UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FRANCISCO RODRIGO PARENTE DA PONTE

FRAPE: A FRAMEWORK FOR RISK ASSESSMENT, PRIORITIZATION AND EXPLAINABILITY OF VULNERABILITIES

FORTALEZA

2023

FRANCISCO RODRIGO PARENTE DA PONTE

FRAPE: A FRAMEWORK FOR RISK ASSESSMENT, PRIORITIZATION AND EXPLAINABILITY OF VULNERABILITIES

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Redes de Computadores

Orientador: Prof. Dr. Emanuel Bezerra Rodrigues

Coorientador: Prof. Dr. César Lincoln Cavalcante Mattos

FORTALEZA

2023

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Sistema de Bibliotecas
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

P857f Ponte, Francisco Rodrigo Parente da.
FRAPE: A Framework for Risk Assessment, Prioritization and Explainability of Vulnerabilities / Francisco Rodrigo Parente da Ponte. – 2023.
179 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2023.
Orientação: Prof. Dr. Emanuel Bezerra Rodrigues.
Coorientação: Prof. Dr. César Lincoln Cavalcante Mattos.

1. Cibersegurança. 2. Gestão de Risco. 3. Aprendizado Ativo. 4. Aprendizado de Máquina. I. Título.
CDD 005

FRANCISCO RODRIGO PARENTE DA PONTE

FRAPE: A FRAMEWORK FOR RISK ASSESSMENT, PRIORITIZATION AND EXPLAINABILITY OF VULNERABILITIES

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Redes de Computadores

Aprovada em: 13 de Outubro de 2023

BANCA EXAMINADORA

Prof. Dr. Emanuel Bezerra Rodrigues (Orientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. César Lincoln Cavalcante Mattos (Coorientador)
Universidade Federal do Ceará (UFC)

Prof. Dr. João Paulo Pordeus Gomes
Universidade Federal do Ceará (UFC)

Prof. Dr. Rafael Lopes Gomes
Universidade Estadual do Ceará (UECE)

Dr. Renato Rodrigues Marinho
Pesquisador Chefe na Accenture Brasil

Prof. Dr. Paulo Romero Martins Maciel
Universidade Federal de Pernambuco (UFPE)

À minha família, por sempre acreditar em mim nessa longa caminhada de aprendizado e por todos os sacrifícios que fizeram para que eu pudesse estar aqui. Sem o amor e o apoio de vocês nada disso teria sido possível.

AGRADECIMENTOS

Aos meus pais, Raimunda Suely e Benedito Machado, por todo amor, carinho e apoio incondicional. Espero deixá-los orgulhosos e retribuir tudo o que fizeram por mim.

À minha companheira, Sayonara Santos, por estar ao meu lado, sempre me apoiando e incentivando. Seu amor e afeto foram determinantes nessa jornada.

Aos meus amigos de Pós-Graduação, Luciano Martins e Belmondo Rodrigues, pela parceria nos estudos e pela amizade que cultivamos no processo.

A todos os meus amigos. Em especial, Cláudio Victor e Rafael Araújo, que se fizeram presentes, mesmo nos momentos de minha ausência dedicados aos estudos.

Aos que colaboraram de alguma forma com este trabalho. Em especial, Joel Teixeira, Juan Veloso e Anderson Bezerra pelo apoio na rotulação das vulnerabilidades.

Aos meus orientadores, Emanuel Bezerra e César Lincoln, pelos ensinamentos, aconselhamentos e confiança durante esse período de formação.

A todos os professores que compõem o MDCC pelos conhecimentos trocados e aos membros da secretaria do programa, pela prontidão e atenção que sempre tiveram.

Aos membros da banca, professores João Paulo, Rafael Lopes, Renato Marinho e Paulo Romero, por terem aceitado o convite e pelas colaborações valorosas dadas ao trabalho.

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001.

RESUMO

Práticas inadequadas de segurança, como o uso de métricas únicas, por exemplo, considerar apenas o Sistema Comum de Pontuação de Vulnerabilidades – *Common Vulnerability Scoring System* (CVSS) – no processo de Gestão de Vulnerabilidades – *Vulnerability Management* (VM) –, podem causar a superestimação do risco de exploração dos ativos. Idealmente, os analistas devem usar informações sobre a vulnerabilidade, inteligência de ameaças e contexto para avaliar a probabilidade e o risco de exploração de falhas de segurança. A falta de ferramentas especializadas torna essa tarefa complexa e passível de erros, pois os analistas precisam correlacionar manualmente as informações de diversas fontes de segurança com os milhares de ativos presentes na organização. Embora o Aprendizado de Máquina – *Machine Learning* (ML) – possa auxiliar nessa tarefa, sua aplicação na área de VM tem sido pouco explorada na literatura. Diante deste contexto, essa tese propõe o *FRAPE*, um *framework* de Gestão de Vulnerabilidades Baseada no Risco – *Risk-Based Vulnerability Management* (RBVM) – que utiliza uma técnica de rotulação de dados chamada de Aprendizado Ativo – *Active Learning* (AL) – em conjunto com a técnica de aprendizado supervisionado para criar um modelo de ML capaz de emular a experiência de especialistas de segurança na análise e avaliação do risco de exploração das vulnerabilidades. *FRAPE* é composto por 4 módulos que são: (*i*) Coleta de Dados, responsável por agregar as informações necessárias para a avaliação do risco; (*ii*) Rotulação das Vulnerabilidades, onde o aprendizado ativo será utilizado para rotular as vulnerabilidades com as características mais significativas; (*iii*) Classificação e Priorização das Vulnerabilidades, onde as falhas de segurança serão classificados e consequentemente, priorizadas para correção considerando os seus riscos; e por fim, (*iv*) Interpretação dos Resultados, onde oferecemos uma visão detalhada do porquê as vulnerabilidades foram selecionadas. Assim, este trabalho desenvolverá uma solução que consiga auxiliar os analistas de segurança a identificar as vulnerabilidades mais críticas da empresa e com isso, possam se defender de potenciais ataques de usuários mal-intencionados.

Palavras-chave: cibersegurança; gestão de risco; aprendizado ativo; aprendizado de máquina.

ABSTRACT

Inadequate security practices, such as using single metrics, for instance, considering only the Common Vulnerability Scoring System (CVSS) in the Vulnerability Management (VM) process, can lead to an overestimation of the risk of asset exploitation. Ideally, security analysts should use vulnerability information, threat intelligence, and context to assess the likelihood and risk of exploiting security flaws. The lack of specialized tools makes this task complex and error-prone, as analysts must manually correlate information from multiple security sources with the thousands of assets present in the organization. Although Machine Learning (ML) can help in this task, researchers haven't thoroughly explored its application in the VM process. Given this context, this thesis proposes FRAPE, a Risk-Based Vulnerability Management (RBVM) framework. FRAPE uses a data labeling technique called Active Learning (AL) combined with a Supervised Learning approach to create an ML model capable of emulating the experience of security experts in analyzing and assessing the risk of exploiting vulnerabilities. FRAPE is composed of 4 modules which are: (i) Data Collection, responsible for aggregating the necessary information for risk assessment; (ii) Vulnerability Labeling, where active learning is used to label vulnerabilities with the most significant characteristics; (iii) Classification and Prioritization of Vulnerabilities, where security flaws will be classified and consequently prioritized for correction considering their risks; and finally, (iv) Results Interpretation, where we provide a detailed analysis of why the vulnerabilities were considered critical. Thus, this work seeks to develop a solution capable of helping security analysts identify the most critical vulnerabilities so that they can defend themselves from potential attacks by malicious users.

Keywords: cybersecurity; risk assessment; active learning; machine learning.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AI	*Artificial Intelligence*
AL	*Active Learning*
ANN	*Artificial Neural Network*
API	*Application Programming Interface*
AUC	*Area Under the Curve*
CAPES	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
CCE	*Common Configuration Enumeration*
CD	*Critical Distance*
CENAPAD	Centro Nacional de Processamento de Alto Desempenho
CIA	*Confidentiality, Integrity and Availability*
CIS	*Center for Internet Security*
CMDB	*Configuration Management Database*
CPE	*Common Platform Enumeration*
CVE	*Common Vulnerabilities and Exposures*
CVSS	*Common Vulnerability Scoring System*
CWE	*Common Weakness Enumeration*
DDoS	*Distributed Denial-of-Service*
DL	*Deep Learning*
DM	*Data Mining*
DMZ	*Demilitarized Zone*
DoS	*Denial-of-Service*
DT	*Decision Tree*
ECE	*Expected Calibration Error*
EPSS	*Exploit Prediction Scoring System*
ES	*Explorability Score*
FIRST	*Forum of Incident Response and Security Teams*
FN	*False Negatives*
FP	*False Positives*
FPR	*False Positive Rate*

FRAPE	*Framework for Risk Assessment, Prioritization and Explainability of Vulnerabilities*
GB	*Gradient Boosting*
HTTPS	*Hyper Text Transfer Protocol Secure*
IDS	*Intrusion Detection System*
InfoSec	*Information Security*
INPI	Instituto Nacional de Propriedade Industrial
IoC	*Indicator of Compromise*
IS	*Impact Score*
KL	*Kullback-Leibler Divergence*
LC	*Least Confidence*
LR	*Logistic Regression*
MISP	*Malware Information Sharing Platform*
MiTM	*Man-in-the-Middle*
ML	*Machine Learning*
MLOps	*Machine Learning Operations*
MLP	*Multilayer Perceptron*
MVP	*Minimum Viable Product*
NIST	*National Institute of Standards and Technology*
NVD	*National Vulnerability Database*
OWASP	*Open Web Application Security Project*
PAD	Processamento de Alto Desempenho
PCA	*Principal Component Analysis*
PoC	*Proofs-of-Concept*
RAT	*Remote Access Tool*
RBVM	*Risk-Based Vulnerability Management*
RF	*Random Forest*
ROC	*Receiver Operating Characteristic*
SCAP	*Security Content Automation Protocol*
SH	*Smart Homes*
SHAP	*Shapley Additive Explanations*
SIEM	*Security Information and Event Management*

SINAPAD	Sistema Nacional de Processamento de Alto Desempenho
SMS	*Short Message Service*
SOC	*Security Operations Center*
SQL	*Structured Query Language*
SQLi	*SQL injection*
SVC	*Support Vector Classification*
SVM	*Support Vector Machine*
TN	*True Negative*
TP	*True Positive*
TPR	*True Positive Rate*
TTPs	*Tactics, Techniques and Procedures*
UFC	Universidade Federal do Ceará
UML	*Unified Modeling Language*
UTM	*Unified Threat Management*
VM	*Vulnerability Management*
VPN	*Virtual Private Network*
VPNs	*Virtual Private Networks*
WAP	*Wi-Fi Protected Access*
XAI	*Explainable Artificial Intelligence*
XP	*Extreme Programming*
XSS	*Cross-Site Scripting*

SUMÁRIO

1 INTRODUÇÃO

Segundo a Base de Vulnerabilidades Nacional – *National Vulnerability Database* (NVD) – mantida pelo Instituto Nacional de Padrões e Tecnologia – *National Institute of Standards and Technology* (NIST) – e amplamente utilizada por profissionais de segurança, nos últimos cinco anos, o número de vulnerabilidades únicas reportadas aumentou em 195% (NIST, 2021). Além disso, um estudo publicado pela empresa de segurança Skybox mostrou que no ano de 2020 a quantidade de *ransomwares* cresceu 106% e de *trojans* 128% (SKYBOX SECURITY, 2021).

Aliado a isso, as infraestruturas de grandes organizações estão ficando cada vez mais complexas e podem conter, em média, 80 mil ativos, incluindo *notebooks*, servidores, roteadores e *smartphones* conectados à internet (JASON, 2020). Um estudo feito pela empresa de segurança *Kenna* mostrou que as redes das organizações possuem aproximadamente 40 milhões de vulnerabilidades e que, em média, somente uma em cada dez serão remediadas (KENNA SECURITY; CYENTIA INSTITUTE, 2019).

Assim, dado o grande volume de falhas que podem ser exploradas e a escassez de mão de obra especializada (FURNELL *et al.*, 2017), fica claro que os analistas de segurança precisam adotar estratégias que os ajudem a identificar e priorizar a correção das vulnerabilidades mais críticas. Evitando assim, que cibercriminosos explorem essas fraquezas, causando danos à infraestrutura e sistemas das organizações.

Tradicionalmente, os analistas empregam a disciplina de Gestão de Vulnerabilidades – *Vulnerability Management* (VM) – para analisar e corrigir as falhas de segurança. VM é uma prática cíclica que pode ser utilizada para identificar, classificar, corrigir e mitigar vulnerabilidades (MELL *et al.*, 2005). Essa atividade é essencial, pois permite melhorar as defesas da organização através da identificação de sistemas, processos e estratégias vulneráveis.

Embora não seja uma disciplina nova, VM é considerada imatura dada a pequena quantidade existente de ferramentas robustas e prontas para uso organizacional. Além disso, segundo Foreman (2019), os usuários dessa disciplina são ingênuos, por não reconhecerem quão importante é que essas soluções sejam integradas com processos de gestão organizacionais bem definidos, que os auxiliem no processo de "endurecimento" das defesas de segurança.

Outras ferramentas que têm sido bastante utilizadas na área de Segurança da Informação – *Information Security* (InfoSec) – são as técnicas de Inteligência Artificial – *Artificial Intelligence* (AI) –, Aprendizado de Máquina – *Machine Learning* (ML) – e Mineração de Dados

– *Data Mining* (DM) (HANDA *et al.*, 2019). Em especial, ML tem sido aplicado com sucesso por pesquisadores para resolver diversos problemas de segurança, dada sua alta eficácia e baixo tempo de resposta (GELUVARAJ *et al.*, 2019). Além disso, ML pode mitigar o problema da escassez de mão de obra qualificada (SHAUKAT *et al.*, 2020).

Entre as contribuições de ML apresentadas na literatura, podemos citar os sistemas de detecção de intrusão (GHARAEE; HOSSEINVAND, 2016; SHAPOORIFARD; SHAMSINEJAD, 2017), de *spam* (SHAH; KUMAR, 2018; CHANDRASEKAR; PRIYATHARSINI, 2018) e de *malware* (SUN *et al.*, 2018; AFEK *et al.*, 2019). No entanto, o uso de ML para solucionar problemas de VM ainda não foi completamente explorado.

1.1 Motivação e Justificativa do Trabalho

Durante o processo de VM, é comum que os analistas utilizem o Sistema Comum de Pontuação de Vulnerabilidades – *Common Vulnerability Scoring System* (CVSS) – como métrica para priorizar a correção das falhas de segurança. O CVSS é um *framework* de código aberto que define a severidade das vulnerabilidades através de suas características (FIRST, 2019). Cada vulnerabilidade publicada no NVD, recebe uma nota de zero a dez e pode ser classificada em um das quatro classes existentes de severidade, que são: *LOW*, *MEDIUM*, *HIGH* e *CRITICAL*. Assim, os analistas costumam focar seus esforços na correção das vulnerabilidades com notas de CVSS mais altas, ou seja, aquelas com severidade *CRITICAL* (BAKER, 2022).

Essa é uma estratégia equivocada, pois pesquisas apontam que a maioria das vulnerabilidades exploradas não são consideradas críticas, como exemplo, podemos citar o estudo publicado pela empresa de consultoria *RecordFuture* que mostra que apenas 4 das 10 vulnerabilidades mais exploradas em 2020 tinham CVSS *CRITICAL* (RecordFuture, 2021). De fato, o CVSS é amplamente mal utilizado, pois foi desenvolvido para comunicar a severidade das vulnerabilidades, mas os analistas o usam como um indicador de risco (SPRING *et al.*, 2021).

A avaliação do risco é um conceito importante na cibersegurança, sendo inclusive um dos pontos centrais da ISO 27001, padrão internacionalmente utilizado para a implementação, manutenção e melhoria contínua de sistemas de gestão de segurança da informação (ISO Central Secretary, 2022). Essa norma estabelece um conjunto de requisitos, processos e controles que buscam proteger a confidencialidade, integridade e disponibilidade das informações em uma organização. É importante então definir, que no contexto de cibersegurança, o risco de exploração de uma vulnerabilidade é dado pela probabilidade de um evento adverso de segurança acontecer

(por exemplo, uma campanha de *malware* em andamento) e seu impacto na organização (por exemplo, o vazamento de informações sensíveis) (ANDRESS, 2019).

Assim, é possível perceber que o risco é algo que o CVSS não pode estimar, por desconsiderar informações além das características das vulnerabilidades. De fato, pesquisadores afirmam que políticas de remediação de vulnerabilidades baseadas em uma única métrica, como aquelas que utilizam apenas o CVSS, não são ideais (DEY *et al.*, 2015). Adicionalmente, de acordo com Spring *et al.* (2021), as vulnerabilidades não devem ser analisadas isoladamente, mas no contexto da organização e do seu risco de exploração, o que os especialistas chamam de inteligência de ameaças.

Dessa problemática surgiu a disciplina de Gestão de Vulnerabilidades Baseada no Risco – *Risk-Based Vulnerability Management* (RBVM) – (JASON, 2020; TENNABLE, 2020a; CEZARINA, 2021), uma estratégia de VM que visa priorizar a correção de vulnerabilidades conforme o risco que elas apresentam para a organização e não somente quanto a sua severidade. Isso é feito considerando três importantes conjuntos de informações, sendo: (*i*) o uso das características intrínsecas das vulnerabilidades para estimar a severidade e o impacto que a sua exploração terá na organização; (*ii*) o emprego da disciplina de inteligência de ameaças para identificar o que os atacantes estão discutindo, para se avaliar a probabilidade de exploração de uma determinada vulnerabilidade (discutido em detalhes na Subseção 2.5.1); e (*iii*) a identificação do contexto do negócio, para definir a criticidade dos ativos, uma vez que a invasão de certos segmentos de rede e ativos pode ser mais prejudicial que outros (discutido na Subseção 2.5.2).

Na prática, para realizar a RBVM, os analistas devem correlacionar informações de diversas fontes de segurança, com os ativos presentes na organização e sua experiência profissional (MEHRI *et al.*, 2021). Essa atividade é difícil e demorada, pois como dito anteriormente, as redes das empresas estão ficando mais complexas e existe uma falta de mão de obra qualificada no mercado. Além disso, essa tarefa costuma ser realizada manualmente pelos analistas, o que a torna propensa a erros causados, por exemplo, pela fadiga (ELBAZ *et al.*, 2021). Vale destacar que existem algumas soluções proprietárias de RBVM (discutidas em detalhes na Subseção 4.3.2) que podem auxiliar no processo, no entanto, elas são pouco transparentes em relação a como o risco é calculado, não consideram a experiência dos analistas e possuem um custo alto de contratação[1].

[1] O custo de contratação de uma solução de RBVM é feito pela quantidade de ativos gerenciados e pode facilmente chegar à dezena de milhares de dólares. Por exemplo, a solução desenvolvida pela empresa *Tenable* custa $50 mil dólares por apenas 250 ativos: https://www.tenable.com/buy.

Fica claro, que a RBVM desempenha um papel importante, pois pode sustar ameaças de segurança. Assim, ao fornecer ferramentas que auxiliem os analistas a remediar as falhas que apresentam maiores riscos, não só diminuímos as vulnerabilidades de segurança, como também poupamos tempo dos analistas e dinheiro das organizações. Por fim, é importante que essas soluções ofereçam um processo de gestão contínuo e considere fatores internos e externos na avaliação de risco, como define o guia de boas práticas do Centro de Segurança para Internet – *Center for Internet Security* (CIS) – (CIS, 2021).

Após uma extensa pesquisa bibliográfica e um estudo detalhado dos trabalhos relacionados (apresentado no Capítulo 4), concluímos que nenhuma das soluções encontradas utiliza simultaneamente informações sobre a vulnerabilidade, inteligência de ameaças e de contexto para avaliar o risco. Além disso, as soluções são complexas de se utilizar e muitas vezes, necessitam da interação constante com o analista, algo contraprodutivo. Adicionalmente, essas soluções desconsideram a competência dos especialistas, o que acreditamos ser indispensável e que poderia ser alcançado através do uso de técnicas de ML capazes de emular a experiência de profissionais de segurança no processo de avaliação das vulnerabilidades.

1.2 Objetivos

Diante do contexto de cibersegurança apresentado e dos desafios em torno do processo de RBVM determinamos os seguintes objetivos deste trabalho.

1.2.1 Objetivo Geral

Desenvolver um *framework* de segurança, que auxilie os analistas no processo de RBVM. Esse *framework* deve utilizar informações sobre as características da vulnerabilidade, inteligência de ameaças e o contexto onde a falha está inserida, para avaliar o risco de sua exploração e o impacto que isso trará para a organização. Além disso, é imprescindível que a experiência profissional do analista seja considerada durante o processo.

Para alcançar tal objetivo, este trabalho propõe uma solução baseada em ML capaz de avaliar o risco de exploração das vulnerabilidades, equivalente à intuição e experiência dos profissionais de segurança. Isto é feito através do uso de uma técnica de ML chamada de Aprendizado Ativo – *Active Learning* (AL) – (detalhada na Seção 3.2), capaz de consultar interativamente o analista durante o processo de treinamento do modelo de ML (SETTLES,

2009).

1.2.2 Objetivos Específicos

- Considerar múltiplas bases de dados, internas e externas à organização, na coleta das informações de segurança;
- Oferecer uma metodologia capaz de classificar e priorizar as vulnerabilidades de acordo com sua probabilidade de exploração e impacto na infraestrutura da organização;
- Desenvolver um *framework* de RBVM que considere a experiência dos analistas de segurança e torne o processo de gestão de vulnerabilidades mais rápido e transparente.

1.3 Metodologia

Para atingir os objetivos apresentados na Seção 1.2, adotamos para esta tese uma metodologia dividida em três etapas, que são:

- **Concepção:** onde definimos os objetos de estudo deste trabalho, que são as técnicas e *frameworks* que auxiliam os analistas no processo de classificação do risco das vulnerabilidades de segurança. A seguir, definimos as "*strings* de busca" e bases de pesquisa, para, então, realizar uma abrangente revisão da literatura. Por fim, realizamos uma extensa análise dos artigos e *whitepapers* encontrados, a fim de identificar e selecionar aqueles mais pertinentes para o desenvolvimento deste trabalho;
- **Execução:** nessa etapa, definimos os requisitos e as especificações do *framework*, bem como os algoritmos de processamento de dados e os modelos de aprendizado de máquina utilizados no desenvolvimento da tese. Em seguida, implementamos o *framework* utilizando a linguagem de programação *Python* e bibliotecas de código aberto que nos auxiliaram na tarefa de processamento e análise de dados;
- **Avaliação:** na última etapa, primeiro realizamos um planejamento das avaliações necessárias, e então analisamos o desempenho das diferentes técnicas de aprendizado de máquina que podem ser utilizadas pelo *framework*; por fim comparamos a metodologia de RBVM proposta nesta tese em relação à priorização padrão utilizada por analistas, que utiliza apenas a nota de CVSS na classificação de risco das vulnerabilidades.

1.4 Estrutura do Trabalho

O restante da tese está estruturada da seguinte forma. Nos Capítulos 2 e 3 apresentamos o embasamento teórico necessário para a melhor compreensão da nossa proposta, isto é, os principais conceitos sobre cibersegurança e ML. No Capítulo 4 descrevemos as pesquisas mais relevantes encontradas que abordam a construção de *datasets* de segurança e o tema de RBVM, bem como suas diferenças em relação ao proposto nesta tese. No Capítulo 5 descrevemos a proposta deste trabalho, que corresponde a um *framework* para avaliação de risco de vulnerabilidades de segurança da informação, chamado de *Framework for Risk Assessment, Prioritization and Explainability of Vulnerabilities* (FRAPE). Nos Capítulos 6, 7 e 8 descrevemos os resultados obtidos, que envolvem a criação de um conjunto de dados de segurança, uma metodologia de classificação de risco das vulnerabilidades e um simulador de redes que pode ser utilizado para comparar diferentes técnicas de VM. Por fim, no Capítulo 9 concluímos esta tese com as considerações finais, uma síntese dos resultados obtidos e os trabalhos futuros.

2 CIBERSEGURANÇA

Em 2020, o número estimado de dispositivos conectados à internet foi de 31 bilhões e projeções sugerem que esse número pode chegar a 75 bilhões no ano de 2025 (GEORGIEV, 2021). Com o aumento no número de dispositivos conectados, aumenta-se também o número de pontos de falhas e consequentemente o número de possíveis ataques. Estima-se que o custo desses ataques cresça em 15% ao ano e que em 2025, as empresas acabem gastando cerca de $10.5 trilhões de dólares americanos devido a eles (MORGAN, 2020).

Dessa forma, providências devem ser tomadas para proteger as empresas, os usuários e seus dispositivos contra esses ataques. Assim, surgiu a disciplina de cibersegurança, que é o conjunto de medidas e mecanismos de defesa capazes de proteger computadores, dispositivos móveis, servidores e dados contra ataques maliciosos (KARPERSKY, 2020).

Para sumarizar, podemos dizer que o principal foco da cibersegurança é fazer com que os dados e sistemas sejam confiáveis, íntegros e estejam sempre disponíveis (THOMAS *et al.*, 2020). Assim, Confidencialidade, Integridade e Disponibilidade – *Confidentiality, Integrity and Availability* (CIA) – são os três principais pilares da cibersegurança.

2.1 Vulnerabilidade x Ameaça x Risco

Antes de nos aprofundarmos no tema de cibersegurança, é importante definir o significado das palavras vulnerabilidade, ameaça e risco no contexto de segurança. Visto que, elas são frequentemente utilizadas erradamente, dificultando a construção de políticas eficazes de segurança, na comunicação com membros do time e no entendimento de ferramentas de gestão de vulnerabilidades.

Uma vulnerabilidade é uma fraqueza na infraestrutura, rede ou *software*, que potencialmente expõem os dispositivos e sistemas a ameaças de segurança (GLOVER, 2020). As vulnerabilidades podem ser físicas (e.g. um dispositivo de rede exposto publicamente à internet), de *software* (e.g. o estouro da memória temporária – *buffer* – em navegadores), ou mesmo humanas (e.g. falhas de configuração de um *software*). Assim, uma organização pode conter milhões de vulnerabilidades e corrigir todas elas não é uma tarefa factível, dada a velocidade com que novas vulnerabilidades surgem e o time reduzido de especialistas (KENNA SECURITY; CYENTIA INSTITUTE, 2019).

Ameaça é qualquer tipo de perigo que possa explorar uma vulnerabilidade e a

partir dela, consiga infectar a rede ou os dispositivos das organizações, com o intuito de causar algum malefício (LIFARS, 2020). As ameaças são praticadas por indivíduos ou grupos de indivíduos mal-intencionados com várias origens e motivações. No geral, essas pessoas procuram criar interrupções, danificar ou roubar dados das empresas, visando o enriquecimento próprio. Exemplos comuns de ameaças incluem *malwares*, *phishing*, violações de dados sensíveis e até mesmo funcionários desonestos. Algumas ameaças possuem maior potencial de exploração e podem causar danos mais graves que outras, por isso, devem receber uma atenção maior durante o processo de gestão e remediação.

Por fim, risco corresponde ao impacto que pode ser causado a um ativo ou a infraestrutura de uma organização, dada a probabilidade de um evento de segurança ocorrer, por exemplo, a exploração de uma vulnerabilidade (ALEXANDER, 2021). Cada organização possui um perfil diferente de risco determinados por fatores internos (e.g. a disposição dos ativos na rede) e externos (e.g. a existência de um *exploit* público). E embora o risco não possa ser completamente eliminado, os profissionais de segurança devem mantê-lo em um nível baixo e conhecido. Para auxiliar os analistas a gerenciar os riscos de segurança, foram desenvolvidas as ferramentas de VM, que serão abordadas em detalhes na Seção 2.5.

2.2 Ataques cibernéticos

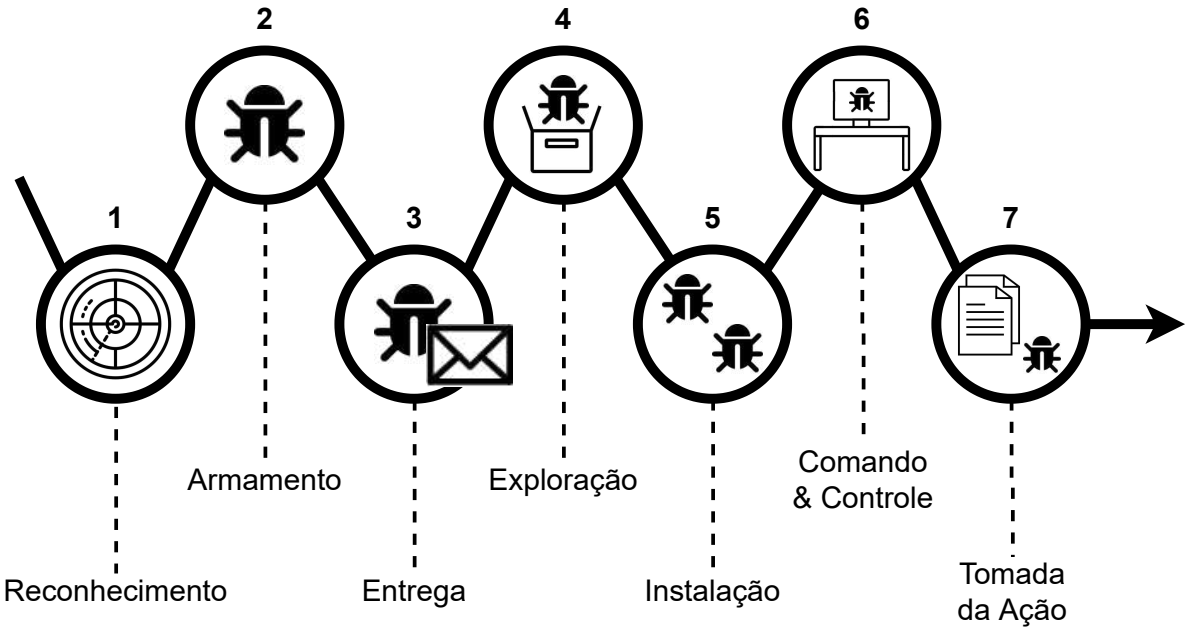
Um ciberataque é qualquer tipo de ação ofensiva que tem como alvo infraestruturas, redes de computadores, sistemas ou dispositivos pessoais, utilizando diversas técnicas para destruir, interromper ou controlá-los (HATHAWAY *et al.*, 2012). Além disso, eles permitem roubar e destruir a integridade de informações sensíveis e sigilosas (NIST, 2012). Esses ataques têm se tornado frequentes, sofisticados e apesar de os profissionais de segurança estarem desempenhando bem seus papéis na defesa das organizações, os atacantes estão ficando cada vez melhores na prática do ataque (MADNICK, 2017).

Não existe uma forma garantida de proteger as organizações de todos os ataques cibernéticos, mas existem várias ferramentas (como as que serão abordadas na Seção 2.5) e práticas de segurança que podem ajudar a reduzir o risco de ataques. Entre as práticas de segurança que devem ser tomadas, compreender as fases, os tipos e as diferentes técnicas empregadas pelos atacantes nos ataques é fundamental para que os profissionais de segurança possam desenvolver processos eficazes de segurança. Por isso, abordaremos as diferentes fases de um ataque (Subseção 2.2.1) e os ataques de cibersegurança mais comuns (Subseção 2.2.2).

2.2.1 Killchain

O termo *kill chain* é um conceito militar que visa descrever a cadeia de um ataque, que vai desde a identificação até a destruição do alvo (GREENERT, 2013). Esse conceito foi adaptado para o contexto de cibersegurança e gerou um *framework* composto por 7 fases, que definem as etapas de um ataque em uma rede de computadores (MARTIN, 2015). Segundo os autores Yadav e Rao (2015), o *killchain* é um modelo para respostas de incidentes, que permite que profissionais de segurança saibam como reagir a um ataque independente de seu estado na rede. A seguir, abordaremos em detalhes cada uma das fases do *killchain*, representadas na Figura 1.

Figura 1 – Geralmente, os ataques cibernéticos podem ser classificados em sete diferentes estágios. Essa cadeia de eventos é conhecida como *killchain* e auxiliam os profissionais de segurança a identificar e se proteger de ataques.



Fonte: elaborado pelo autor.

A primeira fase, chamada de reconhecimento, do inglês *reconnaissance*, é a fase onde o atacante reúne todas as informações possíveis sobre um determinado alvo, que pode ser um indivíduo ou uma organização (DARGAHI *et al.*, 2019). O reconhecimento é feito por buscas realizadas em sites, listas de *e-mails*, conferências, blogs, redes sociais, etc. Nessa fase, o atacante busca identificar vulnerabilidades que possam ser exploradas para atacar seus alvos.

A segunda fase, chamada de armamento, do inglês *weaponize*, é a fase onde o atacante desenvolve um *malware* capaz de burlar procedimentos de autenticação e autorização, garantindo assim, acesso indevido à rede do alvo. Isso é feito utilizando as informações coletadas

na fase de reconhecimento e o principal objetivo do atacante é instalar programas maliciosos que lhe garantam acesso remoto à rede. Esses programas são conhecidos como Ferramentas de Acesso Remoto – *Remote Access Tool* (RAT) – e utilizam vulnerabilidades dos sistemas ou *softwares* para executar (KONDALWAR; SHELKE, 2014).

A terceira fase, chamada de entrega, do inglês *delivery*, é uma fase importante e define se o ataque será bem-sucedido ou não. Essa fase trata de como será efetuada a entrega e execução do *malware*. O atacante utiliza as informações coletadas na fase de reconhecimento para desenvolver formas de entrega (e.g. um anexo malicioso em um *e-mail* falso) que possuem maiores chances de serem executadas. Note que, geralmente, a ação do usuário é necessária para a execução desses programas (UCCI *et al.*, 2019).

A quarta fase, chamada de exploração, do inglês *exploitation*, é a fase onde o *malware* explora uma falha de segurança para instalar e executar silenciosamente os programas que garantem ao atacante o acesso remoto à rede (KONDALWAR; SHELKE, 2014). Normalmente, essa fase necessita da interação com o usuário e ocorre após a fase de entrega. O intuito é garantir o mínimo de acesso necessário para o atacante invadir o ambiente alvo.

A quinta fase, chamada de instalação, do inglês *installation*, é a etapa onde o atacante tenta expandir seu acesso a mais nós (i.e. ativos) da rede, para persistir sua presença no ambiente (DARGAHI *et al.*, 2019). Isso é feito por meio de duas técnicas chamadas de: (*i*) escalação de privilégios, onde o atacante tenta obter mais privilégios, frequentemente buscando acesso a contas de administradores; e (*ii*) movimentação lateral, onde os atacantes se movem lateralmente para outros computadores, sistemas e contas, com o intuito de conseguir mais dados, acessos e privilégios.

A sexta fase, chamada de comando & controle, do inglês *command & control* (ou simplesmente C2), é a fase onde o atacante se sente confortável para montar uma "base de operações" na rede alvo. Nesse ponto do ataque, o cibercriminoso está estabelecido e pode começar a emitir comandos aos *malwares* instalados para causar dano às máquinas, encriptar arquivos ou fazer o *download* de dados sensíveis do ambiente (DARGAHI *et al.*, 2019).

A sétima e última fase, chamada de tomada da ação, do inglês *act on objective*, é a fase onde o atacante executa as ações necessárias para alcançar seus objetivos. Entre eles, podemos citar a negação de serviços ou o roubo de informações sigilosas (DARGAHI *et al.*, 2019). Ao concluir o ataque e antes de deixar o ambiente, o atacante costuma encobrir seus rastros, para evitar que a invasão seja descoberta. Além disso, é comum que deixem "*backdoors*"

que os permitem voltar à rede no futuro caso seja necessário.

É importante destacar que o *killchain* não é um *framework* linear e progressivo, podendo o atacante estar em mais de uma fase simultaneamente. Por exemplo, ele pode já ter conseguido acesso à rede interna do alvo e com as informações coletadas esteja desenvolvendo um novo *malware* que será lançado em uma nova frente de ataque. Assim, o *killchain* funciona como um guia que visa fornecer uma ideia do avanço do atacante e do que pode ser feito para se defender de seu ataque.

