Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de comportamentos maliciosos de usuários internos em redes utilizando análise de fluxos de dados

Rafael Bruno Peccatiello

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Orientador
Prof. Dr. Luis Paulo Faina Garcia

Brasília
2023

Ficha catalográfica elaborada automaticamente,
com os dados fornecidos pelo(a) autor(a)

BP365d

Bruno Peccatiello, Rafael
Detecção de comportamentos maliciosos de usuários internos em redes utilizando análise de fluxos de dados / Rafael Bruno Peccatiello; orientador Luis Paulo Faina Garcia. -- Brasília, 2023.
70 p.

Dissertação(Mestrado Profissional em Computação Aplicada) -- Universidade de Brasília, 2023.

1. ameaças internas. 2. fluxo de dados. 3. aprendizado de máquina. 4. ciência de dados. 5. desbalanceamento. I. Paulo Faina Garcia, Luis, orient. II. Título.

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Detecção de comportamentos maliciosos de usuários internos em redes utilizando análise de fluxos de dados

Rafael Bruno Peccatiello

Dissertação apresentada como requisito parcial para conclusão do
Mestrado Profissional em Computação Aplicada

Prof. Dr. Luis Paulo Faina Garcia (Orientador)
Presidente

Prof. Dr. Marcos Fagundes Caetano
Membro Interno

Prof. Dr. Sylvio Barbon Junior
Membro Externo

Prof. Dr. Thiago de Paulo Faleiros
Suplente

Prof. Dr. Gladston Luiz da Silva
Coordenador do Programa de Pós-graduação em Computação Aplicada

Brasília, 11 de Setembro de 2023

Dedicatória

Dedico este trabalho aos meus pais, irmãos, esposa e filha, pois são minha fonte inesgotável de motivação para atingir todos os objetivos que traço na vida.

Agradecimentos

Primeiramente, agradeço a Deus por soprar o dom da vida nos meus pulmões, me fazendo forte e estando presente em todos os momentos, sejam eles bons ou ruins, vezes me ouvindo agradecer e vezes ouvindo minhas súplicas. Sempre me concedendo muito mais graças do que eu realmente mereço.

Agradeço à minha família. Minha esposa Melissa e filha Giovanna por me aceitarem com meus defeitos e me amarem mesmo assim, entendendo os meus momentos de ausência. Meus pais Italo e Sônia pelo esforço em me preparar para o mundo. Aos meus irmãos Renata e Rodolfo por serem meu porto seguro junto aos nossos pais.

Agradeço ao meu orientador, o Prof. Dr. Luís P. F. Garcia por ter sido os olhos por onde enxerguei o mundo acadêmico, pelo seu profissionalismo, por ter compartilhado comigo o seu conhecimento e pela firmeza na condução do nosso trabalho.

Por fim, agradeço aos meus companheiros de turma, ao Exército Brasileiro e a Universidade de Brasília por terem fornecido os subsídios para que esse trabalho se concretizasse.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES), por meio do Acesso ao Portal de Periódicos.

Resumo

Uma ameaça interna é qualquer pessoa que tenha acesso legítimo à rede interna de uma determinada organização e utiliza esse acesso para executar ações danosas a essa organização. As ameaças internas podem agir com ou sem intenção, quando agem intencionalmente, geralmente também possuem alguma motivação específica. Essa motivação pode variar entre descontentamento pessoal, questões financeiras, coerção e outros. A dificuldade em enfrentar ameaças internas com soluções de segurança tradicionais reside na limitação dessas soluções ao paradigma de detecção baseada em assinatura. Para superar essa restrição, pesquisadores da área têm proposto o uso do aprendizado de máquina, com a intenção de abordar o problema das ameaças internas de maneira mais abrangente. Alguns deles utilizando a abordagem de aprendizado em lote e outros utilizando o aprendizado em fluxo. Abordagens em lote são mais simples de implementar, porém o problema é como aplicá-las no mundo real. Isso ocorre porque os cenários reais de ameaças internas têm características peculiares que não se adaptam bem ao aprendizado em lote. Como por exemplo, temos a impossibilidade de se obter um conjunto de dados finito pois os *logs* sobre as atividades de usuários internos não cessam. Embora mais complexas, as abordagens de fluxo são mais abrangentes e viáveis de implementar. Os estudos também se dividem no uso de técnicas de *Machine Learning* não supervisionadas e supervisionadas, entretanto a dificuldade de obtenção de amostras rotuladas em tempo de execução, prejudica a implementação de soluções totalmente supervisionadas. Este estudo propõe uma estrutura que combina diferentes técnicas de ciência de dados para abordar a detecção de ameaças internas. Entre elas estão o uso de aprendizado de máquina não supervisionado e supervisionado, análise de fluxo de dados e procedimentos periódicos de retreinamento do modelo. Os algoritmos utilizados na implementação foram *Isolation Forest, Elliptic Envelop, Local Outlier Factor e Random Forest*. Este estudo avaliou os resultados de acordo com os valores obtidos pelas métricas de precisão, revocação, F1-Score e kappa. Os melhores resultados foram obtidos pelo algoritmo *Isolation Forest*, com 0,78 para a revocação da classe positiva (maligna) e 0,80 para a revocação da classe negativa (benigna).

Palavras-chave: ameaças internas, fluxo de dados, aprendizado de máquina, ciência de dados, desbalanceamento

Abstract

An insider threat is anyone who has legitimate access to a particular organization's network and uses that access to harm that organization. Insider threats may act with or without intent, but when they have an intention, they usually also have some specific motivation. This motivation can vary, among other, includes personal disgruntlement, financial issues, and coercion. It is hard to face insider threats with traditional security solutions because those solutions are limited to the signature detection paradigm. To overcome this restriction, researchers have proposed using Machine Learning which can address Insider Threat issues more comprehensively. Some of them have used batch learning, and others have used stream learning. Batch approaches are simpler to implement, but the problem is how to apply them in the real world. That is because real insider threat scenarios have complex characteristics to address by batch learning. Although more complex, stream approaches are more comprehensive and feasible to implement. Some studies have also used unsupervised and supervised Machine Learning techniques, but obtaining labeled samples makes it hard to implement fully supervised solutions. This study proposes a framework that combines different data science techniques to address insider threat detection. Among them are using unsupervised and supervised machine learning, data stream analysis, and periodic retraining procedures. The algorithms used in the implementation were Isolation Forest, Elliptic Envelop, and Local Outlier Factor. This study evaluated the results according to the values obtained by the precision, recall, and F1-Score metrics. The best results were obtained by the Isolation Forest algorithm, with 0.78 for the positive class (malign) recall and 0.80 for the negative class (benign) recall.

Keywords: insider threat, data stream, machine learning, data science, unbalance

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

ADNN Autoencoder Deep Neural Network.

AM Aprendizado de Máquina.

AUC Area under the ROC Curve.

CERT Computer Emergency Response Team.

DL Deep Learning.

DNN Deep Neural Network.

DR Taxa de Detecção.

DT Decision Trees.

EV Elliptic Envelop.

FPR Taxa de Falsos Positivos.

HMM Hidden Markov Models.

IA Inteligência Artificial.

IDS Intrusion Detection Systems.

IPS Intrusion Prevention Systems.

ISOF Isolation Forest.

ITC Insider Threat Center.

ITD Insider Threat Dataset.

LOF Local Outlier Factor.

LR Logistic Regression.

MLP Multi Layer Perceptron.

NB Naive Bayes.

NBN Naive Bayesian Network.

NN Neural Network.

OOB Oversamplingbased Online Bagging.

PCA Principal Component Analisys.

RADISH Real-time Anomaly Detection in Streaming Heterogeneity.

RF Random Forest.

RNN Recurrent Neural Network.

SEI Software Engineering Institute.

SMOTE Synthetic Minority Oversampling Technique.

SMOTEBoost Synthetic Minority Over-sampling Technique Boost.

SOM Self Organizing Maps.

SVM Support Vector Machine.

TNR Taxa de Verdadeiros Negativos.

TPH Test of Page-Hinkley.

TPR Taxa de Verdadeiros Positivos.

UOB Undersampling-based Online Bagging.

Capítulo 1

Introdução

Este trabalho abordará o problema de detecção de ameaças internas utilizando técnicas de aprendizado de maquina em um contexto de fluxos de dados. Serão expostos os conceitos envolvidos no domínio do problema e apresentada uma proposta de abordagem do mesmo. Este Capítulo apresentará a contextualização do problema na Seção 1.1, a justificativa da pesquisa na Seção 1.2, a hipótese de pesquisa na Seção 1.3, os objetivos na Seção 1.4 e na Seção 1.5 será exposta a forma como este trabalho está estruturado.

1.1 Contextualização

Ameaça interna é o potencial que um usuário legítimo de uma rede possui para usar seu nível de acesso de forma prejudicial a organização detentora de tal rede [1]. Esse dano pode incluir atos maliciosos, complacentes ou não intencionais que afetam negativamente a integridade, a confidencialidade e a disponibilidade da organização. Um usuário interno é qualquer pessoa que possua ou tenha fornecido algum nível de acesso às informações de uma organização, incluindo aquelas sobre pessoal, instalações, equipamentos, redes e sistemas [1]. Os agentes maliciosos internos elevam o nível de complexidade de detecção por serem indivíduos que realizam atividades hostis, como roubo de informações e sabotagens, sem a necessidade de burlarem as proteções estabelecidas nos ambientes e sistemas [2]. Esses agentes são igualmente danosos quando comparados aos agentes maliciosos externos [3], além disso, geralmente não necessitam da mesma capacidade técnica para executarem suas atividades, conseguindo fazer com que seus ataques se pareçam com atividades normais [2].

As ameaças internas diferem das externas nas questões relacionadas ao acesso às informações, pois geralmente, os usuários internos que realizam atividades maliciosas possuem acesso legítimo aos sistemas, infraestrutura de redes e às informações. Eles também possuem conhecimento dos ativos críticos e da política de segurança da organização com a

qual têm relação [4]. Assim sendo, nota-se que a forma de detecção necessita ser melhor elaborada, pois atividades maliciosas realizadas por um usuário interno, causarão pouca ou nenhuma interferência que possa ser detectada por sistemas convencionais de segurança de infraestrutura de rede [5]. Além disso, como as atividades maliciosas realizadas por um usuário interno são proporcionalmente muito menores do que a soma das atividades não maliciosas realizadas pelo mesmo usuário e pelos demais, a percepção entre o que seria um comportamento malicioso e um comportamento normal dentro da rede, acaba sendo ofuscada devido ao desbalanceamento entre os tipos de comportamentos [5].

De acordo com o relatório *Cost of Insider Threat, Global Report* de 2022 [6], o tempo médio para contenção de um incidente causado por uma ameaça interna é de 85 dias e apenas 12% dos incidentes representados no relatório foram contidos em menos de 30 dias [6]. Nos últimos anos, a frequência da ocorrência de incidentes causados por ameaças internas tem aumentado. Em 2018, 53% das empresas pesquisadas haviam experienciado esse tipo de incidente. Em 2020, esse número subiu para 60% e em 2022, 67% das companhias pesquisadas relataram ter sofrido danos relacionados à ameaças internas [6]. Mesmo com o aumento das ocorrências, as ameaças internas permanecem menos comuns quando comparadas às externas, porém os danos causados por um agente interno podem ter efeitos tão graves quanto [7]. Isso se deve ao fato do elevado nível de acesso que o agente malicioso interno possui sobre informações e sistemas [7]. Em decorrência disso, agentes internos geralmente necessitam de menos esforço e conhecimento técnico para realizar uma ação maliciosa [7].

As soluções convencionais de segurança como, *firewalls*, *Intrusion Prevention Systems (IPS)*, *Intrusion Detection Systems (IDS)* e programas antivírus, possuem limitações para lidar com ameaças internas. Isso se deve, principalmente, por serem orientadas a prevenção de ações de agentes maliciosos externos e por implementarem um paradigma de detecção voltado para assinaturas e regras [8]. Sendo assim, novas soluções para detecção de ameaças baseadas em Inteligência Artificial (IA), têm sido propostas para responder às ameaças internas [9–13], uma vez que algoritmos de AM fornecem condições para criação de modelos capazes de aprender, se atualizar e se adaptar às alterações de cenários de ataque [14].

Dentre as propostas apresentadas, pode-se citar o uso de AM supervisionado [10, 11, 13, 15, 16] e AM não supervisionado [8, 13, 15–17], variando a abordagem entre AM em lote [5, 7, 8, 10, 11, 16, 18] e AM em fluxo de dados [9, 13, 17, 19]. Algumas abordagens holísticas, que envolvem domínios do conhecimento diferentes dos tecnológicos, também são sugeridas como estratégias para abordar a questão das ameaças internas, sempre inserindo o AM como um dos componentes da pesquisa. Como exemplo, podemos citar: (*i*) a análise de fatores psicológicos e comportamentais dos recursos humanos [8]; (*ii*) a

análise desses fatores e comportamentos com o auxílio das mídias sociais [16]; e (*iii*) a utilização de conjunto de dados compostos por informações psicossociais relacionadas aos indivíduos [20].

Dentre os trabalhos apresentados, aqueles que utilizaram AM não supervisionado [8, 16] e AM com abordagem em fluxo [9, 13, 17, 19] foram os que melhores se alinharam a proposta deste estudo. Isso porque, o desbalanceamento do conjunto de dados acarreta na abundância de amostras de umas das classes (benigna), favorecendo a aplicação do AM não supervisionado. Já a analise de fluxo se enquadra adequadamente ao problema de detecção em ambientes de rede, onde os dados são gerados de forma contínua e em grande volume, podendo apresentar mudanças de conceito e com a necessidade de análise e geração de alertas no momento em que os fluxo de dados é analisado. Dado o exposto, este estudo propõe um *framework* para um sistema de detecção de ameaças internas, possível de ser implementado no mundo real. O sistema é vocacionado para o uso de algoritmos de classificação de uma classe, processamento de dados em fluxo com retreinamento periódico do modelo visando adaptar o modelo as possíveis mudanças de conceito.

1.2 Justificativa

As operações diárias de empresas e órgãos governamentais dependem cada vez mais do emprego de recursos de tecnologias de informação que conectam seus ativos e consequentemente seus respectivos dados [21]. Com isso, questões relacionadas a proteção dessas infraestruturas contra ataques desferidos por agentes maliciosos crescem em importância [20, 21]. Apesar dos avanços nas pesquisas voltadas para a identificação de ameaças internas, a detecção desse tipo de ameaça tem se tornado cada vez mais complexo em decorrência do aumento na produção de dados pelas redes e a diversidade de tecnologias que as mesmas utilizam. Nos casos mais emblemáticos de agentes internos como o de Edward Snowden [1], são nítidas a gravidade das consequências de uma ação executada por uma ameaça interna e o quão vulneráveis as organizações estão à esse tipo de ameaça [20]. A causa dessa defasagem pode ser devido a um ou mais dos seguintes motivos [21]:

- As soluções existentes geralmente não são capazes de detectar os primeiros indícios de um agente malicioso interno, uma vez que a maioria dessas soluções não emite alertas até que comportamentos prejudiciais tenham ocorrido;
- A maioria das soluções considera apenas uma fonte de dados de auditoria individual, diminuindo os *insights* sobre as ameaças; e

[1] `https://www.bbc.com/news/world-us-canada-23123964`

- A análise de dados convencional depende muito do conhecimento do ambiente de rede para extrair atributos ou estabelecer regras, resultando em uma capacidade limitada contra ameaças em evolução.

Em geral, as ameaças internas se constituem de usuários autorizados que têm acesso legítimo a material sensível/confidencial e podem conhecer as vulnerabilidades dos sistemas e processos de negócios implantados [8,9]. Os ataques desferidos por agentes internos são mais difíceis de detectar em comparação com os de invasores externos, cujos os rastros são mais difíceis de esconder [4]. Além disso, tem havido uma tendência crescente de ameaças internas não intencionais nos últimos anos, que são aquelas nas quais o agente interno causa o dano de forma não intencional [6]. Todas essas questões reunidas tornam as ameaças internas uns dos modelos de ataque mais desafiadores para se lidar na prática. Portanto, a motivação por parte das organizações para lidar com essas ameaças é muito alta e tende a aumentar nos próximos anos [4].

Visando a realização de estudos para detecção desse tipo de ameaça, foi selecionado um conjunto de dados sintético *Insider Threat Dataset (ITD)* publicado pelo *Insider Threat Center* of *Computer Emergency Response Team (CERT) Division*, do *Software Engineering Institute (SEI)* da *Carnegie Mellon University*, capaz de fornecer insumos para a execução da pesquisa. Essa opção se deve à escassez de conjuntos com dados atuais, compostos por cenários reais de incidentes causados por ameaças internas [22].

O ITD é gerado por meio das informações, sobre ameaças internas, que vêm sendo reunidas pelo *Insider Threat Center* desde a sua criação [23]. Como pontos positivos do conjunto de dados, temos a variedade de cenários, os atributos disponibilizados, a documentação detalhada, a extensa utilização pela comunidade de pesquisadores do assunto [22] e o fato de ser oriundo de um dos mais completos bancos de informações de ataques internos [4]. Uma característica importante presente no conjunto, que deriva da própria natureza do problema [24], é o elevado desbalanceamento entre as atividades malignas e benignas presentes no conjunto.

