sources were configured. Each analyzing, respectively: TCP connections, Traffic Statistics and Application logs.

A set of 1,000 connection samples and their respective features were chosen. In this way, as we defined K = 6, the clustering process formed six clusters for each detector.

C. Results

When analyzing the samples, Detector 1 was found a conflict among two classifiers. Both the IBk and NBtree algorithms presented 28 true positives, 2 false negatives and 11 true negatives. Thus, both presented an accuracy of 95.112195%, as Table II illustrates. Therefore, leading to an advice request. When the classifiers were evaluated, two of them presented the same level of performance.

The same results among well-rated classifiers is not a problem; on the contrary, this only strengthens the confidence in the classification process. However, when two classifiers disagree on the classification of an unknown sample, detection engine considers that a conflict has occurred. This situation is challenging and needs to be solved as quickly as possible to prevent false alarms from being triggered, or alarms to be omitted when attacks are occurring.

TABLE II
CONFLICT IN CLASSIFIER SELECTION FOR DETECTOR 1.

Classifier	Accuraccy	VP	FN	VP	VN	Output
IBk	**95,12195%**	**28**	**2**	**0**	**11**	**Attack**
J48	85,36585%	24	6	0	11	-
NBTree	**95,12195%**	**28**	**2**	**0**	**11**	**Normal**
Naive Bayes	60,97561%	15	15	1	10	-
ADTree	85,36558%	24	6	0	11	-
ADABostM1	82,92683%	23	7	0	11	-
RepTree	48,78049%	9	21	0	11	-
KStar	92,68293%	27	3	0	11	-
Random Forest	75,60975%	20	10	0	11	-

During the analysis of a teardrop attack sample – a subclass of Denial of Service – that was analyzed by Detector 1, a disagreement occurs between the two well-rated classifiers: The IBk classifier classified this sample as an attack, while the NBTree classifier identified this sample as being a normal, as shown in Table II.

For evaluation purposes, we consider this sample as an unknown sample. In this way, to choose the correct classifier output, it is necessary that Detector 1 requests an advice to the counselors network. So, we ask to Detector 2, which is analyzing TCP connections statistics. Detector 2, that is already running, receive a request from Detector 1, containing a timestamp. Based on this timestamp, Detector 2 will check for anomalies in its logs within this time context. In this experiment, a log indicating the existence of teardrop attack was found. For this detection, Detector 2 has selected J48 as best classifier without conflicts, as shown in III.

TABLE III
CLASSIFIER SELECTION AND OUTPUT FROM BEST CLASSIFIER FROM DETECTOR 2.

Classifier	Accuraccy	VP	FN	VP	VN	Output
IBk	63,41463%	15	15	0	11	-
J48	**100%**	**30**	**0**	**0**	**11**	**Attack**
NBTree	63,41463%	15	15	0	11	-
Naive Bayes	80,48780%	30	0	8	3	-
ADTree	63,41463%	15	15	0	11	-
ADABostM1	63,41463%	15	15	0	11	-
RepTree	63,41463%	15	15	0	11	-
KStar	63,41463%	15	15	0	11	-
Random Forest	63,41463%	15	15	0	11	-

Therefore, the final result from Detector 1 is based on the advice coming from Detector 2: the unknown sample is classified as attack. In addition, the signature database is updated with the new sample. From this point on, the IBk classifier will be selected as the best classifier for this cluster, until new information changes the knowledge base.

This happens because its updated accuracy will be higher than NBTree and others classifiers.

D. Discussion

As previously shown, our approach is feasible to solve conflicts in Dynamic Classifier Selection. In scenarios similar to the one discussed, the request for information available in heterogeneous sources may be decisive for the detectors to work properly.

However, we recognize that there are still points that need to be addressed. Supposing a third detector analyzing another data source, presents an advice with a different result from detector 2. In this case, some methods that evaluate the reliability of detectors must be implemented.

Therefore, among the possible solutions that exist for this problem, we enumerate two approaches: (i) The level of accuracy of the best classifier of the detector 2 could be compared to the level of accuracy of the best classifier of the detector 3, choosing the detector with the highest classifier accuracy; (ii) Detector 1 can maintain a reliability history, where even detectors are ranked according to the accuracy of their results – in this approach, test requests can be performed through the use of known samples to evaluate peers detectors.

V. Conclusion

Since in the real world of computer networks and computer systems there is a wide variety of attacks, this issue needs to be carefully addressed. Data classification can be helpful to deal with this. In this context, the literature demonstrates that the use of multiple classifier algorithms is more efficient than a single algorithm. On the other hand, we observed that many existing works are only concerned with choosing the best algorithm for correctly classifying samples from a single data source, consequently sometimes presenting errors or conflicts.