2.2.2 Tipos de Ataques

Ciberataques acontecem a todo momento e como disse o ex-CEO da empresa Cisco[1] John Chambers: "Existem dois tipos de empresas: as que foram *hackeadas* e as que ainda não sabem que foram *hackeadas*" (CISCO, 2018). Há uma abundância de ataques que podem ser realizados, no entanto, para fins de referencial teórico, descreveremos a seguir os 7 principais ataques perpetrados por usuários mal-intencionados contra sistemas de computadores (HUMAYUN *et al.*, 2020).

Ataques de *zero-day* ocorrem quando cibercriminosos exploram uma vulnerabilidade, seja em um *software* ou sistema operacional, antes que sua existência venha a público e possa ser corrigida mediante um *patch* de segurança (ZOPPI *et al.*, 2021). Em muitos casos a vulnerabilidade pode passar despercebida por meses e os desenvolvedores só ficarem sabendo dela após a ocorrência de um incidente. Para se prevenir desse tipo de ataque é necessário verificar os *softwares* por vulnerabilidades e realizar auditorias de segurança, que possam detectar falhas através da simulação de cenários de ataque.

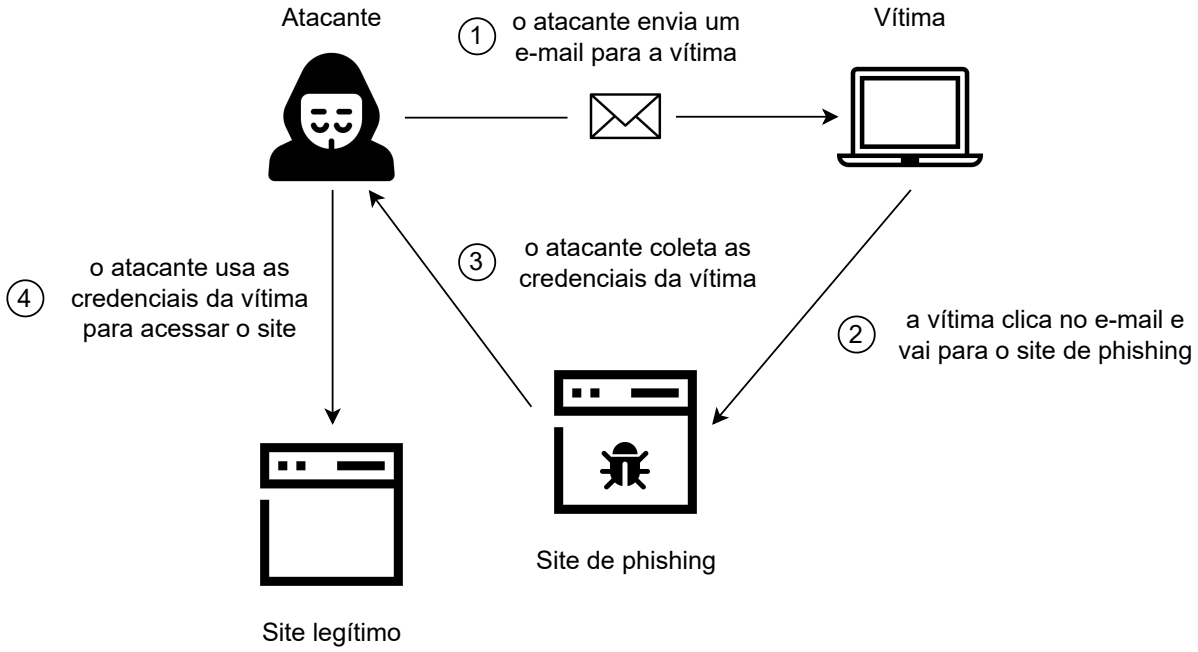
Malwares são aplicações desenvolvidas para interromper o funcionamento normal de dispositivos. São distribuídos no formato de *scripts* ou executáveis e no geral, precisam da interação com o usuário. Existem diversos tipos de *malwares*, entre eles, podemos citar os *ransomwares*, bastante utilizados para extorquir dinheiro das vítimas e os *vírus*, capazes de se propagar autonomamente e causar estrago nas máquinas infectadas (KOK *et al.*, 2019). A melhor maneira de se proteger contra esse tipo de ataque é utilizar antivírus, manter os sistemas atualizados e baixar apenas *softwares* legítimos de lojas confiáveis.

Phishing é a prática de enviar comunicações fraudulentas que parecem vir de uma fonte confiável (RASTOGI *et al.*, 2021). Por ser fácil de realizar e extremamente eficaz, é

[1] Empresa multinacional americana que oferece soluções de rede e segurança: https://www.cisco.com/.

provavelmente a forma mais comum de ataque cibernético. *Phishing* geralmente ocorre por Serviços de Mensagens Curtas – *Short Message Service* (SMS) – ou *e-mail*, cujo funcionamento é demonstrado na Figura 2. Esse tipo de ataque visa roubar dados confidenciais, como senhas de cartões de crédito e informações de login, ou instalar *malwares* na máquina da vítima. Para combater ataques de *phishing* é importante verificar os remetentes dos *e-mails*, as *URLs* encurtadas e os anexos antes de interagir com eles.

Figura 2 – O ataque de *phishing* ocorre quando o atacante envia uma mensagem fraudulenta para a vítima contendo o endereço de um site malicioso usado para roubar informações pessoais.

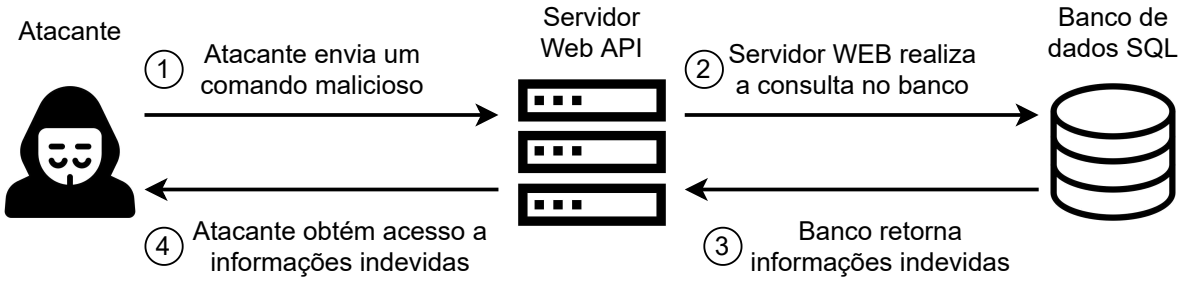


Fonte: elaborado pelo autor.

SQL injection (SQLi) é uma técnica utilizada por atacantes para explorar vulnerabilidades conhecidas na Linguagem de Consulta Estruturada – *Structured Query Language* (SQL) – e fazer com que servidores executem códigos maliciosos nos bancos de dados SQL (LATCHOUMI *et al.*, 2020). Esse tipo de ataque resulta em ações inesperadas por parte dos sistemas e figura em uma posição de destaque na lista das 10 principais ameaças à segurança de aplicações *WEB* do Projeto Aberto de Segurança em Aplicações Web – *Open Web Application Security Project* (OWASP)[2]. A Figura 3 descreve como um atacante pode explorar esse tipo de falha para obter acesso a dados confidenciais. A prevenção para esse tipo de ataque requer o uso de práticas robustas de programação, como a higienização das entradas dos usuários e a configuração correta dos bancos de dados.

[2] Projeto *open-source* que define técnicas e disponibiliza documentos que possibilitam desenvolver aplicações *WEB* seguras: https://owasp.org/www-project-top-ten/.

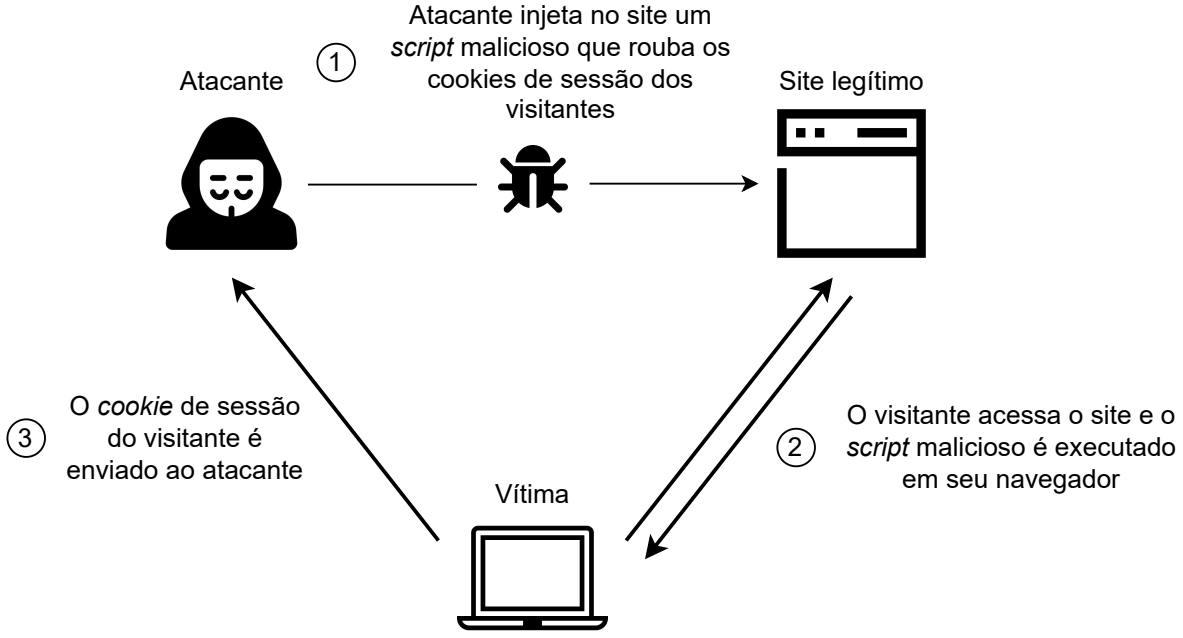
Figura 3 – O ataque de SQLi ocorre quando o atacante envia um comando malicioso para o banco SQL, que devido a não higienização dos dados, acaba executando-o e permitindo o acesso a informação indevidas.



Fonte: elaborado pelo autor.

Cross-Site Scripting (XSS) é um tipo de ataque onde o atacante injeta códigos ou *scripts* maliciosos diretamente em um site confiável, sem atacar o próprio site (RODRÍGUEZ *et al.*, 2020). Dessa forma, sempre que um usuário visita o site comprometido, o navegador do cliente executa o *script*, visto que ele está sendo enviado de uma fonte confiável, como pode ser visto na Figura 4. Isso não só prejudica a reputação do site, mas também permite que o atacante sequestre informações confidenciais como credenciais de usuários, informações de cartão de crédito, *cookies* de sessão, etc. Esse tipo de ataque pode ser evitado através do emprego de boas práticas de programação como a sanitização, validação e "escape" das entradas dos usuários.

Figura 4 – O ataque de XSS ocorre quando o atacante consegue obter informações pessoas de um usuário, a partir da execução automática de um *script* malicioso inserido em um site legítimo acessado pela vítima.



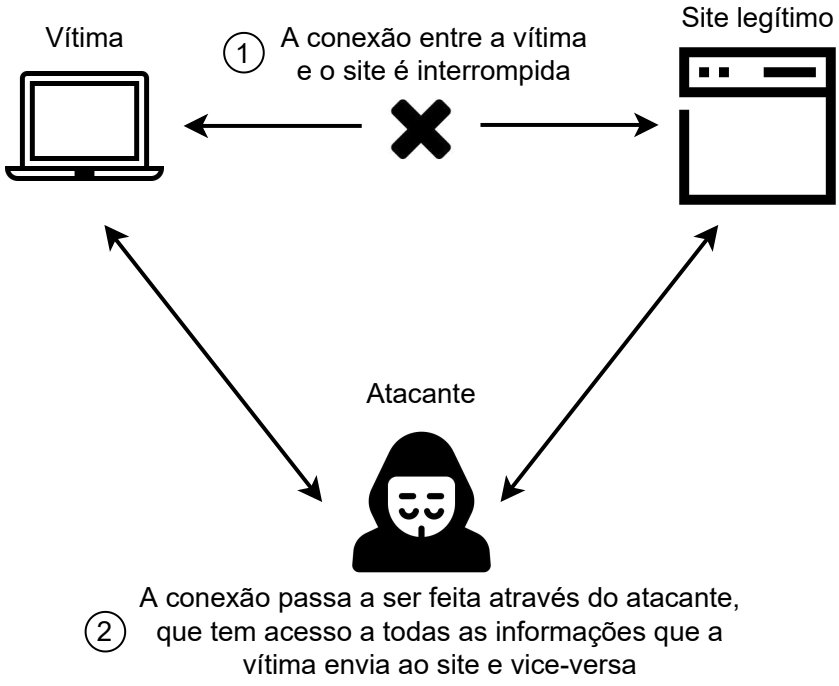
Fonte: elaborado pelo autor.

Negação de Serviço – *Denial-of-Service* (DoS) – e Negação de Serviço Distribuída

– *Distributed Denial-of-Service* (DDoS) – são ataques que buscam tornar inativo serviços na internet. Esse tipo de ataque envolve inundar servidores com solicitações excessivas, que eventualmente esgotam recursos, causam travamentos e impedem a resposta de solicitações legítimas (HUSEINOVIĆ *et al.*, 2020). Embora DoS e DDoS não ofereçam nenhum benefício direto ao atacante, eles podem ser utilizados em conjunto com outras técnicas para perpetrar ataques maiores. Combater um ataque de DoS é bastante difícil, no entanto, existem práticas que podem auxiliar a evitá-los, como, por exemplo, a filtragem de pacotes, do inglês *ingress filtering*, técnica usada para garantir que os pacotes recebidos sejam provenientes de tráfego legítimo.

Homem no Meio – *Man-in-the-Middle* (MiTM) – é quando um atacante intercepta a comunicação entre duas partes na tentativa de espionar a vítima, roubar informações pessoais, credenciais de acesso ou alterar a mensagem entre as partes (CHORDIYA *et al.*, 2018), como pode ser visto na Figura 5. Esse tipo de ataque é menos comum hoje em dia, pois a maioria dos sistemas de *e-mail* e de mensagens instantâneas usa criptografia de ponta-a-ponta, que evita que terceiros adulterem os dados transmitidos pela rede, independentemente dela ser segura ou não. Mesmo assim, a utilização do protocolo *Wi-Fi Protected Access* (WAP), de Redes Privadas Virtuais – *Virtual Private Networks* (VPNs) – e do protocolo *Hyper Text Transfer Protocol Secure* (HTTPS) são práticas de segurança recomendadas para evitar tais ataques.

Figura 5 – O ataque de MiTM ocorre quando o atacante intercepta a comunicação entre um site legítimo e um usuário que, sem perceber, passa a enviar suas informações para o atacante.



Fonte: elaborado pelo autor.

2.3 Classificação das Vulnerabilidades quanto à sua Severidade

Fornecedores de produtos de segurança (e.g. *firewalls*, anti-vírus, etc.) e empresas privadas (e.g. *Gartner*[3]) podem desenvolver suas próprias metodologias para avaliar a severidade das vulnerabilidades, no entanto, esse esforço é visto como algo que deve ser *open-source* e realizado pela comunidade. Assim, o NIST desenvolveu o Protocolo de Automação de Conteúdo de Segurança – *Security Content Automation Protocol* (SCAP) –, que define padrões que auxiliam as organizações a automatizar a maneira como monitoram as vulnerabilidades presentes em seus ambientes (WALTERMIRE *et al.*, 2018). Os principais componentes do SCAP são:

- Vulnerabilidade Comum e Exposição – *Common Vulnerabilities and Exposures* (CVE): identificador único atribuído às vulnerabilidades no momento de sua descoberta. Esse identificador começa com a sigla CVE, seguido por um hífen, o ano de sua descoberta, outro hífen e um número com pelo menos quatro dígitos. Como exemplo, podemos citar o *CVE-2022-22965*, que identifica uma vulnerabilidade que afeta o Java Spring MVC[4];
- Enumeração de Configuração Comum – *Common Configuration Enumeration* (CCE): lista de problemas encontrados durante a configuração de sistemas que pode ser utilizada para desenvolver orientações de segurança;
- Enumeração de Plataforma Comum – *Common Platform Enumeration* (CPE): método utilizado para identificar produtos (i.e. *hardwares*, *softwares* ou sistemas operacionais). Os CPEs são utilizados para descrever quais CVEs e/ou CCEs estão presentes nos ativos;
- Enumeração de Fraqueza Comum – *Common Weakness Enumeration* (CWE): é um sistema de categorização utilizado para se identificar fraquezas de *hardware* e *softwares* comuns a diferentes produtos. Seu objetivo é criar ferramentas automatizadas que possam ser utilizadas para prevenir a exploração dessas falhas;
- Sistema Comum de Pontuação de Vulnerabilidades – *Common Vulnerability Scoring System* (CVSS): nota utilizada para medir a severidade das vulnerabilidades definidas através do CVE.

A seguir, descreveremos em detalhes o CVSS, pois seu entendimento é de suma importância para a compreensão deste trabalho e por ele ser o sistema de pontuação tradicionalmente utilizado em soluções de VM (WANG *et al.*, 2020).

[3] Empresa de pesquisa e consultoria especializada em tecnologia da informação: https://www.gartner.com/en.
[4] *Framework* Java para o desenvolvimento *WEB*: https://spring.io/.

2.3.1 Sistema Comum de Pontuação das Vulnerabilidades (CVSS)

O CVSS é um padrão gratuito e aberto, capaz de capturar as principais características de uma vulnerabilidade e determinar sua gravidade. O CVSS utiliza um sistema de pontuação que varia entre [0, 10], para avaliar a severidade das vulnerabilidades (FIRST, 2019). Assim, quanto maior a nota, maior a gravidade associada. Além disso, essa nota possui um valor textual correspondente, como pode ser visto no Quadro 1, utilizado por profissionais de segurança para priorizar as vulnerabilidades que devem ser corrigidas primeiro.

Quadro 1 – A nota de CVSS varia entre [0, 10] e pode ser mapeada para quatro classes distintas (LOW, MEDIUM, HIGH e CRITICAL) que definem textualmente a severidade das vulnerabilidades.

Severidade	**Pontuação no *CVSS***
LOW	$0.0 \sim 3.9$
MEDIUM	$4.0 \sim 6.9$
HIGH	$7.0 \sim 8.9$
CRITICAL	$9.0 \sim 10.0$

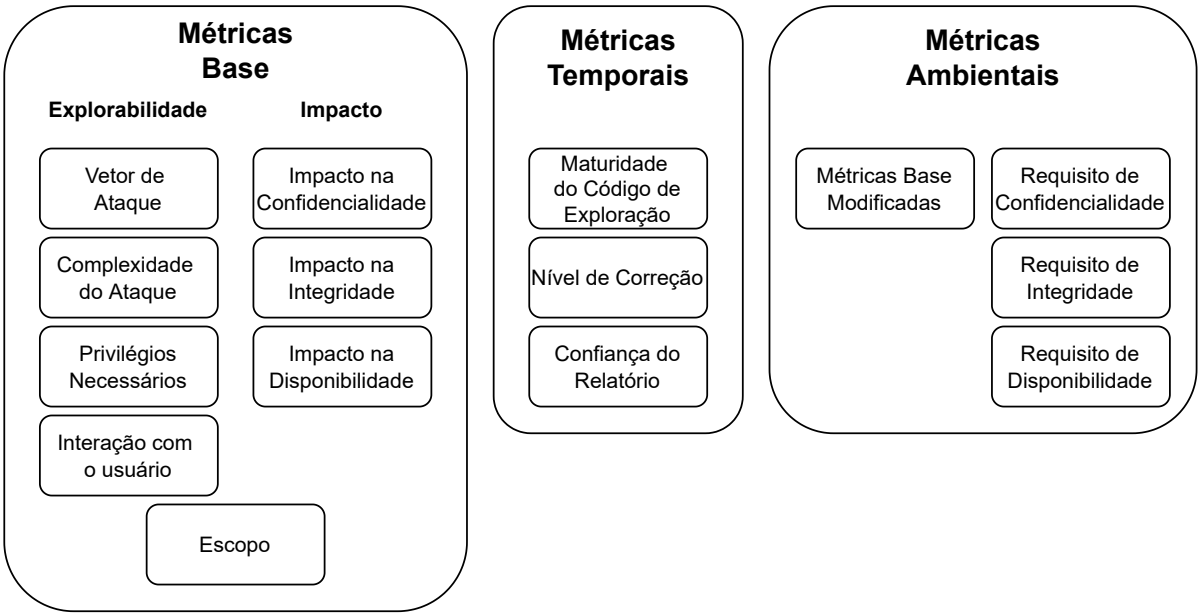
Fonte: elaborado pelo autor.

O CVSS é composto por três grupos de métricas que são: base, temporal e ambiental, como podem ser vistos na Figura 6. O grupo de métricas base é o mais utilizado e reflete as características intrínsecas das vulnerabilidades, constantes temporalmente. O grupo base é composto por dois subgrupos de métricas: (*i*) as métricas de explorabilidade, que refletem a facilidade e os meios técnicos pelos quais uma vulnerabilidade pode ser explorada; e (*ii*) as métricas de impacto, que refletem a consequência direta de uma exploração bem-sucedida. Além disso, o grupo base possui uma variável de escopo que determina se a exploração da vulnerabilidade em um sistema ou componente pode ter, ou não, impacto sobre outro sistema ou componente (FIRST, 2019).

O grupo de métricas temporais reflete as características de uma vulnerabilidade que podem mudar com o tempo, mas não entre ambientes organizacionais diferentes (FIRST, 2019). Esse grupo é responsável por ajustar a pontuação gerada pelo grupo base, considerando fatores externos que afetam a vulnerabilidade. Por exemplo, a presença de um *exploit* público aumentaria o risco de exploração de uma determinada vulnerabilidade e consequentemente sua pontuação de CVSS, enquanto a criação de um *patch* de segurança diminuiria sua nota.

Por fim, temos o grupo de métricas ambientais, que representam as características de uma vulnerabilidade que são relevantes e exclusivas de um determinado ambiente organizacional (FIRST, 2019). As características consideradas por esse grupo incluem a presença de controles de

Figura 6 – O CVSS é composto por três grupos de métricas, a saber, base, temporal e ambiental, que trazem informações sobre as características das vulnerabilidades, a presença de *exploits* públicos e o ambiente afetado por elas.



Fonte: elaborado pelo autor.

segurança que possam mitigar algumas ou todas as consequências de um ataque bem-sucedido e a importância do ativo vulnerável na infraestrutura da organização. Esse grupo ajusta a pontuação gerada pelo grupo base e temporal.

É importante destacar que apesar do CVSS oferecer três grupos de métricas, o grupo base é o mais utilizado por profissionais de segurança e ferramentas de gestão de vulnerabilidades (WANG *et al.*, 2020). A utilização dos grupos temporal e ambiental faria com que a severidade da vulnerabilidade estivesse mais próxima da realidade, pois consideraria no cálculo informações adicionais relevantes. No entanto, como esses valores não são triviais de se conseguir e dado que a fórmula utilizada no cálculo do CVSS não está devidamente justificada, os profissionais acabam não as utilizando (SPRING *et al.*, 2021).

2.3.1.1 Limitações do CVSS

– **A fórmula utilizada no cálculo do CVSS não está devidamente justificada (SPRING *et al.*, 2021).** Para obtenção do CVSS, o analista precisa responder uma série de perguntas qualitativas (e.g. qual o impacto na confidencialidade, integridade e disponibilidade dos sistemas), cujas respostas serão convertidas em valores quantitativos e utilizados em operações algébricas para calcular a nota. Assim, assume-se que os valores ordinais possuem importância relativa entre si e que expressões do tipo "$LOW + HIGH = 6$" são

verdadeiras, sem nenhum critério objetivo;

- **A construção da fórmula não é transparente (SPRING *et al.*, 2021).** A fórmula do *CVSS* foi derivada a partir da classificação feita por especialistas em uma base de dados contendo 100 vulnerabilidades. Nada sobre esse processo de classificação está documentado na especificação[5]. Apesar dos autores originais comentarem sobre cada uma das métricas, eles não explicam o porquê delas terem sido escolhidas para compor a nota;
- **O *CVSS* indica a severidade e não o risco de exploração das vulnerabilidades para as organizações (WALKOWSKI *et al.*, 2021).** O CVSS foi desenvolvido para identificar tecnicamente a severidade de uma vulnerabilidade, assumindo valores que independem do contexto e são constantes temporalmente. No entanto, os analistas costumam utilizá-lo como uma representação de risco e como métrica que define com que rapidez eles devem corrigir as vulnerabilidades. Por isso, ele é amplamente mal utilizado;
- **Existe um problema de dispersão no *CVSS* que faz com que a maioria das vulnerabilidades esteja concentrada em poucas classes (WU *et al.*, 2019).** Esse fato dificulta o processo de gestão de vulnerabilidade. Pois, se muitas vulnerabilidades possuem a mesma severidade (e.g. o CVSS *CRITICAL*), os analistas não conseguirão focar seus esforços em corrigir as vulnerabilidades de maior risco, tornando ineficiente todo o processo de RBVM.

2.4 Gestão de Vulnerabilidades

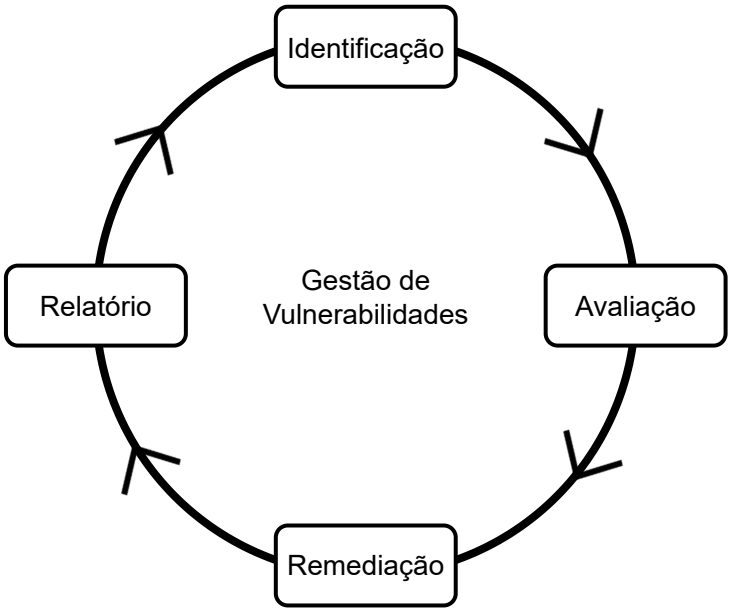
Milhares de novas vulnerabilidades são reportadas anualmente, exigindo que as organizações utilizem diversas medidas de segurança para tornar seus sistemas mais seguros. Essa é uma tarefa árdua e complexa, dado o grande volume de falhas de segurança que podem ser exploradas e a escassez de mão de obra especializada que possa ser utilizada no processo (FURNELL *et al.*, 2017). Além disso, os atacantes estão realizando ataques cada vez mais complexos, de tal forma, que os profissionais de segurança precisam utilizar as mais avançadas ferramentas e técnicas para tentar se defender (CHEUNG; BELL, 2021).

Uma das soluções para esse problema é a adoção de estratégias que auxiliem os analistas a corrigirem as vulnerabilidades proativamente, isto é, antes que elas possam ser exploradas por usuários mal-intencionados. Essa disciplina é chamada de Gestão de Vulnerabilidades – *Vulnerability Management* (VM) –, e é um processo contínuo que busca identificar, categorizar,

[5] Documento de especificação do CVSS v3: https://www.first.org/cvss/v3.0/specification-document.

priorizar e remediar vulnerabilidades em *softwares*, aplicações corporativas e sistemas operacionais (FOREMAN, 2019). O processo de VM pode ser dividido em 4 etapas, representados na Figura 7, sendo elas: a identificação, avaliação, remediação e a produção de relatórios.

Figura 7 – VM é um processo de gestão contínuo, dividido em quatro etapas e que busca auxiliar os analistas de segurança no processo de identificação, avaliação e remediação das vulnerabilidades de segurança.



Fonte: elaborado pelo autor.

A primeira etapa do processo de VM é identificar as vulnerabilidades existentes na rede da organização. Para isso, é necessário utilizar ferramentas conhecidas como varredores de vulnerabilidade, do inglês *vulnerability scanners* (MARTINEZ *et al.*, 2019). Como exemplo podemos citar o *Nessus*[6] da *Tenable* e o *OpenVas*[7] da *Greenbone*. Essas ferramentas conseguem avaliar redes, sistemas e computadores em buscas de falhas conhecidas. O ato de verificar as redes em busca de vulnerabilidades é uma parte essencial do processo de VM e pode ser dividido em quatro estágios:

- primeiro, varre-se a rede da organização, utilizando os protocolos *ICMP* e *TCP/UDP*, para localizar os ativos acessíveis;
- segundo, identifica-se as portas abertas e os sistemas vulneráveis nos ativos encontrados;
- terceiro, se possível, as ferramentas acessam os ativos, utilizando o protocolo *SSH*, para coletar informações adicionais;
- e quarto, correlaciona-se as informações coletadas com a base de vulnerabilidades existente,

[6] Programa proprietário de verificação de vulnerabilidades: https://pt-br.tenable.com/products/nessus.
[7] Programa *open-source* de verificação de vulnerabilidades: https://www.openvas.org/.

para identificar as falhas que afetam o ativo.

Os varredores são um componente essencial de qualquer solução de VM e conforme o CIS [8], as organizações devem realizar varreduras de vulnerabilidades automatizadas pelo menos uma vez por semana (CIS, 2021). No entanto, esses *scanners* podem, às vezes, interromper ou deixar instáveis as redes e serviços, visto que eles sondam ativamente os ativos na rede em busca de vulnerabilidades (RAPID7, c2022). Assim, é comum que a periodicidade dessas varreduras seja maior, acontecendo, normalmente, uma vez por mês.

O segundo passo no processo de VM é avaliar as vulnerabilidades encontradas e determinar a melhor forma de gerenciá-las. Normalmente, as soluções de VM classificam as vulnerabilidades utilizando um sistema de notas, como, por exemplo, o CVSS. Esses sistemas de pontuações são bastante úteis, no entanto, eles falham ao não considerar fatores internos e externos que possam ser utilizados para identificar o real risco de exploração das vulnerabilidades (GOLDSTEIN, 2021). A seguir alguns desses fatores:

- com que facilidade alguém pode explorar esta vulnerabilidade?
- existe um *exploit* público ou uma campanha de *malware* para ela?
- alguém poderia explorar essa vulnerabilidade diretamente da internet?
- qual seria o impacto nos negócios se essa vulnerabilidade fosse explorada?
- há quanto tempo esta vulnerabilidade está presente na rede da organização?

Como qualquer ferramenta de segurança, os varredores de vulnerabilidade não são perfeitos. As taxas de falso-positivo e falso-negativo na detecção de vulnerabilidade, embora baixas, ainda são maiores que zero. Por causa disso, a realização de testes de penetração pode ajudar a melhorar a identificação de vulnerabilidades e focar a atenção dos profissionais de segurança nas falhas mais perigosas (RAPID7, c2022).

Depois que as vulnerabilidades são avaliadas e priorizadas, a próxima etapa é decidir como será realizada a remediação. Esse processo deve envolver as partes interessadas e seguir as diretrizes internas da empresa (GOLDSTEIN, 2021). No geral, existem três diferentes formas de se corrigir uma vulnerabilidade, que são:

- a remediação, que envolve a aplicação de uma correção que corrige completamente a vulnerabilidade e impedem dela ser explorada;
- a mitigação, que reduz temporariamente a probabilidade ou o impacto de exploração de uma vulnerabilidade. Normalmente isso é feito quando não existe um *patch* de segurança;

[8] Organização sem fins lucrativos cuja missão é tornar a internet mais segura: https://www.cisecurity.org/.

– nenhuma ação, reconhecer e aceitar a vulnerabilidade. As organizações normalmente só fazem isso quando o custo de remediar a vulnerabilidade é muito maior do que as consequências dela ser explorada;

A remediação pode ser tão simples quanto aplicar um *patch* de segurança ou tão complexa quanto segmentar a rede e colocar ativos atrás de um *firewall*. No entanto, nem todas as vulnerabilidades precisam ser corrigidas. Por exemplo, se o varredor identificar uma vulnerabilidade no *Adobe Flash Player*, mas o uso do *flash* estiver desabilitado nos navegadores da organização, essa vulnerabilidade pode ser considerada mitigada. Por fim, quando as atividades de remediação forem concluídas, é uma boa prática executar uma nova varredura na rede, confirmando que as vulnerabilidades foram corrigidas de fato (RAPID7, c2022).

A última etapa do processo é a relatória, essencial para as organizações poderem melhorar sua percepção sobre a eficácia, velocidade e o custo associado ao processo de VM (GOLDSTEIN, 2021). É comum que as ferramentas de gestão ofereçam diferentes opções de exportação e visualização dos resultados, entre elas, podemos citar as versões sumarizados para executivos da empresa e os relatórios detalhados para os analistas de segurança. Esses relatórios auxiliam os profissionais a medir o nível de risco da empresa, a compreender quais técnicas de remediação foram mais eficazes e a garantir que a organização esteja consoante suas diretrizes (RAPID7, c2022).

2.5 Gestão de Vulnerabilidades Baseada em Risco (RBVM)

Muitas organizações ainda contam com o hábito de concentrar seus esforços em corrigir as vulnerabilidades com nota de CVSS *CRITICAL*. Além disso, é comum ver organizações utilizando extensas planilhas para correlacionar informações sobre as falhas de segurança presentes em seus ambientes (KENNA SECURITY, 2020). Essas abordagens são ineficientes, visto que os analistas acabam tendo que corrigir listas com milhares de vulnerabilidades, que muitas vezes não representam um risco real caso exploradas (TENNABLE, 2020b). Isso gera um desperdício enorme de recursos que poderiam ser direcionados a tarefas mais estratégicas.

Dado esse contexto, a empresa de pesquisa e consultoria Gartner, chamou a atenção dos profissionais de segurança para a necessidade de se avaliar e priorizar a correção das vulnerabilidades considerando o risco que a sua exploração traria para a organização (LAWSON *et al.*, 2019). De fato, sem uma metodologia capaz de comunicar quais vulnerabilidades representam o maior risco à organização, não há como saber se as vulnerabilidades que estão sendo corrigidas

são as que poderiam causar maiores danos se exploradas.

Assim, surgiu a disciplina de RBVM que propõe que se classifique as falhas de segurança considerando não só as características das vulnerabilidades, mas também as informações de inteligência de ameaças e contexto (discutido em detalhes nas Subseções 2.5.1 e 2.5.2), mudando assim a perspectiva dos sistemas clássicos de VM. Apesar de ser um paradigma novo, pesquisas já sugerem que em 2022, organizações que utilizarem a disciplina de RBVM sofrerão 80% menos ataques (BHAJANKA *et al.*, 2019).

2.5.1 Inteligência de Ameaças

Os cibercriminosos aprimoraram suas Táticas, Técnicas e Procedimentos – *Tactics, Techniques and Procedures* (TTPs) –, a ponto de tornarem seus ataques difíceis de detectar, analisar e remediar. Seus TTPs são menos previsíveis, mais persistentes, engenhosos e organizados (ABU *et al.*, 2018). Para lidar com esse problema, os pesquisadores criaram o conceito de inteligência de ameaças, do inglês *threat intelligence*, que se refere ao "conjunto de dados coletados, analisados e aplicados em relação a ameaças de segurança, atores maliciosos, *exploits*, *malwares*, vulnerabilidades e indicadores de comprometimento" (CONTI *et al.*, 2018).