Os mecanismos tradicionais de detecção de anomalias em redes não representam as melhores soluções para detecção de ameaças internas, devido a sua dependência de implementação de regras e assinaturas para a tarefa de detecção [8]. As detecções baseadas em regras além de se limitarem a descoberta de ameaças já conhecidas, são pouco flexíveis e exigem a manutenção de profissionais com conhecimento profundo do ambiente de rede para sua implementação de modo eficaz [8]. Por tais motivos, a proposta de utilização de AM para detecção de ameaças internas, tem sido a base para realização de diversos estudos nessa área [4,5,8,10,12,13,15,16,21,22,25,26]. No presente estudo, foi considerado como ideal a utilização de AM não supervisionado, uma vez que, dada a quantidade de amostras benignas é possível treinar um modelo apenas com esse tipo de amostra e com

isso realizar a identificação de amostras que destoem das treinadas. Além disso, o conjunto de dados utilizado na pesquisa fornece mecanismos para a identificação de amostras benignas necessárias para compor o conjunto de treino inicial.

Algumas soluções encontradas na literatura utilizam o aprendizado em lote para a detecção de ameaças internas [5,7,8,10,11,16,18]. Essa abordagem é baseada no aprendizado a partir de um conjunto de treinamento finito para desenvolver um modelo de decisão que é então utilizado para classificar instâncias do conjunto de teste também finito. O AM baseado em lotes de dados estáticos pode não ser adequado para desenvolver métodos de detecção de ameaças internas que sejam escaláveis e efetivos [27], já que o comportamento do tráfego de rede muda de forma contínua e novos casos de ameaças internas têm surgido [6]. Para lidar com essa limitação, as abordagens que tratam a detecção de ameaças internas como uma análise de fluxo de dados [9,12,13,19] parecem ser as mais promissoras para a detecção em ambientes reais, uma vez que os fluxos são gerados sequencialmente, podendo ou não, apresentar relação temporal e mudanças na sua distribuição ao longo do tempo [28].

Este trabalho usa aprendizado supervisionado e não supervisionado. O primeiro é utilizado na fase *offline* do *framework* proposto, visando fornecer os subsídios iniciais para a execução. O segundo é utilizado na fase *online* do *framework*, onde ocorre a predição das amostras e as tarefas de retreino do modelo. O *framework* foi desenvolvido dessa forma para se adaptar à algumas restrições impostas pelo problema, dentre elas, a escassez de amostras da classe maligna e como consequência o alto desbalanceamento do conjunto de dados. Nesse estudo, além da utilização de algoritmos de uma classe, também é utilizado um algoritmo de classificação multiclasse para fins de comparação de desempenho entre os dois tipos de algoritmos. Essa comparação visa verificar o real benefício do emprego de algoritmos de uma classe em detrimento dos multiclasse. O desempenho é analisado por meio da detecção das atividades maliciosas, detecção dos cenários maliciosos e detecção dos usuários maliciosos, presentes no conjunto de dados utilizados na pesquisa.

1.3 Hipótese

A utilização de técnicas de AM não supervisionadas em fluxos de dados desbalanceado com a aplicação de retreino periódico do modelo, permite predizer comportamentos maliciosos internos.

1.4 Objetivos

O objetivo dessa pesquisa é realizar a predição de comportamentos maliciosos de usuários internos de uma rede, utilizando técnicas de AM não supervisionadas com o retreino periódico, aplicadas em um fluxos de dados composto pela caracterização das atividades executadas por usuários da rede.

Esta pesquisa possui os seguintes objetivos secundários:

- Construir um conjunto de dados de pesquisa, por meio da extração de atributos do conjunto de dados original (ITD), de forma a melhor caracterizar o problema contextualizado pelo conjunto e tornar possível a sua análise.
- Propor um *framework* de detecção de ameaças internas que seja sensível aos óbices envolvidos na detecção de ameaças internas a partir da análise de fluxo de dados.

1.5 Estrutura do Documento

Este documento compreende mais cinco capítulos. O Capítulo 2 apresenta uma visão mais detalhada dos temas que circundam a detecção de ameaças internas. O Capítulo 3 expõe os trabalhos relacionados com uma visão do estado da arte usado na detecção dessas ameaças. O Capítulo 4 aborda os detalhes da *framework* proposto para de detecção de ameaças internas. O Capítulo 5 detalha o experimento realizado. O Capítulo 6 mostra os resultados acompanhados de suas respectivas análises. Finalmente, o Capítulo 7 apresenta as conclusões da pesquisa e as propostas para trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste Capítulo serão apresentados os assuntos que permeiam a presente pesquisa, de forma a caracterizá-los e enfatizar a sua importância para o estudo. Na Seção 2.1 serão apresentados os conceitos que envolvem as ameaças internas. A Seção 2.2 tratará de fluxos de dados, enquanto a Seção 2.3 tratará de fluxos de dados desbalanceados.

2.1 Ameaças Internas

Em Homoliak, I. *et al.* (2017) [4] são apresentadas algumas definições sobre o que caracteriza um agente interno ou usuário legítimo. Ao reuni-las, percebe-se que trata-se de uma pessoa interna ou externa à organização, que possui conhecimento que permita algum nível de acesso às informações e sistemas que compõem o ambiente organizacional. Os conceitos que definem um agente interno malicioso ou ameaça interna também são expostos. Em suma, pode-se afirmar que trata-se de uma pessoa ou uma ação que seja responsável pelo mau uso dos acessos de um usuário legítimo, de forma intencional ou não, ocasionando algum tipo de dano a uma determinada organização [4].

Segundo Saxena, A. *et al.* (2020) [29] agentes maliciosos internos podem ser classificados em três tipos:

- **Malicioso:** Uma pessoa que intencionalmente abusa dos seus privilégios de acesso para executar ações danosas à organização na qual possui tais privilégios, visando ganhos pessoais. Um exemplo seria a situação em que um funcionário insatisfeito deliberadamente vende informações as quais têm acesso.
- **Comprometido:** Aquele que foi alvo de alguma investida maliciosa, cujo o resultado seja o comprometimento de informações pessoais que forneçam acesso aos

sistemas internos a terceiros. Essa situação pode ser exemplificada com o caso de um funcionário vítima de engenharia social, chantagem ou *phishing*[1].

- **Descuidado:** São pessoas que não se preocupam ou desconhecem as normas de segurança vigentes em sua organização, desta forma agem cometendo erros que acarretam na exposição de dados e consequentemente em danos à organização. Como exemplo, pode ser citado a situação em que um funcionário ao anotar suas credenciais não atenta para a correta preservação dessa informação, que acaba sendo de conhecimento de terceiros.

Quanto ao tipo de atividade maliciosa interna, as atividades executadas por esses agentes podem ser classificadas em quatro grupos [30], a saber:

- **Sabotagem em Tecnologia da Informação:** Classificação onde, geralmente, a principal motivação é a vingança decorrente de algum tipo de insatisfação gerada no ambiente de trabalho.
- **Fraude:** É bastante comum que um conluio de pessoas executem as ações classificadas como fraude. Normalmente, a finalidade principal é a obtenção de ganhos pessoais em favor dos envolvidos.
- **Roubo de Propriedade Intelectual:** Nessa classificação são enquadradas ações de profissionais que, de alguma forma, ajudaram a desenvolver uma tecnologia ou conhecimento e que por esse motivo se acham merecedores de algo além do já recebido da organização na qual trabalham.
- **Espionagem:** Essa classificação é destinada às ações de espionagem em nível de segurança nacional, podendo ser patrocinada por estados ou empresas estrangeiras.

A separação entre os tipos de atividades maliciosas não significa que um agente deva ser enquadrado em um tipo apenas, uma vez que, mais de um tipo de ação pode ser executada por um mesmo agente [30].

Quanto à intenção, ameaças internas podem ser caracterizadas como atividades não intencionais e intencionais [30]. Na primeira, o ator malicioso inadvertidamente faz uso de forma descuidada dos recursos computacionais disponíveis no seu ambiente de trabalho, causando danos de forma não intencional [30]. Como exemplos de agentes maliciosos não intencionais, temos aqueles que, por descuido se tornam vítimas de ataques de engenharia social[2]. Na segunda, o agente malicioso interno realiza atividades maliciosas de forma

[1] `https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/phishing-spear-phishing`

[2] `https://www.enisa.europa.eu/topics/csirts-in-europe/glossary/what-is-social-engineering`

intencional, lançando mão dos seus privilégios de acesso ou da apropriação de privilégios de terceiros. Os atores maliciosos intencionais, não necessariamente são apenas funcionários contratados da empresa. Podem ser também pessoas que executam trabalho terceirizado, parceiros comerciais ou qualquer pessoa que participe do ambiente organizacional e que, em decorrência disso, possua algum tipo de acesso privilegiado [30].

Quanto as motivações, podemos citar três grupos principais [4]:

- **Motivação Financeira:** Onde o agente malicioso é recrutado por terceiros para executar as atividades maliciosas.
- **Motivação Política:** Na qual o agente malicioso possui convicções políticas diferentes daquelas possuídas por seus empregadores, a tal ponto, que o leva a executar atividades danosas ao seu local de trabalho.
- **Motivações Pessoais:** Decorrem de insatisfações em questões particulares relacionadas à empresa, podendo também ocorrer por meio de coação feita por um terceiro a um elemento interno à organização.

As consequências de incidentes maliciosos internos podem ser substanciais, incluindo perdas financeiras, impactos operacionais, danos à reputação e danos a indivíduos [22]. Esses danos vão desde algumas horas de trabalho perdidas até o encerramento de operações [30]. Quando a vítima se trata de empresa que presta algum tipo de serviço público, os danos podem ter repercussões que extrapolam a fronteira organizacional, causando interrupção de operações ou serviços essenciais para um setor específico ou criando sérios riscos para a segurança pública e nacional [30].

Os agentes maliciosos internos, para obterem êxito em suas ações, tendem a seguir uma série de etapas até alcançar seu objetivo final. Essas etapas são conhecidas na literatura como *Kill Chain*, um termo em inglês que remete a uma ideia de cadeia de eventos sequenciais críticos para se atingir um objetivo [21, 22, 29]. Nessa cadeia de eventos cada um dos passos são essenciais, de modo que, ao se conseguir impedir a evolução de quaisquer etapa da cadeia, toda a cadeia colapsa e o ataque não consegue ser bem sucedido [29]. Essa cadeia de eventos críticos possui as seguintes fases [29]:

- **Recrutamento/Conversão:** Fase em que um agente interno passa a ser malicioso por conta própria ou por ter sido aliciado.
- **Reconhecimento:** Momento no qual o agente malicioso interno passa a buscar por alvos compensatórios dentro da própria rede, bem como elabora formas de realizar suas ações, evadindo-se dos mecanismos de segurança e utilizando sua legitimidade de acesso.

- **Exploração:** Nessa fase o agente acessa as informações levantadas anteriormente e prepara as formas de exfiltração das informações que são desejadas.
- **Aquisição:** Com o acesso estabelecido, o agente malicioso realiza e extração dos dados de interesse.
- **Exfiltração:** De posse das informações desejadas, o agente as transporta do seu local de trabalho. Nesse contexto estão inseridos o envio de informações para serviços de nuvem, utilização de dispositivos removíveis de armazenamento e qualquer outra forma que resulte no transporte das informações para fora do ambiente organizacional ao qual as informações pertencem.

Uma importante característica dos usuários caracterizados como ameaças internas é que os mesmos não passam todo seu tempo de trabalho executando atividades nocivas [5]. Pelo contrário, passam apenas poucos períodos do dia realizando atividades indesejadas, aproveitando janelas de oportunidades e de forma furtiva [31]. Sendo assim, na maior parte do tempo, os agentes maliciosos internos executam suas atividades laborais normais e até instantes antes de se tornarem uma ameaça, não executam nenhuma atividade que possa ser considerada suspeita [24]. Somente a partir do disparo de algum gatilho (*e.g.*, motivado por condições de insatisfações de natureza diversa, descuidos relacionados à segurança da informação ou coações realizadas por terceiros) que esses usuários passam a assumir um estado malicioso [20]. As consequências óbvias desse comportamento culminam na complexidade que estão envolvidos os mecanismos empregados para a detecção, principalmente devido ao desbalanceamento do conjunto de dados gerado por suas atividades [31].

Uma vez que, os dados de usuários internos de uma rede, inclusive os de usuários maliciosos, são gerados e devem ser analisados de forma contínua [13], é possível que possam ser classificados como fluxo. Esses dados podem ser informações de utilização de estações de trabalho, como acesso a determinadas páginas, tráfego suspeito na rede, compartilhamento de arquivos ou a utilização de periféricos como dispositivos de armazenamento. Além disso, é importante considerar que esses eventos, quando maliciosos, são raros em relação aos eventos normais. Ou seja, existe um desbalanceamento intrínseco entre as atividades benignas e malignas no fluxo gerado por esses eventos. Unindo o fato do desbalanceamento e a classificação como fluxo de dados, definimos a forma como o problema será abordado por esta pesquisa: detecção de ameaças internas em fluxo de dados desbalanceados.

2.2 Fluxo de Dados

Fluxos são dados que, devido à dinamicidade e à interatividade dos ambientes do mundo real, são gerados de forma contínua, em grandes volumes, recebidos à altas taxas e com possibilidade de alteração de suas características durante a sua geração [32, 33]. Esses dados também são caracterizados por serem produzidos sequencialmente, tendendo a serem gerados infinitamente, podendo apresentar alterações na sua distribuição [34].

Por essas características, ao se trabalhar com fluxo de dados, deve-se levar em consideração o consumo de recursos computacionais e as formas de lidar com alterações que ocorrem no conjunto de dados [33, 34], uma vez que não é possível, por exemplo, o armazenamento completo dos dados na memória principal em tempo de processamento ou aguardar a recepção de todo conjunto de dados para processamento em lote [34]. Alguns exemplos de fontes de fluxos de dados são as redes de computadores, redes de sensores, aplicativos da web e monitoramento de transporte [33]. Nessas fontes, é provável que a distribuição subjacente que gera seus respectivos dados mude naturalmente ao longo do tempo. Essas mudanças são denominadas mudanças de conceito [19].

Formalmente, mudanças de conceito são alterações na probabilidade conjunta $P(x, y) = P(y|x) * P(x)$, onde y é o rótulo de classe do vetor de atributos x [35]. A fórmula compreende a probabilidade do atributo $P(x)$ e probabilidade condicional de rótulo de classe $P(y|x)$, de modo que a mudança na probabilidade conjunta é melhor compreendida através das mudanças em qualquer um desses dois componentes. Assim, existem três possibilidades para a geração de desvios de conceito [35]:

- **Alteração de Atributo:** $P(x)$ muda, mas $P(y|x)$ permanece o mesmo. Nesse caso, alguns vetores de atributos anteriormente infrequentes tornam-se mais frequentes e vice-versa, *e.g.*, durante algum período, um tipo específico de transação com cartão de crédito pode aparecer com mais frequência.
- **Mudança condicional:** $P(x)$ permanece o mesmo, mas $P(y|x)$ muda. Nesse caso, a relevância dos atributos e rótulos da classe mudam, *e.g.*, os atributos utilizados para identificação de uma fraude bancária podem mudar de tempos em tempos. Essas mudanças podem ser decorrentes de inserção de um novo serviço, por exemplo, o qual necessitaria de novos processos e novas tecnologias para torná-lo viável, com isso fazendo com que novas formas de tentativas de fraude também ocorram.
- **Mudança dupla:** Tanto $P(x)$ quanto $P(y|x)$ mudam, combinando a mudança de atributo com a mudança condicional.

As mudanças de conceito podem ser reais ou virtuais. As mudanças de conceito reais ocorrem quando existem uma alteração significativa na probabilidade condicional de rótulo

de classe $P(y|x)$. As mudanças de conceito virtuais ocorrem quando existem alterações na distribuição que geram os dados $P(x)$ porém sem afetar probabilidade condicional $P(y|x)$ [36].

Um ambiente de fluxo de dados têm premissas que são diferentes das apresentadas no cenário tradicional de aprendizado em lote. As mais significativas são as seguintes [37]: (i) processe um exemplo por vez e inspecione apenas uma vez; (ii) use uma quantidade limitada de memória; (iii) analise um período limitado de tempo; e (iv) esteja pronto para prever a qualquer momento.

Como propostas para atender a essas premissas, estudos têm apresentado técnicas de abordagem como o uso de mecanismos de esquecimento, algoritmos adaptativos e testes estatísticos para a detecção de mudanças de conceito [10, 38–40]. Mecanismos de esquecimento são formas de fazer com que os dados mais recentes tenham maior peso, ou seja, sejam mais relevantes para análise. Como exemplo, pode-se citar a aplicação de janelas deslizantes [34].

Os testes estatísticos se constituem na observação de alguma variável, onde alguma medida é adotada caso o valor para essa variável comece a degradar [34]. Um teste bastante conhecido é o *Test of Page-Hinkley (TPH)*, que é uma técnica de análise sequencial tipicamente usada para monitorar a detecção de alterações no processamento de sinais. Ela permite a detecção eficiente de mudanças no comportamento normal de um processo que é estabelecido por um modelo. O TPH é projetado para detectar uma mudança na média de um sinal gaussiano [41]. Este teste considera uma variável cumulativa, definida como a diferença acumulada entre os valores observados para uma determinada métrica e sua média até o momento atual. O valor mínimo dessa métrica também é calculado. Assim, o teste monitora a subtração entre a diferença acumulada da métrica e seu valor mínimo. Quando essa diferença é maior que um determinado limiar, uma mudança na distribuição é alarmada. O limiar dependerá da taxa admissível de falsos alarmes. Aumentar o limiar acarretará menos alarmes falsos, mas poderá ocasionar a perda ou atraso na detecção de mudanças [41].