This work proposed a dynamic architecture that allows multiple classifiers to analyze multiple heterogeneous data sources in order to obtain a more robust final decision by advices exchanging. Our experiments demonstrate that dynamic classifier selection can be strongly empowered when multiple sources are analyzed, where we are able to solve conflicts and improve final decision reliability. So, our proposed architecture is promising for use in scenarios where multiple possibilities of attacks may occur.

In future works, we plan to deal with scenarios where there are conflicts between two or more counselors. One possibility to approach this topic could be to construct a method that measures the reliability of each detector pair, always choosing the one with the highest reliability. In addition, we will exploit other values for K in order to define the ideal configuration for the scenario where the architecture is employed.

References

[1] R. Singh, H. Kumar, and R. Singla, "An intrusion detection system using network traffic profiling and online sequential extreme learning machine," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8609–8624, 2015.

[2] E. W. Ngai, L. Xiu, and D. C. Chau, "Application of data mining techniques in customer relationship management: A literature review and classification," *Expert systems with applications*, vol. 36, no. 2, pp. 2592–2602, 2009.

[3] A. S. Britto, R. Sabourin, and L. E. Oliveira, "Dynamic selection of classifiersa comprehensive review," *Pattern Recognition*, vol. 47, no. 11, pp. 3665–3680, 2014.

[4] T. Woloszynski and M. Kurzynski, "A measure of competence based on randomized reference classifier for dynamic ensemble selection," in *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2010, pp. 4194–4197.

[5] K. Woods, W. P. Kegelmeyer, and K. Bowyer, "Combination of multiple classifiers using local accuracy estimates," *IEEE transactions on pattern analysis and machine intelligence*, vol. 19, no. 4, pp. 405–410, 1997.

[6] J. Kevric, S. Jukic, and A. Subasi, "An effective combining classifier approach using tree algorithms for network intrusion detection," *Neural Computing and Applications*, pp. 1–8, 2016.

[7] A. Mi and H. Sima, "Classifier selection method based on clustering and weighted mean," *Journal of Intelligent & Fuzzy Systems*, vol. 31, no. 4, pp. 2335–2340, 2016.

[8] M. Sabourin, A. Mitiche, D. Thomas, and G. Nagy, "Classifier combination for hand-printed digit recognition," in *Document Analysis and Recognition, 1993., Proceedings of the Second International Conference on*. IEEE, 1993, pp. 163–166.

[9] R. Zuech, T. M. Khoshgoftaar, and R. Wald, "Intrusion detection and big heterogeneous data: a survey," *Journal of Big Data*, vol. 2, no. 1, p. 3, 2015.

[10] A. K. Ganame, J. Bourgeois, R. Bidou, and F. Spies, "A global security architecture for intrusion detection on computer networks," *computers & security*, vol. 27, no. 1, pp. 30–47, 2008.

[11] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.

[12] K. Bartos and M. Rehak, "Self-organized mechanism for distributed setup of multiple heterogeneous intrusion detection systems," in *Self-Adaptive and Self-Organizing Systems Workshops (SASOW), 2012 IEEE sixth international conference on*. IEEE, 2012, pp. 31–38.

[13] S. Marchal, X. Jiang, R. State, and T. Engel, "A big data architecture for large scale security monitoring," in *Big data (BigData Congress), 2014 IEEE international congress on*. IEEE, 2014, pp. 56–63.

[14] K. Hwang, H. Liu, and Y. Chen, "Cooperative anomaly and intrusion detection for alert correlation in networked computing systems," *IEEE Transaction on Dependable and Secure Computing*, 2004.

[15] C.-M. Chen, G.-H. Lai, and P.-Y. Young, "Defense joint attacks based on stochastic discrete sequence anomaly detection," in *Information Security (AsiaJCIS), 2016 11th Asia Joint Conference on*. IEEE, 2016, pp. 74–79.

[16] C. J. Fung and Q. Zhu, "Facid: A trust-based collaborative decision framework for intrusion detection networks," *Ad Hoc Networks*, vol. 53, pp. 17–31, 2016.

[17] A. A. Olusola, A. S. Oladele, and D. O. Abosede, "Analysis of kdd99 intrusion detection dataset for selection of relevance features," in *Proceedings of the World Congress on Engineering and Computer Science*, vol. 1, 2010, pp. 20–22.

[18] (1999) Kdd-cup-99 task description. [Online]. Available: https://archive.ics.uci.edu/ml/machine-learning-databases/kddcup99-mld/task.html

[19] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009. IEEE Symposium on*. IEEE, 2009, pp. 1–6.