Em outras palavras, inteligência de ameaças é a disciplina que coleta e analisa informações em tempo real para identificar o cenário global de ameaças e apontar quais redes, sistemas e computadores correm um maior risco de serem explorados (WAGNER *et al.*, 2019). Essas informações são coletadas de diversas fontes, como, por exemplo, *feeds* de inteligência, mídias sociais, análise do tráfego da internet, da *deep* e *dark web*, etc.

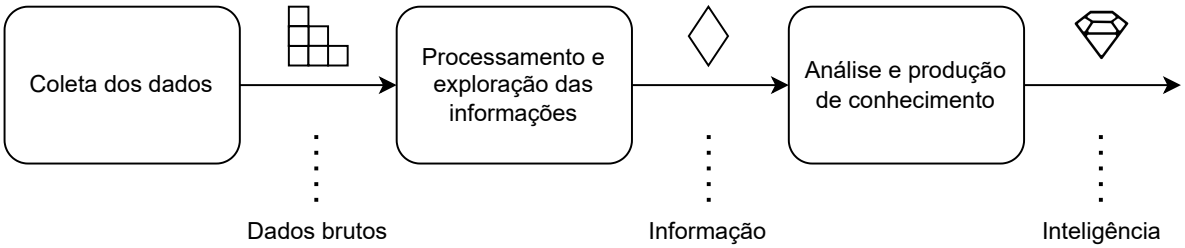
Um exemplo desses *feeds* é a Plataforma de Compartilhamento de Informações de *Malware*[9] – *Malware Information Sharing Platform* (MISP) – um projeto *open-source* utilizado por especialistas de segurança para compartilhar informações sobre Indicadores de Comprometimento – *Indicator of Compromise* (IoC) –, que são artefatos identificados em uma rede de computadores ou em um sistema operacional, que indicam que uma invasão ocorreu. Exemplos comuns de IoC são: assinaturas de vírus e IPs, *hashes* MD5 de *malwares* ou domínios e URLs associadas a *botnets*.

Assim, o que diferencia essa disciplina de uma simples coleta e disseminação de informação é a análise desses dados. Especialistas, devem processar todas as informações coletadas e fornecer o conhecimento necessário para ser possível identificar os atacantes e

[9] Plataforma de compartilhamento de informações de inteligência de ameaças: https://www.misp-project.org/.

solucionar as ameaças de segurança em tempo hábil (BROMANDER, 2021). Como pode ser visto na Figura 8, inteligência de ameaças é o ato de transformar informações brutas em conhecimento, que pode ser utilizado para se defender de ataques cibernéticos.

Figura 8 – Inteligência de ameaças é um processo que combina o processamento de dados brutos, com a experiência dos profissinais de segurança, para produzir conhecimento que auxiliará na defesa das organizações.



Fonte: elaborado pelo autor.

2.5.2 Contexto da Vulnerabilidade

Outro importante conjunto de informações que pode ser usado pelos analistas para identificar o risco de uma vulnerabilidade é o contexto onde ela ocorre. A definição mais conhecida para contexto é dada pelos autores em Abowd *et al.* (1999), que definem contexto como qualquer informação usada para caracterizar a interação entre um usuário e uma aplicação.

Essa definição é centrada no usuário, pois seu foco está no usuário e no serviço realizado por alguma entidade. Por exemplo, uma lâmpada inteligente (entidade) que se acende quando alguém (usuário) entra na sala. Por outro lado, nosso trabalho considera o contexto como qualquer informação usada para categorizar a criticidade de um ativo na rede.

Por exemplo, considere um servidor com um banco de dados que armazena informações confidenciais de usuários e uma estação de trabalho. Explorar uma vulnerabilidade que permite que um invasor execute um código remotamente no servidor é mais crítico do que explorar a mesma falha em uma estação de trabalho.

Neste exemplo, o tipo de ativo (servidor ou estação de trabalho) e os dados que ele armazena são características usadas para classificar o contexto da vulnerabilidade, isto é, a criticidade do ativo vulnerável. O contexto é relevante, pois possibilita a classificação de precedência entre os ativos.

2.5.3 As etapas de um RBVM

O RBVM facilita a atividade de avaliação de risco, pois estabelece um processo que pode ser seguido pelos analistas para avaliar com precisão o risco de exploração das falhas de segurança. Esse processo pode ser dividido em três etapas, que são: (*i*) o mapeamento de ativos e vulnerabilidades; (*ii*) a priorização das vulnerabilidades baseada no risco; e (*iii*) a remediação das falhas de segurança.

Na primeira etapa, onde ocorre o mapeamento de ativos e vulnerabilidades, os analistas devem listar dispositivos (por exemplo, servidores, estações de trabalhos, sistemas de computadores, etc.) usados na organização e as vulnerabilidades que os afetam (LAWSON, 2020). Essa etapa ajuda a fornecer uma visão geral do seu ambiente e a determinar a superfície de ataque da organização. Esse processo é conduzido através do uso de varredores de vulnerabilidades (e.g. Nessus[10]) e de ferramentas de Gerenciamento de Configuração – *Configuration Management Database* (CMDB).

Na segunda etapa, onde ocorre a priorização das vulnerabilidades baseada no risco, os profissionais de segurança analisam e selecionam as vulnerabilidades cujo impacto de exploração trará maiores danos para a organização (LAWSON, 2020). Para isso, cruzam-se os dados de contexto obtidos na etapa anterior, com as informações de inteligência de ameaças. Assim, avalia-se a importância do ativo vulnerável para a organização e as informações acerca de suas vulnerabilidades. Como, por exemplo, se a vulnerabilidade possui ou não *exploit* público; se ela pode ser explorada remotamente; se grupos maliciosos estão falando dela, na *deep* e *dark web*; se ela aparece em alguma lista de vulnerabilidades críticas, como, por exemplo, o *Mitre*[11]; etc.

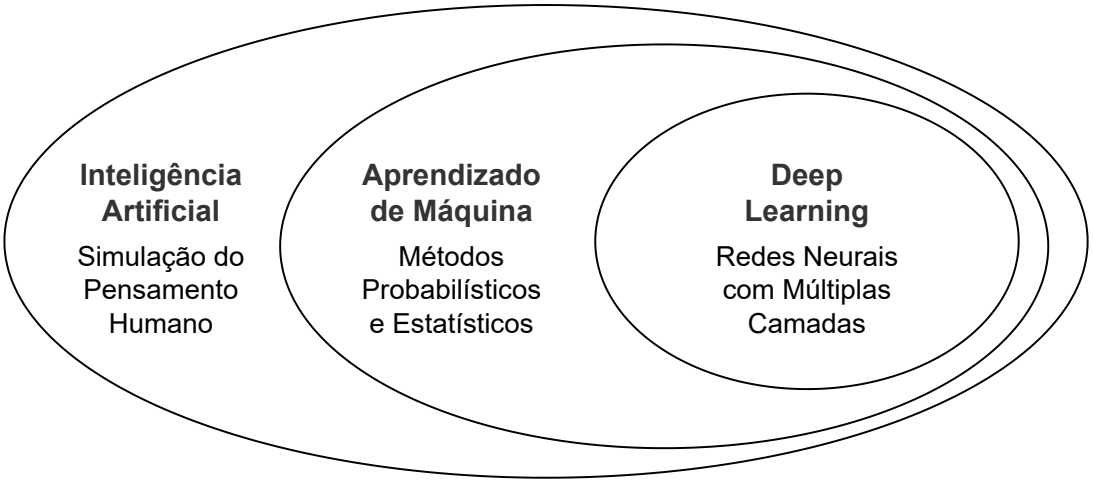
Por fim, a remediação das vulnerabilidades é o processo de mitigação ou correção das vulnerabilidades que afetam os ativos da organização (LAWSON, 2020). Essa atividade pode parecer corriqueira, mas em grandes organizações, com milhões de vulnerabilidades, ela pode ser desafiadora. Na melhor dos cenários, os fabricantes lançaram um *patch* de segurança que corrige a vulnerabilidade. No entanto, para as vulnerabilidades de *zero-day*, que não possuem *patches*, é necessário tomar medidas que visam diminuir o impacto de sua exploração, como, por exemplo: manter antivírus atualizados; utilizar *proxies* e VPN; criar regras de acesso no *firewall*; segmentar as redes da organização; etc.

[10] Varredor de vulnerabilidades desenvolvido pela empresa Tenable: https://tenable.com/products/nessus.
[11] Lista com as fraquezas de *software* consideradas mais críticas pelo Mitre: https://cwe.mitre.org/top25/.

3 APRENDIZADO DE MÁQUINA APLICADA À SEGURANÇA

Aprendizado de Máquina – *Machine Learning* (ML) – é uma técnica que visa desenvolver algoritmos computacionais capazes de emular a inteligência humana na resolução de problemas, através do reconhecimento de padrões em grandes conjuntos de dados (HELM *et al.*, 2020). Como pode ser visto na Figura 9, ML é um subconjunto da disciplina de Inteligência Artificial – AI – por isso, muitas vezes, suas definições se confundem. No entanto, podemos dizer que ML é a técnica cujos algoritmos utilizam probabilidade e estatística para melhorar o desempenho de uma determinada tarefa (MICHALSKI *et al.*, 2013).

Figura 9 – As aplicações de Aprendizado Profundo são um subconjunto das aplicações de Aprendizado de Máquina, que por sua vez, são um subconjunto das aplicações de Inteligência Artificial, como representado pelo diagrama de Veen.



Fonte: elaborado pelo autor.

Os algoritmos de ML, diferentemente dos programas convencionais, são genéricos e podem ser utilizados para resolver uma grande variedade de problemas, sem terem sido especificamente programados para isso (NAQA; MURPHY, 2015). Dada a capacidade desses algoritmos de modificar seu comportamento através da experiência, eles conseguem aprender e melhorar o seu desempenho de forma automática. Assim, uma das definições de ML é:

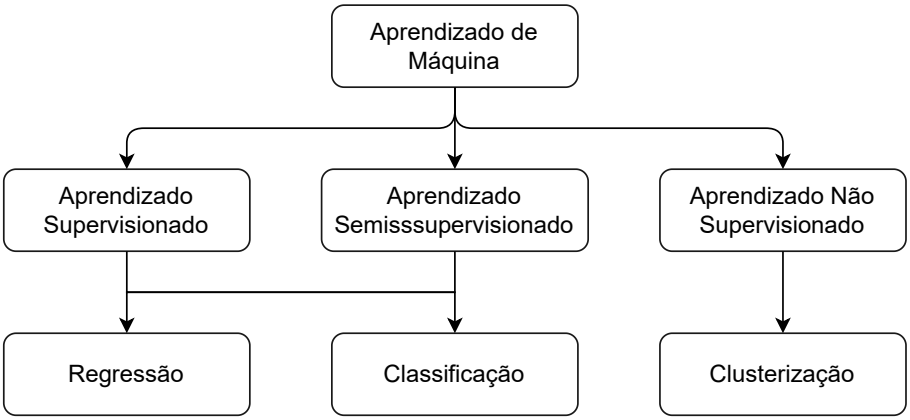
> Programa de computador que aprende com a experiência E em respeito a alguma tarefa T e alguma medida de desempenho P, se seu desempenho em T, conforme medido por P, melhora com a experiência E. (MITCHELL, 1997, p. 2)

Esse processo de modificação do comportamento de um algoritmo de ML é chamado de treinamento. Para que isso ocorra, é necessário que lhe seja fornecido um conjunto de dados de entrada, com os resultados desejados (o que chamamos de rótulos). O algoritmo aprende

com esses dados e configura seu comportamento, produzindo o resultado esperado mesmo para dados não vistos anteriormente. Este treinamento é a parte de "aprendizado" do aprendizado de máquina (ALPAYDIN, 2020).

Como pode ser visto na Figura 10, os algoritmos de ML podem ser divididos em três categorias que são: aprendizado supervisionado, semissupervisionado e o não supervisionado (SHOBHA; RANGASWAMY, 2018). A diferença entre essas categorias é em relação à presença ou não de rótulos. No aprendizado supervisionado, os rótulos de todos os dados de treinamento são conhecidos. No aprendizado semissupervisionado, apenas uma pequena porção dos rótulos é conhecida. Já no aprendizado não supervisionado, nenhum rótulo é conhecido.

Figura 10 – Os algoritmos de ML podem ser divididos em três grupos diferentes, dado a presença e a quantidade de dados rotulados para resolução do problema. Os grupos são: aprendizado supervisionado, semissupervisionado e não supervisionado.



Fonte: elaborado pelo autor.

Ainda em relação à Figura 10, podemos ver que o aprendizado supervisionado e semissupervisionado podem ser divididos em duas categorias: (*i*) a regressão, utilizada para prever valores numéricos; e (*ii*) a classificação, utilizada para prever a qual categoria pertence uma determinada amostra do problema (MOHRI *et al.*, 2018). Já o aprendizado não supervisionado, dado a ausência de rótulos, visa encontrar relações entre os dados, agrupando-os em conjuntos chamados de *clusters*. Esse processo é conhecido como clusterização.

Por fim, é importante mencionar o Aprendizado Profundo – *Deep Learning* (DL) –, que são um subconjunto de algoritmos de ML capazes de extrair automaticamente atributos de interesse de forma hierárquica, enquanto mantém uma estrutura escalável e diferenciável de ponta a ponta (LECUN *et al.*, 2015). Os algoritmos de DL utilizam o princípio das redes neurais para imitar o funcionamento do cérebro humano, em especial, como o ser humano aplica o pensamento lógico e analítico para resolver problemas a partir de informações recém-adquiridas.

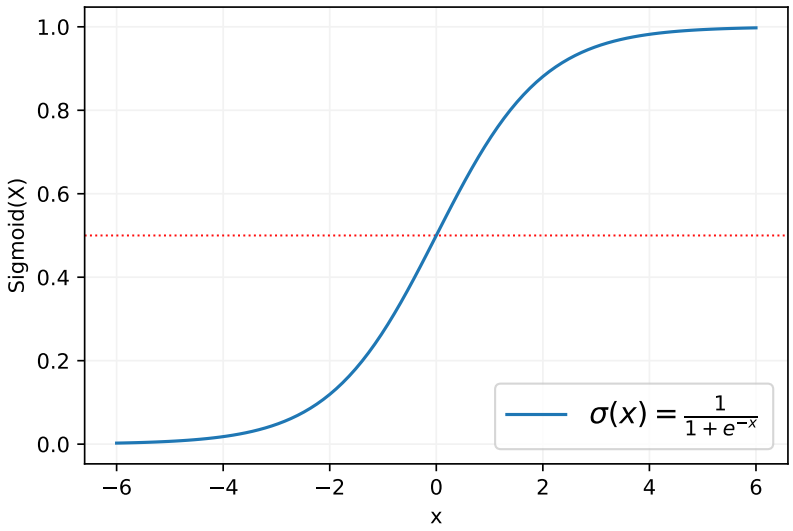
Além disso, os modelos de DL apresentam um desempenho melhor quando o número de amostras disponíveis para treinamento é grande (PURUSHOTHAM *et al.*, 2018; FENG *et al.*, 2019), diferentemente de outros modelos de ML, que possuem uma melhor desempenho em conjuntos de dados menores. Por isso, os modelos de DL são soluções ideais para aplicações como o processamento de imagens e análise de linguagem natural, que lidam com grandes volumes de dados.

3.1 Principais algoritmos de ML

Regressão Logística – *Logistic Regression* (LR) – é uma técnica de ML, que utiliza conceitos do campo da estatística, bastante utilizada para solucionar problemas de classificação binária (BROWNLEE, 2016). LR possui esse nome devido o uso da função logística ou função sigmoide, que é uma curva em forma de S que consegue mapear qualquer valor real para o intervalo entre 0 e 1, como pode ser visto na Figura 11. LR modela a probabilidade de uma entrada pertencer a uma determinada classe. Por exemplo, se estivermos estudando a presença de uma doença coronária cardíaca associada a idade dos indivíduos, podemos modelar o problema de regressão como a probabilidade da pessoa estar doente (1), dado sua idade (X), ou formalmente: $P(Y = 1|X)$. Isso quer dizer que estamos interessados na probabilidade da entrada (X) pertencer à classe padrão $(Y = 1)$, que equivale à classe doente. Assim, uma entrada cuja probabilidade esteja próxima de 1 (ou acima do limiar), será classificada como pertencente à classe doente e uma entrada cuja probabilidade esteja próxima de 0 como não doente.

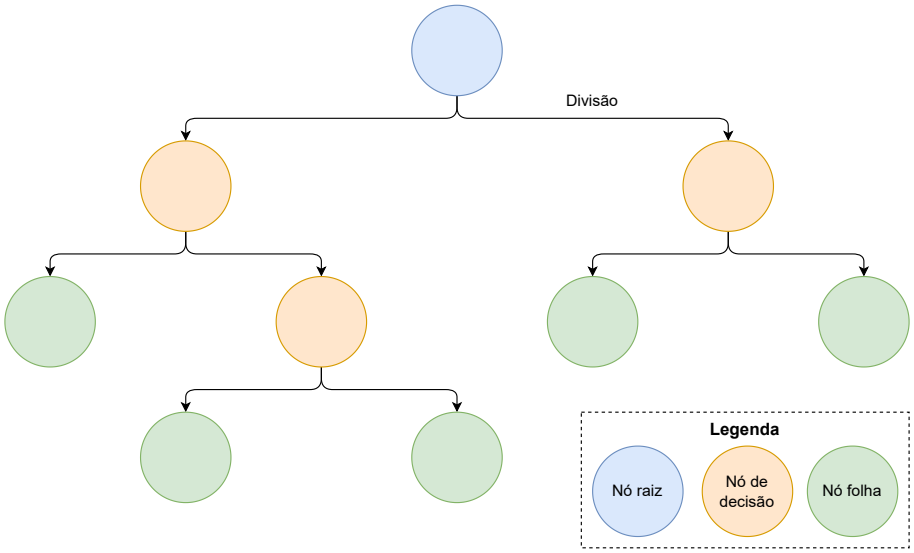
Árvores de Decisão – *Decision Tree* (DT) – são considerados os mais clássicos algoritmos de ML e utilizam uma estrutura em forma de árvore binária para resolver problemas de regressão e classificação (BROWNLEE, 2016). As DT são compostas por três elementos: o nó raiz, os nós de decisão e os nós folha, conforme ilustrado na Figura 12. O nó raiz da árvore representa um objeto que está sendo analisado. Os nós de decisão representam pontos de controle de fluxo, onde condições em relação ao objeto são verificadas (por exemplo, se a pessoa mede mais que x cm). Os nós folha representam a categoria a que o objeto pertence. Portanto, cada caminho do nó raiz até um nó folha descreve as características ou atributos desse objeto. Como podem existir múltiplos caminhos que levam à mesma classe, medidas de entropia e de ganhos de informação – *information gain* – são utilizadas para selecionar o melhor nó de decisão a se tomar. *CART* (PRODROMIDIS; STOLFO, 2001), *C4.5* (QUINLAN, 2014) e *ID3* (GARCIA *et al.*, 2006) são considerados algoritmos importantes de árvore de decisão.

Figura 11 – A função sigmoide (também conhecida como função logística) é uma equação matemática capaz de mapear qualquer valor real para o intervalo [0, 1]. Na figura, a função sigmoide está representada pela curva azul e o limiar pela reta tracejada em vermelho.



Fonte: elaborado pelo autor.

Figura 12 – Árvores de Decisão são algoritmos de ML representados em forma de árvores. O nó raiz representa o objeto analisado, os nós intermediários as tomadas de decisão e os nós folhas a classe que pertence o objeto.

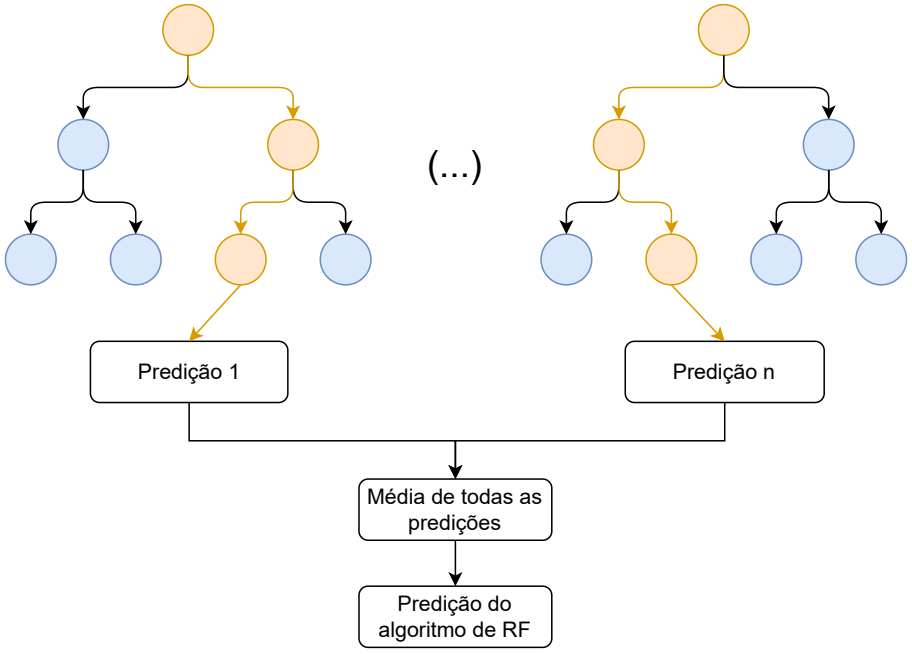


Fonte: elaborado pelo autor.

As Florestas Aleatórias – *Random Forest* (RF) – se encaixam na categoria de algoritmos de ML chamados de métodos de comitê – *ensemble methods* – (MAHESH, 2020). Os algoritmos dessa categoria combinam múltiplas DTs, como pode ser visto na Figura 13, para produzir um modelo preditivo com forte capacidade de generalização. É possível utilizar subconjuntos distintos de dados no treinamento das várias DTs, de forma que cada uma delas pode apresentar um resultado diferente (NARAYAN; SHANMUGAPRIYA, 2022). A média de

todos os resultados será a predição final fornecida pelo modelo. As RF são consideradas uma evolução do algoritmo *CART* (PRODROMIDIS; STOLFO, 2001), por isso, também podem ser utilizadas para fins de classificação e regressão. As RF apresentam um melhor desempenho em problemas não lineares e seu custo computacional durante a fase de treinamento do modelo é menor, se comparada com outros algoritmos de ML (SHAUKAT *et al.*, 2020).

Figura 13 – Florestas Aleatórias são algoritmos de ML que combinam diferentes classificadores para produzir um modelo preditivo ideal, capaz de solucionar problemas tanto de regressão como de classificação. Na figura, vários modelos são combinados para se atingir a predição.

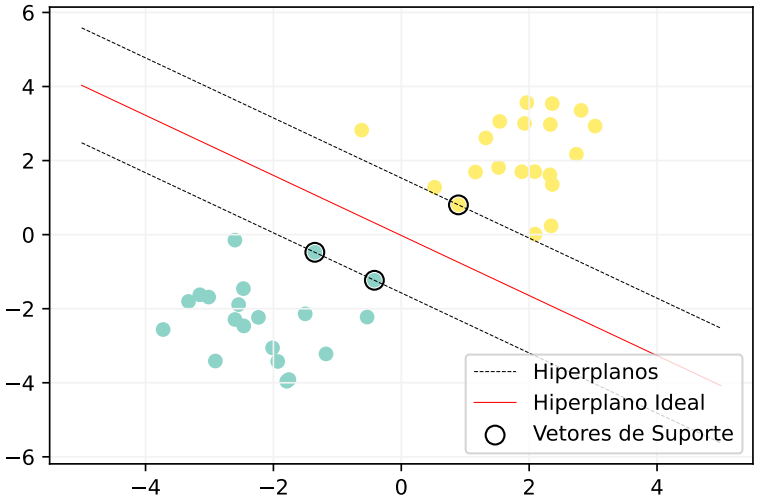


Fonte: elaborado pelo autor.

Máquina de Vetores de Suporte – *Support Vector Machine* (SVM) – são algoritmos que realizam a classificação dos dados traçando hiperplanos que dividem os espaços entre os conjuntos analisados (BONACCORSO, 2017), podendo ser utilizado na solução de problemas com uma classe (CHEN *et al.*, 2005) ou multiclasse (SCHÖLKOPF *et al.*, 1999). Assim, para aumentarmos a precisão, é necessário encontrar o hiperplano que maximiza as margens entre os diferentes conjuntos. Esse plano é conhecido como hiperplano ideal e pode ser visto na Figura 14. Os pontos mais próximos do hiperplano são chamados de vetores de suporte e são os mais importantes, pois, se removidos, alterariam a inclinação dos hiperplanos. O SVM pode resolver problemas lineares e não-lineares, dependendo do *kernel* selecionado para sua função, que pode ser: linear, polinomial, radial, entre outros (BROWNLEE, 2016). O SVM requer muita memória para processamento e consome bastante tempo no seu treinamento.

Rede Neural Artificial – *Artificial Neural Network* (ANN) – é uma abordagem fre-

Figura 14 – Máquina de Vetor de Suporte são algoritmos de ML que traçam hiperplanos para tentar dividir os espaços entre as classes analisadas. Esses planos são traçados com a ajuda dos vetores de suporte, mostrados na figura, e devem maximizar a margem entre as classes.



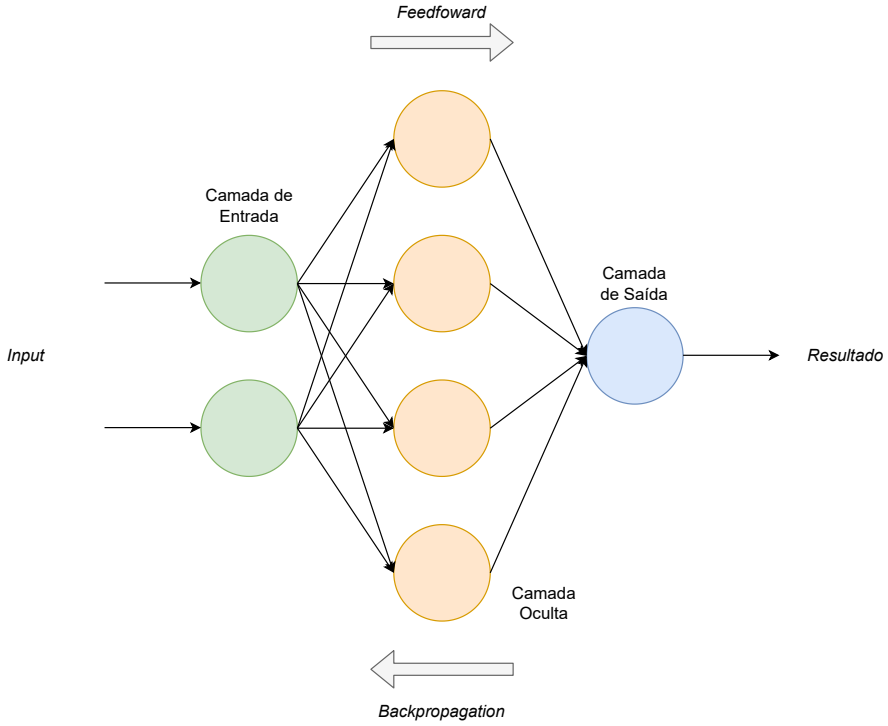
Fonte: elaborado pelo autor.

quentemente usada para fins de classificação por algoritmos como *Hopfields Networks* (JAGOTA, 1991), Perceptron Multicamadas – *Multilayer Perceptron* (MLP) – (AUGUSTEIJN; FOLKERT, 2002) e Árvores Neurais (MARTINEZ, 1998). Os três principais elementos que compõem uma ANN são: as camadas de entrada, oculta e de saída. A Figura 15 apresenta uma ANN simples, com apenas 7 nós dispostos em três camadas. ANNs são treinadas por meio de uma sequência de interações de *feedforward* e *backpropagation*. No processo de *feedforward*, os dados são inseridos nos nós da camada oculta. Uma função de ativação é aplicada e o valor resultante é passado para as camadas seguintes. Na camada final calcula-se o erro, dado pela diferença entre a saída da rede neural e o valor desejado (que em problemas de classificação corresponde a classe do objeto analisado). Essa diferença é enviada de volta até a camada de entrada, pelo processo chamado de *backpropagation*, para os pesos serem ajustados. Este processo é repetido até que o valor desejado seja alcançado (JOSHI, 2017). As ANNs são consideradas fáceis de usar e robustas em relação a ruídos. No entanto, necessitam de um grande volume de dados para atingir um bom desempenho.

3.2 Aprendizado Ativo

O Aprendizado Ativo – *Active Learning* (AL) – é um subcampo do aprendizado de máquina, cuja principal hipótese diz que: se o algoritmo de ML tiver permissão para escolher os

Figura 15 – Redes Neurais Artificiais são algoritmos de ML que procuram modelar como o pensamento humano funciona. Na figura, vemos um exemplo de uma rede neural simples com neurônios dispostos em três camadas, sendo uma de entrada, uma oculta e uma de saída.



Fonte: elaborado pelo autor.

dados com os quais aprende, ele terá uma fase de treinamento menor e um melhor desempenho (SETTLES, 2009). Reduzir a fase de treinamento é uma propriedade desejável para qualquer algoritmo de ML, visto que, muitas vezes, algoritmos de aprendizado supervisionado precisam ser treinados com milhares de instâncias rotuladas para apresentarem bons resultados.

Para alguns problemas, o processo de conseguir dados rotulados é fácil, como, por exemplo, a classificação de uma mensagem como *spam*. Nesse cenário, os usuários, ao marcarem um e-mail como *spam*, estão gratuitamente fornecendo uma instância rotulada que poderá ser usada no treinamento de um modelo de ML. No entanto, existem outros casos onde não é tão fácil obter dados rotulados. Como, por exemplo, no reconhecimento da fala, subárea da computação que desenvolve tecnologias que permitem o reconhecimento e a transcrição da linguagem falada de maneira automática. Zhu (2005) relata que um minuto de fala pode levar até dez minutos para se rotular, e que anotar fonemas pode demorar até 400 vezes mais.

Os preços associados à rotulação manual variam em relação à quantidade e o tipo de objeto rotulado. Por exemplo, no serviço de rotulação fornecido pela *Amazon*, a classificação de um texto custa 12 centavos de dólar, enquanto a classificação dos *pixels* de uma imagem (segmentação semântica), custa 82 centavos de dólar (AMAZON, 2021). Rotular imagens e

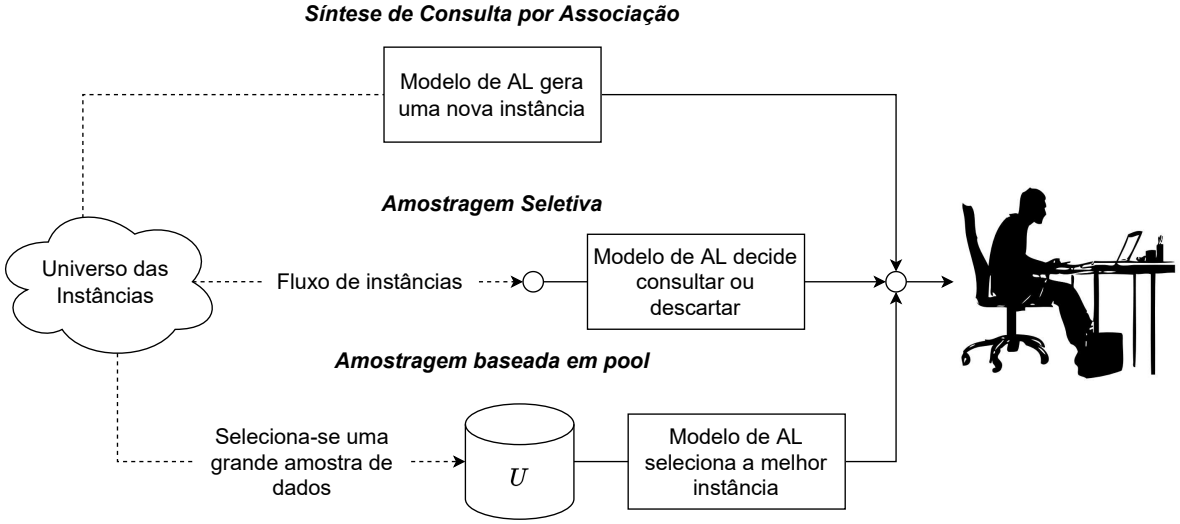
classificar textos é um processo relativamente fácil e que qualquer pessoa pode fazer. No entanto, classificar vulnerabilidades (o cerne da presente pesquisa) é um processo complexo que só pode ser realizado por especialistas com anos de experiência na área. Assim, fica claro como o processo de rotulação pode ser complexo, caro e requer muitas vezes a atenção de pessoas qualificadas (SOROKIN; FORSYTH, 2008).

O AL tenta superar essa limitação da falta de dados rotulados, selecionando a instância com as características mais significativas (obtida do conjunto de valores não rotulados) e submetendo-a a um especialista para rotulação. Esse processo se difere do aprendizado passivo, onde todos os rótulos são conhecidos desde o princípio, visto que no AL os dados são escolhidos interativamente durante a fase de treinamento (SETTLES, 2009). A seguir discutiremos as diferentes abordagens do aprendizado ativo e os métodos de seleção das instâncias (estratégias de consulta).

3.2.1 Diferentes Abordagens do Aprendizado Ativo

Nesta subseção explicaremos em detalhes as três diferentes abordagens de AL, encontradas na literatura, que podem ser utilizadas para resolver problemas computacionais. Além disso, podemos ver o fluxo resumido de cada abordagem na Figura 16, a seguir.

Figura 16 – O aprendizado ativo pode selecionar as instâncias a serem rotuladas de três formas diferentes, que são: síntese de consulta por associação, amostragem seletiva ou amostragem baseada em *pool*. Essas abordagens diferem entre si de acordo com a forma que novas instâncias são "ofertadas" ao modelo de AL.



Fonte: adaptado de Settles (2009).

A primeira abordagem é a Síntese de Consulta por Associação – *Membership Query*

Synthesis –, que permite que o modelo de AL selecione aleatoriamente qualquer instância não rotulada, bem como, gere uma nova instância sinteticamente (SETTLES, 2009). Este cenário não se aplica a todos os casos, pois envolve a geração de dados arbitrários, que podem ser difíceis de rotular por seres humanos. Por exemplo, os autores em Baum e Lang (1992) utilizaram essa abordagem para rotular dados em um problema de classificação de caracteres manuscritos. Muitos dos caracteres gerados pelo modelo de AL foram irreconhecíveis pelos seres-humanos que estavam participando do processo.

A segunda abordagem é a Amostragem Seletiva – *Selective Sampling* – (também chamada de amostragem em fluxo). Nela assume-se que não custa nada conseguir um dado não-rotulado. Dessa forma, todos os dados são apresentados ao modelo de AL de forma sequencial e cabe-lhe decidir se uma instância deve ou não ser rotulada. Essa decisão pode ser feita de diversas maneiras, como, por exemplo, a utilização de uma estratégia de consulta (descrita na Subseção 3.2.2) que garanta que as instâncias mais significativas tenham maiores probabilidades de serem selecionadas.

Por último temos a abordagem chamada de Amostragem Baseada em *Pool – Pool Sampling* –, que assume a existência de um pequeno conjunto de dados rotulados L e um grande conjunto de dados não rotulados U (SETTLES, 2009). Nessa abordagem, aplica-se no conjunto de dados não rotuladas U uma fórmula capaz de medir o ganho que a rotulação de cada instância traria para o modelo. Em seguida, a instância mais relevante é selecionada e apresentada ao especialista para ser rotulada. Essa abordagem é a mais utilizada por modelos de AL, visto que muitos problemas de aprendizado se encaixam nesse cenário.

3.2.2 Estratégias de Consulta

A principal diferença entre o aprendizado ativo e o passivo é a capacidade de o modelo selecionar os dados mais significativos para o treinamento, com base nos rótulos aplicados às instâncias passadas. Isso é feito por meio de técnicas capazes de medir a informatividade das instâncias, isto é, quanto a rotulação de uma determinada instância melhora o desempenho do modelo de AL como todo. A seguir, descreveremos as duas estratégias mais populares para medição da informatividade, que são a consulta por incerteza e por comitê.