Já os algoritmos adaptativos são uma forma de utilizar técnicas tradicionais de AM no cenário de fluxo de dados [34]. Esses algoritmos empregam mecanismos para atualização de modelos preditivos *on-line*, durante sua operação para lidar com as mudanças de conceito. [34]. Em [42], são apresentadas duas categorias de abordagens para lidar com a mudança de conceito, sendo classificadas em:

- Abordagens que adaptam o modelo em intervalos regulares, sem considerar se as mudanças realmente ocorreram.
- Abordagens que primeiro detectam mudanças de conceito e, em seguida, o modelo é adaptado a essas mudanças.

Como exemplos da primeira categoria, citam-se as abordagens ponderadas e as janelas de tempo de tamanho fixo. Abordagens ponderadas baseiam-se na simples ideia de que a importância de uma amostra deve diminuir com o tempo. Já no caso das janelas de tempo, o modelo é induzido apenas com as amostras incluídas na janela. A principal dificuldade enfrentada ao se utilizar janelas de tempo é saber dimensionar o tamanho mais apropriado da janela, pois uma janela pequena pode garantir uma adaptabilidade rápida em períodos que contenham mudanças de conceito, porém em períodos mais estáveis pode afetar o desempenho do algoritmo; enquanto isso, uma janela grande pode produzir resultados bons e regulares em fases estáveis, mas pode não reagir rapidamente às mudanças de conceito [42].

Na segunda categoria, alguns indicadores (*e.g.*, medidas de desempenho, propriedades dos dados, entre outros) são monitorados ao longo do tempo. Se durante o processo de monitoramento for detectada uma mudança considerada anormal nesses indicadores, em referência a um limiar pré-estabelecido, é considerada a ocorrência de uma mudança de conceito e ações para adaptar o modelo a essas mudanças podem ser tomadas. Quando uma janela de tempo de tamanho adaptável é usada, essas ações geralmente levam ao ajuste do tamanho da janela de acordo com a extensão do desvio de conceito. Como regra geral, se um desvio de conceito for detectado, o tamanho da janela diminui, caso contrário, o tamanho da janela aumenta [42].

O procedimento de avaliação de um algoritmo de aprendizado determina quais exemplos são usados para treinar o algoritmo e quais são usados para testar a saída do modelo [37]. O procedimento usado historicamente no aprendizado em lote depende parcialmente do tamanho dos dados. À medida que o tamanho dos dados aumenta, as limitações de tempo prático evitam procedimentos que repetem o treinamento por muitas vezes. É comumente aceito, com fontes de dados consideravelmente maiores, a necessidade de reduzir o número de repetições, para permitir que os experimentos sejam concluídos em um tempo razoável. Ao considerar o procedimento a ser usado na configuração do fluxo de dados, uma das preocupações é como determinar os valores para as métricas ao longo do tempo [37]. Uma forma de se realizar a avaliação de classificadores de fluxo envolve intercalar testes com treinamento, para fluxos em evolução é sugerido o procedimento de avaliação de denominado *prequencial* [33]. Tal procedimento tem objetivo de destacar o desempenho atual do classificador, em vez do desempenho geral. Como isso, permite a identificação das mudanças no fluxo com mais clareza, o que é especialmente importante para a detecção de desvios. Através do procedimento de avaliação *prequencial* é possível calcular periodicamente o valor de uma métrica selecionada e traçar seu valor em um gráfico, que representará o desempenho do classificador ao longo do tempo para a métrica escolhida [33].

2.3 Fluxos de Dados Desbalanceados

Um problema bastante comum a ser enfrentado no aprendizado de fluxos de dados diz respeito ao desbalanceamento das classes [35]. Na aprendizagem em lote, entre as técnicas possíveis para tratamento do desbalanceamento dos conjuntos de dados, está a aplicação da reamostragem. Essa técnica visa obter amostras suficientes para delimitar a classe rara a partir do incremento sintético de amostras da classe minoritária ou redução de amostras da classe majoritária [43].

Em se tratando de fluxos de dados, as amostras das classes raras podem demorar muito tempo para surgir ou se apresentar em intervalos de tempo e quantidades totalmente aleatórios, impossibilitando a obtenção de dados suficientes para sua identificação, a depender do período de tempo analisado [43]. Esse contexto influencia o desempenho de modelos de AM, principalmente enquanto as primeiras amostras estão sendo consumidas, pois as amostras positivas podem não estar presentes ou podem estar presentes em pouca quantidade. Sendo assim, os modelos voltados para fluxos inicialmente tendem a ter um desempenho inferior aos modelos que tratam dados em lote [19].

Alguns dados importantes para lidar com desbalanceamento de conjunto de dados, *e.g.* a proporção do desbalanceamento, só é conhecida para determinadas faixas de tempo [19]. Eventos da classe rara, que aparecem com pouca frequência em uma distribuição estática, podem estar disponíveis em quantidades ainda menores por unidade de tempo em fluxos de dados [43]. Além disso, esse dado pode sofrer alterações conforme o fluxo é gerado, sendo decorrente das mudanças de conceitos que, quando ocorrem em conjunto com o desbalanceamento, agravam sobremaneira o desafio de classificar fluxos de dados [43].

A reamostragem do conjunto de treinamento, forma mais comum de se lidar com o problema de desbalanceamento, é dividida em dois tipos: (i) a sobre-amostragem, a qual realiza o aumento de amostras da classe minoritária; e (ii) a sub-amostragem, que realiza a redução de amostras da classe majoritária [44]. Outra forma empregada para lidar com o desbalanceamento é a realização de ajustes a nível de algoritmo. Esses ajustes utilizam penalizações ou ponderação no aprendizado das classes, fazendo com que o algoritmo procure um equilíbrio ao realizar as classificações [10]. Ainda é possível o tratamento do desbalanceamento através da combinação de técnicas de reamostragem e ajustes a nível de algoritmos [10]. Nesse terceiro tipo podemos citar o algoritmo *Synthetic Minority Over-sampling Technique Boost (SMOTEBoost)* que combina técnicas de reamostragem e técnicas de *Boosting*. Os algoritmos que se baseiam em técnicas de *boosting*, fazem uso de uma combinação de classificadores ditos fracos, visando criar um classificador mas eficiente. A técnica de reamostragem denominada *Synthetic Minority Oversampling Technique (SMOTE)* visa a sobre-amostragem da classe minoritária [10].

Em Wang, S. *et al.* (2015) [44], temos um exemplo de aplicação da reamostragem combinada com a ponderação. Eles propõem versões ponderadas de dois algoritmos para tratamento de fluxos desbalanceados: o *Oversamplingbased Online Bagging (OOB)* e o *Undersampling-based Online Bagging (UOB)* [45]. Ambos originalmente implementam técnicas de reamostragem, porém, o OOB realiza a sobre-amostragem da classe minoritária, enquanto o UOB realiza a sub-amostragem da classe majoritária. Esses algoritmos escolhem qual amostra irá compor o conjunto de treino, a partir de uma variável que determina um valor de contaminação para cada classe e a montagem do conjunto de treino ocorre de forma dinâmica [44]. Na proposta apresentada pelo estudo, os algoritmos sofrem modificações para inserção de um peso para cada classe, visando influenciar o resultado da classificação. Segundo os resultados obtidos, as versões ponderadas obtiveram melhores resultados quando comparadas com as versões originais.

Gao, J. *et al.* (2008), propõem a aplicação de reamostragem, reduzindo a classe majoritária em conjunto com a propagação de amostras minoritárias. Essa propagação é feita repetindo-se amostras minoritárias antigas ao longo do fluxo. A técnica de propagação de amostras da classe minoritária está presente em outros estudos [43, 45, 46], sendo que a diferença entre elas encontram-se na forma com a qual são realizadas. No trabalho realizado por Chen, S. & He, H. (2009) [45] o fluxo é separado em partes (*chunks*) e apenas as amostras que estiverem presentes na última *chunk* observada pelo modelo serão propagadas. Lichtenwalter, R. & Chawla, N. (2010) [43], propõem a propagação das amostras da classe minoritária e das amostras da classe majoritária que foram classificadas de forma equivocada, objetivando a melhora na definição da fronteira entre as classes. Hoens, T. & Chawla, N. (2012) [46] realizaram uma melhoria em seu estudo anterior [47], propondo uma forma de propagar amostras da classe rara baseando-se na similaridade entre o contexto em que a amostra a ser propagada está inserida e o contexto atual do fluxo que está em análise, ao invés de propagar as amostras raras mais recentes.

2.4 Algoritmos Classificadores de Uma Classe

Aplicativos modernos para detecção de ameaças normalmente têm empregado algoritmos de uma classe para desempenhar suas funções. Esses aplicativos possuem a necessidade de analisar fluxos de dados, que são gerados a altas taxas, podendo conter mudanças de conceito, o que resulta em um ambiente desafiador de detecção. [48].

O aprendizado não supervisionado aplicado por meio de algoritmos de uma classe, assume que, para o treinamento do classificador, existem apenas amostras de uma classe [48]. Ao executar o classificador, uma nova amostra de classe diferente da treinada pode aparecer, com isso o classificador deve identificar essa amostra derivando um limite de

classificação que pode separar os objetos conhecidos dos desconhecidos [49]. Assim, o aprendizado não supervisionado pode ser aplicado, para classificação binária de fluxos de dados ou identificação de classes inovadoras, sem conhecer todos os outros tipos de amostras (classes) [49].

Em problemas de classificação multiclasse convencionais, dados de duas (ou mais) classes estão disponíveis e o limite de decisão é baseado no que foi inferido a partir da análise das amostras de cada classe. A maioria dos classificadores multiclasse assumem que as classes presentes em um conjunto de dados estejam mais ou menos equilibradas e com isso, não costumam funcionar bem quando quaisquer uma das classes é severamente subamostrada ou está completamente ausente [50]. Além disso, os algoritmos multiclasse quando aplicados de forma supervisionada necessitam de um conjunto de dados rotulado, o que não ocorre no aprendizado não supervisionado. Essa característica limita a aplicação de soluções baseadas em supervisão na vida real, pois a rotulação de dados em tempo real é muito complexa de ser obtida [51].

Dado o exposto, vemos que a aplicação de algoritmos de uma classe reúne condições para se adequar aos óbices apresentados pela detecção de ameaças internas. Essas barreiras são provenientes do desbalanceamento no contexto de fluxos de dados, com excessiva presença da classe benigna, a necessidade de prever assim que os dados estejam disponíveis e a ocorrência de mudança de conceitos.

2.4.1 Isolation Forest

O classificador ISOF é um algoritmo para detecção de anomalias e se diferencia de outros algoritmos por construir um conjunto de árvores isoladas para um determinado conjunto de dados, então anomalias são aquelas instâncias que possuem os menores comprimentos médios de caminhos entre as árvores isoladas [52]. O termo isolamento neste contexto significa separar uma instância do resto das instâncias e, como os valores discrepantes são poucos e diferentes, isso deveria implicar que eles são mais propensos ao isolamento. O algoritmo da floresta de isolamento funciona particionando recursivamente os dados até que todas as instâncias sejam isoladas; geralmente são necessárias mais partições para isolar instâncias normais, enquanto o oposto é verdadeiro para anomalias [8].

2.4.2 Elliptic Envelope

O EV modela os dados como uma distribuição Gaussiana multivariada. O modelo tenta detectar correlações entre os dados antes de distorcer o Gaussiano visando a formação de uma forma elipsoide. *Outliers* podem ter impacto significativo na montagem da curva,

o modelo é independente para descartar amostras para criar uma forma elipsoide com o menor volume desejável [8].

2.4.3 Local Outlier Factor

O algoritmo LOF define a probabilidade de uma amostra ser atípica com base na densidade de sua vizinhança local [8]. Cada amostra no conjunto de dados recebe uma pontuação de anomalia, que é chamada de *local outlier factor* (LOF) [53]. Ao atribuir a cada amostra uma pontuação baseada na densidade e distância de seus vizinhos, podemos comparar amostras e identificar amostras que possuem densidade substancialmente menor, tendo assim uma maior probabilidade de serem discrepantes [53].

Capítulo 3

Trabalhos Relacionados

Alguns estudos [5,7,8,10,11,16,18] usaram o aprendizado em lote para abordar a detecção de ameaças internas. O problema das implementações em lote é que elas não correspondem aos cenários reais. Embora alguns desses estudos apresentem bons resultados, suas implementações se distanciam da aplicação em cenários reais.

Os dados gerados por ameaças internas geralmente são contínuos, além disso, os padrões de ameaças evoluem com o tempo, sendo possível imaginá-los como um fluxo de comprimento ilimitado [13]. Sendo assim, modelos de classificação eficazes devem ser adaptativos e altamente eficientes, de forma a permitir seu emprego em um contexto que envolva grandes quantidades de dados em constante evolução [13].

Do exposto, é possível verificar que o problema em questão necessita de mecanismos para gerir o consumo de recursos computacionais e métodos para lidar com desbalanceamento entre as classes. Os fluxos de dados são por natureza gerados de forma ilimitada, porém os recursos computacionais que os manipulam possuem limitação. O desbalanceamento das classes é oriundo do fato dos dados sobre atividades maliciosas, realizadas por usuários internos, estarem envolvidos por uma quantidade extremamente superior de dados sobre de atividades benignas.

As abordagens empregadas para lidar com esses problemas vão desde o balanceamento do conjunto de treinamento [10], passando por atribuir um custo mais elevado para amostras classificadas de forma incorreta [13], até aplicar adaptações em algoritmos visando um menor consumo de recursos [25].

A depender do conjunto de treinamento, podem ser utilizadas abordagens supervisionadas e não supervisionadas. Essa decisão influencia diretamente na escolha dos algoritmos e na montagem dos modelos. Em Parveen, P. *et al.* (2013) [13] é feita uma comparação entre as abordagens que demonstraram uma superioridade para a abordagem supervisionada. O estudo também aponta que abordagens *ensemble-based* apresentam melhores resultados que as *single-model based*.

Neste capítulo são apresentados os trabalhos relacionados explorando algumas abordagens utilizadas para a detecção de ameaças internas. Este capítulo estará dividido em duas Seções, a Seção 3.1 apresentará os trabalhos voltados para a detecção baseada em lote e a Seção 3.2 apresentará os trabalhos que se basearam em fluxo.

3.1 Detecção de Ameaças Internas em Lote

Em Gavai, G. *et al.* (2015) [16] são aplicadas técnicas supervisionadas e não supervisionadas. Foi empregado aprendizado supervisionado através do algoritmo *Random Forest (RF)* e não supervisionado utilizando o algoritmo *Isolation Forest (ISOF)*, sendo que ambos foram aplicados em um conjunto de dados do mundo real, com eventos de ameaças internas introduzidos artificialmente. Nesse estudo foi obtido AUC de 0,77 para a abordagem não supervisionada e acurácia de 73,4% para a abordagem supervisionada. Os autores não abordaram o tratamento para conjunto de dados desbalanceados, além disso a métrica acurácia não é a mais ideal para problemas em que o desbalanceamento de conjunto está fortemente presente. O acerto demasiado na identificação da classe majoritária pode levar a valores elevados da métrica acurácia mesmo que nenhuma amostra da classe minoritária tenha sido detectada. No tocante a detecção de ameaças internas é fundamental a identificação da classe minoritária.

Em Kim, D. *et al.* (2018) [18] foram utilizadas propriedades das cadeias de Markov para listar o comportamento dos usuários ao longo do tempo e para classificar a qual estado eles pertencem, maligno ou benigno. Conjuntos de dados sequenciais foram gerados de acordo com a influência das n ocorrências dos atributos Markov (estados criados de acordo com as atividades executadas pelos usuários) e classificados por algoritmos de AM. O estudo faz uma seleção aleatória de usuários presentes no conjunto de dados ITD para realizar seus experimentos, utilizando-o de forma parcial. Os algoritmos utilizados foram os seguintes: *Naive Bayes (NB)*; *Support Vector Machine (SVM)*; *Multilayer Perceptron Multi Layer Perceptron (MLP)* e *RF*. Todos obtiveram resultados muito próximos para as métricas acurácia, *Taxa de Falsos Positivos (FPR)*, revocação e *F1-score*, com valores variando de 0,96 a 0,97. A única exceção foi o algoritmo NB que obteve resultados inferiores. Além do aprendizado em lote, este estudo se diferencia do presente no tocante à metodologia e por utilizar parcialmente o conjunto de dados.

Le, Duc & Zincir-Heywood, A. (2018) [15] utiliza as abordagens supervisionadas e não supervisionadas, nas quais foram aplicados os algoritmos *Self Organizing Maps (SOM)* e *Hidden Markov Models (HMM)*, ambos não supervisionados, além do algoritmo supervisionado C4.5 *Decision Trees (DT)*, para fins de comparação. Neste estudo foi utilizada a versão 4.2 do conjunto de dados disponibilizado pelo CERT *Insider Threat Center*, cujas

atividades foram sintetizadas por dia e por semana. As métricas utilizadas foram *Taxa de Detecção (DR)*, *FPR* e acurácia, as quais obtiveram, respectivamente, os seguintes resultados 82,51%, 0,01% e 99,93%. Esses foram os melhores resultados apresentados, sendo produzidos através de aprendizado supervisionado e utilizando os dados sumarizados por dia. Para alcançar os resultados foi selecionado um extrato do conjunto de dados. Todos os dados referentes as vinte primeiras semanas de atividades foram removidos, por conterem apenas atividades não maliciosas.

Uma análise baseada nos cenários disponibilizados pelo conjunto de dados ITD foi feita em Chattopadhyay, P. *et al.* (2018) [10]. Nesse estudo, os autores utilizam uma abordagem baseada em janela deslizante empregada sobre um novo conjunto de dados, composto por um conjunto de atributos estatísticos, os quais foram extraídos do conjunto de dados original. Foram estabelecidos atributos específicos para cada um dos três cenários de ameaça interna implementado no conjunto de dados utilizado no estudo. Dessa forma, a detecção foi trabalhada especificamente para cada cenário. Nesse estudo foi utilizado o balanceamento dos dados de treino com redução da classe majoritária e implementada uma *Autoencoder Deep Neural Network (ADNN)*. Os melhores resultados foram obtidos na detecção do cenário de número 2, que diz respeito a usuário que rouba informações organizacionais. Para esse cenário foram alcançados os valores de 92,88% para a métrica precisão, 99,48% para a revocação e 96,06% para o *F1-score*. Foi aplicada uma janela deslizante referente a 40 dias para treino e o conjunto foi balanceado utilizando a técnica SMOTE, de tal forma que, a classe minoritária (maliciosa) possuísse mais amostras nesse conjunto que a classe majoritária (benigna), na proporção de 2:1. Apesar de utilizar alguns conceitos aplicados a fluxo de dados como janelas deslizantes esse estudo foi considerado como lote por alterar o balanceamento do conjunto de dados como um todo antes de realizar os experimentos ou invés de tratar o desbalanceamento em tempo de execução.

Em Johannessen, B. S. (2018) [8] foi utilizado aprendizado não supervisionado através da implementação dos seguintes algoritmos para detecção de anomalias: ISOF, *Elliptic Envelop (EV)* e *Local Outlier Factor (LOF)*. Foi utilizada a abordagem em lote e o desbalanceamento foi tratado por meio da configuração dos hiper-parâmetros relativos a taxa de desbalanceamento dos algoritmos utilizados, exceto para o LOF. O conjunto de dados foi organizado de modo que, nos 10% do conjunto utilizado para treino, existissem pelo menos cinco amostras consideradas maliciosas. Nesse estudo foi utilizada a versão 4.2 do conjunto de dados disponibilizado pelo CERT. O estudo sugere a utilização de informações das mais diversas fontes de dados organizacionais, como acessos físicos à locais e análise de componentes psicossociais descritos pelo método de análise de traços de personalidade *BIG FIVE*. Esse método busca descrever a personalidade de um indivíduo através da investigação dos seguintes traços: abertura a experiências (*Openness*), conscienciosidade

ou escrupulosidade (*Conscientiousness*), extroversão (*Extraversion*), agradabilidade ou simpatia (*Agreeableness*) e neuroticismo (*Neuroticism*). Por se tratar de um estudo que explora várias nuances da detecção de ameaças internas, seus resultados relacionados ao AM não foram profundamente explorados. Apesar disso, trata-se de um estudo bastante informativo e abrangente, fornecendo formas diversificadas de se visualizar o problema, com o foco na construção de uma solução.

Em Kim, J. *et al.* (2019) [7] foi proposta uma estrutura de detecção de ameaças internas baseada na modelagem do comportamento do usuário. Foram criados três conjuntos de dados distintos, respectivamente baseados em: atividades diárias dos usuários em suas estações de trabalho; redes de troca de *e-mails* entre os usuários; e conteúdo dos *e-mails* trocados entre os usuários. Nesse estudo foi utilizada a versão 6.2 do conjunto de dados disponibilizado pelo CERT e foram avaliados os algoritmos *Gauss, Parzen, kMeans* e *Principal Component Analisys (PCA)*, empregados como classificadores semi e não supervisionados. A análise do conjunto de dados foi feita em lote. Foram utilizados para o conjunto de treino 90% dos dados normais, selecionados randomicamente. O conjunto de teste foi composto pelos 10% de dados normais restantes mais todas as amostras anormais. Para cada conjunto de dados foi criado um *ranking* de anomalia, no qual as atividades foram classificadas. O estudo apontou que utilizando o *framework* proposto, foi possível detectar 53.67% das atividades maliciosas presentes no top 1% do *ranking* de atividades maliciosas e 90% das atividades maliciosas presentes no top 30%.

Uma metodologia para processamento dos conjuntos de dados organizacionais foi apresentada em Hall, A. *et al.* (2019) [11]. O estudo tratou o problema com a abordagem supervisionada e agregou o conjunto de dados por dia e por usuário construindo um modelo para a identificação cada cenário malicioso. Somente foram considerados para montagem do conjunto de treino, os dados de usuários com potencial para execução de alguns dos cenários maliciosos e os dados provenientes de um único mês. Além disso, as amostras negativas foram retiradas até que o desbalanceamento atingisse uma taxa de 15:1 (subamostragem). Com essas medidas, os autores visaram reduzir a desproporcionalidade apresentada pelo ITD entre as atividades maliciosas e não maliciosas e entre usuários maliciosos e não maliciosos. Para obtenção dos resultados, o estudo utilizou os classificadores *Neural Network (NN)*, *Naive Bayesian Network (NBN)*, *Support Vector Machine (SVM)*, *RF*, *Decision Trees (DT)*, e *Logistic Regression (LR)* para compor modelos únicos e comparou os resultados com a melhor versão *Boosting* de tais classificadores (no caso do estudo a versão Boosting do NBN foi utilizada na comparação). O estudo concluiu que o modelo que utilizou a composição de classificadores foi superior, obtendo uma AUC de 0,988, contra uma AUC de 0,980 obtida pela versão *Boosting* do NBN.

Dado o exposto, a partir da informações obtidas dos trabalhos em lote foram reunidas

as contribuições consideradas relevantes para apresente pesquisa. Johannessen (2018) [8] maximizou o uso do conjunto de dados ao se utilizar ao máximo as várias fontes de informação fornecidas pelo conjunto, para extração de atributos. O estudo de Chattopadhyay *et al.* (2018) [10] foi a referência para extração de alguns dos atributos utilizados neste trabalho. Kim *et al.* (2019) [7] influenciou na aplicação de aprendizado não supervisionado por meio de algoritmos de classificação de uma classe. Este tipo de algoritmo é capaz de realizar seu treinamento com apenas uma classe e detectar a presença de anomalias. Assim, pode-se aproveitar a alta disponibilidade de amostras da classe benigna no conjunto de dados, que será utilizado neste estudo, para realizar o treinamento do algoritmos.

3.2 Detecção de Ameaças Internas em Fluxo

O estudo realizado em Senator, T. *et al.* (2013) [12] utilizou dados privados de redes reais empresariais, para pesquisar a detecção de ameaças internas. Para coletar esses dados, programas especializados nesse tipo de atividade foram instalados em estações de trabalho das empresas participantes do estudo. Já os dados de cenários maliciosos foram inseridos de forma sintética por um time de especialistas em ações cibernéticas de caráter ofensivo. O estudo apresenta um sistema de detecção de ameaças internas com o uso de algoritmos para detecção de anomalias baseados em: cenários suspeitos de comportamento malicioso interno; indicadores de atividades incomuns; padrões estatísticos de alta dimensão; sequências temporais; e grafos de evolução. Apesar de citar como seu melhor resultado a AUC de 0,979, os autores citam como um trabalho futuro, estudar a factibilidade da aplicação de seu método em um cenário real. Diferentemente do presente estudo, o trabalho de Senator, T. *et al.* (2013) [12] é focado em descoberta de cenários específicos e não cita tratamento para conjuntos de dados desbalanceados.

Um sistema criado a partir da implementação de um modelo baseado em *Deep Learning (DL)*, não supervisionado e em tempo real foi utilizado em Tuor, A. *et al.* (2017) [17]. Como o comportamento de ameaça interna varia amplamente, o estudo se preocupou em modelar o reconhecimento das características de cada usuário na rede, utilizando *Deep Neural Network (DNN)* e *Recurrent Neural Network (RNN)*, para desta forma determinar se o comportamento é normal ou anômalo. Os resultados foram comparados com os obtidos pelos algoritmos *ISOF*, *SVM* e *PCA*, demonstrando superioridade para as redes neurais empregadas na comparação. O estudo focou na criação e análise de perfis para cada usuário, baseado em suas funções dentro da empresa. Os modelos de rede neurais aprendem os comportamentos normais para cada usuário baseado em um vetor de atributos referentes as atividades de cada usuário e realizam a previsão para um vetor futuro.

Quando o vetor verdadeiro chega, é comparado com o vetor predito se o erro na predição for considerado alto o comportamento que gerou o vetor real é considerado anômalo.

O trabalho apresentado em Bose, B. *et al.* (2017) [9], propõe um sistema denominado *Real-time Anomaly Detection in Streaming Heterogeneity (RADISH)*, o qual possui dois componentes principais. O primeiro, RADISH-L é o responsável por aprender os comportamentos definindo novos modelos e limiares para comportamentos considerados normais, a partir da análise dos fluxos de dados. O segundo, RADISH-A é o sistema responsável por comparar os novos fluxos com os limiares existentes. Para isso, utiliza-se os modelos criados no módulo L, aplicando-os aos novos fluxos e gerando alertas para aqueles que ultrapassarem o limiar de normalidade previamente definido. Como resultados o estudo apresenta a revocação em torno de 0,5 e precisão de 0,08, que foram obtidos sem que a questão do desbalanceamento do conjunto de dados fosse abordada. O estudo se autojustifica a partir dos cálculos relacionados aos prejuízos sofridos pelas empresas vítimas de ameaças internas e a quantidade de falsos positivos. O estudo sugere que poderia ser mais vantajoso, financeiramente, verificar uma quantidade substancial de alertas falsos do que dispor de recursos para responder a um incidente não alarmado. Assim como no presente estudo, o problema de ameaças internas foi tratado através da análise de fluxos de dados e também foi utilizado o conjunto de dados fornecido pelo CERT.

Um *framework* para reamostragem de conjuntos de dados desbalanceados foi proposto em Zhang, H. *et al.* (2019) [19]. Nesse estudo foram empregadas técnicas para reamostragem em conjuntos oriundos de fluxos de dados, visando reduzir o impacto do desbalanceamento da classe e das mudanças de conceitos. Foram utilizadas oito bases de dados sintéticas, geradas através do *software* MOA, e quatro bases de dados reais, a PAKDD sobre fraudes bancárias, a *Give Me Some Credit* (GMSC) sobre riscos relacionados à empréstimos, a *Forest Covertype* (Covtype) que é relacionada a cobertura de florestas e a Poker23 que diz respeito a jogo de cartas. O estudo compara seu próprio *framework*, o *Resample-based Ensemble Framework for Drifting Imbalanced Stream* (RE-DI) com outros existentes a saber: *Over- sampling Online Bagging* (OOB); *Undersampling Online Bagging* (UOB); Adaptive Random Forests (ARF); e Lever- aging Bagging (LB). A métricas utilizadas para a avaliação foram a *Prequential AUC* e acurácia, o *framework* proposto obteve os melhores resultados para a maioria dos conjuntos em que foi aplicado. Entre as várias técnicas que foram empregadas, cita-se a seleção de amostras com o objetivo de inseri-las conjunto de treino. Essa técnica será reproduzida no presente estudo para retreinamento do modelo, aplicada ao contexto de detecção de ameaças internas.

Assim como nos trabalhos em lote, algumas técnicas mencionadas nos estudos que abordaram o problema utilizando fluxo de dados, foram selecionadas para compor a presente pesquisa. O estudo realizado por Krawczyk & Wozniak (2015) [49], trouxe como con-

tribuição técnicas para adaptar o modelo de forma incremental às mudanças de conceito no fluxo de dados. Entre elas, podemos citar procedimentos para retreinar o algoritmo, para construir um conjunto de dados de retreinamento e para descarte de amostras.

Bose *et al.* [9] e Senator *et al.* [12] abordam o problema de ameaças internas por meio de uma análise de fluxo de dados. O primeiro utilizou o ITD, porém na versão 2, e o segundo lançou mão de um conjunto de dados privado. No entanto, eles não abordaram a questão do desequilíbrio do conjunto de dados em seus estudos. Bose *et al.* [9] atualizou seus modelos treinando-os com um conjunto de treinamento composto de amostras benignas recentes e um buffer que armazenava amostras malignas. No presente estudo é empregado um procedimento similar ao citado anteriormente. Parveen *et al.* [13] concluiu que seu experimento supervisionado, apesar de ter melhores resultados, não ocorreria no mundo real, apresentando um indício de que aplicações baseadas em fluxos de dados seriam mais adequadas nessas situações. Finalmente, Zhang *et al.* [19] usou novas instâncias que chegaram para atualizar dinamicamente seus modelos conforme proposto por este estudo. A seção a seguir apresentará como este estudo usou as técnicas citadas para construir um *framework* para detecção de ameaças internas.

Como apresentado neste Capítulo, notamos que muitos trabalhos não citam o tratamento para o desbalanceamento da base de dados. Os estudos que relatam algum tratamento, utilizam como abordagem o aprendizado supervisionado em lote, o que não seria o mais adequado em se tratando de ambientes reais. Isso ocorre pela dificuldade de se obter um mecanismo para rotulação de amostras em tempo real. Os estudos que trataram o assunto com a abordagem em fluxo de dados ou não citaram métodos para evolução do aprendizado para lidar com as mudanças de conceito ou não trataram o desbalanceamento da base, diferindo da proposta apresentada por este trabalho.

Capítulo 4

Método Proposto

Neste Capítulo será apresentado o *framework* proposto. Na primeira fase do *framework* foi utilizada uma abordagem *offline* em lote de forma supervisionada, para definir os parâmetros iniciais a serem utilizados pelos algoritmos durante os experimentos. Para isso, este estudo utilizou o algoritmo *Grid Search* sobre um conjunto de treinamento rotulado. Em um segundo momento partiu-se para a adoção da abordagem *online*, para retreinamento e classificação das amostras. Os dados relativos às atividades internas devem ser analisados no momento de sua geração, como no caso do fluxo de dados, visando produzir alarmes oportunos. A análise de dados gerados por usuários internos em lote ou fragmentos pode levar a notificações atrasadas. Os fluxos de dados possuem algumas peculiaridades como os desvios de conceito que necessitam de tratamento visando manter a efetividade do modelo. Todas essas questões foram endereçadas pela proposta que será apresentada a seguir.

4.1 *Insider Threat Detection Framework*

A estrutura proposta vem da combinação de abordagens usadas em estudos anteriores [7,9,10,12,19,49] e visa permitir a implantação do sistema de detecção em um ambiente do mundo real. Esse ambiente exige o manejo das premissas impostas pelo problema. Os mais significativos são os seguintes [37]: (i) processar um exemplo por vez e inspecionar apenas uma vez; (ii) usar uma quantidade limitada de memória; (iii) analisar um período limitado; e (iv) estar pronto para prever a qualquer momento. Além dessas premissas, existem alguns obstáculos também impostos pelo problema de detecção de ameaças internas, como o desequilíbrio extremo da distribuição de classes [31], tendo a classe maligna como menor, a necessidade de identificar com precisão essa classe, a possibilidade da ocorrência de desvio de conceito e a semelhança entre os dados gerados por *insiders* malignos e benignos [13].

O presente estudo estabeleceu uma abordagem de análise híbrida para satisfazer essas premissas, com uma fase *offline* em lote para o primeiro treinamento do algoritmo e uma fase *online* em fluxo para retreinamento e classificação das amostras. Para executar suas tarefas de retreinamento, o modelo deve ser capaz de selecionar novas amostras, implementar um mecanismo de esquecimento para descartar as antigas e adaptar os parâmetros dos algoritmos. A figura 4.1 ilustra a visão geral do *framework*.

O módulo de pré-processamento representa a primeira etapa do sistema que trata da extração de atributos das fontes de dados. Em implementações para detecção de ameaças internas em redes reais, cada fonte de dados deve ter seu *parser*, ou seja, um trecho de código capaz de estruturar as informações contidas em sua linha de *log*. Neste trabalho, o tratamento dos arquivos de *log* do ITD com a finalidade de compor o conjunto de dados utilizado na pesquisa abstraiu a etapa de extração de atributos.

O módulo de avaliação classifica sequencialmente as amostras conforme elas chegam e realiza o retreinamento periódico do classificador. Nesta etapa, o modelo escolhe quais amostras farão parte do conjunto de treinamento. A cada nova inserção de amostra de treinamento, o modelo remove a mais antiga. Para manter o modelo resiliente às mudanças de conceito que possam ocorrer durante a geração de dados, este estudo realiza tarefas periódicas de retreinamento. As amostras incluídas no retreinamento usam os rótulos preditos pelo modelo após passarem pelo algoritmo de classificação.

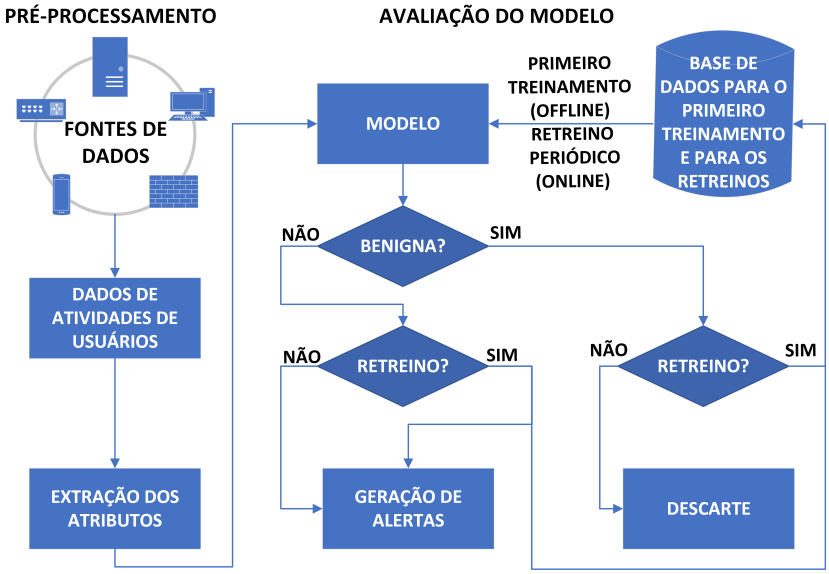

Figura 4.1: Framework.