3.2.2.1 Consulta por Incerteza

Consulta por incerteza é provavelmente a técnica mais simples e utilizada para medir a informatividade dos dados (SETTLES, 2009). Nessa técnica, o modelo de AL seleciona a instância que possui a menor certeza em relação a qual rótulo aplicar. A intuição por trás desse processo é de que, após a rotulação dessa instância, o modelo aprenderá a rotular um novo conjunto de dados que até então não conseguia, dado a falta de informações sobre ela. Para explicar o funcionamento das duas principais estratégias de consulta por incerteza, utilizaremos as informações contidas no Quadro 2, que apresenta a probabilidade de certeza em relação a cada rótulo que pode ser aplicado às instâncias i_1 e i_2.

Quadro 2 – Duas instâncias fictícias i_1 e i_2, com os respectivos valores de certeza (probabilidade) em relação aos rótulos *A*, *B* e *C*, que podem ser aplicados pelo modelo de AL.

Instância	**Rótulo** *A*	**Rótulo** *B*	**Rótulo** *C*
i_1	0,9	0,09	0,01
i_2	0,2	0,5	0,3

Fonte: elaborado pelo autor.

Na primeira estratégia, chamada de Menor Confiança – *Least Confidence* (LC) –, o modelo de AL selecionará a instância em que possui a menor confiança em relação ao seu rótulo mais provável. O valor de menor confiança é calculado por

$$\hat{y} = \underset{x}{argmax}(1 - P_\theta(\hat{y}|x)), \tag{3.1}$$

onde $P_\theta(\hat{y}|x)$ representa a probabilidade do rótulo $\hat{y}$ ser aplicado a instância analisada x. Se analisarmos o Quadro 2, podemos perceber que para instância i_1, o modelo possui 90% de certeza de que o rótulo a ser aplicado é o *A*. Já para instância i_2, o rótulo mais provável apresenta apenas 50% de certeza. Assim, essa estratégia escolheria a instância i_2 para ser rotulada, pois é a que possui menor confiança.

A segunda estratégia, chamada de Amostragem por Entropia – *Entropy Sampling* –, que utiliza as probabilidades de todos os possíveis rótulos para selecionar a instância mais informativa. Essa técnica utiliza uma medida bastante popular chamada de entropia, que é a média de incerteza inerente aos possíveis resultados da instância analisada (SETTLES, 2009). O valor de entropia pode ser calculado por

$$H = -\sum_k P(y_k) \log P(y_k), \tag{3.2}$$

que representa o somatório da multiplicação entre os valores de probabilidade de todos os rótulos, representados por $P(y_k)$, multiplicados pelos logaritmos dessas probabilidades. Assim,

se calcularmos os valores de entropia para as instâncias i_1 e i_2, presentes no Quadro 2, obteremos os valores de 0.155 e 0.447 respectivamente. Dessa forma, mais uma vez, o modelo de AL selecionaria a instância i_2 para ser rotulada.

3.2.2.2 Consulta por Comitê

A consulta por comitê é uma abordagem que envolve a utilização de um conjunto de modelos de AL treinados em hipóteses concorrentes, evitando qualquer tipo de viés que exista no treinamento de um modelo único. Para isso, utiliza-se um comitê de modelos, $C = \{\theta^{(1)}, ..., \theta^{(n)}\}$, treinados com o mesmo conjunto de dados rotulados L, para selecionar novas instâncias a serem rotuladas (SETTLES, 2009). Nessa abordagem, as instâncias não rotuladas do conjunto U são apresentadas aos membros do comitê C, que devem sugerir um rótulo para ser aplicado. A instância cujos membros mais discordarem em relação a qual rótulo aplicar, será a instância selecionada para ser rotulada pelo especialista. As duas principais técnicas para se calcular essa discordância são a Entropia dos Votos – *Vote Entropy* – e a divergência média de *Kullback-Leibler – Kullback-Leibler Divergence* (KL).

A Entropia dos Votos (DAGAN; ENGELSON, 1995) é uma generalização da estratégia de Amostragem por Entropia utilizada na consulta por incerteza. O valor de entropia para a instância analisada pode ser calculada por

$$H = \underset{x}{\text{argmax}} - \sum_{k} \frac{V(y_k)}{C} \log \frac{V(y_k)}{C}, \tag{3.3}$$

onde y_k corresponde a um dos rótulos possíveis, $V(y_k)$ é a quantidade de votos que um determinado rótulo recebeu entre todos os membros do comitê e C é a quantidade de membros no comitê $\mathscr{C}$.

Já a divergência média de *Kullback-Leibler* (MCCALLUMZY; NIGAMY, 1998) quantifica o quanto duas distribuições de probabilidade diferem entre si. Assim, se considerarmos y_k como todos os possíveis rótulos que podem ser aplicados à instância x, $\theta^{(c)}$ como um dos modelos pertencentes ao comitê e $\mathscr{C}$ como o próprio comitê, podemos calcular o valor médio de KL através da fórmula

$$x^*_{\text{KL}} = \underset{x}{\text{argmax}} \frac{1}{C} \sum_{c=1}^{C} KL(P_{\theta^{(c)}} || P_{\mathscr{C}}), \tag{3.4}$$

dado que:

$$\text{KL}(P_{\theta^{(c)}} || P_{\mathscr{C}}) = \sum_{k} P_{\theta^{(c)}}(y_k|x) \log \frac{P_{\theta^{(c)}}(y_k|x)}{P_{\mathscr{C}}(y_k|x)}, \tag{3.5}$$

onde $P_{\theta^{(c)}}(y_k|x)$ corresponde à probabilidade do rótulo y_k ser aplicado à instância x pelo modelo θ^c e $P_{\mathscr{C}}(y_k|x)$ é a probabilidade acordado em consenso pelo comitê $\mathscr{C}$ de que o rótulo y_k é o certo. Portanto, esta medida de discordância considera a instância mais significativa como sendo a que apresenta a maior diferença média entre as distribuições do rótulo aplicado por qualquer membro do comitê e o consenso.

3.3 Calibração de Probabilidades

Alguns modelos de ML podem fornecer estimativas ruins de probabilidade, isto é, quando as probabilidades das previsões, em média, não são próximas das probabilidades verdadeiras. Além disso, existem alguns modelos que não oferecerem suporte para esse tipo de previsão, como, por exemplo, o SVM. Para solucionar essa limitação, utilizamos uma técnica chamada de calibração de probabilidade. Essa técnica permite calibrar as probabilidades de um determinado modelo ou adicionar suporte para previsão de probabilidade caso o mesmo não possua (NICULESCU-MIZIL; CARUANA, 2005).

Abordaremos nessa seção, duas técnicas de calibração bastante utilizadas. A primeira delas, utiliza uma regressão sigmoide, baseada no modelo logístico de Platt (PLATT, 2000) e representada por

$$p(y_i = 1|f_i) = \frac{1}{1 + exp(Af_i + B)}, \tag{3.6}$$

onde y_i é o rótulo verdadeiro da amostra i e f_i é a saída do classificador não calibrado para amostra i. A e B são números reais a serem determinados de modo a ajustar a saída dada para que ela possua máxima verossimilhança com o rótulo verdadeiro.

Já a segunda, é uma técnica não paramétrica que pode ser utilizada para construir um mapeamento entre valores de acurácia e probabilidades utilizando uma função de regressão isotônica (ZADROZNY; ELKAN, 2002) representada, por

$$y_i = m(f_i) + \varepsilon_i, \tag{3.7}$$

onde m é uma função de mapeamento monotonicamente crescente. Assim, dado um conjunto de treinamento (f_i, y_i), o problema de regressão isotônica é encontrar o valor de $\hat{m}$, tal que

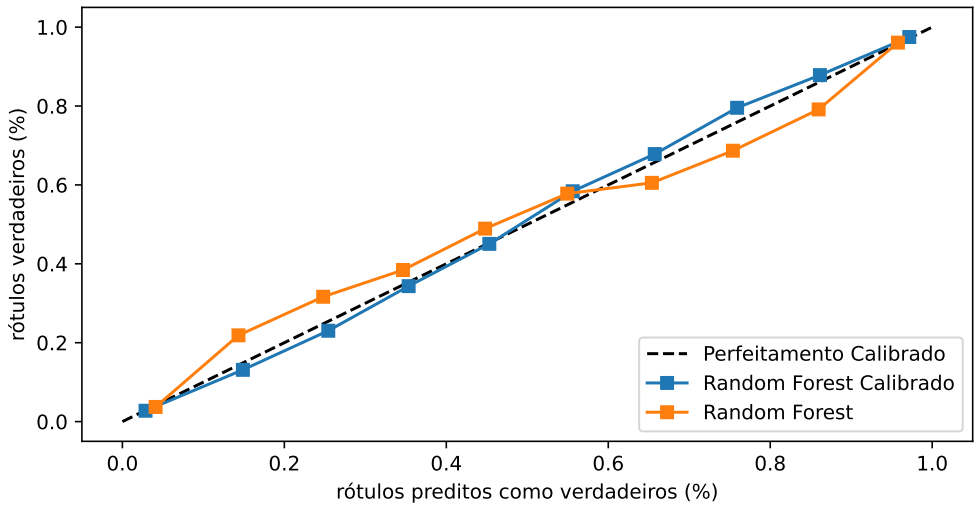
$$\hat{m} = \underset{z}{\operatorname{argmin}} \sum (y_i - z(f_i))^2, \tag{3.8}$$

onde y_i é o rótulo verdadeiro da amostra i e f_i é a saída do classificador não calibrado para amostra i.

É importante destacar que ambas as técnicas de calibração suportam apenas dados unidimensionais, por exemplo, cuja saída do modelo é binária. No entanto, estas técnicas podem ser estendidas para classificação multiclasse, utilizando uma estratégia conhecida como "um contra todos" – *one-vs-all* –. Nesta estratégia, treinamos um classificador para cada classe e ajustamos sua probabilidade contra todas as outras. Dessa forma, as probabilidades calibradas para cada classe são previstas separadamente e como elas não somam necessariamente um, um pós-processamento nos utilizando uma função *softmax* é realizado para normalizá-las. A vantagem dessa técnica é que, como cada classe é representada por apenas um classificador, fica fácil interpretar o resultado das análises feitas.

Por fim, para verificarmos se o resultado do nosso modelo está calibrado, basta desenharmos um gráfico relacionando a porcentagem de rótulos verdadeiros (eixo y) em comparação a porcentagem de rótulos previstos como verdadeiro (eixo x). Esse tipo de gráfico é chamado de diagrama de confiabilidade ou curva de calibração. Um modelo bem calibrado possuirá pontos próximos da diagonal principal, partindo do canto inferior esquerdo ao canto superior direito do gráfico, como pode ser visto na Figura 17.

Figura 17 – A curva de calibração relaciona a porcentagem de rótulos verdadeiros (eixo y) com a porcentagem de rótulos previstos como verdadeiro (eixo x). Modelos bem calibrados possuem valores próximos da reta $y = x$.



Fonte: elaborado pelo autor.

3.4 Explicabilidade da AI

Um dos maiores desafios da área de AI é fornecer soluções que possam ser compreendidas pelos seres humanos. Visto que muitos modelos de ML funcionam como "caixas pretas", que impossibilitam a explicação do porquê da AI ter chegado em conclusões específicas (BURKART; HUBER, 2021). Isso é um problema em muitos domínios, principalmente naqueles onde tais soluções não funcionam de forma independente e sim como apoio às decisões tomadas por especialistas. Como exemplo, temos o domínio da saúde, onde uma compreensão profunda da solução é essencial para garantir maior confiabilidade e completude nos diagnósticos realizados (HOLZINGER *et al.*, 2019).

Dessa necessidade surgiu a Inteligência Artificial Explicável – *Explainable Artificial Intelligence* (XAI) –, que é um conjunto de métodos e ferramentas que podem ser aplicados para auxiliar os usuários a entender ou prever, consistentemente, os resultados dados pelas soluções de AI (MOLNAR, 2022). A explicabilidade, ou interpretabilidade, pode ser abordada de duas formas: (*i*) globalmente, onde se analisa como os atributos das instâncias afetam coletivamente o resultado do modelo de AI; ou (*ii*) localmente, onde se explica, individualmente, como cada atributo contribuiu para a classificação de uma dada instância (BURKART; HUBER, 2021).

Além disso, o uso da XAI permite que os modelos de AI sejam validados antes de serem implantados em um ambiente de produção. Algo bastante importante, principalmente quando esses modelos são utilizados na tomada de decisões em áreas críticas como saúde e segurança (LIPTON, 2018). Além de explicar o funcionamento do modelo, a interpretabilidade confere às soluções de AI três importantes características:

- **Confiabilidade:** ao se explicar o processo de tomada de decisão, evidenciando os pontos fracos e fortes, a XAI ajuda a apoiar os resultados dos modelos de AI com evidências concretas. Assim, fica mais fácil para os usuários confiar nas soluções apresentadas (LIPTON, 2018);
- **Responsabilidade:** quando um modelo de AI toma uma decisão errada ou desonesta, ele deve ser responsabilizado pelas suas ações. A XAI garante que as decisões sejam explicadas, justificadas e rastreadas, oferecendo assim as ferramentas necessárias para que essas falhas não ocorram novamente no futuro (FREITAS, 2014);
- **Conformidade:** com o avanço das leis de proteção de dados pessoais, é imprescindível que os modelos de AI sejam transparentes e informativos em relação ao uso das informações. Dessa forma, a XAI garante-se que as soluções de AI estejam em conformidade com as

políticas da empresa, padrões da indústria e regulamentos governamentais (GOODMAN; FLAXMAN, 2017).

3.4.1 Explicações Aditivas de Shapley

É importante destacar que as técnicas de explicabilidade podem ser aplicadas em todos os tipos de modelos, desde os mais simples, como os modelos lineares, até os mais complexos, como as redes neurais. Além disso, a maioria dessas ferramentas são consideradas soluções agnósticas, por concentrarem suas análises apenas nos pares de entrada e saída dos modelos de AI. Assim, por questão de brevidade, concentraremos nossa discussão em apenas uma dessas ferramentas, a chamada Explicações Aditivas de *Shapley – Shapley Additive Explanations* (SHAP) (LUNDBERG; LEE, 2017).

O SHAP é uma ferramenta de XAI, capaz de obter o impacto de cada atributo de uma determinada instância para a escolha do rótulo pelo modelo de AI (LUNDBERG; LEE, 2017). Isso é feito através do cálculo dos valores de *Shapley*, um conceito da teoria dos jogos cujo objetivo é medir a contribuição de cada jogador para a vitória de um determinado jogo (FUDENBERG; TIROLE, 1991). A intuição é que se n jogadores participam coletivamente para obter uma recompensa p, então se encontrarmos o quanto cada jogador contribuiu para a vitória, conseguiríamos dividir justamente a recompensa do jogo.

Assim, para calcular o valor de *Shapley* de um determinado atributo j de uma instância de nosso problema, precisamos obter o valor médio de sua contribuição marginal. Nesse contexto, contribuição marginal é a diferença que a presença ou ausência desse atributo tem para o resultado das possíveis combinações (coalizões) que podem ser geradas com os N atributos da instância. Esse valor pode ser obtido substituindo o modelo de ML original, por um modelo local $g(z')$, representado por

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j, \tag{3.9}$$

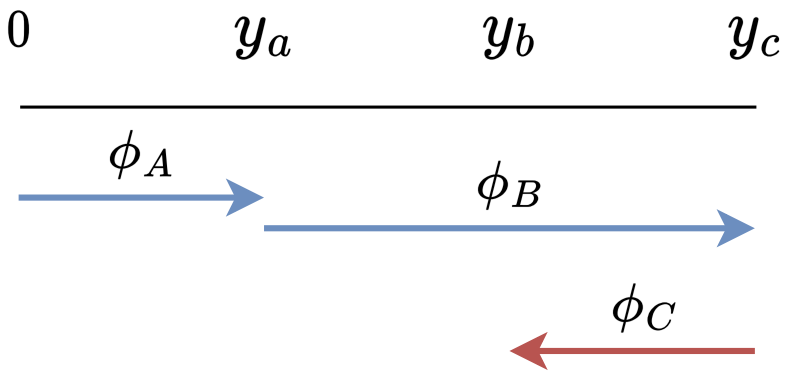
onde $z' \in \{0,1\}^M$ representa o vetor de coalizões, M define o número máximo de coalizões e $\phi_j \in R$ corresponde ao valor de *Shapley* do atributo j (MOLNAR, 2022). Dessa forma, um vetor de coalizões com valor igual a 0 indica que o atributo está ausente e com o valor 1 indica que ele está presente. Ademais, o valor de *Shapley* do atributo, ϕ_j, é dado por

$$\phi_j = \sum_{S \subseteq N \setminus \{j\}} \frac{|S|!(M-|S|-1)!}{M!} [f_x(S \cup \{j\}) - f_x(S)], \tag{3.10}$$

onde S corresponde ao subconjunto de todos os N atributos que não contém o atributo j, $\frac{|S|!(M-|S|-1)!}{M!}$ contabiliza o número de permutações de S, $f_x(S)$ é a saída esperada para o subconjunto S e $f_x(S \cup \{j\}) - f_x(S)$ representa a diferença associada a inserção do atributo j nas diferentes coalizões (LUNDBERG; LEE, 2017).

Para ilustrar graficamente como os valores de *Shapley* podem explicar o funcionamento do modelo de ML, considere uma instância que possa ser classificada em três classes distintas (y_a, y_b e y_c) e que possua três atributos (A, B e C), com três valores de *Shapley* (ϕ_A, ϕ_B e ϕ_C). Como mostrado na Figura 18, os atributos A, B e C tiveram impactos diferentes na classificação da instância, o que pode ser visto pelo comprimento e cor das setas ϕ_A, ϕ_B e ϕ_C. Os atributos A (ϕ_A) e B (ϕ_B) tiveram um impacto positivo (seta em azul), sendo que o impacto de B foi maior que o impacto de A (comprimento das setas). Por outro lado, o atributo C (ϕ_C) teve um impacto negativo (seta em vermelho). Como esse método é aditivo, se considerássemos apenas os atributos A (ϕ_A) e B (ϕ_B), o rótulo aplicado seria o y_c. No entanto, graças ao atributo C (ϕ_C), o rótulo final da instância será y_b.

Figura 18 – Interpretação gráfica de como os valores de *Shapley*, obtidos através da utilização do *Shap Kernel*, podem ajudar no entendimento do resultado. Na figura, podemos ver que os atributos A (ϕ_A) e B (ϕ_B) contribuíram positivamente e o atributo C (ϕ_C) negativamente para classificação final da instância como sendo da classe y_c.



Fonte: adaptado de Lundberg e Lee (2017).

Por fim, vale destacar que essas explicações são obtidas apenas para instâncias individuais. No entanto, graças ao caráter aditivo do SHAP, é possível combinar os valores locais de *Shapley* e obter explicações globais para o modelo de ML (LUNDBERG *et al.*, 2020). A ideia é bastante simples, atributos com altos valores absolutos de *Shapley* são mais importantes para o modelo e vice-versa. Assim, para encontrar o valor de importância global para um determinado atributo, basta calcular a média do somatório dos valores absolutos de *Shapley* desse atributo,

para cada uma das instâncias existentes, o que pode ser obtido através da equação

$$I_j = \frac{1}{n}\sum_{i=1}^{n} |\phi_j^i|, \tag{3.11}$$

onde I_j representa o valor de importância global para o atributo j, n o número total de instâncias e ϕ_j^i o valor de *Shapley* para o atributo j da instância i. Assim, ao se calcular os valores globais de *Shapley* para todos os atributos existentes no problema, é possível mensurar aqueles que mais contribuem para a classificação das instâncias.

3.5 Aplicações de ML em Segurança

Segundo Shaukat *et al.* (2020), as organizações costumam utilizar muitos sistemas tradicionais de segurança para se proteger de ataques cibernéticos. Entre esses sistemas, podemos citar soluções de Gerenciamento de Eventos e Informações de Segurança – *Security Information and Event Management* (SIEM) – (SHREAD, 2021), Sistemas de Detecção de Intrusão – *Intrusion Detection System* (IDS) (GUERCIO, 2021) –, Gerenciamento Unificado de Ameaças – *Unified Threat Management* (UTM) (AGHAM, 2016) –, *firewalls*, antivírus, etc. Apesar dessas soluções oferecerem um grau relativo de segurança para as organizações, elas não são adaptativas, pois funcionam a partir de regras pré-definidas de rede.

Esse funcionamento estático introduz pontos de falha na segurança das organizações, que podem ser explorados por agentes maliciosos que utilizam as mais avançadas técnicas de ataque. De acordo com um estudo realizado por Sharma *et al.* (2011), mais de 60% dos ataques cibernéticos são identificados após já terem ocorrido e causado danos às organizações. Assim, é necessário o desenvolvimento de novos métodos automatizados de segurança, que consigam lidar com esses desafios.

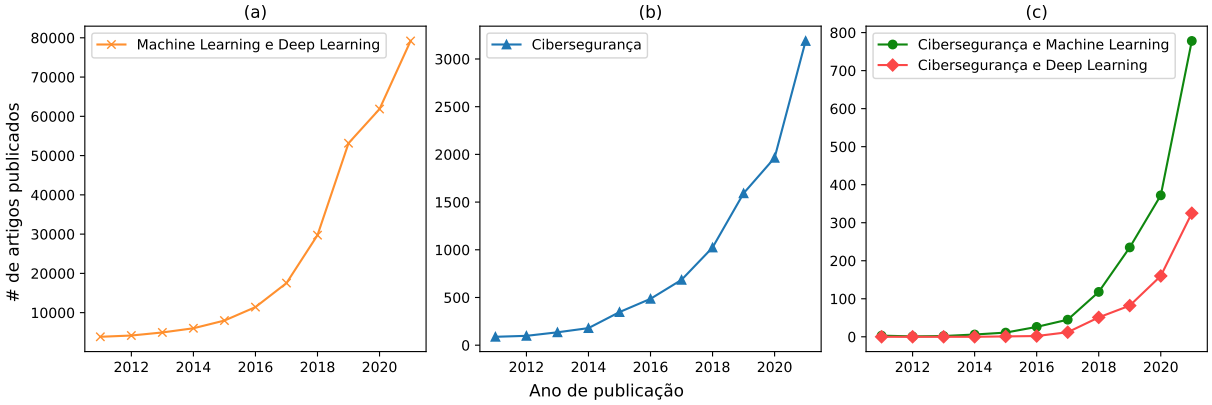
Sistemas que utilizam AI apresentam melhores resultados em relação à taxa de erro, ao desempenho e no tempo de resposta a ataques, em relação a sistemas tradicionais de segurança (GELUVARAJ *et al.*, 2019). Isso faz desses sistemas uma boa solução para ser aplicada em problemas de cibersegurança. Além disso, ML pode resolver o problema de escassez de mão de obra qualificada, visto que os algoritmos podem aprender com a experiência e agir de forma automática na presença de ameaças não detectadas anteriormente (SHAUKAT *et al.*, 2020).

A fim de descobrir a tendência na publicação de artigos nas áreas de ML, DL e cibersegurança nos últimos dez anos, realizamos uma pesquisa no site Scopus[1] (acessado em

[1] Banco de dados de resumos e citações de artigos para jornais/revistas acadêmicos: https://www.scopus.com/.

2 de julho de 2022) e o sumário dos achados pode ser visto na Figura 19. A quantidade de artigos publicados, independente da área, que utilizam ML ou DL cresceu consideravelmente nos últimos anos, como pode ser visto na Figura 19 (a). Observa-se também essa tendência de crescimento para a área de cibersegurança, como comentam os autores Prasad e Rohokale (2020) em seu artigo "*Artificial intelligence and machine learning in cyber security*" e como mostra a Figura 19 (b). Além disso, é possível notar na Figura 19 (c), que a quantidade de artigos publicados em cibersegurança que utilizam ML é maior que aqueles que utilizam DL, provavelmente por DL ser uma tecnologia emergente.

Figura 19 – Na última década, a quantidade de publicações na área de aprendizado de máquina e cibersegurança cresceram consideravelmente. Na figura, podemos observar esse crescimento, bem como, o aumento no número de publicações de segurança que utilizam ML e DL.



Fonte: elaborado pelo autor.

Fica claro que ML é uma técnica que está sendo amplamente utilizada na literatura para lidar com problemas de cibersegurança. Como não seria possível abordar todos os temas dessa área, decidimos entrar em detalhes nos três principais, que são: detecção e classificação de intrusão, *SPAM*s e *malwares*. Vale destacar que esses temas são abordados apenas neste capítulo de referencial teórico, a fim de se exemplificar o uso de importantes técnicas de ML para resolver problemas de cibersegurança, sendo o tema principal da tese a gestão de vulnerabilidades baseado no risco (conforme explicado na Seção 2.5).

3.5.1 Detecção de Intrusão

Sistema de Detecção de Intrusão – *Intrusion Detection System* (IDS) –, são capazes de identificar o acesso indevido a redes e sistemas de computadores. Eles podem ser classificados em três categorias: (*i*) detecção baseada no uso indevido, do inglês *misuse-based*, que monitora ataques conhecidos; (*ii*) detecção que monitora o comportamento normal, do inglês *anomaly-*

based, que procura diferenciá-lo do uso anormal da rede; e por fim, (*iii*) detecção que combina as abordagens anteriores, do inglês *hybrid-based*, para aumentar a acurácia durante a detecção de anomalias (KUMAR *et al.*, 2018).

Técnicas tradicionais de IDS se mostraram ineficientes, uma vez que elas procuram detectar ataques conhecidos e dada a complexidade que é para um ser humano modelar o comportamento normal de uma rede de computadores (SHAUKAT *et al.*, 2020). Assim, os invasores conseguem explorar, com êxito, falhas nesses sistemas e adentrar as redes, causando sérios problemas organizacionais. Por isso, para superar essas limitações, técnicas de ML capazes de detectar e prever de imediato invasões e até mesmo ataques futuros, estão sendo utilizadas nesse contexto.

Diversas técnicas de aprendizado supervisionado foram aplicadas em sistemas IDS do tipo *misuse-based* (ELHAG *et al.*, 2015; GHARAEE; HOSSEINVAND, 2016), *anomaly-based* (CASAS *et al.*, 2012; LIN *et al.*, 2015) e *hybrid-based* (CHANDRASEKHAR; RAGHUVEER, 2013; KIM *et al.*, 2014; SHAPOORIFARD; SHAMSINEJAD, 2017). Além disso, alguns trabalhos utilizaram técnicas de aprendizado não supervisionado, como a clusterização (AMOLI *et al.*, 2016) e redes neurais (CHOI *et al.*, 2019), para melhorar a acurácia dos sistemas. Os algoritmos de ML mais utilizados são ANN, DT e SVM, que foi considerada uma das melhores técnicas para desenvolver ferramentas de IDS (SHAUKAT *et al.*, 2020).

3.5.2 Detecção de SPAM

Correio eletrônico ou *e-mail* é um método de compartilhamento de mensagens entre indivíduos que sofre de um grave problema chamado de *SPAM* (LEE *et al.*, 2010; SELVARAJ, 2019). Os *SPAM*s são qualquer tipo de mensagens irrelevantes, não solicitados e indesejados que consomem tempo dos usuários, armazenamento e largura de banda na internet, diminuindo significativamente a eficiência da rede (ALSMADI; ALHAMI, 2015; SHAH; KUMAR, 2018). Em 2018, aproximadamente 230 bilhões de *e-mail*s foram trocados diariamente (MONITOR, 2019), dos quais, cerca de 55% eram *spams* (JOSEPH, 2021).

*E-mail*s são considerados o principal meio utilizado por criminosos (conhecidos como *spammers*) para realizar ataques de *SPAM*. No entanto, *Facebook*, *Twitter*, *YouTube* e outras plataformas sociais, também são explorados. Assim, pesquisadores desenvolveram inúmeras técnicas de filtragem de *SPAM* (GOMES *et al.*, 2004; ABDELRAHIM *et al.*, 2013). No entanto, essas técnicas costumam ser facilmente burladas, pois os *spammers* são inteligentes o suficiente

para alterar as palavras-chave bloqueadas nesses filtros (CHAKRABORTY *et al.*, 2016).

Para solucionar esse problema, pesquisadores têm aplicado várias técnicas de ML que conseguem evitar os ataques de *SPAM* com mais eficiência. Entre os domínios aplicados, podemos citar a classificação de *SPAM* (ALSMADI; ALHAMI, 2015; SHAH; KUMAR, 2018; CHANDRASEKAR; PRIYATHARSINI, 2018), filtragem de *SPAM* (GOMES *et al.*, 2004; ELHADI *et al.*, 2012) e identificação de *SPAM* (CASTILLO; DAVISON, 2010; STERN, 2008). Em relação às técnicas, são comumente utilizados os algoritmos DT, RF e SVM.

3.5.3 Detecção de Malware

Um programa malicioso, mais conhecido como "*malware*" (combinação entre "mal" de "malicioso" com "*ware*" de "*software*"), é um trecho de código inserido secretamente em sistemas ou redes de computadores com a intenção de explorar vulnerabilidades de segurança. O *malware* compromete a integridade, confidencialidade e disponibilidade do *hardware* ou *software* de suas vítimas. Vírus, *ransomwares*, *worms*, *trojans*, *spyware* e *adware* são exemplos comuns de *malwares* (PATHAK, 2016; GUO *et al.*, 2016)

Em 2019, o número de *malwares* conhecidos ultrapassou a marca de 800 milhões, conforme o relatório técnico apresentado pela McAfee (BEEK *et al.*, 2019) e espera-se que esse mercado gere uma receita de aproximadamente 8 bilhões de dólares americanos até 2025 (JOHNSON, 2021). É importante destacar que os *malwares* não são um problema exclusivo de grandes empresas. Nos últimos anos, o número de usuários desavisados com seus computadores infectados e transformados em *bots* (usados na mineração de criptomoedas e em ataques de DDoS) vem crescendo bastante (DOSSETT, 2021).

No passado, diversas técnicas foram usadas para a detecção de *malware* baseado em sua assinatura (uma sequência contínua de *bytes* que permite identificar um artefato malicioso). No entanto, essas técnicas não conseguem detectar um tipo de *malware* chamado de *zero-day* (aqueles que ainda são desconhecidos). Por serem incapazes de identificar esse tipo de *malware*, além daqueles que foram apenas "ofuscados", as técnicas de ML começaram a ser utilizadas nessa área (GANDOTRA *et al.*, 2014; SUN *et al.*, 2018; AFEK *et al.*, 2019). SVM é a abordagem de classificação de ML mais estudada, com uma porcentagem de uso de 29%, seguido pelas DT com aproximadamente 17% (KUMPULAINEN *et al.*, 2008).

3.6 Os Desafios do uso de ML em Cibersegurança

A vasta maioria das aplicações de aprendizado de máquina pressupõem que o ambiente que operam é seguro e não se preocupam em adotar medidas de segurança para se protegerem (PITROPAKIS *et al.*, 2019). No entanto, é ingênuo achar que usuários mal-intencionados não possam explorar falhas nesses projetos de ML para lançar ataques devastadores contras tais soluções. Portanto, é importante que os cientistas de dados implementem medidas de segurança para proteger suas aplicações. Para isso, é necessário conhecer os principais tipos de ataques perpetrados contra modelos de ML. A seguir, discutiremos três dos principais ataques.

O primeiro ataque, conhecido como Envenenamento de Dados – *Data Poisoning* –, ocorre quando um atacante tenta poluir o conjunto de dados de treinamento, com o intuito de produzir um mau classificador (GOLDBLUM *et al.*, 2022). Nesse ataque, o atacante pode inserir novas instâncias, editar ou remover instâncias existentes do conjunto de treinamento. Esses dados manipulados são consideradas fontes de ruído que podem fazer com o modelo de aprendizado de máquina classifique erroneamente novas instâncias, diminuindo assim sua acurácia e confiabilidade.

O segundo ataque, conhecido como Roubo de Modelo – *Model Stealing* –, é quando um atacante explora o acesso fornecido a aplicação de ML, por exemplo, através de uma Interface de Programação de Aplicação – *Application Programming Interface* (API) – pública, para duplicar a funcionalidade do modelo, ou seja, "roubá-lo" (TRAMÈR *et al.*, 2016). Para isso, o usuário mal-intencionado realiza diversas consultas ao modelo para obter respostas para um conjunto de amostras específico. Em seguida, utiliza um modelo substituto, por exemplo, uma rede neural, para aprender a relação entre as amostras e as respostas dadas pelo modelo original. Com uma quantidade suficiente de amostras, é possível criar um modelo que possua um desempenho semelhante ao que está sendo roubado.

O terceiro ataque, conhecido como Inversão de Modelo – *Model Inversion* –, é quando o atacante, com acesso ao modelo de ML, consegue, por meio de engenharia reversa, descobrir os atributos que geraram a classificação da classe alvo (FREDRIKSON *et al.*, 2015). Neste caso, é possível expor informações do conjunto de dados de treinamento, que podem ser potencialmente privadas. Os ataques de inversão de modelo podem ser classificados como caixa branca – *white-box* –, que é quando o atacante conhece o funcionamento do modelo utilizado na aplicação, ou caixa preta – *black-box* –, onde o funcionamento é desconhecido.

Apesar de reconhecermos a importância de proteger as soluções de AI de ataques

que possam ser realizados por usuários mal-intencionados, esse tema não será abordado na proposta deste trabalho, devido à necessidade de focarmos nossa atenção e esforços no cerne do nosso problema: o desenvolvimento de uma solução que auxilie os profissionais de segurança no processo de gestão de vulnerabilidades baseado no seu risco de exploração.

4 TRABALHOS RELACIONADOS

Neste capítulo, descreveremos as bases de dados utilizadas na revisão da literatura (Seção 4.1), os principais trabalhos relacionados a construção de *datasets* de segurança (Seção 4.2) e sistemas RBVM na área de cibersegurança (Seção 4.3), bem como suas respectivas *strings* de buscas.

4.1 Bases de Pesquisa

Buscamos por artigos publicados que possuam títulos, resumos e palavras-chave que se encaixam nas "*strings* de busca", nas seguintes bases de dados: *IEEE Xplore*[1], *ACM digital library*[2], *SpringerLink*[3] e *ScienceDirect*[4]. Além disso, para garantir a completude dos resultados encontrados, realizamos a mesma pesquisa nos agregadores *Web of Science*[5], *Google Scholar*[6] e *Scopus*[7]. Essas bases de dados foram escolhidas por serem bastante conhecidas e oferecerem os trabalhos revisados por pares, o que lhes confere uma maior credibilidade.

4.2 *Datasets* de Segurança

Esta seção descreve os trabalhos relacionados que desenvolveram conjuntos de dados relacionados à vulnerabilidade na última década. Discutimos seus pontos fortes e fracos e como eles diferem do conjunto de dados desenvolvido no nosso trabalho. Finalmente, a Tabela 1 resume nossas descobertas.

4.2.1 String de Busca

Para identificar artigos relevantes que propuseram a criação de *datasets* contendo vulnerabilidades de segurança, buscamos pelas *strings* ("*SECURITY VULNERABILITIES*" *OR* "SOFTWARE VULNERABILITIES"), ("*COMMON VULNERABILITIES AND EXPOSURES*" *OR* "*CVE*"), ("*NATIONAL VULNERABILITY DATABASE*" *OR* "*NVD*") e (*"DATASET"*) nas bases mencionadas no começo deste capítulo. No total, foram encontrados 473 trabalhos. Após

[1] Banco de dados de jornais e artigos acadêmicos publicados pelo *IEEE*: https://ieeexplore.ieee.org/.
[2] Banco de dados de jornais e artigos acadêmicos publicados pela *ACM*: https://dl.acm.org/.
[3] Banco de dados de jornais e artigos acadêmicos publicados pela *Springer*: https://link.springer.com/.
[4] Banco de dados de jornais e artigos acadêmicos publicados pela *ScienceDirect*: https://www.sciencedirect.com/.
[5] Agregador de diversas bases de dados contendo publicações acadêmicas: https://www.webofscience.com.
[6] Agregador de diversas bases de dados de publicações acadêmicas: https://scholar.google.com.
[7] Agregador de diversas bases de dados de publicações acadêmicas: https://www.scopus.com/.

a remoção de trabalhos duplicados e que tivessem sido publicados antes de 2016, sobraram 443 artigos. Analisamos os títulos e resumos desses 443 trabalhos, removendo aqueles que não fossem relevantes. Ao todo, selecionamos 6 artigos, discutidos em detalhes a seguir. Acreditamos que os termos selecionados cobrem a maioria, senão todos, os trabalhos de criação de *dataset* de vulnerabilidades de segurança.