O primeiro treinamento em lote do modelo ocorre *offline* com amostras benignas conhecidas e com uma contaminação inserida deliberadamente. Este trabalho inseriu a contaminação porque é improvável que as amostras provenientes de comunicações de rede

real não contenham nenhuma atividade maligna. Essa característica é uma consequência da semelhança de ações realizadas por agentes internos malignos e benignos [13]. O objetivo da fase *offline* é estabelecer os hiperparâmetros iniciais dos algoritmos que o modelo utilizará. O algoritmo Grid Search foi aplicado ao conjunto de dados de treinamento inicial para selecionar a melhor combinação dentre um conjunto de hiperparâmentros estabelecido para cada algoritmo.

Na fase *online*, o modelo classifica cada amostra de fluxo assim que elas são recebidas. Para realizar a avaliação, o modelo usa algoritmos de classificação de uma classe. O modelo salva as predições e suas respectivas pontuações de anomalia, para dar suporte ao processo de retreino. As pontuações de anomalia ou *anomaly scores* são fornecidas pelo algoritmo durante a avaliação de uma amostra. Ainda na fase *online* ocorre o retreinamento. O modelo implementa um mecanismo de retreinamento visando adaptar, constantemente, o modelo aos possíveis desvio de conceito. O modelo compila os conjuntos de retreinamento usando o primeiro conjunto de treinamento como ponto de partida. O conjunto de retreinamento consiste em substituir amostras antigas do primeiro conjunto de treinamento por novas amostras oriundas do fluxo de dados analisado pelo modelo. Durante sua execução, o modelo avalia as pontuações de anomalia e rótulos das amostras e escolhe aqueles que farão parte do conjunto de retreinamento.

Após os algoritmos classificarem uma amostra e o modelo avaliar sua pertinência para compor o conjunto de treinamento, ela é descartada ou reutilizada. O modelo descartará amostras em duas situações: (i) quando forem classificadas como benignas pelos algoritmos, e o modelo não seleciona-las para compor o conjunto de retreinamento, ou (ii) toda vez que elas forem substituídas por novas amostras no conjunto de retreinamento. O modelo reutilizará amostras sempre que forem consideradas benignas ou malignas e o modelo seleciona-las para compor o conjunto de retreinamento. Sempre que o algoritmo classificar uma amostra como maligna, o modelo a utilizará para gerar um alerta.

4.2 Avaliação do Modelo

As próximas Seções apresentarão detalhes sobre as fases *offline* e *online* do *framework* representado na Figura 4.2. Esta figura mostra um fluxograma da avaliação do modelo onde a fase *offline* fornece os hiperparâmetros iniciais dos algoritmos e o conjunto de dados de treinamento inicial como entradas para a fase *online*. A fase *online* depende dos resultados da fase *offline* para iniciar sua execução. A fase *offline* é executada uma vez e a *online* é executada indefinidamente.

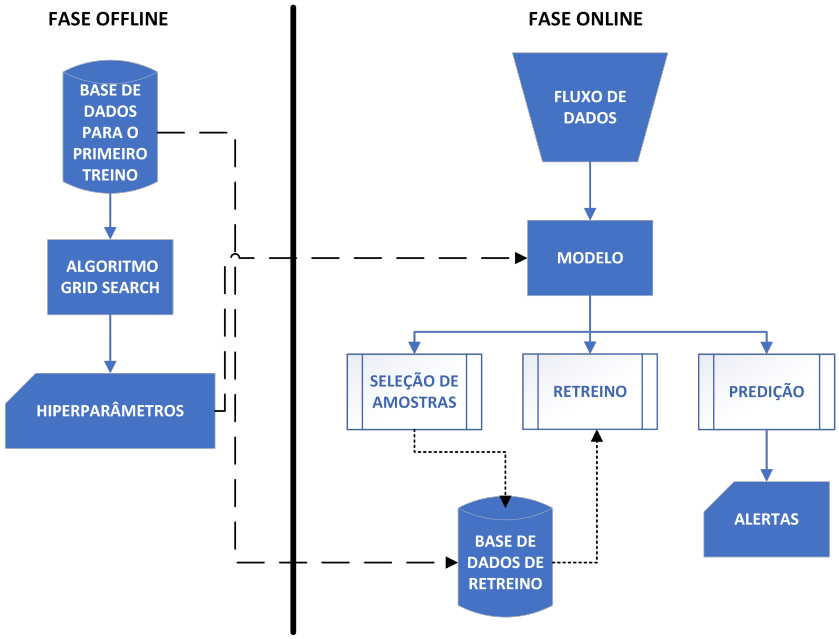

Figura 4.2: Avaliação do modelo.

4.2.1 Fase *Offline*

O objetivo principal da fase *offline* é fornecer os valores dos hiperparâmetros iniciais a serem utilizados pelos algoritmos no modelo. Para isso, o presente estudo realiza uma implementação do algoritmo Grid Search sobre um conjunto de treinamento rotulado. O algoritmo Grid busca a melhor combinação de um subconjunto de hiperparâmetros [54]. Cada algoritmo tem seu próprio subconjunto de hiperparâmetros. O Grid Search conta com a pontuação obtida por uma métrica para escolher o melhor conjunto de hiperparâmetros. Neste trabalho, a revocação média entre as classes benigna e maligna foi escolhida com a métrica de referência. O algoritmo calcula os valores das métricas para cada rótulo e a média aritmética entre eles. Quanto maior a média entre os valores da revocação das classes maligna e benigna, maior o valor individual de cada métrica de revocação. Usamos os rótulos das amostras de contaminação para realizar os testes do *RF* com o Grid Search, pois RF é um algoritmo multiclasse.

4.2.2 Fase *Online*

Após definir os hiperparâmetros iniciais dos algoritmos e realizar o treinamento dos mesmos, inicia-se a fase *online*. Esta fase executa a seleção das amostras para a construção do conjunto de retreinamento, as predições das amostras e os procedimentos de retreinamento do modelo.

O modelo usa o conjunto de treinamento inicial para construir o de retreinamento. Antes de fazer isso, ele separa o conjunto inicial em dois conjuntos de acordo com os rótulos de suas amostras, benignas e malignas (contaminação). O modelo constrói uma distribuição de pontuação de amostra de ambos os conjuntos para identificar em qual quartil as amostras malignas e benignas provavelmente estão. Depois disso, dois limiares são definidos pelo modelo, o superior e o inferior. Quando o algoritmo começa a receber e classificar as amostras, o modelo compara cada pontuação obtida por cada amostra com os limites superior e inferior da seguinte maneira:

1. Se o algoritmo prediz uma amostra como benigna e sua pontuação é maior que o limite superior, ela é inserida no conjunto de treinamento e o modelo descarta a amostra mais antiga no conjunto de treinamento.

2. Se o algoritmo prediz uma amostra como maligna e sua pontuação é maior que o limite inferior, ela é inserida no conjunto de treinamento e o modelo descarta a amostra mais antiga do conjunto de treinamento.

3. Em qualquer outro caso, o modelo não reaproveita a amostra classificada.

O modelo irá treinar novamente o algoritmo usando o conjunto resultante da operação acima. A ideia é pegar amostras provavelmente benignas e as malignas com pontuação alta, para compor a contaminação no conjunto de retreinamento. Essa escolha visa obter um conjunto que possui contaminação, porém com um relativo equilíbrio na composição das amostras. Desta forma, principalmente no tocante a contaminação, não são inseridas amostras malignas muito discrepantes das benignas, o que acarretaria em inserção de ruídos muito acentuados para que serem tratados pelos algoritmos de uma classe. Ambos os limites cintados anteriormente sempre serão positivos devido ao alto desequilíbrio do conjunto de dados em favor da classe benigna.

Para retreinar, o modelo verifica se o intervalo de retreinamento foi atingido. O intervalo de retreinamento pode ser definido pelo tempo ou pelo número de amostras analisadas. Este trabalho calcula um intervalo de tempo usando o número de amostras como referência, 1.200 amostras equivalem a aproximadamente 01 dia no ITD. Sempre que as verificações do intervalo de retreinamento retornam positivamente, o modelo redefine os limites superior e inferior e retreina o algoritmo usando o novo conjunto de retreinamento. O modelo redefine os limites usando o mesmo procedimento explicado no segundo parágrafo desta Seção. A contaminação é o único hiperparâmetro do algoritmo que pode mudar no processo de retreinamento. Isso ocorre de acordo com a distribuição dos escores obtidos pelas amostras do conjunto de treinamento/retreinamento. Usando a distribuição de pontuação do conjunto de treinamento/retreinamento da amostra maligna (contaminação), a seguinte comparação pode é feita: Se o valor do percentil 75 da distribuição

da amostra maligna do conjunto de treinamento for positivo, o valor do hiperparâmetro de contaminação do algoritmo permanece o igual ao previsto pelo algoritmo Grid Search. Caso contrário, o hiperparâmetro é definido como automático ou com o valor padrão, dependendo do classificador.

Vale ressaltar que, apesar do modelo selecionar amostras benignas e malignas com seus respectivos rótulos e pontuações para compor o conjunto de retreinamento, tal conjunto é entregue aos algoritmos para treinar sem rótulos, visto que este trabalho utilizou classificadores de uma classe. Apesar de não utilizar o rótulo para treinamento/retreinamento, o modelo salva os rótulos para construir a distribuição de escores para classes benignas e malignas separadamente. Somente nos testes com o algoritmo RF foi necessário o uso de rótulos.

Capítulo 5

Metodologia

O Capítulo 5 apresentará a metodologia seguida para executar o experimento. Será apresentado o conjunto de dados, a forma de extração dos atributos e a execução do experimento com base no *framework* apresentado na Seção 4.1.

5.1 O conjunto de dados

O conjunto de dados usado neste estudo foi o *Insider Threat Dataset (ITD)* [23], publicado pelo *Insider Threat Center (ITC)* da Carnegie Mellon University [1]. Esta pesquisa selecionou o conjunto de dados por meio da análise da bibliografia pesquisada sobre o assunto. Os critérios de seleção foram a presença generalizada do conjunto de dados em pesquisas relacionadas à detecção de ameaças internas, a diversidade de domínios informacionais presentes no conjunto de dados e a qualidade da documentação fornecida com o conjunto de dados. O conjunto de dados consiste em seis arquivos e uma pasta. Todos os arquivos têm em comum os campos *id* (id do evento), *date* (grupo data-hora do evento), *user* (id do usuário) e *pc* (id do computador pessoal).

- ***LDAP***: Pasta com arquivos de *log* do *Lightweight Directory Access Protocol* (18 arquivos).
- ***device.csv***: Arquivo de log do dispositivo removível (405.380 linhas). Campo específico: *activity*.
- ***email.csv***: Arquivo de *log* de e-mail (2.629.979 linhas). Campos particulares: *to; cc; bcc; from; size; attachment_count; content*.
- ***http.csv***: Arquivo de *log* de navegação na Web (28.434.423 linhas). Campos particulares: URL e *content*.

[1] `https://kilthub.cmu.edu/articles/dataset/Insider_Threat_Test_Dataset/12841247/1`

- ***logon.csv***: Arquivo de *log* de *logon/logoff* de usuários (854.859 linhas). Campo específico: activity.
- ***file.csv***: Arquivo de *log* de manipulação de arquivo (445.581 linhas). Campos particulares: *filename* e *content*.
- ***psychometric.csv***: Arquivo de informações de traços de personalidade do usuário (1000 linhas, não utilizado).

A única pasta fornecida pelo conjunto de dados refere-se aos dados LDAP. Dentro desta pasta estão os arquivos com dados atualizados mensalmente sobre os funcionários. Cada funcionário tem os seguintes atributos: *employee_name*, *user_id*, *email*, *role*, *business_unit*, *functional_unit* , *department*, *team* e *supervisor*.

O conjunto de dados implementa três cenários de atividades malignas. A documentação do conjunto de dados descreve cada cenário conforme mostrado abaixo:

1. **Cenário 1:** O usuário que anteriormente não usava unidades removíveis ou trabalhava após o expediente começa a fazer login após o expediente, usando uma unidade removível e carregando dados no wikileaks.org. Deixa a organização pouco tempo depois.
2. **Cenário 2:** O usuário começa a navegar em sites de empregos e a solicitar emprego de um concorrente. Antes de deixar a empresa, eles usam um *flash drive* (em taxas consideravelmente mais altas do que em suas atividades anteriores) para roubar dados.
3. **Cenário 3:** O administrador do sistema fica descontente. Baixa um *keylogger* e usa um *flash drive* para transferi-lo para a máquina de seu supervisor. No dia seguinte, ele usa os *logs* de chaves coletadas para fazer login como se fosse seu próprio supervisor e envia um e-mail alarmante em massa, causando pânico na organização. Ele deixa a organização imediatamente após executar essa ação.

As informações dos arquivos/diretórios simulam atividades em uma rede corporativa com 1.000 usuários por aproximadamente 17 meses. Dos 1.000 usuários do conjunto de dados, apenas 70 são responsáveis por alguma atividade maliciosa. Os usuários mal-intencionados são distribuídos pelos cenários da seguinte forma: 30 usuários maliciosos internos executaram o Cenário 1, outros 30 executam o Cenário 2 e 10 executam o Cenário 3.

5.2 Extração de Atributos

O dataset tem suas informações divididas em dias, mas para a execução da pesquisa, os atributos foram extraídos e organizados por sessão de usuário. Considera-se uma sessão o período entre o *logon* e *logoff* realizado por um usuário. O ITD possui alguns eventos de logon sem seus respectivos *logoffs*. Portanto, foi necessário inserir *logoffs* implícitos toda vez que um usuário registrasse eventos de logon consecutivos na mesma estação de trabalho sem que um evento de *logoff* intercalasse esses eventos de logon. Outra intervenção que precisou ser feita, visando à extração de atributos, foi a definição do horário de início e término da jornada de trabalho. Este trabalho definiu esses tempos analisando o comportamento dos *logons* e *logoffs* registrados no dataset. Assim, pode-se afirmar que, para as análises realizadas no presente estudo, o horário de início do expediente é às 7:00 AM e o horário de término é às 17:00 (Figura 5.1). Abaixo estão os atributos extraídos e suas respectivas definições.

- ***diff_begin_first***: Diferença entre o início do horário de trabalho e o primeiro login.
- ***device_count_out***: O número de dispositivos de armazenamento removíveis usados fora do horário de trabalho.
- ***device_count***: O número de dispositivos de armazenamento removíveis usados.
- ***count_dwn_exe_file***: O número de arquivos .exe baixados.
- ***url_blocklist***: O número de URLs bloqueadas acessadas.
- ***unit_code***: Código da unidade funcional do usuário.
- ***tfidf_jobsites***: Comparação TF-IDF entre corpus de páginas de empregos e as páginas acessadas pelos usuários.

O conjunto de dados obtido após a extração dos atributos tem as seguintes características: 470.608 linhas, das quais 1.460 correspondem à classe positiva (maligna) e 469.148 à classe negativa (benigna). Esses valores geram uma relação de desbalanceamento em torno de 1:321. Essa taxa de desequilíbrio reflete como o problema se apresenta na vida real.

5.3 Experimento

Este trabalho realizou dois tipos de experimentos, um com e outro sem o retreinamento. Os experimentos sem retreinamento foram realizados para determinar se os procedimentos

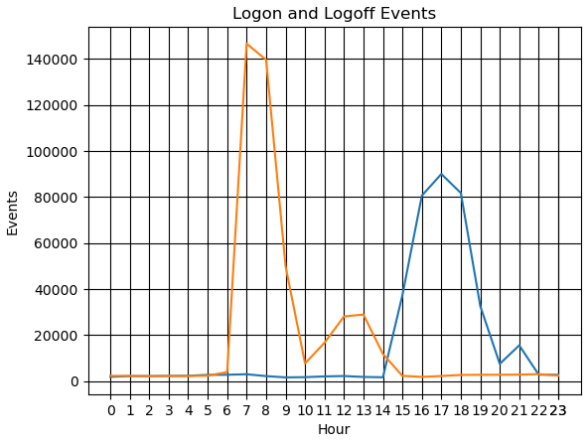

Figura 5.1: Horário de início e término de expediente.

de retreinamento poderiam melhorar a detecção de ameaças internas. Esses experimentos foram realizados utilizando todas as técnicas apresentadas nesse trabalho, inclusive com a utilização do algoritmo Grid Search e todas as demais características abordadas neste estudo menos o retreinamento.

Dadas as características do modelo, para a execução dos experimentos, alguns parâmetros metodológicos precisaram ser definidos. Como o primeiro treinamento ocorre *offline* e após isso o modelo evolui continuamente, foi necessário configurar o tamanho do conjunto de treinamento e o intervalo em que os retreinamentos irão ocorrer. O tamanho do conjunto de treinamento/retreinamento não varia durante a execução do modelo. Neste trabalho, o tratamento dos arquivos de *log* do conjunto de dados abstraiu a etapa de extração dos atributos.

Conforme afirmado na Seção 4.1, o primeiro conjunto de treinamento (fase *offline*), que será a base da geração dos conjuntos de retreinamento na fase *online*, foi predominantemente composto por amostras benignas e uma pequena contaminação de amostras malignas. A Tabela 5.1 mostra a taxa de contaminação de cada tamanho do conjunto de treinamento inicial usado nos experimentos.