4.2.2 Discussão dos Artigos

Bhandari *et al.* (2021) criaram uma ferramenta chamada de *CVEfixes* que busca automaticamente por todos os registros de CVEs, disponíveis no NVD, que possuam um repositório de controle de versão (*GitHub*) associado. Os autores analisaram os *commits* presentes nesses repositórios e enriqueceram o conjunto de dados acerca das vulnerabilidades, com metadados, tais como, a linguagem de programação e a presença de *patches* de correção. A versão inicial do *CVEfixes* contém informações sobre 5.365 CVEs extraídos de 5.495 *commits* de 1.754 projetos de código aberto.

Fan *et al.* (2020) criaram um conjunto de dados chamado *Big-Vul*, contendo 3.754 CVEs publicados entre 2002 e 2019. O *Big-Vul* abrange um total de 348 projetos hospedados no *GitHub* e escritos utilizando as linguagens de programação C/C++. Além disso, os autores extraíram e adicionaram informações sobre os CVEs recuperadas do *site CVEdetails.com*, um banco de dados gratuito de informações sobre vulnerabilidades de segurança. O resultado é um conjunto de dados contendo trechos de código antes e depois da aplicação de *patchs* de segurança, bem como informações do CVSS.

Jimenez *et al.* (2018) apresentaram o *VulData7*, um *framework* capaz de coletar automaticamente vulnerabilidades de segurança presentes em quatro projetos de código aberto escritos em C, que são: *Linux Kernel*, *WireShark*, *OpenSSL* e *SystemD*. O conjunto de dados resultante contém informações sobre as vulnerabilidades (por exemplo, CVE, descrição, severidade, etc.), código (lista de versões afetadas) e *patches* de segurança correspondentes. Os autores conseguiram obter dados acerca de 2.800 vulnerabilidades e 1.600 *patches* de correção. No entanto, a fonte de dados para os CVEs, são arquivos do tipo XML, que não são mais fornecidos pelo NVD.

Gkortzis *et al.* (2018) apresentaram o *VulinOSS*, um conjunto de dados contendo 17.738 vulnerabilidades de 153 projetos de código aberto. Os dados foram coletados mapeando os metadados do projeto (por exemplo, tags de *commits*, *branches*, etc.) para as informações do

CVE obtidas através do NVD. O conjunto de dados fornece informações sobre o CVE, métricas de código e dados relacionados às tendências de desenvolvimento modernas, como integração e teste contínuos. Os autores consideram apenas uma versão mais antiga do CVSS, ou seja, a versão 2.0[8].

Alves *et al.* (2016) analisaram os sistemas de rastreamento de *bugs* de cinco projetos de código aberto escritos em C/C++ (*Linux Kernel*, *Mozilla*, *Xen Hypervisor*, *httpd* e *glibc*) e procuraram por ocorrências de CVEs. Ao todo, 2.875 falhas de segurança foram encontradas. Então, os autores usaram os *commits* associados aos CVEs para criar um conjunto de dados, rotulados manualmente, de códigos vulneráveis e não vulneráveis. Em seguida, os autores investigaram se as métricas de *software* (por exemplo, número de linhas de código, profundidade de aninhamento de função, etc.) se correlacionavam com o número de vulnerabilidades presentes nas aplicações.

Ao contrário dos trabalhos anteriores, Ponta *et al.* (2019) criaram manualmente um conjunto de dados compreendendo 624 vulnerabilidades divulgadas publicamente que afetam 205 projetos Java de código aberto. O conjunto de dados apresenta informações sobre o CVE, descrição, severidade, *patches* de segurança, etc. Os autores alegaram que 29 vulnerabilidades encontradas não possuem um CVE e 46 não tinham informações adicionais além do identificador CVE no NVD no momento da publicação.

A pesquisa bibliográfica mostrou trabalhos relevantes que buscam criar conjuntos de dados relacionados à cibersegurança. No entanto, eles têm um escopo limitado, concentrando suas análises em vulnerabilidades presentes em grandes projetos de código aberto desenvolvidos em uma determinada linguagem de programação. Além disso, os conjuntos de dados não abordam vulnerabilidades de *hardware* e sistema operacional (exceto aquelas que afetam o *kernel* do Linux). Ademais, poucos consideram múltiplos feeds de segurança e nenhum traz dados sobre inteligência de ameaças, o que é essencial para avaliar o risco de exploração das vulnerabilidades e o impacto na infraestrutura das organizações. Assim, nosso trabalho propõe superar essas limitações criando um conjunto de dados de segurança de propósito geral com características das vulnerabilidade e informações de inteligência de ameaças coletadas de dez fontes diferentes, incluindo avisos de segurança de fabricantes e de redes sociais.

[8] Guia da versão 2.0 do CVSS: https://www.first.org/cvss/v2/guide.

Tabela 1 – Comparação entre os trabalhos relacionados que desenvolvem conjuntos de dados contendo informações sobre vulnerabilidades de segurança e a nossa proposta. A tabela mostra que nosso trabalho é o único que considera vulnerabilidades de *hardware* e traz informações sobre inteligência de ameaças. Além disso, nosso trabalho é o que utiliza a maior quantidade de fontes de dados na coleta de informações e o que possui o maior número de vulnerabilidades.

Referência	**Característica das Vulns.**	**Inteligência de Ameaças**	**# fontes**	**Plataformas**	**Tamanho**
Bhandari *et al.* (2021)	×		2	Software & OS (Linux Kernel)	5.365
Fan *et al.* (2020)	×		2	Software & OS (Linux Kernel)	3.754
Jimenez *et al.* (2018)	×		2	Software & OS (Linux Kernel)	2.800
Gkortzis *et al.* (2018)	×		2	Software	17.738
Alves *et al.* (2016)	×		4	Software & OS (Linux Kernel)	2.875
Ponta *et al.* (2019)	×		2	Software	624
Este Trabalho	×	×	10	Hardware, Software & OS	200.000+

4.3 Ferramentas de RBVM

Nessa seção, discutiremos os principais trabalhos que desenvolvem soluções de RBVM. A Tabela 2 resume o foco dos trabalhos encontrados na literatura, destacando as principais diferenças entre eles e a nossa proposta, o FRAPE.

4.3.1 String de Busca

Para identificar as contribuições relevantes para a área de gestão de riscos, buscamos pelas *strings* (“*RISK ASSESSMENT*” *AND* “*CYBERSECURITY*”), (“*VULNERABILITY MANAGEMENT*” *AND* “*CYBERSECURITY*”), (”*VULNERABILITY PRIORITIZATION*” *AND* “*CYBERSECURITY*”) e (“*IMPROV**” *AND* “*CVSS*”) nas bases mencionadas no começo deste capítulo. No total, foram encontrados 1,224 trabalhos. Após a remoção de trabalhos duplicados

e publicados antes de 2016, sobraram 850 artigos. Analisamos os títulos e resumos desses 850 trabalhos, removendo aqueles que não fossem relevantes e que não tratassem da gestão e risco de vulnerabilidades de *hardware*, *software* ou sistemas operacionais (o ponto central da nossa pesquisa). Ao todo, selecionamos 11 trabalhos (8 artigos e 3 *whitepapers*) discutidos em detalhes a seguir. Acreditamos que os termos selecionados cobrem a maioria, senão todos, os trabalhos de gestão de risco em cibersegurança.

4.3.2 Discussão dos Artigos

Kure *et al.* (2022) desenvolveram uma abordagem para avaliação de risco que possui três partes. Na primeira parte, os ativos são classificados, usando lógica difusa, de acordo com sua criticidade. Para isso, os analistas precisam responder cinco perguntas sobre o impacto que a exploração do ativo teria na Confidencialidade, Integridade e Disponibilidade – *Confidentiality, Integrity and Availability* (CIA) –, do mesmo e quanto tempo seria necessário para se recuperar do ataque. Na segunda parte, um modelo de ML é utilizado para prever se o ativo está vulnerável a algum dos dez ataques cibernéticos analisados. E por fim, na terceira parte, os analistas de segurança avaliam se os controles de segurança empregados pela empresa estão em conformidade com o ISO 27005 (FIROIU, 2015). Experimentos demonstraram a eficácia da classificação da criticidade dos ativos e da avaliação dos controles de segurança. No entanto, o modelo de ML obteve uma acurácia inferior a 40%. Além disso, as partes funcionam independentemente e não se comunicam entre si, obrigando os especialistas a realizar a análise final de risco manualmente.

Elbaz *et al.* (2021) utilizaram a técnica de aprendizado ativo para rotular um conjunto de dados, contendo informações de CPE (que são os identificadores únicos de *softwares*), extraídas das descrições de vulnerabilidade recém-publicadas no NIST. Os dados rotulados foram utilizados para treinar um modelo de ML capaz de classificar as vulnerabilidades em três níveis: (*i*) *LOG*, onde é feito um registro sobre a vulnerabilidade recém descoberta, mas ninguém é alertado sobre ela; (*ii*) *TICKET*, onde um chamado é criado e espera-se que ele seja resolvido no horário de trabalho; e por fim, (*iii*) *ALERT*, onde um alerta é feito para o responsável e espera-se que a vulnerabilidade seja resolvida imediatamente. A solução proposta pode classificar as vulnerabilidades e optar por alertar ou não o analista. Esta solução é limitada, pois considera apenas as informações do *CPE*, que muitas vezes, no momento de publicação de uma vulnerabilidade, são inexistentes. Além disso, desconsidera informações importantes sobre vulnerabilidades, inteligência de ameaças e contexto.

Granadillo *et al.* (2021) propõem uma ferramenta de VM, capaz de identificar, analisar e gerenciar vulnerabilidades na infraestrutura das organizações. A ferramenta, chamada de *Vulnerability Discovery Manager* (*VDM*), possui 4 componentes, que são: (*i*) *scanner* de vulnerabilidades; (*ii*) banco de dados para armazenar informações sobre a rede; (*iii*) um classificador de risco; e por fim, (*iv*) um módulo de relatória. O classificador utiliza uma fórmula matemática, que considera a causa raiz, o impacto da exploração e o tempo de remediação, para calcular o risco das vulnerabilidades. Experimentos mostraram ser possível recalcular o risco das vulnerabilidades considerando as informações adicionais. No entanto, a utilização da solução é complexa, pois os analistas precisam responder perguntas subjetivas, como, por exemplo, se o esforço de correção é grande ou baixo e se o custo é alto ou moderado. Além disso, os autores não consideram informações de inteligência de ameaças e contexto.

Walkowski *et al.* (2021) propuseram um método para avaliar o risco das vulnerabilidades combinando a nota de CVSS, com o impacto que a exploração do ativo vulnerável teria na CIA dos sistemas organizacionais. Para isso, os autores primeiro classificaram os ativos da organização em uma escala que mede seu impacto na CIA em alto, médio alto, médio baixo ou baixo. Em seguida, eles identificaram quais falhas deveriam ser remediadas primeira na organização, por meio de uma fórmula que mede quantos porcentos dos ativos da organização são afetados por cada vulnerabilidade. Essa fórmula funciona com um multiplicador para a nota de CVSS, de tal forma que se uma vulnerabilidade afeta muitos ativos na rede, sua nota de CVSS irá aumentar e vice-versa. Assim, os autores sugerem que os especialistas foquem seus esforços na correção das vulnerabilidades com as maiores notas de CVSS, nos ativos mais críticos. Experimentos realizados demonstraram que a metodologia proposta acabou concentrando a maioria dos ativos em médio alto e fez com que boa parte das vulnerabilidades ficassem com nota MEDIUM de CVSS. Essa falta de variabilidade na classificação torna a priorização das vulnerabilidades impraticável. Além disso, a solução não considera nenhum atributo de inteligência de ameaças.

Wang *et al.* (2020) afirmam que a nota de risco de uma vulnerabilidade não deve ser calculada considerando apenas seus atributos, como a Nota de Explorabilidade – *Explorability Score* (ES) – e a Nota de Impacto – *Impact Score* (IS) –, ambas dadas pelo CVSS; mas também os ativos afetados por ela. Assim, para calcular o risco de uma vulnerabilidade, os autores propuseram duas fórmulas: (*i*) a primeira é responsável por calcular o risco do ativo, através do somatório do risco de todas as vulnerabilidades presentes no dispositivo, mais o risco dos

dispositivos adjacentes a ele; (*ii*) a segunda é responsável por calcular o risco da vulnerabilidade, dado pelo somatório do risco dos dispositivos que ela afeta. Os experimentos demonstraram que o risco de certas vulnerabilidades aumentaram conforme o contexto onde elas aparecem. No entanto, essa solução é limitada, pois ela não considera informações adicionais sobre a vulnerabilidade e sobre inteligência de ameaças.

Chawla *et al.* (2019) argumentam que, apesar do CVSS ser um padrão universal bastante utilizado em cibersegurança, ele necessita de melhorias para conseguir expressar corretamente o risco das vulnerabilidades. Os autores identificaram dois fatores importantes que devem ser considerados no cálculo de risco: (*i*) representação do ambiente – *environment representative* –, que indica qual sistema operacional é afetado pela vulnerabilidade (e.g. *Linux*, *Windows* ou *macOS*); e o (*ii*) tipo da vulnerabilidade – *vulnerability type* –, que identifica o tipo de fraqueza utilizada para explorar a vulnerabilidade (e.g. *authentication weaknesses*, *buffer overflow*, *unvalidated input*, etc.). Os autores propõem uma nova fórmula para se calcular a nota do CVSS base que considera esses atributos. Apesar de experimentos realizados mostrarem que esses dois fatores melhoram o cálculo de risco, a proposta falha em não considerar informações de inteligência de ameaças e de contexto. Além disso, é difícil para o ser humano captar aspectos subjetivos e mensurá-los numericamente (e.g. quanto vale em uma escala de 0 a 10 um risco alto e outro muito alto), de tal forma que, soluções baseadas em fórmulas matemáticas costumam ser complexas de se utilizar.

Wu *et al.* (2019) afirmam que todo sistema de pontuação deve ser objetivo e considerar o princípio de dispersão. O CVSS sofre com problemas de dispersão, uma vez que, a maioria das vulnerabilidades se concentra em duas classes. Dessa forma, os autores propõem uma versão melhorada do CVSS chamada de *CVSS_PCA*. Essa nova métrica utiliza a Análise do Componente Principal (MAĆKIEWICZ; RATAJCZAK, 1993) – *Principal Component Analysis* (PCA) – para recombinar linearmente os valores que compõem o CVSS, melhorando a distribuição das vulnerabilidades entre as classes. O processo ocorre da seguinte forma: primeiro os autores obtêm um novo valor para cada um dos atributos que constitui o CVSS base utilizando o *PCA*. Em seguida, esses valores são normalizados e utilizados para calcular as notas de ES e IS. Finalmente, esses valores são utilizados nas fórmulas do CVSS base para calcular a severidade da vulnerabilidade. Ainda que solucione o problema de dispersão, a proposta apresentada é incapaz de calcular corretamente o risco das vulnerabilidades, pois desconsidera informações além do CVSS.

Baseado no padrão *GB/T 30279-2013*[9], que define os elementos que devem ser considerados no processo de VM e no CVSS, Lei *et al.* (2018) propõem uma métrica dinâmica para avaliar o risco das vulnerabilidades nas Casas Inteligentes – *Smart Homes* (SH). Na metodologia proposta, consideram-se três elementos: (*i*) o vetor de acesso, que identifica o meio pelo qual a exploração ocorreu (físico, local, adjacente ou remoto); (*ii*) a complexidade do ataque (baixo, médio ou alto); e (*iii*) o grau de impacto da exploração da vulnerabilidade no sistema em relação à confidencialidade, integridade e disponibilidade. Os autores demonstram que utilizando o método proposto, uma mesma vulnerabilidade (de *buffer overflow*) assume nota de risco diferente dependendo do contexto onde se encontra, isto é, se está presente em um roteador *D-link DIR-645* ou *D-link DIR-818W*. Isso ocorre, pois a complexidade de ataque em um dos roteadores é menor que no outro. Mesmo apresentando uma melhoria em relação ao CVSS, a solução é limitada, pois considera apenas duas informações de contexto. Adicionalmente, ela não avalia as informações de inteligência de ameaças no cálculo do risco.

Keramati (2016) afirma que uma das maiores limitações do CVSS é não considerar características das vulnerabilidades que mudam ao longo do tempo para calcular a nota. Nesse sentido, é proposta uma fórmula para calcular o risco das vulnerabilidades considerando: (*i*) a probabilidade de ocorrer um evento adverso (e.g. o surgimento de um *exploit* público); e (*ii*) o impacto desse evento nos sistemas afetados. Segundo o autor, a probabilidade de um evento adverso acontecer pode ser calculado utilizando a complexidade para realizar o ataque e uma distribuição de Pareto, onde a variável aleatória é o intervalo em dias entre a descoberta da vulnerabilidade e o momento de sua análise. Experimentos mostraram que a fórmula proposta conseguiu com sucesso alterar os valores de risco das vulnerabilidades. No entanto, por ser uma estratégia puramente matemática e não considerar *feeds* de segurança, não é possível afirmar que a solução representa adequadamente o que acontece no mundo real. Além disso, não se utilizam informações de contexto.

Existem também soluções, desenvolvidas por empresas da área de cibersegurança, como o *Vulnerability Priority Rating* (*VPR*) da Tenable (TENABLE, 2020), o *Real Risk Score* (*RRS*) da Rapid7 (RAPID7, 2021) e o *Risk Score* (*RS*) da Kenna (Kenna Security, 2020), que os autores afirmam utilizar dados de inteligência de ameaças e algoritmos sofisticados de ML para calcular o risco de exploração das vulnerabilidades. Esses *softwares* são soluções proprietárias que não divulgam como o risco é estimado ou quais atributos são utilizados no processo. Além

[9] Padrão adotado pela República Popular da China para classificação de vulnerabilidades de segurança e seu grau de severidade, em sistemas de informação: https://www.chinesestandard.net/PDF/English.aspx/GBT30279-2013.

disso, embora os autores afirmem reduzir a lista de vulnerabilidades críticas em 97%, os usuários não possuem estudos que indiquem que a metodologia aplicada seja correta ou eficiente.

Os trabalhos analisados desenvolveram soluções para avaliar o risco de exploração de vulnerabilidades. No entanto, nenhum deles considera simultaneamente as características da vulnerabilidade, as informações de inteligência de ameaças e o contexto onde ela está inserida nessa avaliação. Além disso, apenas 2 trabalhos utilizam ML, técnica que pode beneficiar o processo de gestão de vulnerabilidades baseada em risco. Assim, o FRAPE propõe usar esse conjunto de informações e a técnica de *AL* para capturar a experiência do analista de segurança cibernética, a fim de desenvolver um modelo de ML que classifique as vulnerabilidades quanto ao seu risco. Apresentamos na Tabela 2, a seguir, os atributos utilizados para se estimar o risco e a técnica empregada nesse processo pelos trabalhos analisados e a nossa proposta.

Tabela 2 – Comparação entre os trabalhos relacionados que desenvolvem soluções de RBVM e a nossa proposta. Nosso trabalho é o único que considera as características das vulnerabilidades, inteligência de ameaças e contexto na análise do risco, bem como que utiliza a técnica de ML.

Referências	**Característica das Vulns.**	**Inteligência de Ameaças**	**Contexto**	**Ferramentas**
Kure *et al.* (2022)		×	×	Lógica Difusa + ML
Elbaz *et al.* (2021)	×			ML
Granadillo *et al.* (2021)	×			Modelo Estático
Walkowski *et al.* (2021)	×		×	Modelo Estático
Wang *et al.* (2020)	×		×	Modelo Estático
Chawla *et al.* (2019)	×			Modelo Estático
Wu *et al.* (2019)	×			Análise Exploratória
Lei *et al.* (2018)	×		×	Modelo Estático
Keramati (2016)	×	×		Modelo Estático
FRAPE	×	×	×	ML

5 PROPOSTA

A seguir apresentaremos a proposta deste trabalho, que consiste em um *framework* que pode ser utilizado por profissionais de cibersegurança para estimar o risco e o impacto de exploração das vulnerabilidades em sistemas organizacionais. Este capítulo está organizado da seguinte forma: (*i*) na Seção 5.1 contextualizamos o problema e mostramos como a utilização da ferramenta proposta pode auxiliar na atividade de gestão de vulnerabilidades baseada em risco; e (*ii*) na Seção 5.2 apresentamos a arquitetura do FRAPE e descrevemos em detalhes os seus componentes.

5.1 Contextualização

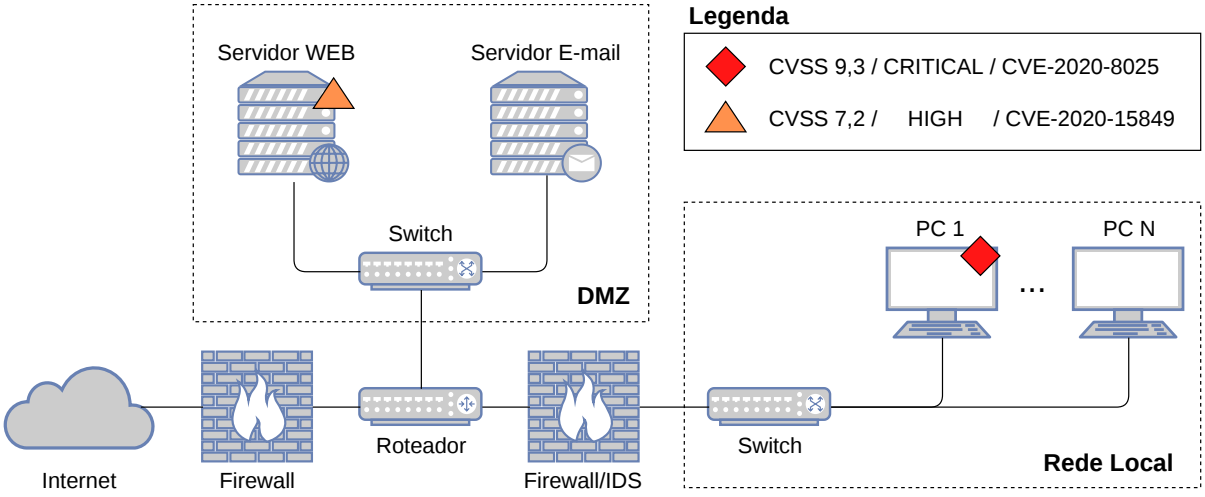
Para ilustrarmos o problema, considere a rede de uma organização fictícia, com duas vulnerabilidades cujos identificadores são CVE-2020-8025[1] e CVE-2020-15849[2], respectivamente, conforme apresentado na Figura 20. O CVE-2020-8025 é uma vulnerabilidade que afeta sistemas operacionais *Linux* desenvolvidos pela empresa *SUSE*[3]. Essa vulnerabilidade possui uma nota de CVSS de 9,3, seu impacto na CIA é alto e sua exploração ocorre a partir da rede local. Já o CVE-2020-15849 é uma vulnerabilidade de *software* que afeta uma ferramenta de atendimento ao cliente chamada *Re:Desk*[4]. Essa vulnerabilidade possui uma nota de CVSS de 7,2, um alto impacto na CIA e pode ser explorada remotamente.

Seguindo a abordagem clássica das ferramentas de VM de corrigir as vulnerabilidades com as maiores notas de CVSS, os analistas priorizariam a correção do CVE-2020-8025. No entanto, essa prática é desaconselhada por especialistas, visto que, para se estimar corretamente o risco de exploração de uma vulnerabilidade, é necessário analisar além suas características, o contexto em que está inserida e o cenário global de ameaças. Dessa forma, antes de escolher qual vulnerabilidade corrigir, os analistas devem coletar e compilar informações provenientes de *feeds* de segurança, das redes sociais, de fóruns da *deep* e *dark web*, etc. Essas informações, em conjunto com o conhecimento da rede da organização e a experiência dos especialistas, permitirá que as vulnerabilidades mais críticas sejam corretamente priorizadas durante o processo de correção.

---

[1] Vulnerabilidade que afeta sistemas operacionais *SUSE*: https://nvd.nist.gov/vuln/detail/CVE-2020-8025.
[2] Vulnerabilidade que afeta uma ferramenta de *help desk*: https://nvd.nist.gov/vuln/detail/CVE-2020-15849.
[3] Empresa de *software* de código aberto: https://www.suse.com/.
[4] Ferramenta de atendimento ao cliente *Re:Desk*: https://www.re-desk.com/.

Figura 20 – Rede fictícia de uma organização contendo duas vulnerabilidades, cujas severidades são *CRITICAL* (nota de CVSS 9,3) e *HIGH* (nota 7,2). A vulnerabilidade *CRITICAL* está em uma rede local, protegida atrás de um robusto *firewall* e um IDS. Enquanto que a vulnerabilidade *HIGH* está na DMZ, protegida apenas por um simples *firewall* utilizado na filtragem de pacotes.



Fonte: elaborado pelo autor.

Depois de um estudo realizado, os especialistas descobriram que o CVE-2020-8025 afeta uma estação de trabalho, que recebe atualizações de segurança com frequência e está em uma rede protegida por um *firewall* robusto e um IDS. Em contrapartida, o CVE-2020-15849 afeta um servidor web na Zona Desmilitarizada – *Demilitarized Zone* (DMZ) –, protegida apenas por um *firewall* básico que realiza a filtragem de pacotes. Além disso, os analistas identificaram uma campanha de *malware* que explora essa falha graças ao surgimento de um novo *exploit*. Nesse cenário, a exploração do CVE-2020-15849 representa um risco maior para a organização, pois ela afeta um dispositivo conectado diretamente com a internet, que pode ser explorado remotamente graças à existência de um *exploit* público. Portanto, os analistas de segurança devem corrigir primeiro a vulnerabilidade com o CVE-2020-15849, mesmo ele possuindo uma nota de CVSS menor que a do CVE-2020-8025, pois sua exploração possibilita ao atacante acessar informações sensíveis da base de dados da aplicação. O atacante, então, poderia utilizar essas informações para extorquir dinheiro da organização ou para obter acesso indevido à rede local da empresa, continuando seu ataque em uma nova frente.

A análise que mostramos considerou poucas características entre as duas vulnerabilidades. Na prática, os analistas de segurança devem correlacionar milhares de ativos e dezenas de informações. Essa atividade é demorada, complexa e propensa a erros, uma vez que, na maioria das vezes, é realizada manualmente pelos especialistas, que contam apenas com a ajuda de planilhas e *scripts* simples de programação. O uso de uma ferramenta especializada capaz de

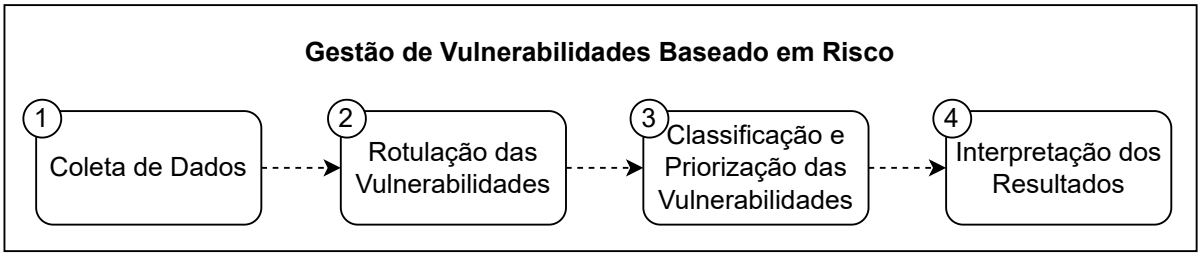
realizar essa tarefa seria bastante útil, pois evitaria que erros humanos fossem cometidos durante a classificação/priorização e, além disso, permitiria que os especialistas focassem seus esforços em outras atividades. Assim, este trabalho visa desenvolver um *framework* que auxilie os analistas de segurança no processo de coleta de informações, classificação do risco e priorização das vulnerabilidades. Essa solução é chamada de *Framework for Risk Assessment, Prioritization and Explainability of Vulnerabilities* (FRAPE) e será descrita em detalhes a seguir.

5.2 FRAPE framework

O *framework* proposto é composto por 4 módulos, que são: (*i*) o módulo de Coleta de Dados, responsável por agregar as informações acerca das vulnerabilidades, inteligência de ameaças e contexto; (*ii*) o módulo de Rotulação das Vulnerabilidades, que através do processo de AL, consultará os especialistas iterativamente para rotular o risco das vulnerabilidades; (*iii*) o módulo de Classificação e Priorização das Vulnerabilidades, que classificará o risco das vulnerabilidades e elegerá aquelas que devem ser corrigidas primeiro, utilizando um modelo de ML; e por último, (*iv*) o módulo de Interpretação dos Resultados, que fornecerá informações importantes sobre o processo de classificação e priorização das vulnerabilidades selecionadas.

A Figura 21 mostra uma visão simplificada dos módulos que compõem o *framework* e como eles interagem entre si. O processo inicia-se com a coleta de informações acerca das vulnerabilidades e dos ativos de rede, e termina com a identificação das vulnerabilidades mais críticas no ambiente organizacional. Nas subseções a seguir descreveremos em detalhes o funcionamento de cada um desses módulos.

Figura 21 – Arquitetura do *framework* proposto, que auxiliará os analistas no processo de RBVM, possui 4 módulos, que são: Coleta de Dados; Rotulação das Vulnerabilidades; Classificação e Priorização das Vulnerabilidades; e por último, Interpretação dos Resultados.



Fonte: elaborado pelo autor.

5.2.1 Coleta de Dados

Como demonstrado na Seção 5.1, para avaliar corretamente o risco de uma vulnerabilidade é necessário considerar várias informações sobre ela. Só assim, a probabilidade de uma falha ser explorada e o impacto desse evento nos sistemas da organização ficará evidente. Nesse contexto, este trabalho propõe que o risco de uma vulnerabilidade seja avaliado considerando três conjuntos de informações, que são: suas características, inteligência de ameaças e o contexto onde ela está inserida.

Assim, o primeiro módulo, representado na Figura 22, é o de Coleta de Dados. Esse módulo é responsável por agregar todas as informações acerca das vulnerabilidades e do ambiente organizacional. As informações das vulnerabilidades são coletadas de diversos *feeds* de segurança e armazenadas em uma base de "Conhecimento sobre as Vulnerabilidades". Já as informações de contexto, coletadas através do uso de ferramentas de gestão de vulnerabilidades e de inventários de rede, são armazenadas em uma base de "Conhecimento sobre a Organização".

Com base nessas informações e na experiência pessoal dos analistas, o *framework* proposto conseguirá classificar e priorizar as vulnerabilidades mais críticas (temas abordados nas Subseções 5.2.2 e 5.2.3). A seguir, descreveremos os três conjuntos de dados coletados e utilizados pelo FRAPE na classificação de risco das vulnerabilidades.

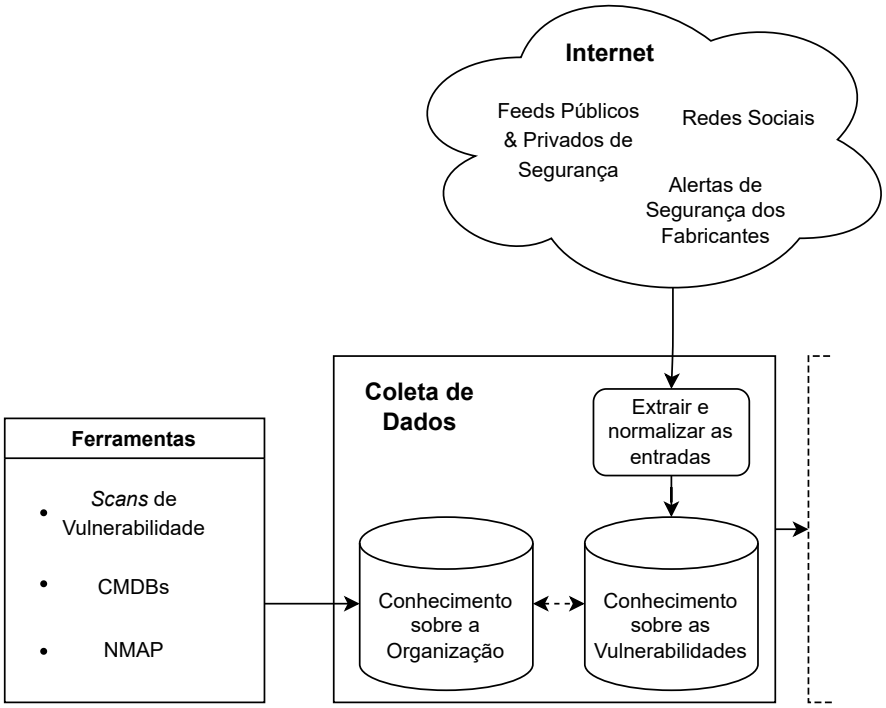
5.2.1.1 Características das Vulnerabilidades, Inteligência de Ameaças e Contexto

O primeiro grupo de dados, o de Características das Vulnerabilidades, fornece informações que não mudam ao longo do tempo e são constantes para todos os ambientes. Com essas informações, os analistas conseguem identificar a vulnerabilidade e o impacto de sua exploração nos sistemas das organizações. Como exemplo de dados pertencentes a esse grupo, podemos citar a nota de CVSS e o produto, versão e fabricante afetado pela falha de segurança. Esses dados estão publicamente disponíveis para coleta, graças a iniciativas públicas como a do *NVD*[5] e privadas, mas sem fins-lucrativos, como a do *Mitre*[6].

O segundo grupo de dados, o de Inteligência de Ameaças, são informações que variam ao longo do tempo, pois são influenciados diretamente pelas ações de atores maliciosos. Esse conjunto de dados é essencial para se conhecer o cenário atual de cibersegurança, fornecendo as informações necessárias para sua caracterização. Com essas informações, por exemplo,

[5] Site do *National Vulnerability Database*: https://nvd.nist.gov/.
[6] Site do *Mitre* CVE: https://cve.mitre.org/cve/.

Figura 22 – O módulo de Coleta de Dados é responsável por coletar e armazenar as informações provenientes de *feeds* de inteligência (características das vulnerabilidades e inteligência de ameaças) e de ferramentas de VM e inventariado de rede (contexto dos ativos vulneráveis).



Fonte: elaborado pelo autor.

é possível identificar a existência de *exploits* públicos e/ou de campanhas de *malware* em andamento que afetam uma determinada vulnerabilidade. Essas informações podem ser coletadas de *feeds* privados, como *Crowdstrike*[7] e *RecordFuture*[8]; *feeds* públicos como *ExploitDB*[9] e *VirusTotal*[10]; de boletins de segurança dos fabricantes, como o "*patch tuesday*" da *Microsoft*[11]; e, finalmente, de redes sociais e buscadores, como o *Twitter* e o *Google*.

Por fim, o terceiro grupo de dados, o de Contexto, fornece informações específicas sobre o ativo vulnerável. Essas informações consideram a rede e o ativo onde a vulnerabilidade foi detectada. Portanto, elas mudam dependendo da organização e do ambiente. A combinação dessas informações permite definir a criticidade dos ativos, útil na hora de priorizar quais vulnerabilidades devem ser corrigidas primeiro. Como exemplo de informações de contexto, podemos citar a existência de dados sensíveis no ativo ou o fato dele não receber mais atualizações de segurança. Essas informações são obtidas manualmente ou por meio de ferramentas de inventário

[7] Empresa de segurança que fornece serviços de defesa para APIs, informações de inteligência contra ameaças e resposta a ataques cibernéticos: https://www.crowdstrike.com.br/.

[8] Empresa de segurança especializada na coleta, processamento, análise e disseminação de informações de inteligência de ameaças: https://www.recordedfuture.com/.

[9] Um arquivo de *softwares* vulneráveis e *exploits*: https://www.exploit-db.com/.

[10] Serviço gratuito que analisa arquivos e URLs em busca de vírus, *worms* e *trojans*: https://www.virustotal.com/.

[11] Boletins de Segurança da *Microsoft*: https://msrc.microsoft.com/update-guide/en-us.

de rede, como os Bancos de Dados de Gerenciamento de Configuração (mais conhecidos como CMDBs), de escaneadores de portas de computadores (as ferramentas de *port scan*), como o *NMAP*[12] ou ainda, através de verificadores de vulnerabilidades, como o Nessus.

5.2.2 Rotulação das Vulnerabilidades

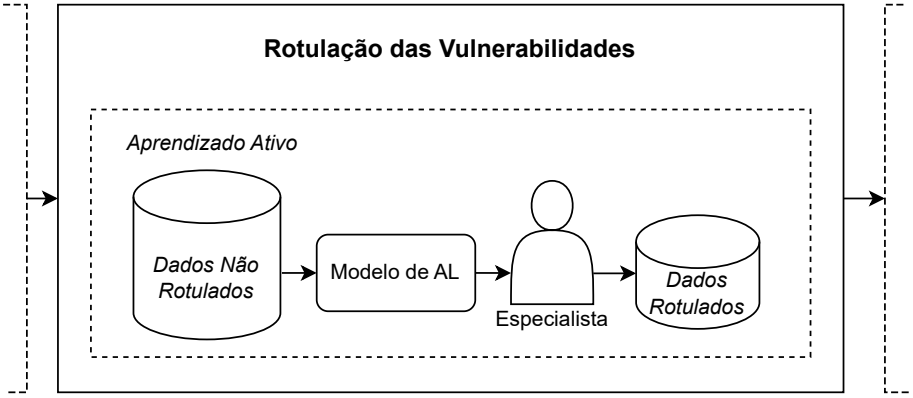
A classificação de risco é um mecanismo de segurança empregado por especialistas para proteger os ativos e as redes de uma organização. Esse processo envolve a correlação entre informações sobre vulnerabilidades, ameaças e ativos (ANDRESS, 2019). Essas informações, aliada com a experiência pessoal dos analistas, fornecem o conhecimento necessário para identificar vulnerabilidades críticas, ou seja, aquelas que causariam o maior impacto a organização se exploradas (BROMANDER, 2021).

É possível utilizar técnicas de ML para treinar um modelo capaz de auxiliar os analistas no processo de classificação do risco das vulnerabilidades. No entanto, não podemos simplesmente aplicar uma técnica de aprendizado supervisionado porque não há dados rotulados e experimentos preliminares com o aprendizado não supervisionado (i.e. clusterização) não mostraram bons resultados. O passo lógico seria rotular esses dados manualmente. Atividade considerada cara, ainda mais para este problema específico, onde temos grandes quantidades de itens para serem rotulados e poucos especialistas qualificados para realizar tal atividade (MILLER *et al.*, 2020).

Para resolver este problema, propomos o uso do aprendizado ativo, uma técnica de ML, que visa aperfeiçoar o processo de rotulação de instâncias não rotuladas em um *dataset* (descrito em detalhes na Seção 3.2). Essa técnica seleciona os itens mais significativos para serem rotulados pelos especialistas, dessa forma, reduz-se o número de entradas desnecessárias rotuladas e, consequentemente, diminui os custos associados a esta atividade (SETTLES, 2011). Assim, o segundo módulo, representado na Figura 23, é o de Rotulação das Vulnerabilidades. Esse módulo utiliza as informações provenientes do módulo de Coleta de Dados (apresentado na Subseção 5.2.1) e a experiência dos especialistas, bem como a técnica de aprendizado ativo, para rotular o risco das vulnerabilidades de segurança presentes na rede da organização.

[12] Ferramenta *open-source* utilizada para mapear serviços e portas: https://nmap.org/.

Figura 23 – O módulo de Rotulação das Vulnerabilidades é responsável por rotular o risco das vulnerabilidades. Para isso, utilizamos uma técnica de ML chamada de aprendizado ativo, que consulta interativamente os especialistas para rotular as instâncias (vulnerabilidades) mais significativas do conjunto de dados.



Fonte: elaborado pelo autor

5.2.2.1 Rótulos Aplicados

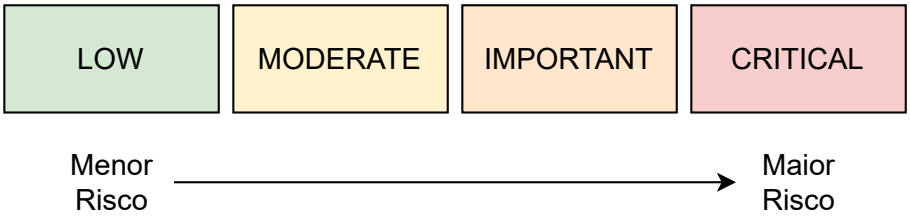
Os rótulos, ou classes, utilizados na classificação de risco, foram escolhidos com base na classificação de severidade elaborada pelas empresas Adobe (ADOBE, 2022) e Microsoft (MICROSOFT, 2022). Essa classificação é familiar para os analistas e oferece uma boa alternativa semântica aos rótulos utilizados pelo CVSS. Assim, evitamos qualquer tipo de viés que os especialistas tenham na hora de classificar as vulnerabilidades, uma vez que, os rótulos diferem dos atribuídos pelo CVSS em sua nota base. No que lhes concerne, os especialistas devem utilizar sua experiência pessoal em conjunto com as informações coletadas acerca das vulnerabilidades, de inteligência de ameaças e do ambiente, para classificar o risco das falhas de segurança seguindo uma ordem crescente de criticidade. Os rótulos utilizados para classificar o risco das vulnerabilidades, com alguns possíveis atributos, são:

- ***CRITICAL*:** atribuído às vulnerabilidades que afetam dispositivos críticos, facilmente exploradas (e.g. dada a presença de *exploits*), que podem causar grande impacto na confiabilidade, integridade e disponibilidade desses ativos, permitindo possivelmente o acesso a informações sensíveis da empresa;
- ***IMPORTANT*:** atribuído às vulnerabilidades que podem afetar dispositivos críticos e que causam algum impacto na confiabilidade, integridade e disponibilidade desses ativos, permitindo ou não o acesso a informações sensíveis da empresa;
- ***MODERATE*:** atribuído às vulnerabilidades que afetam dispositivos comuns, difíceis de serem exploradas e que não causam grande impacto para os sistemas da empresa, além de não possibilitarem o acesso a informações sensíveis;

– ***LOW*:** atribuído às demais vulnerabilidades que não se encaixam em nenhuma das alternativas anteriores.

Dessa forma, uma vulnerabilidade classificada como *CRITICAL* possuirá um risco maior do que uma vulnerabilidade classificada como *IMPORTANT* e assim por diante. Na Figura 24 podemos ver graficamente a disposição desses rótulos, com o *LOW* mais a esquerda, na posição com o menor risco e o *CRITICAL* mais a direita, na posição com o maior risco.

Figura 24 – As vulnerabilidades, segundo o FRAPE, podem ser classificadas em 4 classes, que são: *LOW*, *MODERATE*, *IMPORTANT* e *CRITICAL*. Sendo a classe *LOW* atribuída as vulnerabilidades com os menores riscos e a *CRITICAL* aquelas com os maiores riscos.



Fonte: elaborado pelo autor.

5.2.2.2 Estratégia de Rotulação

Rotular por seleção aleatória instâncias de um conjunto de dados é uma prática comum. Entretanto, nesta estratégia, instâncias com características redundantes e insignificantes têm a mesma probabilidade de serem escolhidas como as mais representativas. Alternativamente, a técnica de AL permite que o modelo selecione quais instâncias devem ser rotuladas em cada iteração, evitando assim desperdício de recursos e agilizando o processo de treinamento.

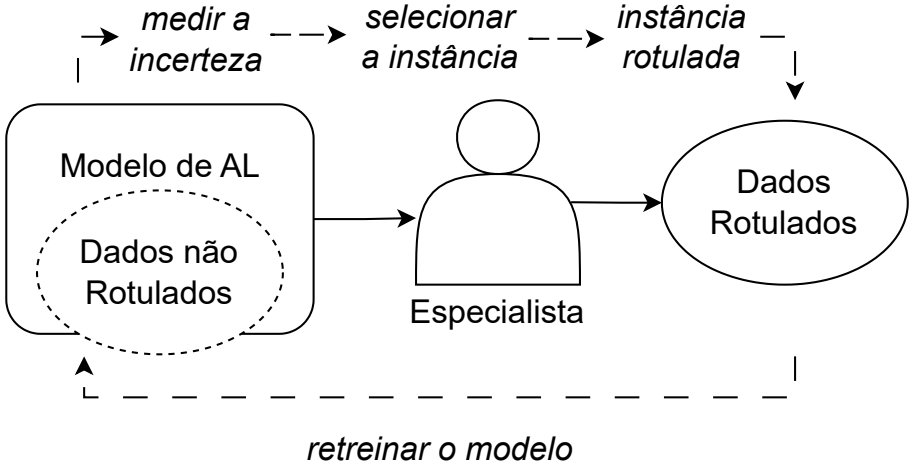
Dessa forma, optamos por utilizar uma técnica de AL, descrita na Seção 3.2, para rotular nossas instâncias. O processo inicia-se com a seleção de uma pequena quantidade de dados, obtidos através do uso de uma técnica de *clusterização* chamada de *k-medoids* (PARK; JUN, 2009). Essa técnica consegue selecionar as instâncias que são os centros de *clusters* (*medoids*) e que, por isso, possuem características que representam bem um conjunto de amostras. Essa pequena quantidade de dados é rotulada manualmente e será usada como entrada para o modelo de AL. Essa etapa é importante, pois garantimos que o modelo de AL comece com um bom desempenho, o que fará com que o processo de treinamento seja mais rápido e eficiente.

Em seguida, o modelo de AL recebe como entrada um grande conjunto de dados não rotulados, que são as vulnerabilidades presentes na rede da organização. Essa estratégia, descrita em detalhes na Subseção 3.2.1, é chamada de *pool-based*. Então, de posse das instâncias,

o modelo de AL seleciona aquela que possui as características mais significativas para ser rotulada. Esse processo é feito através do emprego de uma estratégia de consulta, que pode ser por incerteza ou por comitê. Essas estratégias foram descritas em detalhes na Subseção 3.2.2.

Por fim, o especialista deve escolher o rótulo apropriado para aplicar à instância selecionada, que será usada para treinar novamente o modelo de AL. Este processo é iterativo e será repetido várias vezes. Normalmente, o processo de aprendizado é interrompido após um certo número de elementos serem rotulados ou quando o modelo atinge um valor esperado de acurácia (SETTLES, 2009). A Figura 25 resume o processo de AL descrito anteriormente.

Figura 25 – O processo de aprendizado ativo envolve a seleção de uma instância cujo modelo tem a menor certeza em relação ao rótulo que deve ser aplicado. Essa instância é apresentada ao especialista para ser rotulada e então, utilizada para retreinar o modelo de AL.



Fonte: elaborado pelo autor.

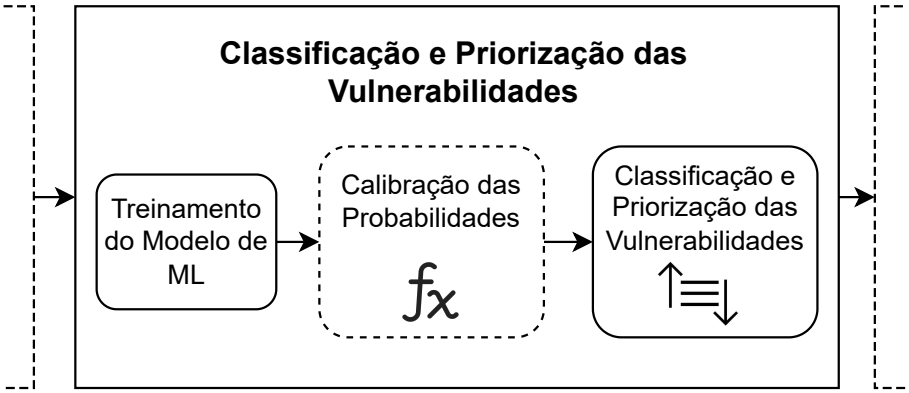
É importante destacar que o modelo treinado irá emular o conhecimento dos especialistas de segurança. Assim, para evitar qualquer dúvida que possa surgir, em relação a qual rótulo uma vulnerabilidade deve receber, o processo de AL deve ser realizado idealmente por mais de um especialista. Nesse caso, se houver alguma divergência entre o rótulo aplicado, os especialistas podem discutir e chegar a um consenso. Além disso, como o AL utiliza os valores de certeza em relação aos possíveis rótulos de uma vulnerabilidade, para selecionar novas instâncias a serem rotuladas, recomenda-se, que o processo de treinamento do modelo AL seja feita em conjunto com uma técnica de calibração de probabilidades (descrita na Seção 3.3).

5.2.3 Classificação e Priorização das Vulnerabilidades

Nessa etapa do processo, após um subconjunto das vulnerabilidades presentes na organização terem sido rotuladas (atividade descrita na Subseção 5.2.2), conseguimos treinar um modelo de ML capaz de emular a experiência dos profissionais de segurança no processo de classificação do risco das vulnerabilidades. Com esse modelo de ML, o *framework* conseguirá classificar o risco das falhas de segurança e consequentemente, oferecer um guia de quais vulnerabilidades devem ser corrigidas primeiro pelos analistas.

Conseguimos, assim, sair de um universo de milhares de vulnerabilidades, com diferentes criticidades, para um conjunto com centenas de vulnerabilidades críticas, ou seja, aquelas mais perigosas para a organização. Assim, o terceiro módulo do *framework*, representado na Figura 26, é o de Classificação e Priorização das Vulnerabilidades. Esse módulo é responsável por classificar e ordenar as vulnerabilidades da mais perigosa para a menos perigosa, dando ao analista um guia de qual vulnerabilidade corrigir a seguir.

Figura 26 – O módulo de Classificação e Priorização das Vulnerabilidades é responsável por classificar e priorizar as vulnerabilidades que devem ser corrigidas primeiro, considerando o seu risco para a organização. Isso é feito através de um modelo de ML capaz de emular a experiência de profissionais de segurança no processo de VM.



Fonte: elaborado pelo autor.

5.2.3.1 Estratégia de Aprendizado e Priorização

De posse dos dados rotulados, provenientes do módulo de Rotulação das Vulnerabilidades (Subseção 5.2.2), podemos usar tanto uma técnica de aprendizado supervisionado quanto de aprendizado semissupervisionado para treinar um modelo de ML capaz de avaliar o risco das vulnerabilidades de segurança. O aprendizado supervisionado é uma subcategoria de algoritmos de ML que usam apenas dados rotulados para treinar modelos de regressão ou

classificação (MURPHY, 2022). Já a técnica de aprendizado semissupervisionado, recebe como entrada um conjunto de dados contendo poucos dados rotulados e muitos dados não rotulados para o treinamento (MURPHY, 2022). Em alguns casos, o aprendizado semissupervisionado pode obter um desempenho melhor que o aprendizado supervisionado.

É importante destacar que a nossa metodologia de aprendizado pode ser aplicada com qualquer algoritmo de ML. No entanto, como decidimos usar o mesmo modelo tanto para o processo de aprendizado ativo, quanto para o treinamento do classificador final, todos os algoritmos de ML utilizados possuem suporte para prever probabilidades. Isso foi feito porque precisamos conseguir estimar, pelo menos aproximadamente, a incerteza das previsões realizadas.

Após o treinamento, aplicamos uma técnica de calibração de probabilidades (discutida em detalhes na Seção 3.3) para ajustar a saída do modelo ML, de modo que o resultado dado seja o mais próximo possível do real. Isso é importante, pois modelos não calibrados podem, muitas vezes, apresentar um viés na classificação, o que significa que as probabilidades são muito ou pouco confiáveis. Já modelos bem calibrados, são considerados classificadores probabilísticos, para os quais, a probabilidade de certeza em relação ao rótulo aplicado pode ser interpretada como o nível de confiança. Assim, para calibrar o modelo de ML, o FRAPE pode utilizar um regressor sigmoide ou isotônico (descritos na Seção 3.3).

Em seguida, com o modelo treinado e calibrado, o *framework* criará uma lista ordenada duplamente pelo rótulo (i.e., LOW, MODERATE, IMPORTANT, e CRITICAL) e pela probabilidade de certeza dado pelo modelo de ML. Assim, as vulnerabilidades com os maiores riscos, ou seja, aquelas cujo o classificador tem a maior certeza (probabilidades próximas a $1,0$) de que são críticas (rótulo CRITICAL) ficaram no topo e aquelas que ele tiver menor certeza (probabilidades próximas a $0,0$) de que são críticas (rótulo LOW) ficaram no final da lista.

Por fim, é oportuno dizer que, nossa proposta de gestão de vulnerabilidades é flexível e permite que outras metodologias possam ser utilizadas na hora de priorizar as vulnerabilidades para correção. Por exemplo, o usuário pode escolher corrigir primeiro todas as vulnerabilidades classificadas com o rótulo CRITICAL que afetem um servidor Windows, pois, segundo as regras de negócio da empresa, esses ativos são considerados sensíveis e devem receber uma maior atenção no processo de VM.

5.2.3.2 Estratégia de Implantação e Manutenção do Modelo

Não está no escopo deste trabalho aprofundar a discussão acerca do processo de implantação e manutenção do *framework*. No entanto, achamos pertinente incluir uma seção falando sobre isso. Visto que, com o passar do tempo e a introdução de novos dados com diferentes características, é comum que os modelos de ML comecem a degradar e perder acurácia. Esse fenômeno, chamado de desvio de conceito, do inglês *concept drift*, ocorre devido a mudanças imprevistas nas propriedades estatísticas dos itens que o modelo tenta prever (ZENISEK *et al.*, 2019).

Para superar essa limitação e lidar com as características dinâmicas das vulnerabilidades de segurança, pode-se adotar uma estratégia de treinamento dividida em duas partes. A primeira parte, que chamaremos de treinamento *offline*, deve acontecer durante a implantação do sistema e permitirá que o *framework*, em conjunto com os especialistas, crie uma base de conhecimento geral sobre as vulnerabilidades. A segunda parte, que chamaremos de treinamento *online*, é uma interação contínua e periódica, onde os especialistas são consultados para rotular novas vulnerabilidades que surgiram nesse período. Essa segunda parte permite que o modelo adquira conhecimento sobre as últimas vulnerabilidades descobertas.

Cada uma dessas fases possui um custo de rotulação independente, com a fase *offline* acontecendo uma única vez e a fase *online* exigindo um esforço contínuo ao longo do tempo. Essas duas fases do treinamento são importantes na implantação da solução, pois permitem que o modelo esteja sempre atualizado conforme as tendências da área de segurança, que costumam mudar rapidamente (MELTZER, 2020).

Para viabilizar esse processo, pode-se utilizar a abordagem de Operações de Aprendizado de Máquina – *Machine Learning Operations* (MLOps) –, que é a união entre ML e a cultura DevOps (ALLA; ADARI, 2021). MLOps fornece um conjunto de práticas que incentiva a colaboração entre cientistas de dados e engenheiros de DevOps, para garantir que a transição entre o desenvolvimento local de um modelo de ML e implantação do mesmo em um ambiente de produção seja feito de forma automática e contínua (MAKINEN *et al.*, 2021).

5.2.4 Interpretação dos Resultados

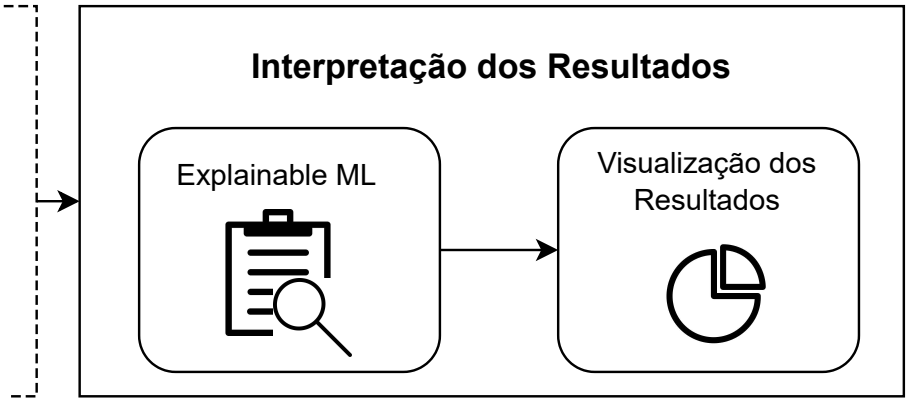
O uso da técnica de ML para resolver o problema de classificação de risco traz consigo inúmeras vantagens. Entre elas, podemos citar a rapidez e precisão com que o modelo

consegue analisar e classificar milhares de vulnerabilidades, uma tarefa que seria impraticável para seres humanos. No entanto, a utilização de ML envolve um compromisso entre acurácia e transparência (ROSCHER *et al.*, 2020). Visto que esses modelos, apesar de apresentarem alta acurácia, funcionam como uma "caixa preta", não oferecendo nenhuma explicação de como se chegou ao resultado apresentado.

Isso é grave, especialmente tratando-se de um problema de segurança, onde os analistas devem compreender o motivo pelo qual o modelo classificou ou não as falhas de segurança como críticas. Portanto, é imprescindível que os resultados sejam claros e para isso, precisamos garantir duas coisas: (*i*) a transparência do classificador, ou seja, o usuário precisa entender como os componentes do modelo funcionam e interagem entre si (detalhado nas seções anteriores); e (*ii*) a explicabilidade dos resultados, isto é, deve ser possível entender como se chegou ao conceito abstrato (o rótulo aplicado) a partir de características intrínsecas da instância analisada (BHATT *et al.*, 2020). Só assim garantimos que a solução seja confiável e possa ser melhorada, caso necessário, pelos especialistas.

Assim, o quarto e último módulo do *framework* FRAPE, apresentado na Figura 27, é o de Interpretação dos Resultados. Esse módulo utiliza XAI (descrito na Seção 3.4) para analisar e interpretar os riscos atribuídos as vulnerabilidades pelo módulo de Classificação e Priorização das Vulnerabilidades (Subseção 5.2.3). Isso é realizado através do uso da ferramenta SHAP (descrita em detalhes na Subseção 3.4.1), que por ser uma solução agnóstica de interpretabilidade, pode ser utilizada em qualquer modelo de ML.

Figura 27 – O módulo de Interpretação dos Resultados é responsável por oferecer uma explicação detalhada do funcionamento do modelo de ML. Isso é possível através do uso de técnicas de XAI, que garantem uma maior confiabilidade e transparência ao *framework*.



Fonte: elaborado pelo autor.

6 CONSTRUÇÃO DO *DATASET* DE SEGURANÇA E ANÁLISE EXPLORATÓRIA

Neste capítulo, será descrita em detalhes a metodologia utilizada na mineração e consequente criação de um conjunto de dados de segurança contendo informações sobre as características das vulnerabilidades e inteligência de ameaças. Assim, na Seção 6.1, o processo de criação do conjunto de dados de segurança (que corresponde ao módulo de Coleta de Dados descrito na Subseção 5.2.1) é apresentado e na Seção 6.2 uma análise exploratória dos dados coletados é conduzida e os achados importantes expostos.

6.1 Conjunto de Dados de Segurança

Nesta seção descreveremos a metodologia utilizada na criação do *dataset* de segurança (Subseção 6.1.1), os atributos que fazem parte dele (Subseção 6.1.2) e como incorporar informações de contexto aos dados coletados (Subseção 6.1.3).

6.1.1 Metodologia de Coleta de Dados

O conjunto de dados foi criado minerando automaticamente *feeds* públicos de segurança. Usamos a versão 3.8 da linguagem de programação Python para coletar, processar e analisar as informações reunidas. Ao todo, consultamos dez bases de dados, listadas a seguir:

- NVD[1]: um repositório mantido pelo Instituto Nacional de Padrões e Tecnologia (NIST) que contém várias informações sobre falhas de segurança, incluindo o CVE, que é um identificador único utilizado para referenciar vulnerabilidades;
- Mitre[2] e OWASP[3]: organizações sem fins lucrativos que enumeram vulnerabilidades perigosas de *software* e *hardware*, as chamadas, CWEs. A presença dessas falhas tornam os sistemas, redes e *hardwares* mais vulneráveis a ataques;
- ExploitDB[4]: um arquivo público de *exploits* e Provas de Conceito – *Proofs-of-Concept* (PoC) –, que pode ser usado para explorar ativos vulneráveis;
- Sistema de Pontuação para Previsão de Exploração – *Exploit Prediction Scoring System* (EPSS)[5]: um sistema de pontuação que usa informações de ameaças e dados reais de exploração para estimar a probabilidade de que uma vulnerabilidade será explorada nos

[1] Site da Base de Dados Nacional: https://nvd.nist.gov/.
[2] Site da corporação Mitre: https://cwe.mitre.org.
[3] Site do OWASP: https://owasp.org.
[4] Um arquivo de *softwares* vulneráveis e *exploits*: https://www.exploit-db.com/.
[5] Métrica para estimar a probabilidade de uma vulnerabilidade ser explorada: https://www.first.org/epss/.

próximos 30 dias;

- Boletins de Segurança da Microsoft[6], Adobe[7], e Intel[8]: essas empresas fornecem, periodicamente, *patches* de segurança ou soluções alternativas para corrigir/mitigar vulnerabilidades críticas identificadas em seus produtos;
- Twitter e Google Trends: as vulnerabilidades que receberam alguma atenção nessas redes, ou seja, aquelas citadas pelos usuários, são de interesse, pois podem ser alvos de campanhas de *malware* ou possuir *exploits* públicos.

Construímos o conjunto de dados em três etapas:

1. Primeiro, coletamos as informações disponibilizadas por esses *feeds* por meio de *web scraping*, consultas de API ou *download* de arquivos com informações de vulnerabilidade no formato JSON. Isso foi possível graças a bibliotecas como *urllib*[9] e *BeautifulSoup*[10], que permitem acessar e manipular o conteúdo de sites a partir de suas URLs;
2. Em seguida, processamos os dados coletados sobre cada vulnerabilidade, excluindo informações não relevantes e mapeando-as para o identificador CVE;
3. Por fim, juntamos todas as informações em um único conjunto de dados, processo feito usando o CVE como ponto de mesclagem. Coletamos informações de vulnerabilidade publicadas entre 2002 e 2022.

Os *scripts* usados na mineração dos dados e criação do *dataset* podem ser encontrados no GitHub[11]. Além disso, uma cópia do *dataset* gerado foi disponibilizado para uso e pode ser encontrado no *figshare* (PARENTE *et al.*, 2022). Todas as informações sobre as vulnerabilidades são armazenadas em um arquivo CSV que pode ser consultado por analistas usando, por exemplo, uma ferramenta de planilha ou a biblioteca Python *pandas*[12]. Ao todo, coletamos informações sobre mais de 200.000 vulnerabilidades. É importante observar que as informações de vulnerabilidade mudam constantemente ao longo do tempo. Assim, os usuários devem executar o *script* periodicamente para manter o conjunto de dados atualizado, pois novas vulnerabilidades são publicadas diariamente.

[6] Boletins de Segurança da Microsoft: https://msrc.microsoft.com/update-guide/en-us.
[7] Boletins de Segurança da Adobe: https://helpx.adobe.com/security.html.
[8] Boletins de Segurança da Intel: https://www.intel.com/content/www/us/en/security-center/default.html.
[9] Pacote Python para trabalhar com URLs: https://docs.python.org/3/library/urllib.html.
[10] Biblioteca Python para extrair informações de páginas da web: https://pypi.org/project/beautifulsoup4/.
[11] Código desenvolvido para criar o *dataset*: https://github.com/rodrigoparente/cvejoin-security-dataset.
[12] Biblioteca Python para análise e manipulação de dados: https://pypi.org/project/pandas/.

6.1.2 Descrição dos Atributos do Conjunto de Dados

O conjunto de dados possui 37 atributos, que podem ser agrupados logicamente em dois conjuntos, que são: características da vulnerabilidade e informações de inteligência de ameaças. A seguir, descrevemos esses atributos, sendo os 21 primeiros pertencentes ao grupo de características de vulnerabilidade e os 16 restantes do grupo de inteligência de ameaças:

1. **CVE-ID (*cve_id*):** identificador único e amplamente utilizado pela comunidade de segurança para se referir às vulnerabilidades descobertas. Todo identificador CVE começa com a palavra CVE, seguido de um hífen, do ano de descoberta da vulnerabilidade, outro hífen e uma sequência aleatória de quatro ou cinco dígitos. Como exemplo, podemos citar o CVE-2022-3723[13], que é uma falha no navegador Google Chrome, que permite ao atacante executar códigos de forma arbitrária no contexto do usuário conectado. Seu tipo é *string* e sua fonte é o NVD;
2. **CWE (*cwe*):** sistema utilizado para categorizar as vulnerabilidades de acordo com suas fraquezas de implementação. Todo identificador CWE começa com a palavra CWE, seguido de um hífen e uma sequência de um à quatro dígitos. Como exemplo, temos o CWE-89, que é uma fraqueza de *software* que ocorre quando um *input* de usuário não foi validado corretamente e pode ser utilizado para realizar um ataque do tipo *SQL Injection*. Seu tipo é categórico e sua fonte é o NVD;
3. **Plataforma (*part*):** identifica qual plataforma a vulnerabilidade afeta. Seus possíveis valores são: *hardware*, *software* ou *operating system*. Seu tipo é categórico e sua fonte é o NVD;
4. **Fabricante (*vendor*):** refere-se ao nome do fabricante do produto vulnerável. Ao todo, no conjunto de dados, existem mais de 10 mil fabricantes com produtos vulneráveis cadastrados. Seu tipo é *string* e sua fonte é o NVD;
5. **Produto (*product*):** refere-se ao nome do produto vulnerável. Ao todo, no conjunto de dados, existem mais de 33 mil produtos vulneráveis cadastrados. Seu tipo é *string* e sua fonte é o NVD;
6. **Descrição (*description)*:** breve descrição sobre a vulnerabilidade, que pode conter, entre outras informações, como o ataque é realizado. Seu tipo é *string* e sua fonte é o NVD;
7. **Tipo do CVSS (*cvss_type*):** informa qual a versão do CVSS. As possíveis versões são 2 para o CVSSv2 e 3 para o CVSSv3. Seu tipo é inteiro e sua fonte é o NVD;

[13] Vulnerabilidade descoberta no navegador Chrome: https://nvd.nist.gov/vuln/detail/CVE-2022-3723.

8. **Vetor de Ataque (*attack_vector*):** esse atributo reflete o contexto de rede pelo qual a exploração de vulnerabilidade ocorre. Seu possível valor é: *NETWORK*, quando o ataque pode ocorrer remotamente; *ADJACENT*, quando o ataque pode ser realizado a partir de uma rede adjacente; *LOCAL*, quando o ataque só pode ser realizado a partir da mesma rede; ou *PHYSICAL*, quando o ataque só pode ser realizado com a manipulação física do componente vulnerável. Seu tipo é categórico e sua fonte é o NVD;
9. **Complexidade do Ataque (*attack_ complexity*):** esse atributo descreve a complexidade necessária para o atacante conseguir explorar com sucesso a vulnerabilidade. Seu possível valor é *LOW*, quando existe uma complexidade baixa ou *HIGH*, quando a complexidade é alta. Seu tipo é categórico e sua fonte é o NVD;
10. **Privilégios Necessários (*privileges_required*):** esse atributo descreve o nível de privilégios que um atacante deve possuir antes de explorar com sucesso a vulnerabilidade. Seu possível valor é: *NONE*, quando o ataque não precisa de qualquer autorização para ser realizado; *LOW*, quando é necessária uma autorização básica de usuário; ou *HIGH*, quando é necessário privilégio de administrador. Seu tipo é categórico e sua fonte é o NVD;
11. **Interação com Usuário (*user_interaction*):** esse atributo exprime a necessidade de um usuário humano, diferente do invasor, participar do comprometimento bem-sucedido do componente vulnerável. Seu valor pode ser *NONE*, quando não é necessário interação com o usuário ou *REQUIRED*, quando um usuário precisa realizar alguma ação para o ataque ser bem-sucedido. Seu tipo é categórico e sua fonte é o NVD;
12. **Escopo (*scope*):** esse atributo informa se uma vulnerabilidade em um componente impacta recursos em componentes além de seu escopo de segurança. Seu valor pode ser *UNCHANGED* quando a exploração da vulnerabilidade só afeta o componente vulnerável ou *CHANGED* quando a exploração da vulnerabilidade afeta outros recursos. Seu tipo é categórico e sua fonte é o NVD;
13. **Impacto na Confidencialidade (*confidentiality_impact*):** esse atributo mede o impacto que a exploração da vulnerabilidade terá na confidencialidade do sistema vulnerável. O possível valor é: *NONE*, quando não existe impacto; *LOW*, quando o impacto é baixo; ou *HIGH*, quando existe um alto impacto. Seu tipo é categórico e sua fonte é o NVD;
14. **Impacto na Integridade (*integrity_impact*):** esse atributo mede o impacto que a exploração da vulnerabilidade terá na integridade do sistema vulnerável. Seu possível valor é: *NONE*, quando não existe impacto; *LOW*, quando o impacto é baixo; ou *HIGH*, quando

existe um alto impacto. Seu tipo é categórico e sua fonte é o NVD;

15. **Impacto na Disponibilidade (*availability_impact*):** esse atributo mede o impacto que a exploração da vulnerabilidade terá na disponibilidade do sistema vulnerável. O possível valor é: *NONE*, quando não existe impacto; *LOW*, quando o impacto é baixo; ou *HIGH*, quando existe um alto impacto. Seu tipo é categórico e sua fonte é o NVD;
16. **Nota de Explorabilidade (*exploitability_score*):** é uma métrica que compõe o cálculo do CVSS e reflete a facilidade e os meios técnicos pelos quais a vulnerabilidade pode ser explorada. O valor de explorabilidade é calculado em função dos valores de *attack_vector*, *attack_complexity*, *privileges_required* e *user_interaction*. Seu tipo é decimal e sua fonte é o NVD;
17. **Nota de Impacto (*impact_score*):** é uma métrica que compõe o cálculo do CVSS e corresponde ao impacto geral sofrido pelo componente, caso a exploração da vulnerabilidade seja bem-sucedida. O valor de impacto é calculado em função dos valores de *scope*, *confidentiality_score*, *integrity_score* e *availability_score*. Seu tipo é decimal e sua fonte é o NVD;
18. **Nota de CVSS (*base_score*):** nota que considera as características intrínsecas da vulnerabilidade para definir sua severidade. Esta nota é constante ao longo do tempo e em diferentes ambiente. O seu valor é decimal e sua fonte é o NVD;
19. **Escala Qualitativa do CVSS (*base_severity*):** representação textual da nota de CVSS. Os possíveis valores são: *LOW*, *MEDIUM*, *HIGH* e *CRITICAL*. Seu tipo é categórico e sua fonte é o NVD;
20. **Data de Publicação CVE (*cve_published_date*):** data de publicação do CVE na base de dados do NVD. Seu tipo é *date* (YYYY-MM-DD) e sua fonte é o NVD;
21. **Data da última modificação CVE (*cve_last_modified_date*):** data da última alteração feita nas informações sobre o CVE presentes no NVD. Seu tipo é *date* (YYYY-MM-DD) e sua fonte é o NVD;
22. **Mitre Top** 25 **(*mitre_top_25*):** CWEs considerados perigosos, pelo Mitre, pela sua exploração implicar em um forte impacto na CIA dos sistemas vulneráveis. Seu tipo é booleano, sendo 1 quando a vulnerabilidade possui um CWE pertence a lista e 0 caso contrário. Seu valor é obtido a partir do site do Mitre.
23. **OWASP Top** 10 **(*owasp_top_10*):** lista de CWEs que afetam soluções *web* e considerados perigosos pelo OWASP. Seu tipo é booleano, sendo 1 quando a vulnerabilidade possui um

CWE pertence a lista e 0 caso contrário. Seu valor é obtido a partir do site do OWASP.