Tamanho do Conjunto de Treinamento	**Taxa de Contaminação**
15 dias	1:382
30 dias	1:552
60 dias	1:1106
150 dias	1:2788

Tabela 5.1: Taxa de contaminação do primeiro conjunto de treinamento.

Como mencionado anteriormente, a contaminação simula um problema comum em ambientes da vida real. Nesses ambientes pode ser difícil afirmar que os *logs* são provenientes de atividades totalmente livres de ações malignas. Além disso, a contaminação é

pertinente para a implementação dos testes com o classificador multiclasse RF. Metodologicamente, é aconselhável ter um conjunto de dados de treinamento com amostras de todas as classes quando classificadores multiclasse são empregados de forma supervisionada. A contaminação simulada é possível porque os classificadores de uma classe não precisam de conjuntos de treinamento rotulados.

Os algoritmos usados no experimento foram ISOF [52], EV [55], LOF [53] e RF [56]. Os três primeiros são algoritmos de uma classe e sua principal característica é que eles podem treinar com amostras de apenas uma das classes presentes no domínio do problema. Após seu treinamento, em qualquer situação, esses algoritmos podem identificar, dentre as amostras testadas, aquelas que diferem das treinadas [50]. Este estudo também utilizou a implementação do RF para comparar os resultados de um algoritmo multiclasse supervisionado com aqueles obtidos pelos algoritmos de uma classe.

Antes de realizar o primeiro treinamento do algoritmo (fase *offline*), este experimento utilizou o algoritmo Grid Search para estabelecer os hiperparâmetros iniciais dos algoritmos. Para isso, aplicou o algoritmo Grid Search ao conjunto de treinamento inicial, que contém amostras de ambas as classes devido à contaminação. A métrica utilizada como *benchmark* pelo Grid Search foi a macro revocação. Tendo escolhido isso, o modelo tentou obter o maior valor médio possível entre a revocação das classes malignas e benignas. O objetivo é encontrar parâmetros que atinjam a maior taxa de verdadeiros positivos para cada uma das classes. O primeiro treinamento do algoritmo é realizado somente após a definição de seus hiperparâmetros iniciais.

O algoritmo Grid Search indicou os valores abaixo como os melhores para os hiperparâmetros dos classificadores. Os demais hiperparâmetros não citados permaneceram com o valor padrão.

- **ISOF:** 0.3 para a *contamination*; 40 para o *n_estimators*; *bootstrap activate*; e 3 para o *max_features*.
- **EV:** 0.5 para a *contamination*; e 1 para o *support_fraction*.
- **LOF:** 0.3 para a *contamination*; 20 para o *leaf_size*; *novelty activate*; *mahathan* para a *metric*; e 18 para o *n_neighbors*.
- **RF:** 200 para o *n_estimators*; *entropy* para o *criterion*; e 6 para o *min_samples_leaf*.

Após o treinamento inicial (fase *offline*), o modelo começa a receber as linhas de amostra do conjunto de dados de teste, uma a uma (fase *online*). Com isso, além de simular um fluxo de informações, foi possível avaliar o desempenho de cada um dos classificadores. A velocidade de análise de dados é um fator crucial em um ambiente com dados gerados e analisados continuamente. Este estudo variou o tamanho do conjunto de dados de

treinamento e os intervalos de retreinamento como parte dos experimentos. Todas essas mudanças visam estabelecer valores ideais para esses parâmetros metodológicos e tirar conclusões sobre a aplicação do sistema em um cenário do mundo real. Para isso, deve-se conhecer a importância desses parâmetros para o desempenho do modelo, e assim decidir se deverão ou não ser atualizados durante a execução do modelo.

Durante sua execução, os experimentos variaram os valores do tamanho da janela de treinamento/retreinamento entre 15, 30, 60 e 150 dias e o intervalo de retreinamento entre 15, 60 e 120 dias. O tamanho do conjunto de treinamento/retreinamento é importante para saber quantas amostras são necessárias para que esse conjunto se torne representativo dos dados do fluxo. A duração do intervalo de retreinamento é crucial para saber se o modelo pode se adaptar às mudanças de conceito, que podem estar presentes no fluxo de dados. Os experimentos que não aplicaram o retreinamento tiveram apenas o primeiro treinamento *offline*, inicializando as predições das amostras do fluxo de dados logo em seguida.

As métricas analisadas foram precisão (*precision*), revocação (*recall*) e *F1-score* pois são as mais indicadas para lidar com conjuntos de dados altamente desbalanceados cuja classificação é binária [10]. A métrica de precisão busca identificar entre todos os modelos classificados como positivos, quais foram classificados corretamente. A revocação indica dentre a quantidade de verdadeiros positivos, quantos deles foram previstos pelo algoritmo. O F1-score é a média harmônica entre precisão e a revocação, essa métrica fornece uma representação unificada dessas métricas [57].

Além dessas, este estudo analisou as métricas *kappa* e média geométrica (gmean). A primeira minimiza o problema de desequilíbrio de classes, pois considera a distribuição de classes em seu cálculo [58] e a última é independente da distribuição de classes e adequada para avaliar o desempenho do classificador em conjuntos de dados desbalanceados [48].

Este estudo também analisou a taxa de verdadeiros positivos Taxa de Verdadeiros Positivos (TPR) com a intenção de conhecer as proporções de predições corretas de classes maliciosas. As TPR foram analisadas por cenário e de forma geral.

Capítulo 6

Resultados

Neste capítulo são apresentados os resultados obtidos com a realização dos experimentos e suas respectivas analises. Os experimentos sem retreino necessitaram apenas da definição do tamanho do conjunto de treinamento. Os tamanhos dos conjuntos de treinamento/retreinamento variam entre 15, 30, 60 e 150 dias de amostra. Os intervalos de retreinamento variam entre 15, 60 e 120 dias. Os melhores resultados são aqueles em que os valores entre a revocação das classes maligna e benigna são altos e equilibrados entre si. Isso porque queremos evitar previsões erradas em excesso para ambas as classes. Dado o desequilíbrio do conjunto de dados, descobrimos que em todas as situações em que a revocação da classe positiva é alta, os resultados falsos negativos também são altos. Esse fato na vida real acaba camuflando os resultados positivos em meio a uma grande quantidade de falsos positivos. O contrário também pode ser afirmado, pois toda vez que temos muitas predições corretas da classe negativa, aumentamos drasticamente os resultados falsos negativos, neste caso, o modelo erra a predição da classe positiva. A métrica gmean também é importante de ser verificada, pois fornece uma visão unificada da revocação e da Taxa de Verdadeiros Negativos (TNR), quanto maior o valor, melhor o desempenho do classificador.

As Tabelas 6.1, 6.2, 6.3 e 6.4 mostram os resultados obtidos pelos algoritmos utilizados nos experimentos. A Tabela 6.1 mostra os resultados obtidos sem a realização do procedimento de retreinamento. Nessa Tabela específica, as colunas rotuladas 15, 30, 60 e 150 representam apenas o comprimento do conjunto de dados utilizado na aplicação do Grid Search e no primeiro treinamento do algoritmo. A Tabela 6.2 mostra os resultados com o período de retreinamento, configurado para ocorrer a cada 15 dias. Em seguida, a Tabela 6.3 mostra os resultados com o período de retreinamento definido para cada 60 dias. Por fim, a Tabela 6.4 mostra os valores com o período de retreinamento definido a cada 120 dias. A primeira coluna da Tabela representa o algoritmo usado nos experimentos, a segunda mostra os nomes das métricas e as demais colunas mostram os

valores obtidos pelas métricas. As colunas rotuladas como 15, 30, 60 e 150 representam o comprimento do conjunto de dados de treinamento/retreinamento. Os intervalos de retreinamento e o tamanho do conjunto de treinamento/retreinamento são expressos em dias e são aproximações baseadas no número de amostras de um dia. Os resultados em negrito representam os melhores resultados de recuperação obtidos por cada algoritmo. Em alguns casos, os algoritmos podem ter dois de seus resultados marcados em negrito. Isso se deve à proximidade entre os dois resultados obtidos por um mesmo algoritmo em situações diferentes.

A Tabela 6.1 mostra os resultados sem a aplicação do procedimento de retreinamento. Nessa Tabela vemos um claro desiquilíbrio entre as métricas revocação da classe benigna e maligna, enquanto uma delas apresenta um resultado bom a outra apresenta um valor muito ruim. Para o algoritmo RF os valores obtidos para a métrica revocação da classe maligna e para as métricas que oferecem uma visão consolidada de ambas as classes são nulos. Esse fato evidencia a dificuldade na adaptabilidade do algoritmo ao alto desbalanceamento presente no conjunto de dados.

A principal diferença entre os resultados obtidos na Tabela 6.1 e os obtidos usando o retreinamento é a redução drástica dos falsos positivos, mesmo com um aumento na detecção de classes malignas. Essa redução é relevante no contexto de segurança e está relacionada ao esforço de análise a ser demandado quando se utiliza uma solução de segurança ruidosa. Alarmes falsos em excesso podem atrasar a identificação de ameaças internas ou, no pior dos casos, resultar na sua não identificação. Alarmes verdadeiros podem estar encobertos pela quantidade excessiva dos falsos, dificultando ainda mais a identificação de ameaças internas. Este problema é recorrente nos domínios da segurança da informação e já foi descrito por Gheyas *et al.* [3] no contexto da detecção de ameaças internas. Como a classe benigna representa 99,5% (287.860 amostras) do total de amostras (289.230 amostras), sua redução de proporcionalidade é muito mais significativa do que a da classe maligna. Por exemplo, em números absolutos, 10% das amostras benignas representam 28.923 amostras, e a mesma porcentagem da classe maligna representa apenas 142 amostras.

Para os melhores resultados de todos os cenários, o procedimento de retreinamento reduziu a diferença entre as revocações das classes malignas e benignas. Como exemplo, tomando o número de alarmes falsos positivos dos melhores resultados, com e sem retreinamento, produzidos pelos três algoritmos, observa-se que o ISOF gera 57.770 deste tipo de alarme, executando o retreinamento, contra 157.352 sem executá-lo. A diferença aproxima-se de cem mil amostras, sendo que, a quantidade de falsos positivos sem execução do retreino é mais que o dobro das apresentadas no melhor caso com retreinamento.

As Tabelas 6.2, 6.3 e 6.4 apresentam os resultados com a execução do retreinamento e

Algoritmo	Classe	Métrica	Tamanho do Conjunto de Treinamento (dias) 15	30	60	150
ISOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.68	**0.67**	0.66	0.63
		F1-score	0.81	0.80	0.80	0.77
	Maligna	precision	0.01	0.01	0.01	0.01
		recall	0.87	**0.88**	0.88	0.89
		F1-score	0.03	0.03	0.03	0.02
	Benigna/Maligna	kappa	0.01	0.01	0.01	0.01
		gmean	0.76	**0.76**	0.76	0.74
EV	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.48	**0.47**	**0.47**	0.45
		F1-score	0.65	**0.64**	**0.64**	0.62
	Maligna	precision	0.02	0.02	0.02	0.01
		recall	0.93	**0.93**	**0.93**	0.94
		F1-score	0.02	0.02	0.02	0.02
	Benigna/Maligna	kappa	0.01	0.01	0.01	0.01
		gmean	0.66	**0.66**	**0.66**	0.64
LOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.77	0.77	0.76	**0.73**
		F1-score	0.87	0.87	0.64	0.84
	Maligna	precision	0.01	0.01	0.01	0.01
		recall	0.54	0.50	0.51	**0.59**
		F1-score	0.02	0.02	0.02	0.02
	Benigna/Maligna	kappa	0.01	0.01	0.01	0.01
		gmean	0.64	0.62	0.62	**0.65**
RF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	1.00	1.00	1.00	1.00
		F1-score	1.00	1.00	1.00	1.00
	Maligna	precision	0.00	0.00	0.00	0.00
		recall	0.00	0.00	0.00	0.00
		F1-score	0.00	0.00	0.00	0.00
	Benigna/Maligna	kappa	0.00	0.00	0.00	0.00
		gmean	0.00	0.00	0.00	0.00

Tabela 6.1: Métricas obtidas sem retreinamento.

mostram que o algoritmo ISOF obteve o melhor resultado de revocação entre os algoritmos utilizados, chegando a **0,80** para a classe benigna e **0,78** para a maligna. Este estudo considerou que o ISOF foi o algoritmo que melhor se adaptou ao problema, isto porque, obteve o melhor equilíbrio entre as métricas de revocação das classes benignas e malignas, alcançando o valor mais alto para a métrica gmean, **0,79**, no seu melhor caso. O ISOF também foi o único que identificou todos os usuários maliciosos com pelo menos um alerta para cada e além disso, foi o algoritmo menos ruidoso. O melhor resultado obtido pelo ISOF está presente na Tabela 6.3. Para obter os resultados apresentados nessa tabela o ISOF utilizou o intervalo de retreinamento e o tamanho do conjunto treinamento/retreinamento correspondentes a aproximadamente dois meses de atividades (72.000 amostras). Essas são as melhores configurações de parâmetros metodológicos para esse algoritmo.

Para o algoritmo EV, os melhores valores metodológicos foram cinco meses para o tamanho do conjunto de treinamento/retreinamento e dois meses para o intervalo de retreinamento. Os valores para revocação das classes benigna e maligna foram **0,69** e **0,82**, respectivamente. O algoritmo EV foi o mais rápido. Foi, em média, nove vezes mais rápido que o ISOF e um pouco mais rápido que o LOF. Os resultados do melhor caso do algoritmo EV estão presentes na Tabela 6.3.

O algoritmo LOF teve seu melhor resultado usando cinco meses para o tamanho do conjunto de treinamento/retreinamento e quatro meses para intervalo de retreinamento. Os valores de revocação com esta configuração foram **0,76** para a classe benigna e **0,58** para a classe maligna. Esses valores estão representados na Tabela 6.4.

Este estudo utilizou o algoritmo de RF para analisar como um algoritmo multiclasse se comportaria quando submetido às dificuldades que envolvem o problema de detecção de ameaças internas, principalmente no que diz respeito ao desequilíbrio do conjunto de dados. Os melhores resultados obtidos pelo RF para a métrica de revocação foram 0,98 para classe benigna e 0,23 para classe maligna. Para alcançar esses resultados, RF usou um quinze dias para o tamanho do conjunto de treinamento/retreinamento e quinze dias para o intervalo de retreinamento. Após a verificação dos resultados de RF, este trabalho concluiu que o algoritmo de RF não se ajusta bem quando aplicado ao ambiente de detecção de ameaças internas. O conjunto de dados altamente desbalanceado pode ser a causa crucial do baixo desempenho de RF. Devido à escassez de amostras de classes positivas, o RF pode não ter sido capaz de identificar esta classe com eficiência.

Outra análise significativa é entender como o desempenho dos algoritmos varia ao longo do tempo. As figuras 6.1, 6.2 e 6.3 mostram o comportamento das métricas de precisão, revocação e F1-score durante a execução do modelo. O eixo y representa os valores das métricas e o eixo x representa o número de amostras analisadas (sessões). A linha amarela corresponde aos resultados ISOF, a linha azul corresponde aos resultados

Algoritmo	Classe	Tamanho do Conjunto de Treinamento/Retreinamento(days) Métrica	15	30	60	150
ISOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	**0.81**	0.80	0.80	0.79
		F1-score	0.89	0.89	0.89	0.88
	Maligna	precision	0.02	0.02	0.02	0.01
		recall	**0.76**	0.75	0.76	0.77
		F1-score	0.04	0.04	0.04	0.02
		kappa	0.03	0.03	0.03	0.02
		gmean	**0.78**	0.77	0.78	0.78
EV	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.87	0.87	**0.86**	0.88
		F1-score	0.93	0.93	0.92	0.94
	Maligna	precision	0.02	0.02	0.02	0.01
		recall	0.50	0.52	**0.53**	0.44
		F1-score	0.04	0.04	0.04	0.02
	Benigna/Maligna	kappa	0.03	0.02	0.03	0.02
		gmean	0.65	0.67	**0.68**	0.70
LOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.79	0.79	**0.78**
		F1-score	0.89	0.88	0.88	0.88
	Maligna	precision	0.01	0.01	0.01	0.01
		recall	0.50	0.45	0.45	**0.55**
		F1-score	0.02	0.02	0.02	0.02
	Benigna/Maligna	kappa	0.03	0.02	0.03	0.02
		gmean	0.63	0.60	0.60	**0.66**
RF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	**0.98**	0.97	0.98	1.00
		F1-score	0.99	0.98	0.99	1.00
	Maligna	precision	0.06	0.03	0.04	0.11
		recall	**0.23**	0.19	0.16	0.12
		F1-score	0.09	0.05	0.07	0.12
	Benigna/Maligna	kappa	0.08	0.04	0.06	0.11
		gmean	**0.47**	0.42	0.39	0.35

Tabela 6.2: Métricas obtidas com o retreinamento a cada 15 dias.