24. **Nome do *Exploit* (*exploit_name*):** nome do *exploit* mais recente que afeta a vulnerabilidade. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será nulo. Seu tipo é *string* e sua fonte é o site do ExploitDB;
25. **Data de Publicação do *Exploit* (*exploit_published_date*):** data de publicação do *exploit* mais recente. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será nulo. Seu tipo é *date* (YYYY-MM-DD) e sua fonte é o site do ExploitDB;
26. **Tipo do *Exploit* (*exploit_type*):** tipo do *exploit* mais recente. Os possíveis valores são: *webapps*, *dos*, *remote*, *local*. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será nulo. Seu tipo é categórico e sua fonte é o site do ExploitDB;
27. **Plataforma do *Exploit* (*exploit_platform*):** plataforma afetada pelo *exploit* mais recente. Seus possíveis valores são: *php*, *windows*, *multiple*, *hardware*, *linux*, *java*, *macos*, *jsp*, *android*, *cgi*, *aspx*, *xml*, *windows_x8664*, *ios*, *windows_x86*, *python*, *unix*, *asp*, *solaris*, *ruby*, *json*, *linux_mips*, *nodejs*, *linux_x86*, *linux_x8664*, *openbsd*, *aix*, *osx*, *netbsd_x86*, *cfm*, *freebsd_x86*, *perl*, *watchos*, *freebsd*, *sco*, *lua*, *ashx*, *vxworks* ou *alpha*. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será nulo. Seu tipo é categórico e sua fonte é o site do ExploitDB;
28. **Número de *Exploit*s (*exploit_count*):** número total de *exploits* que afetam a vulnerabilidade. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será 0. Seu tipo é inteiro e sua fonte é o site do ExploitDB;
29. **Tipo de Ataque (*attack_type*):** tipo de ataque utilizado para explorar a vulnerabilidade. Seus possíveis valores são: *remote code execution*, *arbitrary code execution*, *tampering*, *denial of service*, *spoofing*, *defense in depth*, *elevation of privilege*, *security feature bypass*, *information disclosure*, *xss*, *memory leak*, *sql-injection*, *zero-day* e *proof-of-concepts*. Seu tipo é categórico e suas fontes são o NVD, Mitre, OWASP ou Twitter;
30. **EPSS (*epss*):** é um sistema de pontuação que visa definir a probabilidade de uma vulnerabilidade ser explorada nos próximos 30 dias. A nota assume valores decimais entre 0 e 1, com valores próximos a zero indicando que a probabilidade de exploração é baixa e próximos a um, alta. Seu tipo é decimal e sua fonte é o Fórum de Resposta a Incidentes e Equipes de Segurança – *Forum of Incident Response and Security Teams* (FIRST).
31. **Data do Boletim de Segurança (*advisory_published_date*):** data de publicação do boletim de segurança sobre a vulnerabilidade. Caso não exista nenhum boletim de segurança, o

valor no campo é nulo. Seu tipo é *date* (YYYY-MM-DD) e sua fonte são os boletins de seguranças publicados pelos fabricantes, e.g., Microsoft, Intel e Adobe.

32. **Atualização Disponível (*update_available*):** indica se a vulnerabilidade em questão possui um *patch* de segurança que corrige a falha. Seu possível valor é um, quando existe um *patch* e zero, que não necessariamente indica que não existe um *patch*, mas que não conseguimos encontrar um. Seu valor é booleano e sua fonte são os boletins de segurança dos fabricantes.
33. **Referência (*reference*):** *link* para o *patch* de segurança. Caso o valor do campo *update_available* seja zero, este campo terá valor nulo. Seu tipo é *string* e sua fonte são os boletins de seguranças dos fabricantes.
34. **Tendência no Google (*google_trend*):** tendência nas pesquisas realizadas sobre determinado CVE no Google nos últimos 7 dias. Seus possíveis valores são: *decreasing*, *steady* ou *increasing*. Seu valor é categórico e sua fonte é o Google Trends.
35. **Interesse no Google (*google_interest*):** número de pesquisas normalizadas sobre determinado CVE nos últimos 7 dias. Assume valores decimais entre 0 e 1, com valores próximos a zero indicando que houveram poucas pesquisas e próximos a um indicando muitas pesquisas. Seu tipo é decimal e sua fonte é o Google Trends.
36. **Audiência (*audience*):** número máximo de usuários do Twitter que podem ter visto uma postagem sobre um determinado CVE em seu *feed*. É calculado através da soma de todos os seguidores de cada usuário do Twitter que fez um *tweet* ou *retweet* sobre um determinado CVE. Seu valor é inteiro e sua fonte é o Twitter;
37. **Audiência Normalizada (*audience_normalized*):** valores do campo *audience* normalizados. Assume valores decimais entre 0 e 1, com valores próximos a zero indicando que houveram poucas visualizações e próximos a um indicando muitas visualizações. Seu tipo é decimal e sua fonte é o Twitter.

6.1.3 Incorporando Informações de Contexto no Conjunto de Dados

As informações de contexto são específicas para cada ambiente, portanto, não estão inicialmente presentes em nosso conjunto de dados. Automatizar a coleta dessas informações não é uma tarefa trivial e requer conhecimento de ferramentas de inventário de rede. No entanto, como essas informações são cruciais para o gerenciamento de risco das vulnerabilidades, desenvolvemos um método que facilita a incorporação de dados de contexto em nosso conjunto

de dados.

Para isso, o usuário pode executar um *script*, passando como entrada um arquivo CSV contendo os identificadores dos ativos da rede (por exemplo, seus endereços IP), os dados de contexto associados e os CVEs das vulnerabilidades. A saída é um arquivo CSV contendo características das vulnerabilidades, inteligência de ameaças e informações de contexto mapeadas pelo identificador de ativo, conforme pode ser visto na Figura 28. O *script* pode ser encontrado no GitHub do *dataset*.

Figura 28 – A saída do *script* que une as informações de contexto dos ativos com as informações presentes no conjunto de dados. Como pode ser visto na figura, cada linha do CSV traz informações sobre uma vulnerabilidade associada a um ativo vulnerável (indicado pela coluna *ASSET_ID*).

	ASSET_ID	CVE_ID	PART	VENDOR	BASE_SCORE	...	SECURITY_ADVISORY	OWASP_TOP_10	EXPLOIT_COUNT	EPSS	ATTACK_TYPE
771	ASSET-77	CVE-2020-9496	application	other	6.1	...	0	1	1.0	0.94875	['xss']
2198	ASSET-219	CVE-2021-35464	application	other	9.8	...	1	1	1.0	0.94376	['remote code execution']
657	ASSET-65	CVE-2021-41773	application	other	7.5	...	0	1	2.0	0.93300	['remote code execution']
918	ASSET-91	CVE-2020-13151	application	other	9.8	...	1	1	1.0	0.92212	NaN
788	ASSET-78	CVE-2021-36260	hardware	other	9.8	...	0	1	1.0	0.89301	NaN

Fonte: elaborado pelo autor.

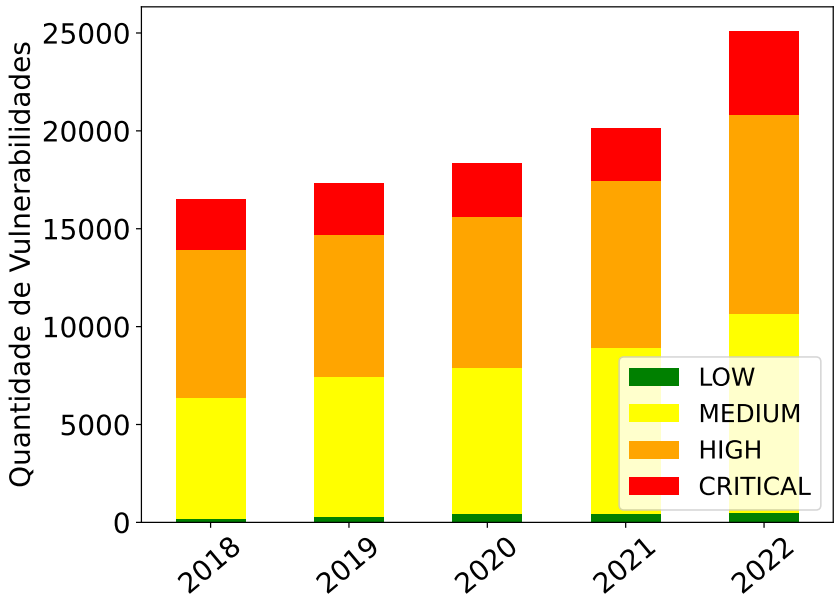
6.2 Análise Exploratória dos Dados Obtidos

As análises a seguir foram feitas considerando apenas as vulnerabilidades publicadas entre 2018 e 2022, ou seja, nos últimos cinco anos. Em 2022, foram publicadas mais de 25.000 vulnerabilidades únicas no NVD. Um aumento de aproximadamente 25% em relação ao ano de 2021. Se analisarmos a quantidade de vulnerabilidades reportadas ao longo dos anos, como mostra a Figura 29, podemos perceber que o número de vulnerabilidades descobertas vem crescendo desde 2018. Aliado a esse fato, se considerarmos a escassez de trabalhadores qualificados na área de cibersegurança (FURNELL *et al.*, 2017), fica clara a importância de usar estratégias para analisar, priorizar e remediar essas vulnerabilidades.

Alguns analistas costumam utilizar o CVSS como métrica única no processo de priorização, pois acreditam que o CVSS indica o risco de exploração das vulnerabilidades. Dessa forma, focam seus esforços na correção de vulnerabilidades com severidade CRITICAL. No entanto, essa estratégia é inadequada, pois o CVSS foi projetado para indicar a severidade das vulnerabilidades e não o seu risco de exploração.

Um estudo realizado pela empresa de segurança RecordFuture mostrou que entre as dez vulnerabilidades mais exploradas de 2020, apenas 4 delas possuíam um CVSS CRITICAL (RecordFuture, 2021). Além disso, podemos observar, na Figura 29, que o número de vulnerabilidades que possuem CVSS MEDIUM e HIGH são superior a 50%. E vale destacar que a severidade CRITICAL aparece apenas para vulnerabilidades publicadas após 2016, quando a versão *v3* do CVSS foi lançada.

Figura 29 – Distribuição de vulnerabilidades por severidade ao longos dos anos. Como pode ser visto na figura, a maioria das vulnerabilidades possui um CVSS HIGH ou MEDIUM. Vulnerabilidades com CVSS CRITICAL aparecem em terceiro lugar em quantidade.

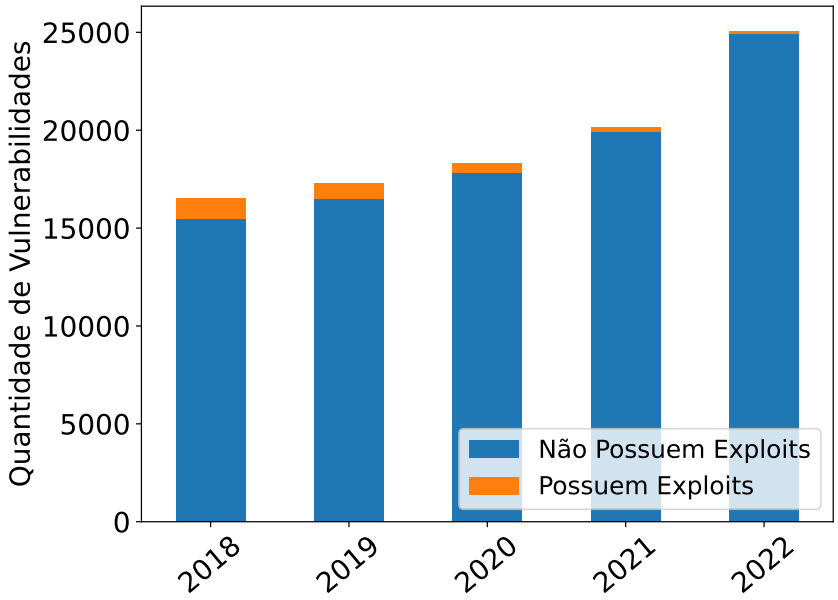


Fonte: elaborado pelo autor.

Assim, além do CVSS, é relevante considerar outras informações, como, por exemplo, a presença de *exploits* públicos, informação pertencente ao grupo de inteligência de ameaças. A Figura 30 mostra o número de vulnerabilidades que possui *exploits* públicos em relação àquelas que não possuem. Podemos perceber que a quantidade afetada pela presença de *exploits* é apenas uma pequena fração do total. No entanto, essas vulnerabilidades são muitas vezes consideradas mais críticas. Outra análise realizada foi a distribuição dos *exploits* por severidade, que mostrou que vulnerabilidades com CVSS HIGH possuem a maior quantidade de *exploits*, aproximadamente 43% do valor total. Assim, ao considerar apenas o CVSS, estamos subestimando o risco das vulnerabilidades serem exploradas.

Adicionalmente, para aprofundar o conhecimento acerca das vulnerabilidades presentes nas redes das organizações, os profissionais de segurança devem reunir e analisar informações sobre suas características. Por exemplo, a Figura 31 mostra que as vulnerabilidades são classifi-

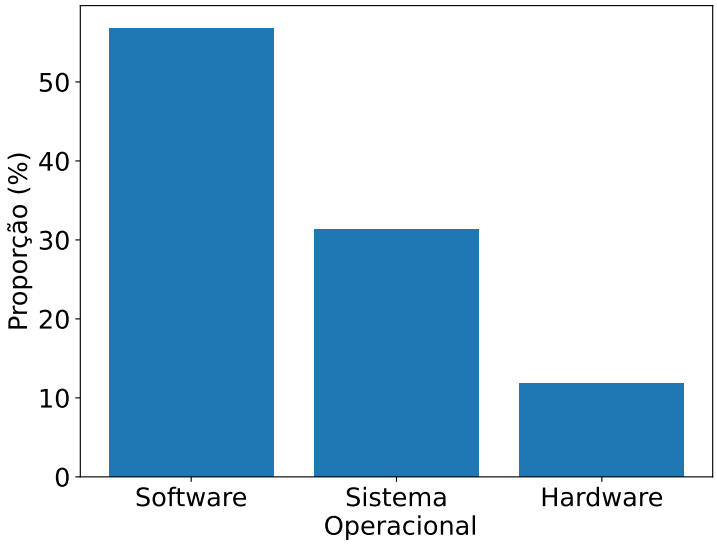
Figura 30 – Distribuição de *exploits* por ano. O número de vulnerabilidades com *exploits* públicos é muito menor do que aquelas que não possuem *exploits*. No entanto, eles são as mais perigosas, pois usuários mal-intencionados podem usar esses *exploits* para explorar as vulnerabilidades com uma maior facilidade.



Fonte: elaborado pelo autor.

cadas em três grupos, que são: *hardware*, *software* e sistemas operacionais. Observe que 88% de todas as vulnerabilidades se concentram nos grupos de *software* e sistema operacional.

Figura 31 – Distribuição das vulnerabilidades por plataforma. Podemos observar na figura, que mais da metade de todas as falhas de segurança publicadas no NVD são de *software*. Em segundo lugar, temos aquelas que afetam os sistemas operacionais e, em terceiro, as vulnerabilidades de *hardware*.

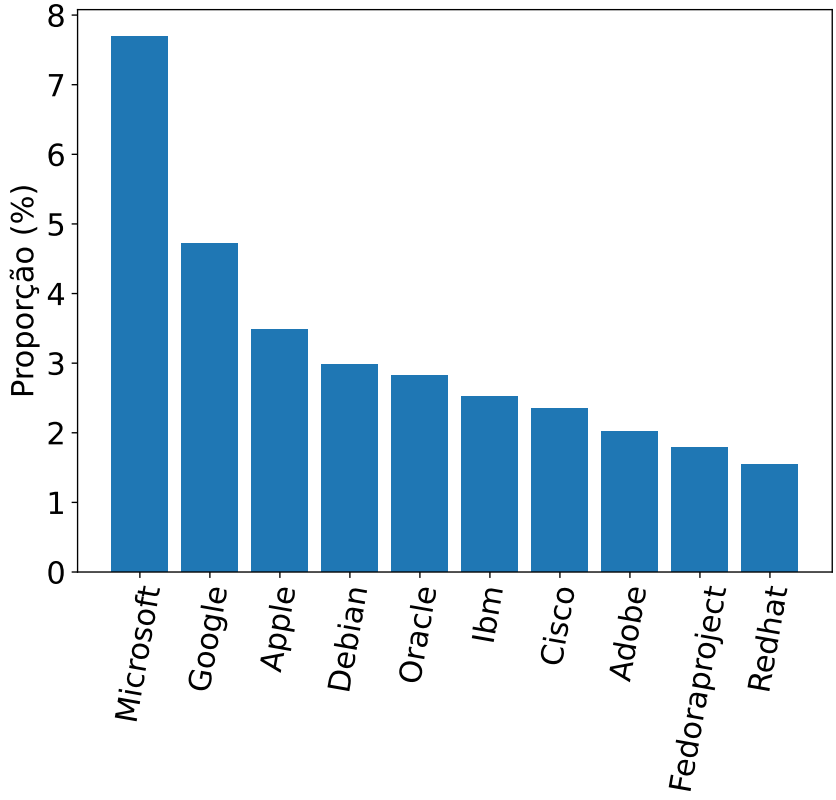


Fonte: elaborado pelo autor.

Outra forma de classificar as vulnerabilidades é quanto ao fabricante do produto

vulnerável. A Figura 32, mostra os dez fabricantes com o maior número de vulnerabilidades. Nota-se que a Microsoft é o fabricante de produtos com o maior número de vulnerabilidades publicadas (com aproximadamente 8%), seguida do Google (com 5%) e da Apple (com 4%). A análise mostrou que esses dez fabricantes são responsáveis por aproximadamente 32% de todas as vulnerabilidades publicadas no NVD.

Figura 32 – Os dez fabricantes mais vulneráveis. Esses 10 fabricantes são a fonte de aproximadamente 32% de todas as vulnerabilidades publicadas no NVD, e a empresa Microsoft lidera o *ranking* com quase 8% das falhas de segurança.



Fonte: elaborado pelo autor.

Além do NVD, existem também organizações de segurança, como Mitre e OWASP, que publicam em seus sites, listas dos pontos fracos mais críticos encontrados em vulnerabilidades de *software*. Essas fraquezas, conhecidas como CWE, são indicadores importantes que auxiliam o analista a entender como ocorre a exploração da vulnerabilidade e o que pode ser feito para se proteger de possíveis ataques. Pensando nisso, compilamos uma lista com os dez principais CWE que mais afetam as vulnerabilidades de *software*, conforme apresentado na Tabela 3. Podemos notar a presença de falhas bem conhecidas, como *cross-site scripting* (XSS) e SQL *Injection*, que podem ser evitados através do emprego de práticas simples de sanitização de *inputs* dos usuários.

Tabela 3 – Os dez CWEs que mais afetam vulnerabilidades de *software*. Em primeiro lugar, temos o CWE-79, uma fraqueza que permite a execução de um ataque do tipo XSS e que afeta 25% de todas as vulnerabilidades de *software*.

Ranking	**CWE-ID**	**Description**
1º	CWE-79	Improper neutralization of input during web page generation (cross-site scripting)
2º	CWE-787	Out-of-bounds write
3º	CWE-89	Improper neutralization of special elements used in an SQL command (SQL injection)
4º	CWE-20	Improper input validation
5º	CWE-125	Out-of-bounds read
6º	CWE-416	Use After Free
7º	CWE-22	Improper limitation of a pathname to a restricted directory
8º	CWE-352	Cross-site request forgery (CSRF)
9º	CWE-200	Exposure of sensitive information to an unauthorized actor
10º	CWE-119	Improper restriction of operations within the bounds of a memory buffer

Por fim, interessa também a análise do que se fala sobre as vulnerabilidades nas redes sociais. Ao consultar o Twitter pelo CVE de uma vulnerabilidade, podemos contabilizar o número de pessoas falando sobre essa falha e seu alcance (ou seja, o número de visualizações). Assim, é possível identificar aquelas que são críticas e merecem atenção. A Tabela 4 mostra as dez vulnerabilidades mais citadas no Twitter em 8 de janeiro de 2022. É possível notar a presença da vulnerabilidade Apache Log4j, na primeira posição da tabela, que foi um *zero-day* que afetou milhões de dispositivos Java em todo o mundo [14]

Além disso, podemos observar na Tabela 4, algumas vulnerabilidades que, apesar de não terem sido publicadas oficialmente no NVD, são conhecidas e consideradas críticas pela comunidade de segurança. Como, no caso do CVE-2021-44142, que ocupa a 7º posição da tabela. Esta vulnerabilidade é uma falha na implementação do protocolo Samba[15]. Vale destacar que, em 21 de janeiro, 13 dias após nossa análise, o CVE-2021-44142 foi publicado no NVD,

[14] Notícia sobre a vulnerabilidade Log4J e como ela foi explorada: https://blog.qualys.com/vulnerabilities-threat-research/2021/12/10/apache-log4j2-zero-day-exploited-in-the-wild-log4shell.

[15] Notícia sobre o CVE-2021-44142: https://www.helpnetsecurity.com/2022/02/02/samba-bug-may-allow-code-execution-as-root-on-linux-machines-nas-devices-cve-2021-44142/.

recebendo uma pontuação base de 8,8 (CVSS HIGH). Isso demonstra como as redes sociais são uma importante fonte de informação a ser utilizada no processo de VM.

Tabela 4 – Vulnerabilidades mais mencionadas no Twitter em 8 de janeiro de 2022. Na primeira posição, podemos observar a vulnerabilidade Apache Log4j, um *zero-day* crítico que afetou milhões de dispositivos Java no mundo todo.

Rank	CVE	Name	Publication date	CVSS Score	Views
1º	CVE-2021-44228	Apache Log4j	Dec. 2021	10	450K
2º	CVE-2022-21882	Windows Server	Jan. 2022	7,8	270K
3º	CVE-2021-4034	Polkit	Jan. 2022	7,8	110K
4º	CVE-2022-0509	–	Feb. 2022	–	110k
5º	CVE-2021-39137	Go Ethereum	Aug. 2021	7,5	100K
6º	CVE-2014-6271	GNU Bash	Sept. 2014	9,8	100K
7º	CVE-2021-44142	–	–	–	97K
8º	CVE-2022-23263	–	Feb. 2022	7,7	95K
9º	CVE-2022-23262	–	Feb. 2022	6,3	95K
10º	CVE-2020-24807	NodeJS	Oct. 2020	7,8	83K

6.3 Conclusão

Neste capítulo, apresentamos um conjunto de dados contendo mais de $200,000$ vulnerabilidades de segurança da informação, publicadas entre 2002 e 2022 no NVD. No *dataset*, é possível encontrar informações sobre as características das vulnerabilidades e inteligência de ameaças, de falhas que afetam *hardwares*, *softwares* e sistemas operacionais. Esses dados foram coletados automaticamente de 10 fontes distintas, através do uso da linguagem de programação Python e de bibliotecas, que permitem acessar e manipular o conteúdo de sites a partir de suas URLs. Além disso, dada a importância das características de contexto na classificação de risco das vulnerabilidades, disponibilizamos um *script* capaz de incorporar essas informações no *dataset*. Esse resultado é importante, pois a pesquisa bibliográfica, discutida na Subseção 4.2, demonstrou que nenhum outro trabalho, que propõe a criação de um *dataset* de segurança, é tão extenso e completo quanto o nosso. Os *scripts* desenvolvidos e utilizados na coleta dos dados foi disponibilizado publicamente no GitHub.

Além disso, realizamos uma análise exploratória das vulnerabilidades publicadas entre os anos de 2018 e 2022. Esse estudo foi importante, pois serviu para: (*i*) demonstrar que o uso exclusivo do CVSS, como política de priorização e correção das vulnerabilidades, é uma metodologia equivocada que pode deixar a rede das organizações vulneráveis; e (*ii*)

identificar características importantes acerca das vulnerabilidades que pudessem ser utilizadas na classificação de risco, como, por exemplo, a presença de *exploits* públicos e o que os usuários estão falando sobre essas vulnerabilidades nas redes sociais.

7 AVALIAÇÃO DE DESEMPENHO DO APRENDIZADO ATIVO

Para avaliar o módulo de Classificação e Priorização das Vulnerabilidades (apresentado na Subsecção 5.2.3), fizemos diversos experimentos envolvendo o aprendizado ativo (descrito na Seção 3.2) em comparação com a rotulação das vulnerabilidades por seleção aleatória. Além disso, analisamos como as diferentes estratégias de consulta utilizadas pelo AL (descritas na Subseção 3.2.2), as técnicas de aprendizado supervisionado e semissupervisionado e os diferentes algoritmos de calibração de probabilidade (descritos na Subseção 3.3) influenciam na acurácia e no desempenho do classificador.

Vale destacar, que em nossos experimentos iniciais testamos diversas técnicas baseadas em modelos generativos semissupervisionados, como *Variational Autoencoder* (ZHANG *et al.*, 2019), *Auxiliary Deep Generative Model* (MAALØE *et al.*, 2016) e a combinação de modelos de variáveis latentes com modelos discriminantes (KINGMA *et al.*, 2014), mais conhecidos como M1 + M2. No entanto, nenhuma dessas técnicas apresentou um ganho significativo no desempenho em relação à técnica de aprendizado supervisionado. Portanto, escolhemos utilizar nos nossos testes, a técnica de aprendizado semissupervisionada chamada de autoaprendizado, em inglês *self-training*, que permite que o modelo aprenda consumindo simultaneamente poucos dados rotulados e muitos dados não rotulados (YAROWSKY, 1995).

A técnica de autoaprendizado permite que diferentes algoritmos de ML sejam utilizados no processo de aprendizado. Assim, um modelo de ML, treinado utilizando os dados rotulados, é usado para prever as pseudo-classes dos dados não rotulados (TANHA *et al.*, 2017). Então, algumas instâncias que satisfaçam um critério pré-estabelecido (e.g., probabilidade de predição maior que 90%) são selecionadas e incorporadas ao conjunto de dados rotulados, que será utilizado para re-treinar o modelo de ML. Esse processo é repetido até que todos os dados sejam rotulados, nenhuma instância satisfaça o critério de pseudo-rotulação ou que um número máximo de iterações tenha sido alcançado.

Assim, este capítulo se divide da seguinte forma: (*i*) na Seção 7.1 descreveremos a infraestrutura utilizada nos nossos testes; (*ii*) na Seção 7.2 descrevemos o processo de criação do conjunto de dados utilizados nos experimentos, a rotulação *offline* dessas vulnerabilidades e o Oráculo Simulado; (*iii*) na Seção 7.3, avaliamos qual a melhor estratégia de consulta utilizado pelo AL; (*iv*) na Seção 7.4 avaliamos o desempenho do aprendizado supervisionado e semissupervisionado ao se utilizar o aprendizado ativo em comparação com uma solução de *baseline* (estratégia de rotulação simples por seleção aleatória das vulnerabilidades); (*v*) na Seção

7.5 analisamos a influência da calibração de probabilidades no resultado do classificador; por fim, (vi) na Seção 7.6 apresentamos o resultado de um experimento de rotulação *online* das vulnerabilidades.

7.1 Cenário de Teste

Os experimentos apresentados nas seções a seguir foram realizados no Centro Nacional de Processamento de Alto Desempenho (CENAPAD) da Universidade Federal do Ceará (UFC). Esse centro, é uma estrutura organizacional ligada à Pró-Reitoria de Pesquisa e Pós-Graduação da UFC, que integra o consórcio Sistema Nacional de Processamento de Alto Desempenho (SINAPAD). Esse consórcio reúne dez centros, espalhados pelo Brasil, que formam uma grade computacional que visa fornecer os recursos de Processamento de Alto Desempenho (PAD) necessários para o desenvolvimento científico e tecnológico do país. Para a execução dos nossos experimentos, o CENAPAD disponibilizou um servidor com 2 processadores Intel X5650 Hexa-Core 2,67 Ghz, 24 GB de memória DDR3@1333MHz e um disco de 250 GB SATA3.

7.2 Conjunto de Dados, Rotulação *Offline* e Oráculo Simulado

O aprendizado ativo é um processo interativo que ocorre entre o modelo de AL e o especialista de segurança, consultado acerca do melhor rótulo a ser aplicado a instância selecionada. No entanto, dado a dificuldade de se conseguir um tempo na agenda dos especialistas para realizar os experimentos em tempo real e a necessidade de executar os testes múltiplas vezes a fim de calcular as estatísticas de interesse, optou-se por rotular previamente um pequeno conjunto de dados que pudesse ser utilizado exaustivamente nos testes. Assim, a seguir, descreveremos o conjunto de dados selecionado, o processo de rotulação *offline* e o mecanismo utilizado para viabilizar os testes com o AL, i.e., o Oráculo Simulado.

7.2.1 Descrição do dataset

Utilizamos os dados coletados acerca das mais de 200.000 vulnerabilidades presentes no NIST (processo apresentado no Capítulo 6), para criar um subconjunto de 260 amostras, contendo informações sobre vulnerabilidades, inteligência de ameaças e contexto. Essa quantidade de amostras foi escolhida de forma arbitrária, garantindo-se apenas que tivesse uma quantidade mínima de dados para se trabalhar nos experimentos (20 amostras iniciais, 100 de treinamento e

40 de testes) e fosse possível rotulá-la manualmente.

O novo *dataset*, possui 24 atributos, sendo eles: 8 características da vulnerabilidade, i.e., plataforma (*platform*), fabricante (*vendor*), nota do CVSS (*base_score*), impacto na confiabilidade (*confidentiality_impact)*, integridade (*integrity_impact*) e disponibilidade (*availability_impact)*, data de publicação (*cve_published_date*) e presença de atualização de segurança (*update_available*); 10 informações de inteligência de ameaças, i.e., mitre top 25 (*mitre_top_25*), OWASP top 10 (*owasp_top_10*), quantidade de *exploits* públicos (*exploit_count*), idade do *exploit* mais recente (*exploit_published_date*), EPSS (*epss*), presença de boletins de segurança dos fabricantes (*advisory_published_date*), tipo do ataque (*attack_type*), tendência e interesse nas pesquisas do Google (*google_trend* e *google_interest*) e audiência no Twitter (*audience*); e por fim, 6 dados de contexto, descritos a seguir, por não estarem presentes no conjunto de dados original.

- **Topologia (*topology*):** indica onde na rede está o ativo, que pode ser na rede local ou na DMZ. As redes locais são mais seguras porque são protegidas por *firewalls* robustos. Além disso, a comunicação com ativos dessas redes só é possível a partir da mesma rede ou por meio de VPNs. Os ativos na DMZ são mais críticos porque geralmente são visíveis a partir da internet, o que permite que todos os usuários, inclusive os mal-intencionados, tenham acesso direto a eles. Esse atributo é do tipo *string* e pode assumir o valor *DMZ* ou *LOCAL*;
- **Tipo do Ativo (*asset_type*):** indica a finalidade do ativo, que pode ser uma estação de trabalho ou um servidor. Os servidores geralmente armazenam sistemas e bancos de dados críticos, que possuem informações confidenciais, tornando-os alvos potenciais de ataques cibernéticos. Esse atributo é do tipo *string* e pode assumir o valor *SERVER* ou *WORKSTATION*;
- **Ambiente (*environment*):** indica o ambiente onde o ativo está localizado e pode ser de desenvolvimento ou produção. Os ativos de desenvolvimento geralmente estão em redes segregadas, atrás de *firewalls* e não possuem dados confidenciais. Assim, eles são menos críticos do que os ativos de produção, que normalmente contêm informações confidenciais e estão acessíveis a mais pessoas. Esse atributo é do tipo *string* e pode assumir o valor *PRODUCTION* ou *DEVELOPMENT*;
- **Dados Sensíveis (*sensitive_data*):** indica se o tipo de dado armazenado no ativo é sensível ou não. Servidores que armazenam informações confidenciais de usuários, como nome completo e números de cartão de crédito, são mais críticos. Esse atributo é do tipo booleano

e pode assumir o valor 0 quando não armazena dados sensíveis ou 1 caso contrário;

- **Fim da vida (*end_of_life*):** indica se o ativo em questão chegou ao fim de sua vida útil, ou seja, se receberá ou não atualizações de segurança. Esse atributo é importante, pois é comum ter ativos legados com *softwares* ou sistemas operacionais que não recebem mais atualizações. Esses ativos são críticos e devem receber atenção especial por suas vulnerabilidades serem mais difíceis de corrigir. Esse atributo é do tipo booleano e pode assumir o valor 0 quando o ativo não está no fim de sua vida útil ou 1 caso contrário;
- **Ativo crítico (*critical_asset*):** indica se o ativo é crítico para a organização e pode ser um alvo potencial de invasores. *Active Directories* (ferramenta que permite o controle de *softwares*, arquivos e contas de usuários na rede das organizações), cofres de senhas ou computadores pessoais de executivos são exemplos de ativos críticos que devem receber atenção especial. Esse atributo é do tipo booleano e pode assumir o valor 0 quando não for um ativo sensível e 1 caso contrário.

É importante observar que, embora estejamos usando esse conjunto específico de dados, a metodologia proposta nesta tese é geral e pode ser aplicada a qualquer *dataset*. No entanto, para se avaliar corretamente o risco de exploração de vulnerabilidades, os dados utilizados devem possuir, necessariamente, informações que possam caracterizar vulnerabilidades, ameaças e contexto.

7.2.2 Rotulação Offline

Pedimos que três especialistas de cibersegurança de diferentes empresas, sendo um do Centro de Operações de Segurança – *Security Operations Center* (SOC) –, um da equipe de Inteligência de Ameaças e um da equipe de Gestão de Vulnerabilidades, realizassem a classificação de risco de todas as 260 vulnerabilidades selecionadas. Devido a conflitos em suas agendas, esse processo foi feito de forma independente e manual por cada um dos especialistas. O processo foi realizado ao longo de múltiplos dias e levou em média 5 horas e meia. É importante destacar que os especialistas foram escolhidos por possuírem anos de experiência na área de cibersegurança, tendo alcançado cargos de diretores e líderes técnicos. Assim, são profissionais capacitados e capazes de estimar com segurança o risco de exploração das vulnerabilidades, dada a grande experiência que possuem no assunto.

Após o processo de rotulação, analisamos os rótulos escolhidos pelos especialistas e removemos todas as vulnerabilidades onde não foi possível formar uma votação majoritária,

ou seja, removemos cada instância votada com 3 rótulos diferentes. Tais vulnerabilidades são consideradas uma fonte de ruído que pode piorar o desempenho do classificador. Como resultado, 52 vulnerabilidades foram removidas. Dessa forma, o *dataset* final utilizado nos testes, contém 208 itens, rotulados da seguinte forma: 60 como LOW ($\approx 28\%$); 45 como MODERATE ($\approx 22\%$); 45 como IMPORTANT ($\approx 22\%$); e 58 como CRITICAL ($\approx 28\%$).

7.2.3 Pré-processamento dos dados

Os atributos presentes no *dataset* são contínuos (3), discretos (4), booleanos (7) e categóricos (10). Assim, antes de executarmos o treinamento dos modelos de ML, foi necessário realizar um pré-processamento nos dados. Esse processo envolveu transformar os dados categóricos em valores numéricos usando a técnica de *one-hot-encoding*, pois os elementos não possuem nenhuma hierarquia entre eles e adicionalmente, quando necessário, normalizar os dados com média zero e desvio padrão unitário.