Algoritmo	Classe	Tamanho do Conjunto de Treinamento/Retreinamento(days) Métrica	15	30	60	150
ISOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.80	**0.80**	0.79
		F1-score	0.88	0.89	0.89	0.89
	Maligna	precision	0.02	0.02	0.02	0.02
		recall	0.76	0.76	**0.78**	0.77
		F1-score	0.04	0.04	0.04	0.04
	Benigna/Maligna	kappa	0.03	0.03	0.03	0.02
		gmean	0.78	0.78	**0.79**	0.78
EV	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.80	0.80	**0.76**
		F1-score	0.89	0.89	0.89	0.86
	Maligna	precision	0.02	0.02	0.02	0.01
		recall	0.65	0.64	0.65	**0.71**
		F1-score	0.03	0.03	0.03	0.03
	Benigna/Maligna	kappa	0.02	0.02	0.02	0.02
		gmean	0.72	0.72	0.72	**0.73**
LOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.79	0.79	0.79	**0.77**
		F1-score	0.88	0.88	0.88	0.87
	Maligna	precision	0.01	0.01	0.01	0.01
		recall	0.50	0.46	0.47	**0.57**
		F1-score	0.02	0.02	0.02	0.02
	Benigna/Maligna	kappa	0.01	0.01	0.01	0.01
		gmean	0.63	0.60	0.61	**0.66**
RF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.99	**0.99**	0.99	1.00
		F1-score	0.99	0.99	0.99	0.99
	Maligna	precision	0.05	0.07	0.06	0.06
		recall	0.14	**0.18**	0.14	0.12
		F1-score	0.08	0.11	0.09	0.08
	Benigna/Maligna	kappa	0.07	0.05	0.09	0.10
		gmean	0.36	**0.42**	0.37	0.36

Tabela 6.3: Métricas obtidas com o retreinamento a cada 60 dias.

Algoritmo	Classe	Tamanho do Conjunto de Treinamento/Retreinamento(days) Métrica	15	30	60	150
ISOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.80	0.80	**0.80**	0.78
		F1-score	0.89	0.89	0.89	0.88
	Maligna	precision	0.02	0.02	0.02	0.02
		recall	0.77	0.76	**0.78**	0.77
		F1-score	0.04	0.04	0.04	0.03
	Benigna/Maligna	kappa	0.03	0.03	0.03	0.02
		gmean	0.78	0.78	**0.79**	0.78
EV	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.69	**0.69**	**0.69**	0.68
		F1-score	0.82	0.82	0.82	0.81
	Maligna	precision	0.01	0.01	0.01	0.01
		recall	0.81	**0.82**	**0.82**	0.82
		F1-score	0.03	0.03	0.03	0.02
	Benigna/Maligna	kappa	0.02	0.01	0.01	0.01
		gmean	0.75	**0.75**	**0.75**	0.74
LOF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.79	0.79	0.79	**0.76**
		F1-score	0.88	0.88	0.88	0.86
	Maligna	precision	0.01	0.01	0.01	0.01
		recall	0.51	0.47	0.49	**0.58**
		F1-score	0.02	0.02	0.02	0.02
	Benigna/Maligna	kappa	0.01	0.01	0.01	0.01
		gmean	0.63	0.61	0.62	**0.66**
RF	Benigna	precision	1.00	1.00	1.00	1.00
		recall	0.99	**0.98**	0.99	1.00
		F1-score	0.99	0.99	0.99	1.00
	Maligna	precision	0.08	0.04	0.09	0.15
		recall	0.14	**0.17**	0.14	0.11
		F1-score	0.11	0.07	0.12	0.15
	Benigna/Maligna	kappa	0.08	0.05	0.10	0.11
		gmean	0.37	**0.41**	0.37	0.32

Tabela 6.4: Métricas obtidas com o retreinamento a cada 120 dias.

EV e a linha vermelha corresponde aos resultados LOF.

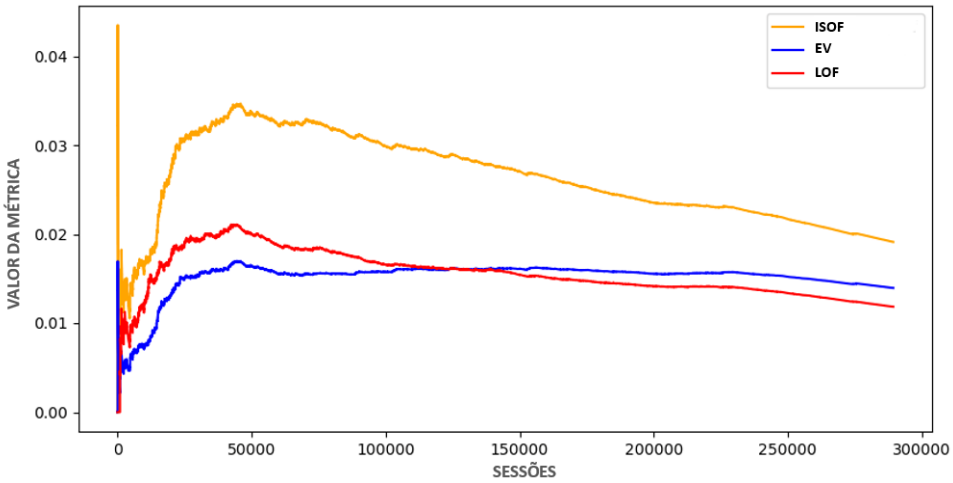

Figura 6.1: Evolução da métrica precisão da classe maligna.

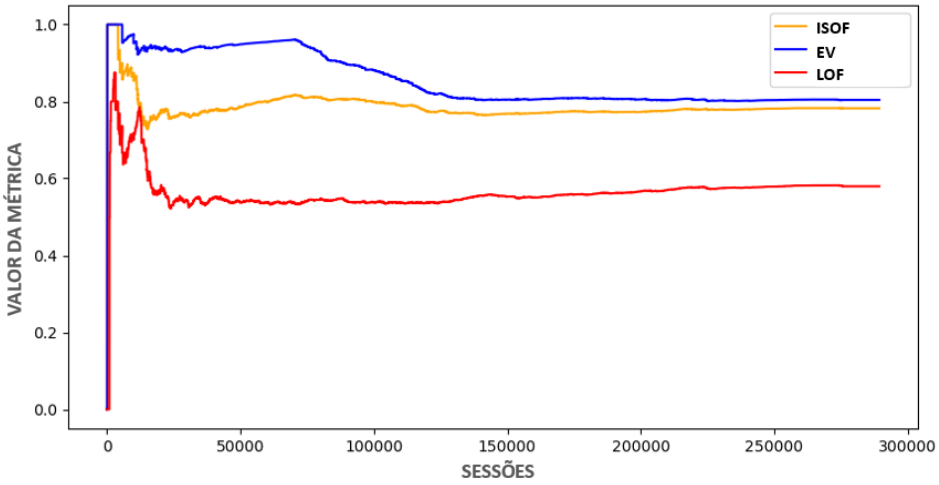

Figura 6.2: Evolução da métrica revocação da classe maligna.

As figuras mostram que sempre que a curva de revocação se estabiliza a curva de precisão tende a diminuir. Na mesma situação, a curva F1-score também diminui ligeiramente, influenciada pela precisão. A provável causa para isso é um aumento na ocorrência de falsos positivos em valores absolutos. Os resultados verdadeiros e falsos crescem em valores absolutos muito distantes devido ao alto índice de desequilíbrio entre as classes presentes no conjunto de dados. Depois de analisar cerca de 50.000 amostras, as linhas dos gráficos param de oscilar drasticamente e se tornam mais suaves, sinalizando um momento em que a classe maligna já possa ter ocorrido em quantidade suficientemente necessária a sua delimitação.

Além disso, este estudo realizou uma análise de cenários maliciosos, conforme mostrado na Tabela 6.5. Nessa Tabela, pode-se observar a matriz de confusão em formato

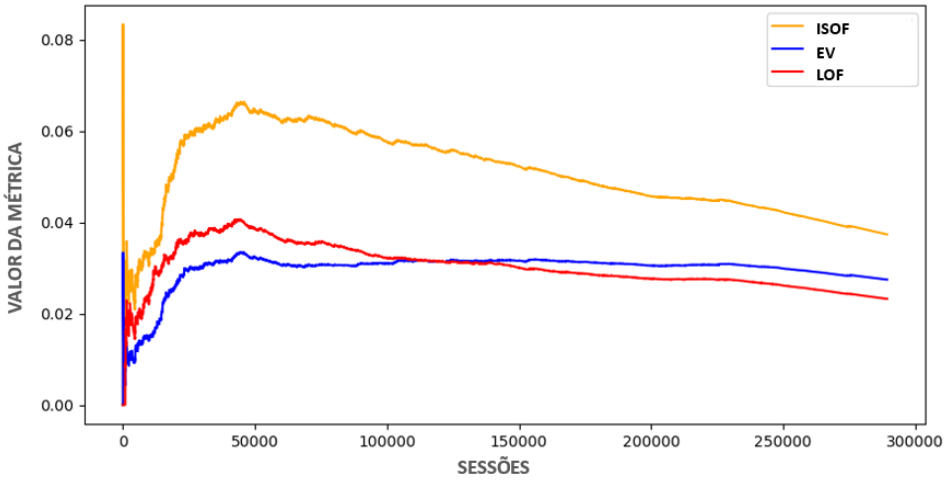

Figura 6.3: Evolução da métrica F1-score da classe maligna.

tabelado, os resultados individualizados por cenário e os respectivos desempenhos obtidos por cada algoritmo. Os resultados apresentados são provenientes do melhor caso para cada algoritmo. A primeira coluna indica o cenário, a segunda mostra as métricas e as demais os algoritmos, exceto a última em negrito que apresenta a totalização. A coluna Total representa a soma de todas as amostras malignas em cada cenário e a última linha é a soma das benignas. A última linha em negrito representa o ruído gerado por resultados falsos positivos, não está dividida por cenário porque representa amostras da classe benigna que foram incorretamente classificadas. O primeiro cenário tem 189 atividades maliciosas executadas por 30 usuários, o segundo tem 1110 atividades maliciosas executadas por 30 usuários e o terceiro tem 121 atividades maliciosas executadas por 10. O algoritmo ISOF foi o menos ruidoso, com 57.770 alarmes falsos positivos.

Cenário	**Métricas**	**ISOF**	**EV**	**LOF**	**Total**
1	VP	89	105	52	
	FN	100	84	137	**189**
2	VP	927	946	741	
	FN	183	164	369	**1110**
3	VP	90	107	30	
	FN	31	14	91	**121**
-	VN	230.090	199.286	222.341	
	FP	**57.770**	**88.574**	**65.519**	**287.860**

Tabela 6.5: Representação em tabela da matriz de confusão individualizada por cenário.

Usando o algoritmo ISOF, o modelo detectou todos os usuários maliciosos emitindo pelo menos um alerta para cada um. Na maioria dos casos, um alarme foi emitido para a primeira atividade maliciosa realizada por um usuário malicioso. Para o primeiro cenário

oito agentes internos não foram identificados na primeira vez em realizaram uma atividade maliciosa. Para o segundo e terceiro cenários três agentes maliciosos internos não tiveram sua primeira atividade maliciosas alertada.

Com o algoritmo EV, o modelo não detectou apenas dois usuários maliciosos. Ambos os usuários não detectados pertenciam ao primeiro cenário. Em relação à detecção da primeira atividade maliciosa realizada por um agente interno malicioso, o algoritmo EV não identificou a primeira atividade maliciosa de sete usuários do primeiro cenário, de cinco usuários do segundo e de um usuário do terceiro.

Por fim, ao utilizar o algoritmo LOF, o modelo não detectou apenas sete usuários maliciosos, todos eles pertenciam ao primeiro cenário. Além disso, não alertou a primeira atividade maliciosa de vinte usuários maliciosos do primeiro cenário, de nove usuários maliciosos do segundo cenário e de sete usuários maliciosos do terceiro.

Cenário	**Métricas**	**ISOF**	**EV**	**LOF**
1	TPR	0.47	0.55	0.28
2	TPR	0.83	0.85	0.67
3	TPR	0.74	0.88	0.25
ALL	TPR	0.78	0.82	0.58

Tabela 6.6: Taxa de verdadeiro positivo por cenário.

A Tabela 6.6 representa a Taxa de Verdadeiros Positivos (TPR), calculada para cada cenário. O objetivo dessa Tabela é identificar, de forma proporcional, qual dos cenários obteve a menor taxa de detecção de suas respectivas atividades maliciosas. Observando a Tabela 6.6, pode-se afirmar que o primeiro cenário foi o mais desafiador de se identificar. O algoritmo EV teve melhor desempenho neste cenário, mas ao custo de uma alta geração de resultados falso positivos e mesmo assim, não conseguiu identificar dois dos trinta usuários maliciosos presentes no cenário. Apesar de um desempenho um pouco pior para o primeiro cenário, o algoritmo ISOF identificou todos os usuários maliciosos envolvidos.

Ao analisar os resultados, pode-se notar que os algoritmos de uma classe se adaptam melhor aos cenários de ameaças internas apresentados no ITD. As escassez de amostras rotuladas penalizaram o algoritmo de RF que não foi capaz de se adaptar adequadamente ao conjunto de dados altamente desbalanceado. Entre os algoritmos testados o ISOF foi o algoritmo de uma classe que melhor se adaptou.

O retreinamento influenciou positivamente nos resultados, esse fato pode ser constato ao compararmos os resultados das Tabelas que utilizaram o retreinamento com a Tabela que apresentou os resultados sem esse procedimento. Um fato que evidencia sobremaneira essa constatação são os resultados obtidos pelo RF que estavam completamente nulos sem

o uso do retreinamento e que após a aplicação desse procedimento passaram a apresentar resultados mensuráveis.

Outro fator importante foi boa adaptabilidade dos algoritmos de uma classe à analise baseada em fluxo de dados desbalanceados. Isso devido a possibilidade de treinamento dos modelos com amostras de apenas uma das classes.

Capítulo 7

Conclusão

Os sistemas de detecção de ameaças internas com base no aprendizado supervisionado em lote podem não ser adequados para cenários reais de detecção. Isso ocorre pois é difícil obter conjuntos de dados rotulados em tempo real e os comportamentos de ameaças internas mudam continuamente ao longo do tempo [6]. Não há assinatura para identificar ameaças internas e quanto mais sofisticado o ataque, mais ele se parece com uma atividade benigna. Considerando a abordagem de aprendizado de máquina não supervisionado focada no fluxo de dados, um modelo capaz de lidar com a detecção de ameaças internas pode ser construído de forma mais prática. Essa abordagem elimina a dependência de amostras rotuladas, tornando um sistema baseado nessa abordagem viável de ser implementado no mundo real.

Baseado nessas informações esse trabalho buscou analisar os principais óbices do problema de detecção de ameaças internas e propor uma abordagem para minimizá-los. Para o desbalanceamento do conjunto de dados propomos o uso de aprendizado não supervisionado por meio da aplicação dos algoritmos de uma classe, visando utilizar a excessiva quantidade de amostras benignas para treinamento do modelo. Complementarmente foi realizada a seleção de amostras buscando obter aquelas mais representativas para compor o conjunto de retreinamento do modelo. Os algoritmos classificadores de uma classe parecem ser uma opção promissora para compor soluções para detecção de ameaças internas que utilize AM. A possibilidade de treinar o modelo com apenas uma das classes de forma não supervisionada permitiu a não dependência do uso de amostras rotuladas.

As mudanças de conceito foram abordadas através da estratégia de retreinamento periódico do modelo, desta forma objetivamos a melhor adaptação do modelos as referidas mudanças ao longo do tempo. O retreinamento contribuiu de forma relevante para a viabilização do *framework* proposto, pois possibilitou a redução acentuada de resultados falso positivos. Essa redução ficou bastante evidente quando comparados os resultados obtidos com e sem a utilização do retreino. Entre os melhores resultados obtidos com

a utilização do retreino vemos pouca variação conforme os parâmetros de tamanho do conjunto e intervalo de retreino eram alterados. Isso demostra que o *framework* consegue manter certa estabilidade mesmo com a alteração desses parâmetros.

Por último para trazer a pesquisa mais próxima de um cenário real, utilizamos a abordagem de analise de fluxo de dados. O principal motivo para essa escolha está fundamentado numa característica do próprio problema, que é o fato da geração dos dados de uma rede em produção, ocorrer de forma contínua e indefinida, tornando inviável o armazenamento de todo o dado gerado ao longo do tempo para análise. Esse fato torna imperiosa uma estratégia de análise que consiga utilizar os recursos computacionais para análise dos dados sem esgotá-los, coincidindo com as características de análise de fluxos de dados.

Os resultados demonstraram a eficácia do *framework* e a viabilidade da sua implementação em um cenário do mundo real. No presente caso, o ISOF foi o algoritmo que melhor se adaptou ao problema. Esse algoritmo alcançou um equilíbrio relativo entre as métricas de revocação das classes benignas (**0,80**) e malignas (**0,78**) adaptando-se melhor ao desequilíbrio acentuado presente no conjunto de dados. Embora o *framework* implementado não tenha detectado todas as amostras maliciosas, ele detectou atividades maliciosas de todos os cenários implementados no conjunto de dados usando o algoritmo ISOF. O modelo também gerou alarmes para todos os usuários mal-intencionados no fluxo de dados. Na maioria dos casos, o modelo identificou os usuários maliciosos em sua primeira atividade maliciosa.

7.1 Contribuições em Produção Bibliográfica

A presente pesquisa gerou a seguinte publicação até presente momento:

- Peccatiello, Rafael Bruno, João José Costa Gondim e Luís Paulo Faina Garcia: Applying one-class algorithms for data stream-based insider threat detection. IEEE Access, 11:70560–70573, 2023. 47 [59].

7.2 Trabalhos futuros

Dado o exposto até aqui, verificamos algumas oportunidades para melhorias, como a utilização de outros tipos de algoritmos pois este estudo limitou-se a alguns tipos de classificadores de uma classe. Ainda há espaço para a proposição de novos atributos, visando aumentar a representatividade do conjunto de dados, reduzindo assim a ocorrência de falsos resultados. Mesmo sendo compensatório a checagem de falsos resultados em

alguns casos [10] a sua redução traria mais racionalização no uso dos recursos destinados à segurança. O presente estudo adota um abordagem de retreino periódico como forma de lidar com as mudanças de conceito, uma possível evolução seria realizar o retreino a partir da detecção das mudanças de conceito, complementarmente poderia ser adicionada uma forma de validação das amostras utilizadas no retreino. Outras possibilidades de ampliação do presente estudo são explorar a utilização de dados psicossociais, como os que são disponibilizados pelo ITD, e realizar um estudo de caso em uma rede real que seja capaz de permitir a coleta de todos os dados utilizados na prospecção do conjunto atributos desta pesquisa.

Os trabalhos futuros propostos visam contornar ou eliminar algumas das limitações enfrentadas na presente pesquisa. Para o caso da detecção de ameaças internas a tarefa de obter a amostras reais totalmente livres de contaminação é bastante complicada. Uma opção pode ser apenas usar dados considerados benignos pelas soluções de segurança de rede. No entanto, mesmo nesses casos, não estaríamos protegidos contra a exploração de vulnerabilidades do tipo *zero-day* ou de ameaças avançadas persistentes. Este fato motivou a inclusão deliberada da contaminação nos experimentos realizados neste estudo. Outra limitação está relacionada ao fluxo de dados que neste estudo foi simulado por meio de um algoritmo utilizando os dados do conjunto de pesquisa. Não foram utilizados geradores de fluxos aleatórios pois os dados gerados deveriam estar relacionados ao domínio do problema e serem provenientes do ITD. Por último, cito que por mais que a detecção tenha sido realizada com a análise baseada em um fluxo, ainda existe um atraso na emissão do alerta que está relacionado ao encerramento da seção por parte do usuário, pois, somente após encerrada a sessão teremos os dados referentes a ela para serem analisados.

Referências

[1] Cybersecurity e Infrastructure Security Agency: *Insider Threat Mitigation.* Relatório Técnico November, Cybersecurity e Infrastructure Agency, 2020. `https://www.cisa.gov/insider-threat-mitigation`. 1

[2] Soh, Charlie, Sicheng Yu, Annamalai Narayanan, Santhiya Duraisamy e Lihui Chen: *Employee profiling via aspect-based sentiment and network for insider threats detection.* Expert Systems with Applications, 135:351–361, 2019, ISSN 09574174. `https://doi.org/10.1016/j.eswa.2019.05.043`. 1

[3] Gheyas, Iffat A. e Ali E. Abdallah: *Detection and prediction of insider threats to cyber security: a systematic literature review and meta-analysis.* Big Data Analytics, 1(1), 2016, ISSN 2058-6345. 1, 38

[4] Homoliak, Ivan, Flavio Toffalini, Juan Guarnizo e Yuval Elovici: *Insight into Insiders and IT: A Survey of Insider Threat Taxonomies, Analysis, Modeling, and Countermeasures.* ACM Computing Surveys, 99(99), 2017. 2, 4, 7, 9

[5] Le, Duc C. e A. Nur Zincir-Heywood: *Machine learning based insider threat modelling and detection.* 2019 IFIP/IEEE Symposium on Integrated Network and Service Management, IM 2019, páginas 1–6, 2019. 2, 4, 5, 10, 18

[6] Insitute, Ponemon: *2022, Cost of Insider Treat, Global Report.* Relatório Técnico, Proofpoint Institute, 2022. `https://www.proofpoint.com/us/resources/threat-reports/cost-of-insider-threats`. 2, 4, 5, 48

[7] Kim, Junhong, Minsik Park, Haedong Kim, Suhyoun Cho e Pilsung Kang: *Insider threat detection based on user behavior modeling and anomaly detection algorithms.* Tese de Doutoramento, School of Industrial Management Engineering, 2019. 2, 5, 18, 21, 22, 25

[8] Johannessen Berdal, Sondre: *A Holistic Approach to Insider Threat Detection.* Tese de Mestrado, University of Oslo, 2018. 2, 3, 4, 5, 16, 17, 18, 20, 22

[9] Bose, Brock, Bhargav Avasarala, Srikanta Tirthapura, Yung Yu Chung e Donald Steiner: *Detecting Insider Threats Using RADISH: A System for Real-Time Anomaly Detection in Heterogeneous Data Streams.* IEEE Systems Journal, 11(2):471–482, 2017, ISSN 19379234. 2, 3, 4, 5, 23, 24, 25

[10] Chattopadhyay, Pratik, Lipo Wang e Yap Peng Tan: *Scenario-based insider threat detection from cyber activities.* IEEE Transactions on Computational Social Systems, 5(3):660–675, 2018, ISSN 2329924X. 2, 4, 5, 12, 14, 18, 20, 22, 25, 36, 50

[11] Hall, Adam James, Nikolaos Pitropakis, William J. Buchanan e Naghmeh Moradpoor: *Predicting malicious insider threat scenarios using organizational data and a heterogeneous stack-classifier*. arXiv, 2019. 2, 5, 18, 21

[12] Senator, Ted E., Henry G. Goldberg, Alex Memory, William T. Young, Brad Rees, Robert Pierce, Daniel Huang, Matthew Reardon, David A. Bader, Edmond Chow, Irfan Essa, Joshua Jones, Vinay Bettadapura, Duen Horng Chau, Oded Green, Oguz Kaya, Anita Zakrzewska, Erica Briscoe, Rudolph L. Mappus, Robert Mccoll, Lora Weiss, Thomas G. Dietterich, Alan Fern, Weng Keen Wong, Shubhomoy Das, Andrew Emmott, Jed Irvine, Jay Yoon Lee, Danai Koutra, Christos Faloutsos, Daniel Corkill, Lisa Friedland, Amanda Gentzel e David Jensen: *Detecting insider threats in a real corporate database of computer usage activity*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Part F128815:1393–1401, 2013. 2, 4, 5, 22, 24, 25

[13] Parveen, Pallabi, Nathan McDaniel, Zackary Weger, Jonathan Evans, Bhavani Thuraisingham, Kevin Hamlen e Latifur Khan: *Evolving insider threat detection stream mining perspective*. International Journal on Artificial Intelligence Tools, 22(5):1–24, 2013, ISSN 02182130. 2, 3, 4, 5, 10, 18, 24, 25, 27

[14] Noble, Jordan e Niall Adams: *Real-time dynamic network anomaly detection*. IEEE Intelligent Systems, 33(2):5–18, 2018, ISSN 15411672. 2

[15] Le, Duc C. e A. Nur Zincir-Heywood: *Evaluating insider threat detection workflow using supervised and unsupervised learning*. Proceedings - 2018 IEEE Symposium on Security and Privacy Workshops, SPW 2018, páginas 270–275, 2018. 2, 4, 19

[16] Gavai, Gaurang, Kumar Sricharan, Dave Gunning, John Hanley, Mudita Singhal e Rob Rolleston: *Supervised and unsupervised methods to detect insider threat from enterprise social and online activity data*. Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications, 6(4):47–63, 2015, ISSN 20935382. 2, 3, 4, 5, 18, 19

[17] Tuor, Aaron, Samuel Kaplan, Brian Hutchinson, Nicole Nichols e Sean Robinson: *Deep learning for unsupervised insider threat detection in structured cybersecurity data streams*. AAAI Workshop - Technical Report, WS-17-01 - WS-17-15:224–234, 2017. 2, 3, 22

[18] Kim, Dong Wook, Sung Sam Hong e Myung Mook Han: *A study on classification of insider threat using markov chain model*. KSII Transactions on Internet and Information Systems, 12(4):1887–1898, 2018, ISSN 22881468. 2, 5, 18, 19

[19] Zhang, Hang, Weike Liu, Shuo Wang, Jicheng Shan e Qingbao Liu: *Resample-based ensemble framework for drifting imbalanced data streams*. IEEE Access, 7:65103–65115, 2019, ISSN 21693536. 2, 3, 5, 11, 14, 23, 24, 25

[20] Kim, Aram, Junhyoung Oh, Jinho Ryu e Kyungho Lee: *A review of insider threat detection approaches with IoT perspective*. IEEE Access, 8:78847–78867, 2020, ISSN 21693536. 3, 10

[21] Liu, Liu, Olivier De Vel, Qing Long Han, Jun Zhang e Yang Xiang: *Detecting and Preventing Cyber Insider Threats: A Survey*. IEEE Communications Surveys and Tutorials, 20(2):1397–1418, 2018, ISSN 1553877X. 3, 4, 9

[22] Al-mhiqani, Mohammed Nasser, Rabiah Ahmad, Z Zainal Abidin e Warusia Yassin: *A Review of Insider Threat Detection: Classification Machine Learning Techniques Datasets Open Challenges and Recommendations*. Applied Sciences (Switzerland), página 41, 2020. 4, 9

[23] Glasser, Joshua e Brian Lindauer: *Bridging the gap: A pragmatic approach to generating insider threat data*. Proceedings - IEEE CS Security and Privacy Workshops, SPW 2013, páginas 98–104, 2013. 4, 31

[24] Azaria, Amos, Ariella Richardson, Sarit Kraus e V S Subrahmanian: *Behavioral Analysis of Insider Threat: A Survey and Bootstrapped Prediction in Imbalanced Data*. IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, páginas 1–25, 2014. `http://www.cert.` 4, 10

[25] Bose, Brock, Bhargav Avasarala, Srikanta Tirthapura, Yung Yu Chung e Donald Steiner: *Detecting Insider Threats Using RADISH: A System for Real-Time Anomaly Detection in Heterogeneous Data Streams*. IEEE Systems Journal, 11(2):471–482, 2017, ISSN 19379234. 4, 18

[26] Janjua, Faisal, Asif Masood, Haider Abbas e Imran Rashid: *ScienceDirect Handling Handling Insider Insider Threat Threat Through Through Supervised Supervised Machine Machine Learning Learning Techniques Techniques*. Procedia Computer Science, 177:64–71, 2020, ISSN 1877-0509. `https://doi.org/10.1016/j.procs.2020.10.012`. 4

[27] Ribeiro, Guilherme Henrique: *Detecção de botnets utilizando classificação de fluxos contínuos de dados*. Tese de Mestrado, Universidade Federal de Uberlândia, 2020. 5

[28] Silowash, George, Dawn Cappelli, Andrew Moore, Randall Trzeciak, Timothy Shimeall e Lori Flynn: *Common Sense Guide to Mitigating Insider Threats, 4th Edition*. Relatório Técnico December, Carnegie Mellon University, 2012. 5

[29] Saxena, Neetesh, Emma Hayes, Elisa Bertino, Patrick Ojo, Kim Kwang Raymond Choo e Pete Burnap: *Impact and key challenges of insider threats on organizations and critical businesses*. Electronics (Switzerland), 9(9):1–29, 2020, ISSN 20799292. 7, 9

[30] Cappelli, Dawn, Andrew Moore e Randall Trzeciak: *The CERT Guide to Insider Threats: How ro Prevet, Detect, and Respond to Information Technology Crimes (Theft, Sabotage, Fraud)*. Addison-Wesley Professional, Westford, Massachusetts, 1ª edição, 2012, ISBN 9780321812575. 8, 9

[31] Yuan, Shuhan e Xintao Wu: *Deep Learning for Insider Threat Detection: Review, Challenges and Opportunities*. arXiv, 2020, ISSN 23318422. 10, 25

[32] Rossi, André Luis Debiaso, André Carlos Ponce de Leon Ferreira de Carvalho, Carlos Soares e Bruno Feres de Souza: *MetaStream: A meta-learning based method for periodic algorithm selection in time-changing data*. Neurocomputing, 127:52–64, 2014, ISSN 09252312. http://dx.doi.org/10.1016/j.neucom.2013.05.048. 11

[33] Watson, Chester C., Brian L. Van Zanten e Steven R. Abt: *Stream classification*. Em Sammut, Claude e Geoffrey I. Webb (editores): *Encyclopedia of Machine Learning*, páginas 1097–1100. Springer, Boston, MA, 2016, ISBN 978-0-387-34558-1. 11, 13

[34] Sá, Jáder M.C. de, Andre L.D. Rossi, Gustavo E.A.P.A. Batista e Luís P.F. Garcia: *Algorithm recommendation for data streams*. Proceedings - International Conference on Pattern Recognition, páginas 6073–6080, 2020, ISSN 10514651. 11, 12

[35] Gao, Jing, Bolin Ding, Wei Fan, Jiawei Han e Philip S. Yu: *Classifying data streams with skewed class distributions and concept drifts*. IEEE Internet Computing, 12(6):37–49, 2008, ISSN 10897801. 11, 14

[36] Gama, J. e A. Zliobait˘ e,˙ I., Bifet, A., Pechenizkiy, M., and Bouchachia: *A Survey on Concept Drift Adaptation*. ACM Computing Surveys, página 35, 2013. https://dl.acm.org/doi/10.1145/2523813. 12

[37] Bifet, Albert, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby e Ricard Gavaldà: *New ensemble methods for evolving data streams*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, páginas 139–147, 2009. 12, 13, 25

[38] Ditzler, Gregory, Manuel Roveri, Cesare Alippi e Robi Polikar: *Learning in Non-stationary Environments: A Survey*. IEEE Computational Intelligence Magazine, 10(4):12–25, 2015, ISSN 1556603X. 12

[39] Gama, Joao: *Knowledge Discovery from Data Streams*. Chapman and Hall/CRC, 1ª edição, 2010, ISBN 978-1439826119. 12

[40] Kuncheva, Ludmila I.: *Classifier Ensembles for Changing Environments*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3077(June 2004):V–VI, 2004, ISSN 16113349. 12

[41] Gama, João, Raquel Sebastião e Pedro Pereira Rodrigues: *Issues in evaluation of stream learning algorithms*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, (June 2014):329–337, 2009. 12

[42] Gama, João, Pedro Medas, Gladys Castillo e Pedro Rodrigues: *Learning with drift detection*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 3171(May 2014):286–295, 2004, ISSN 16113349. 12, 13

[43] Lichtenwalter, Ryan N. e Nitesh V. Chawla: *Adaptive methods for classification in arbitrarily imbalanced and drifting data streams*. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 5669 LNAI:53–75, 2010, ISSN 03029743. 14, 15

[44] Wang, Shuo, Leandro L. Minku e Xin Yao: *Resampling-based ensemble methods for online class imbalance learning*. IEEE Transactions on Knowledge and Data Engineering, 27(5):1356–1368, 2015, ISSN 10414347. 14, 15

[45] Wang, Shuo, Leandro L. Minku e Xin Yao: *A learning framework for online class imbalance learning*. Em *2013 IEEE Symposium on Computational Intelligence and Ensemble Learning (CIEL)*, páginas 36–45, 2013. 15

[46] Hoens, Thomas Ryan e Nitesh V. Chawla: *Learning in non-stationary environments with class imbalance*. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, páginas 168–176, 2012. 15

[47] Hoens, T. Ryan, Nitesh V. Chawla e Robi Polikar: *Heuristic Updatable Weighted Random Subspaces for non-stationary environments*. Proceedings - IEEE International Conference on Data Mining, ICDM, páginas 241–250, 2011, ISSN 15504786. 15

[48] Moulton, Richard Hugh, Herna L. Viktor, Nathalie Japkowicz e João Gama: *Contextual one-class classification in data streams*, 2019. 15, 36

[49] Krawczyk, Bartosz e Michał Woźniak: *One-class classifiers with incremental learning and forgetting for data streams with concept drift*. Soft Computing, 19:3387–3400, dezembro 2015, ISSN 14337479. 16, 23, 25

[50] Khan, Shehroz S. e Michael G. Madden: *One-class classification: taxonomy of study and review of techniques*. The Knowledge Engineering Review, 29(3):345–374, 2014. 16, 35

[51] Kurlej, Bartosz e Michal Wozniak: *Active learning approach to concept drift problem*. Logic Journal of the IGPL, 20(3):550–559, fevereiro 2011, ISSN 1367-0751. `https://doi.org/10.1093/jigpal/jzr011`. 16

[52] Liu, Fei Tony, Kai Ming Ting e Zhi Hua Zhou: *Isolation forest*. Em *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, páginas 413–422, 2008. 16, 35

[53] Breunig, Markus M., Hans Peter Kriegel, Raymond T. Ng e Jörg Sander: *Lof: Identifying density-based local outliers*. Em *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, página 93–104, New York, NY, USA, 2000. Association for Computing Machinery, ISBN 1581132174. `https://doi.org/10.1145/342009.335388`. 17, 35

[54] Liashchynskyi, Petro e Pavlo Liashchynskyi: *Grid search, random search, genetic algorithm: A big comparison for nas*, 2019. 28

[55] Howard, S. Francis: *The elliptical envelope*. arXiv: Differential Geometry, 2007. 35

[56] Breiman, Leo: *Random Forests*. Machine Learning, 45:5–32, 2001. 35

[57] Vakili, Meysam, Mohammad Ghamsari e Masoumeh Rezaei: *Performance Analysis and Comparison of Machine and Deep Learning Algorithms for IoT Data Classification.* 2020. `http://arxiv.org/abs/2001.09636`. 36

[58] Rossi, André: *Meta-aprendizado aplicado a fluxos contínuos de dados.* Tese de Doutoramento, Universidade de São Paulo, 2014. 36

[59] Peccatiello, Rafael Bruno, João José Costa Gondim e Luís Paulo Faina Garcia: *Applying one-class algorithms for data stream-based insider threat detection.* IEEE Access, 11:70560–70573, 2023. 49