7.2.4 Métricas de avaliação

Existem diferentes indicadores e métricas que podem ser utilizadas para avaliar o desempenho de modelos de ML. Nos nossos experimentos, concentramos nossas análises na matriz de confusão e nos valores de acurácia, precisão, revocação e *f1-score*. A seguir, apresentamos uma breve explicação acerca dessas métricas e como calculá-las.

Matriz de Confusão, também chamada de Matriz de Erro, é uma tabela que descreve o desempenho de um modelo de ML através de seus valores de: Verdadeiro Positivo – *True Positive* (TP) –, que corresponde ao número de amostras positivas classificadas corretamente; Verdadeiro Negativo – *True Negative* (TN) –, número de amostras negativas classificadas corretamente; Falso Positivo – *False Positives* (FP) –, número de amostras negativas que são erroneamente classificadas como positivas; e Falso Negativo – *False Negatives* (FN) –, número de amostras positivas classificadas como negativas. A Tabela 5 mostra a disposição desses valores em uma Matriz de Confusão para um problema binário com duas classes, uma positiva e outra negativa.

Acurácia, do inglês *accuracy*, é a razão entre a quantidade de amostras classificadas corretamente e a quantidade total de amostras no conjunto de dados. O valor de acurácia pode ser calculado através da seguinte equação

$$\text{Accuracy} = \frac{TP+TN}{TN+FP+FN+TP}, \tag{7.1}$$

Tabela 5 – Matriz de Confusão é uma métrica que permite a visualização do desempenho de modelos de ML. Na tabela, podemos ver como se daria a disposição dos valores de TP, FN, FP, TN para um problema binário com duas classes, sendo uma positiva e outra negativa.

		Classe Predita	
		Condição Positiva	Condição Negativa
Classe Verdadeira	Condição Positiva	TP	FN
	Condição Negativa	FP	TN

onde TP, TN, FP e FN equivalem aos valores de verdadeiro positivo, verdadeiro negativo, falso positivo e falso negativo respectivamente. Valores maiores de acurácia são desejáveis.

Precisão, do inglês *precision*, é a razão entre a quantidade de amostras classificadas corretamente como positivas e o total de amostras classificadas como positivas. O valor de precisão pode ser calculado através da equação

$$\text{Precision} = \frac{TP}{TP+FP}, \tag{7.2}$$

onde TP equivale ao valor de verdadeiro positivo e FP de falso positivo. Um valor maior de precisão indica um melhor desempenho do modelo de ML.

Revocação, do inglês *recall*, é a razão entre a quantidade de amostras classificadas corretamente como positivas e a quantidade de exemplos de fato positivas. O valor de revocação pode ser calculado através da equação

$$\text{recall} = \frac{TP}{TP+FN}, \tag{7.3}$$

onde TP equivale ao valor de verdadeiro positivo e FN de falso negativo. Assim como na precisão, valores mais altos de revocação indicam um melhor desempenho do modelo de ML.

F1-score é outra forma de se calcular a acurácia do modelo, utilizando os valores de Precisão (Equação 7.2) e Revocação (Equação 7.3). Portanto, essa métrica tende a ser um resumo da qualidade do modelo e pode ser calculada através da equação

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{7.4}$$

Assim, se o valor de *f1-score* for alto, isso quer dizer que o modelo de ML tanto consegue acertar suas predições (alto valor de precisão), quanto recuperar as amostras da classe de interesse (alto valor de revocação).

É importante observar que, como nosso problema é multiclasse (como discutido na Subseção 5.2.2.1), foi necessário considerar uma média ponderada para calcular os valores de precisão, revocação e *f1-score*. Assim, primeiro calculamos as métricas para cada rótulo, depois

encontramos a média ponderada pelo suporte, ou seja, o número de instâncias verdadeiras para cada rótulo.

7.2.5 Oráculo Simulado

Para simular a interação do usuário com o modelo de aprendizado ativo, desenvolvemos um *script* chamado de Oráculo Simulado. Esse programa é capaz de fornecer o rótulo correspondente à instância selecionada pela técnica de consulta utilizada pelo AL. Isso é feito através de uma consulta aos rótulos previamente dados pelos especialistas às vulnerabilidades pertencentes ao conjunto de dados (processo descrito anteriormente na Subseção 7.2.2). Isso permite que os testes sejam executados programaticamente e repetidas vezes.

7.3 Impacto das Estratégias de Consulta Utilizada pelo Aprendizado Ativo no Desempenho do Classificador

Nesta seção, discutiremos o desempenho das diferentes estratégias de consultas que podem ser utilizadas pelo modelo de AL para selecionar as instâncias a serem rotuladas pelos especialistas.

7.3.1 Descrição dos Experimentos

Como discutido na Subseção 3.2.2, existem diferentes técnicas de consulta que podem ser utilizadas pelo AL para selecionar as instâncias a serem rotuladas. No que lhes concerne, essas técnicas podem ser agrupadas em duas abordagens distintas, que são: a amostragem por incerteza e por comitê. Assim, a fim de identificar qual estratégia apresenta o melhor desempenho para nosso problema, analisamos as duas principais técnicas de cada uma dessas abordagens. Da abordagem por incerteza, analisamos as técnicas de amostragem por Entropia e por Menor Confiança. Já na abordagem por comitê, analisamos as técnicas de Divergência Média de Kullback-Leibler e amostragem por Entropia dos Votos.

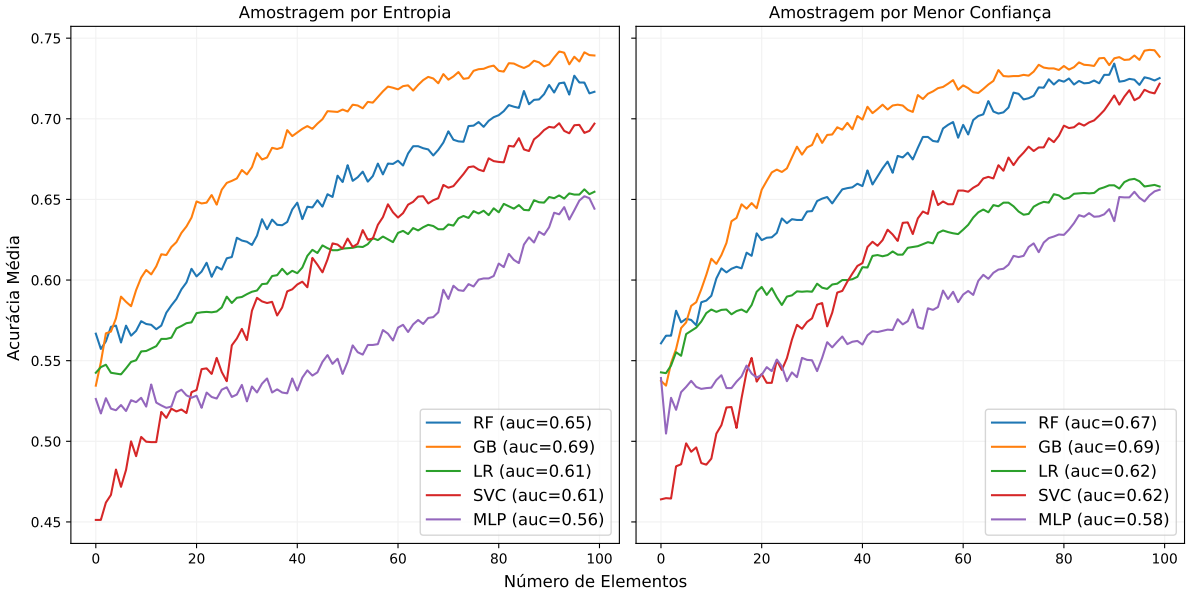
Dessa forma, montamos 4 cenários de experimentos, onde avaliamos o desempenho das técnicas de consulta mencionadas acima, com os seguintes algoritmos de ML: *Random Forest* (RF), *Gradient Boosting* (GB), *Logistic Regression* (LR), *Support Vector Classification* (SVC) e *Multilayer Perceptron* (MLP). Tal escolha de modelos visa incluir estratégias de aprendizagem distintas, desde classificadores lineares até redes neurais artificiais. O *dataset* de treinamento e

teste utilizado pelo modelo de ML foi o conjunto de vulnerabilidades descrito na Subseção 7.2.1. Já para simular a interação do analista com os modelos de AL, utilizamos o Oráculo Simulado, descrito na Subseção 7.2.5. Por fim, vale destacar que cada teste foi repetido 100 vezes, para se calcular as estatísticas de interesse mostradas na discussão a seguir.

7.3.2 Discussão dos Resultados

Nos dois primeiros cenários de teste, analisamos os valores médios de acurácia obtidos utilizando-se as técnicas de Amostragem por Entropia e por Menor Confiança. Podemos observar na Figura 33, que o comportamento das curvas entrem as duas técnicas se assemelham bastante. Isso é corroborado pelos valores das Áreas Sob as Curvas – – *Area Under the Curve* (AUC). Todavia, a técnica de Amostragem por Menor Confiança se saiu melhor que a Amostragem por Entropia, com valores médios de acurácia e AUC maiores para todos os algoritmos de ML testados.

Figura 33 – Após a execução de 100 consultas pelo AL, os diferentes algoritmos testados apresentaram valores de acurácia e AUC próximos para as duas técnicas de consulta por incerteza avaliadas. Isso fica evidenciado pela posição final das curvas de acurácia nos gráficos e pelos valores de AUC nas legendas. No entanto, a amostragem por Menor Confiança foi a técnica de consulta por incerteza que obteve os melhores resultados, se sobressaindo sobre a técnica de amostragem por Entropia.

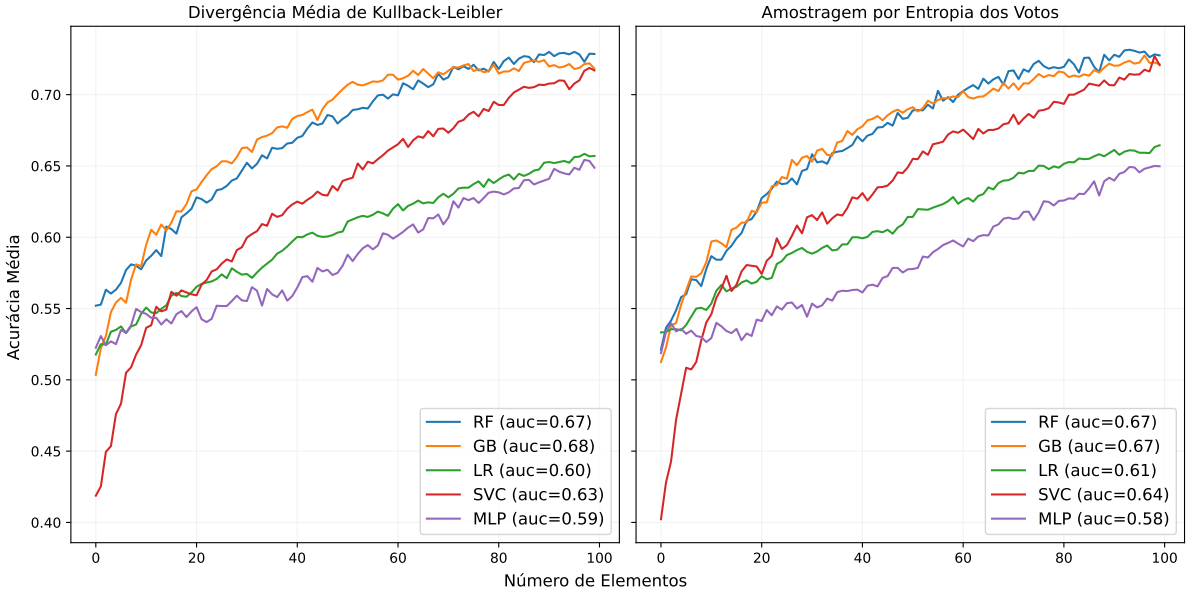


Fonte: elaborado pelo autor.

Já nos dois cenários restantes, analisamos as estratégias de Amostragem por Comitê. Podemos notar, pela análise da Figura 34, que assim como nas técnicas de amostragem por

incerteza, as curvas possuem um comportamento semelhante. No entanto, o desempenho do algoritmo SVC inicialmente foi pior que os demais algoritmos. Além disso, é possível notar, também, que a distância entre as curvas diminuiu, a ponto das curvas dos algoritmos RF e GB se cruzarem diversas vezes. Por fim, vemos que, em média, a técnica que utiliza a Divergência Média de Kullback-Leibler se saiu melhor que a Entropia dos Votos.

Figura 34 – Após a execução de 100 consultas pelo AL, os diferentes algoritmos testados apresentaram valores de acurácia e AUC próximos para as duas técnicas de consulta por comitê avaliadas. Isso fica evidenciado pela posição final das curvas de acurácia nos gráficos e pelos valores de AUC nas legendas. No entanto, a Divergência Média de Kullback-Leibler foi a técnica de consulta por comitê que obteve os melhores resultados, se sobressaindo sobre a técnica de amostragem por Entropia dos Votos.

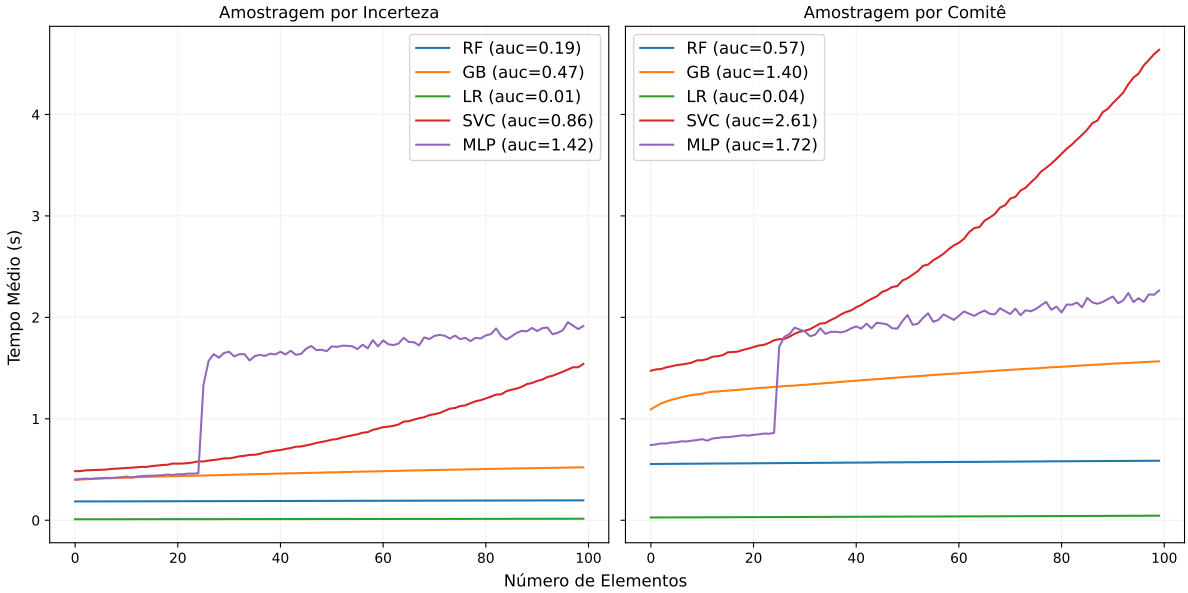


Fonte: elaborado pelo autor.

A Figura 35 apresenta os valores médios dos tempos de execução em segundos das duas abordagens analisadas, isto é, amostragem por incerteza e por comitê. Podemos perceber, pela análise da figura, que as técnicas que utilizam amostragem por comitê consomem mais tempo para executar, isso ocorre devido à necessidade de treinar vários classificadores para compor o comitê. Em média, as técnicas que utilizam amostragem por incerteza foram 33% mais rápidas, com exceção do algoritmo MLP que apresentou um tempo de execução simular para as duas abordagens.

Por fim, a Tabela 6 mostra os valores médios de acurácias e desvio padrão após 100 consultas realizadas pelo AL para os diferentes cenários analisados. Assim, concluímos que, por apresentar os melhores valores de acurácia e os menores tempos de execução para os algoritmos testados, a técnica de amostragem por Menor Confiança apresenta o melhor desempenho para

Figura 35 – Calculamos os valores médios dos tempos de execução em segundos para as duas abordagens de consulta (amostragem por incerteza e por comitê) e plotamos os gráficos a seguir. Podemos perceber, pela análise da figura, que os tempos observados na amostragem por comitê foram maiores, visto a necessidade de se treinar múltiplos classificadores para compor o comitê.



Fonte: elaborado pelo autor.

o nosso problema. Portanto, será a técnica utilizada nos próximos experimentos envolvendo o AL. Na próxima seção, abordaremos em detalhes o comportamento de cada um dos algoritmos apresentados aqui.

Tabela 6 – Valores médios de precisão (μ) e desvio padrão (σ) observados para as diferentes estratégias de consulta utilizadas pelo AL. A técnica de amostragem por Menor Confiança foi a que obteve os melhores valores de acurácia para todos os algoritmos.

	Amostragem por Incerteza		**Amostragem por Comitê**	
	Entropia	**Menor Confiança**	**Divergência Média de KL**	**Entropia dos Votos**
	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
RF	$0{,}72 \pm 0{,}08$	$0{,}73 \pm 0{,}07$	$0{,}73 \pm 0{,}08$	$0{,}73 \pm 0{,}08$
GB	$0{,}74 \pm 0{,}07$	$0{,}74 \pm 0{,}07$	$0{,}72 \pm 0{,}08$	$0{,}72 \pm 0{,}08$
LR	$0{,}65 \pm 0{,}08$	$0{,}66 \pm 0{,}08$	$0{,}66 \pm 0{,}08$	$0{,}66 \pm 0{,}07$
SVC	$0{,}70 \pm 0{,}08$	$0{,}72 \pm 0{,}07$	$0{,}72 \pm 0{,}08$	$0{,}72 \pm 0{,}08$
MLP	$0{,}64 \pm 0{,}08$	$0{,}66 \pm 0{,}08$	$0{,}65 \pm 0{,}08$	$0{,}65 \pm 0{,}08$

7.4 Aprendizado Ativo vs. Rotulação por Seleção Aleatória

Nesta seção, discutiremos o desempenho da estratégia de rotulação utilizando AL em comparação à rotulação de vulnerabilidades a partir da seleção aleatória. Além disso, avaliamos

também a influência do aprendizado supervisionado e semissupervisionado na acurácia final do modelo de ML.

7.4.1 Descrição dos Experimentos

Projetamos nossos experimentos para avaliar como uma abordagem de AL pode ser uma solução eficiente e de baixo custo para a rotulação das vulnerabilidades e consequentemente, na classificação de risco, quando comparada à rotulação por seleção aleatória. Também avaliamos como as técnicas de aprendizado supervisionado e semissupervisionado podem impactar nos resultados dos modelos de ML. Assim, executamos quatro cenários de testes: (*i*) aprendizado ativo com aprendizado supervisionado; (*ii*) aprendizado ativo com aprendizado semissupervisionado; (*iii*) rotulação aleatória com aprendizado supervisionado; e finalmente, (*iv*) rotulação aleatória com aprendizado semissupervisionado.

Para cada cenário, avaliamos os seguintes algoritmos de ML: *Random Forest* (RF), *Gradient Boosting* (GB), *Logistic Regression* (LR), *Support Vector Classification* (SVC) e *Multilayer Perceptron* (MLP). Tal escolha de modelos visa incluir estratégias de aprendizagem distintas, desde classificadores lineares até redes neurais artificiais. O *dataset* de treinamento e teste utilizado pelos modelos de ML, foi o conjunto de vulnerabilidades descrito na Subseção 7.2.1. Já para simular a interação do analista com o modelo de AL, utilizamos o Oráculo Simulado, descrito na Subseção 7.2.5. Além disso, dada as conclusões obtidas na Seção 7.3, nessa rodada de testes, utilizaremos apenas a técnica de consulta de amostragem por Menor Confiança do AL. Por fim, vale destacar que cada teste foi repetido 100 vezes, para se calcular as estatísticas de interesse mostradas na discussão a seguir.

7.4.2 Discussão dos Resultados

A Tabela 7 apresenta os valores médios de acurácia e desvio padrão após 100 consultas realizadas pelos modelos de ML para os quatro cenários analisados. O uso de AL apresentou melhores resultados do que a rotulação aleatória para todos os algoritmos, com exceção do MLP, que apresentou o mesmo desempenho para as duas técnicas analisadas.

Já esperávamos que o MLP não tivesse um bom desempenho, mas o incluímos nos testes para fins de completude. MLP é um tipo de ANN composta por pelo menos uma camada não linear oculta. As ANNs são treinadas por meio de uma sequência de iterações de *feedforward* e *backpropagation*, onde os dados são usados para ajustar os pesos da rede. Assim, um modelo

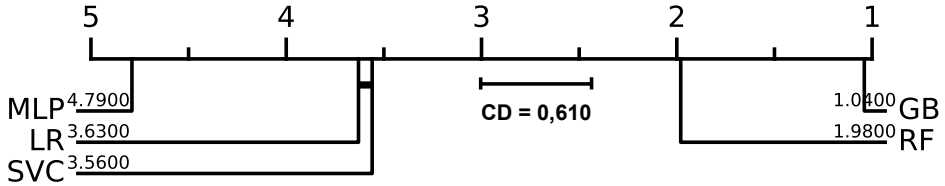
Tabela 7 – Valores médios de precisão (μ) e desvio padrão (σ) obtidos para as diferentes estratégias de rotulação e aprendizados analisados. O algoritmo GB com aprendizado ativo + supervisionado obteve os melhores resultados.

	Active + Semi	**Random + Semi**	**Active + Super**	**Random + Super**
	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$	$\mu \pm \sigma$
RF	0,72 ± 0,08	0,69 ± 0,07	0,73 ± 0,07	0,71 ± 0,08
GB	0,74 ± 0,07	0,70 ± 0,07	0,74 ± 0,08	0,70 ± 0,08
LR	0,66 ± 0,08	0,65 ± 0,07	0,66 ± 0,08	0,65 ± 0,07
SVC	0,71 ± 0,07	0,68 ± 0,08	0,72 ± 0,08	0,69 ± 0,08
MLP	0,65 ± 0,08	0,65 ± 0,07	0,66 ± 0,08	0,66 ± 0,08

de ANN requer muito mais dados para treinar adequadamente seus parâmetros. O cerne do nosso problema envolve a ausência de grandes quantidades de dados rotulados e o alto custo relacionado à atividade de aquisição manual de novas instâncias rotuladas. Assim, seria de se esperar que uma ANN não fosse o modelo mais adequado, pois sofreria de *overfitting*, que ocorre quando o modelo não consegue generalizar muito além dos dados de treinamento.

Para avaliar estatisticamente as diferenças entre os classificadores, foi realizado o teste de classificação de Friedman com nível de significância $\alpha = 0,05$, com a acurácia escolhida como métrica de desempenho. A Figura 36 mostra o diagrama das Distâncias Críticas – *Critical Distance* (CD) – para o teste *post-hoc* Nemenyi do AL combinado com a estratégia de aprendizado supervisionado. Como podemos observar, o GB supera todos os outros algoritmos, sendo o classificador mais preciso, próximo ao ranque 1, enquanto o MLP foi o menos preciso, próximo ao ranque 5, corroborando os resultados apresentados na Tabela 7. Por fim, o valor da distância crítica foi $0,610$, portanto, concluímos que não houve diferenças significativas entre os algoritmos SVC e LR, conforme indicado pela linha horizontal que os conecta na Figura 36.

Figura 36 – Diagrama de Distâncias Críticas para o teste de Nemenyi, realizado sobre os valores médios de acurácia para o cenário que combina o aprendizado ativo e supervisionado. Constatamos que o melhor algoritmo, aquele que aparece próximo ao ranque 1, foi o GB.

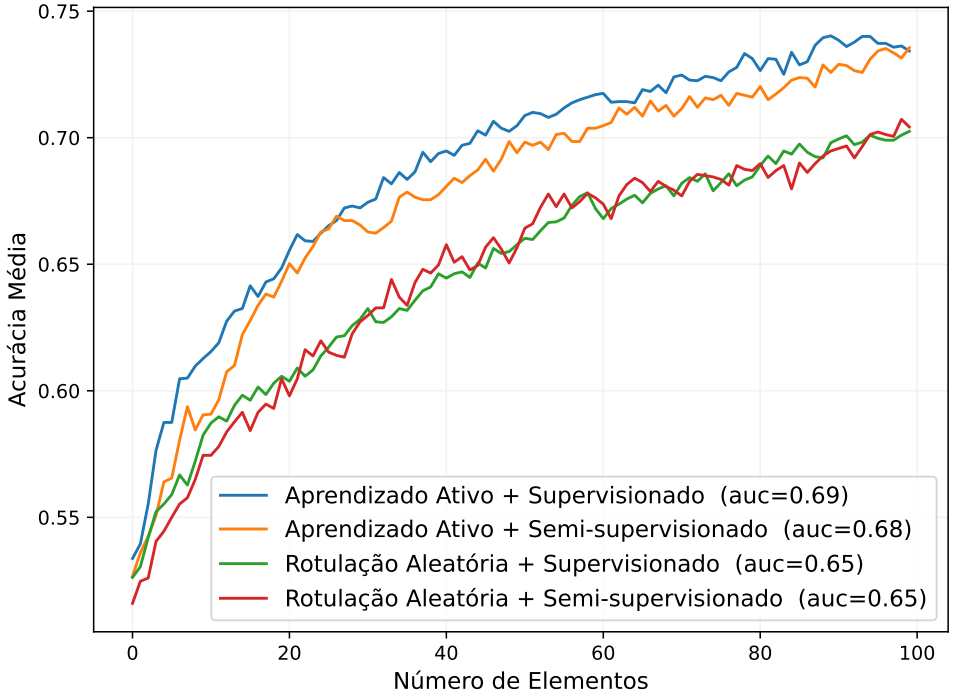


Fonte: elaborado pelo autor.

Portanto, por uma questão de brevidade, focaremos as análises adicionais no GB, pois ele apresentou o maior valor médio de acurácia (Tabela 7) e superou todos os outros

algoritmos no quesito desempenho (Figura 36). A Figura 37 mostra a evolução dos valores de acurácia ao longo do processo de treinamento do algoritmo GB para os quatro diferentes cenários comparados. Como pode ser observado, a técnica que combina aprendizado ativo com aprendizado supervisionado obteve os melhores resultados.

Figura 37 – A figura mostra a evolução dos valores médios de acurácia ao longo de 100 consultas realizadas pelo modelo de AL para os 4 cenários testados. Como podemos observar, o aprendizado ativo + supervisionado usando GB cresce mais rápido e tem um melhor desempenho do que as demais técnicas.



Fonte: elaborado pelo autor.

Ainda sobre a Figura 37, podemos observar pelas curvas, que todas as metodologias começam com valores de acurácia semelhantes. No entanto, aquelas que usam aprendizado ativo crescem mais rápido do que aquelas que selecionam aleatoriamente instâncias para serem rotuladas. Outra forma de interpretar esses resultados é comparando os valores de AUC. As metodologias que utilizam aprendizado ativo apresentaram valores de AUC maiores do que aquelas que não o utilizam, como pode ser observado na legenda da Figura 37. Esses resultados confirmam a importância de se escolher cuidadosamente qual instância deve ser rotulada, i.e., do uso do AL, para melhora da acurácia.

É importante destacar que o *pool* de instâncias utilizado no treinamento do modelo de AL é pequeno e as instâncias estão quase uniformemente distribuídas entre os rótulos (conforme mencionado na Subseção 7.2.2). Assim, a probabilidade de escolher aleatoriamente uma instância

com características relevantes para ser rotulada era bastante alta, o que favoreceu a técnica de rotulação aleatória. Mesmo assim, a Figura 37 mostra que o aprendizado ativo supera rapidamente a metodologia de seleção aleatória. Em um cenário real, com um *pool* de milhares de vulnerabilidades, a diferença entre as curvas deve ser mais acentuada.

A Tabela 8 apresenta os valores médios e o desvio padrão de precisão, revocação e *f1-score* para o algoritmo GB. Podemos observar que os valores de precisão e revocação foram altos (acima de 70%), o que mostra que temos poucos Falsos Positivos e Falsos Negativos. Isso implica que nossa solução não subestima ou superestima o risco das vulnerabilidades. O que é uma característica importante, pois quer dizer que, por exemplo, uma vulnerabilidade CRITICAL, não será classificada como LOW. Já o valor alto de *f1-score* indica que o modelo de ML é capaz tanto de acertar suas predições (alto valor de precisão) quanto de recuperar os exemplos da classe de interesse (alto valor de revocação).

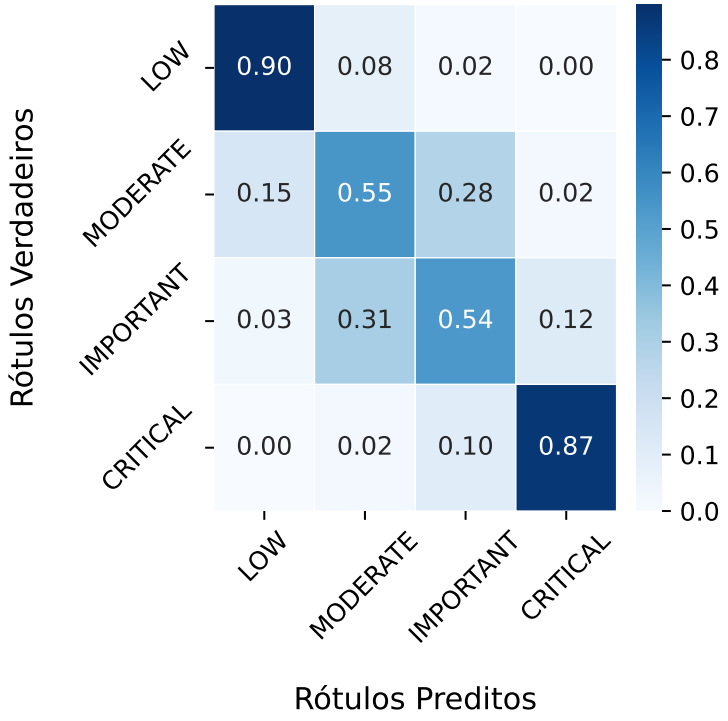
Tabela 8 – Média (μ) e Desvio Padrão (σ) de Precisão, revocação e *F1-Score* para o Algoritmo GB. Os valores altos de Precisão e revocação demonstram que a solução proposta não subestima ou superestima o risco das vulnerabilidades na hora da classificação.

	Active + Semi	**Active + Super**
	$\mu \pm \sigma$	$\mu \pm \sigma$
Precisão	$0{,}75 \pm 0{,}08$	$0{,}75 \pm 0{,}08$
Revocação	$0{,}74 \pm 0{,}07$	$0{,}73 \pm 0{,}08$
F1-score	$0{,}73 \pm 0{,}08$	$0{,}73 \pm 0{,}08$

A Figura 38 apresenta a matriz de confusão média ao se rotular 100 instâncias, no cenário que combina o aprendizado ativo, aprendizado supervisionado e o algoritmo GB. Podemos observar que apesar da acurácia média ser de 74%, o modelo classifica com sucesso, aproximadamente, 87% de todas as vulnerabilidades com rótulo CRITICAL, ou seja, aquelas cujo risco e impacto de exploração é maior. Além disso, quase todas as vulnerabilidades CRITICAL que foram incorretamente classificadas, receberam o rótulo IMPORTANT, que na escala de risco, apresentada na Subseção 5.2.2.1, é o segundo grupo mais crítico.

De modo a entender melhor o comportamento do classificador e o porquê dele ter classificado erroneamente algumas instâncias, analisamos as vulnerabilidades cujos rótulos foram incorretamente atribuídos pelo modelo de ML. Nas Tabelas 9 e 10, apresentamos duas vulnerabilidades cujos rótulos aplicados foram CRITICAL e IMPORTANT, quando, na verdade, seus rótulos verdadeiros eram respectivamente IMPORTANT e CRITICAL. Como podemos perceber, seus atributos são iguais, com exceção dos valores de “Mitre Top 25” e “Tipo de

Figura 38 – Matriz de confusão média ao se rotular 100 instâncias utilizando o aprendizado ativo, em conjunto com o aprendizado supervisionado e o algoritmo GB. Podemos notar, que apesar da metodologia ter alcançado uma acurácia média de 74%, ela foi capaz de classificar corretamente 87% das vulnerabilidades críticas.



Fonte: elaborado pelo autor.

Ataque", destacados em cinza nas tabelas.

As vulnerabilidades analisadas possuem características muito próximas, no entanto, seus rótulos são diferentes. Acreditamos que essa confusão, por parte do modelo, ocorreu, pois a quantidade de instâncias utilizadas durante a fase de treinamento foi pequena (como descrito na Subseção 7.2.1) e porque houve discordância entre os especialistas encarregados da atividade de rotulação das vulnerabilidades (como descrito na Subseção 7.2.2), acerca de qual rótulo aplicar. Isto é, para cada uma das vulnerabilidades, houve um especialista que discordou do rótulo aplicado pelos outros dois companheiros. Assim, fica claro que o erro cometido pelo classificador é justificado, visto que até os próprios especialistas não chegaram a um consenso em relação acerca de risco de exploração da vulnerabilidade.

Para aprofundarmos essa discussão, analisamos como cada analista classificou individualmente as vulnerabilidades presentes no *dataset* criado após a votação majoritária. Como podemos verificar na Tabela 11, os analistas possuem opiniões diferentes em relação à que rótulo aplicar e em alguns casos, a taxa de erro foi bastante alta, chegando a mais de 50%. Isso demonstra como o processo de classificação de risco é difícil, não trivial e bastante influenciado pela experiência pessoal dos analistas. Visto que, até mesmo profissionais com anos de experiência

Tabela 9 – A tabela mostra alguns dos atributos de uma vulnerabilidade identificada pelo CVE-2019-20902. Segundo os especialistas, esta falha de segurança possui um risco IMPORTANT. No entanto, o classificador a rotulou, de forma errônea, como CRITICAL.

CVE-ID			**Rótulo Verdadeiro**		**Rótulo Atribuído**	
CVE-2019-20902			IMPORTANT		CRITICAL	
CVSS	**Fabricante**	**EPSS**	**Nº *Exploits***	**Mitre Top 25**	**OWASP Top 10**	**Tipo de Ataque**
9.8	Outro	0.00885	0	Não	Sim	–
Topologia	**Tipo do Ativo**		**Ambiente**	**Dados Sensíveis**	**Fim da Vida**	**Ativo Crítico**
Local	Estação de Trabalho		Produção	Não	Não	Sim

Tabela 10 – A tabela mostra alguns dos atributos de uma vulnerabilidade identificada pelo CVE-2021-20717. Segundo os especialistas, esta falha de segurança possui um risco CRITICAL. No entanto, o classificador a rotulou, de forma errônea, como IMPORTANT.

CVE-ID			**Rótulo Verdadeiro**		**Rótulo Atribuído**	
CVE-2021-20717			CRITICAL		IMPORTANT	
CVSS	**Fabricante**	**EPSS**	**Nº *Exploits***	**Mitre Top 25**	**OWASP Top 10**	**Tipo de Ataque**
9.8	Outro	0.00885	0	Sim	Sim	*Security Feature Bypass*
Topologia	**Tipo do Ativo**		**Ambiente**	**Dados Sensíveis**	**Fim da Vida**	**Ativo Crítico**
Local	Estação de Trabalho		Produção	Não	Não	Sim

na área de cibersegurança, divergem sobre o assunto.

Por fim, para avaliar o desempenho do modelo de ML em classificar o risco das vulnerabilidades em relação à atividade realizada manualmente pelos especialistas, analisamos a matriz de confusão apresentada na Figura 39. Essa matriz foi obtida a partir da média das matrizes de confusão dos analistas (apresentadas na Tabela 11), que por sua vez foi construída comparando a decisão individual dos especialistas em relação à decisão majoritária deles. Podemos notar que a utilização de um grupo de analistas no processo de rotulação melhorou o desempenho da classificação de risco, reduzindo a quantidade de instâncias rotuladas erroneamente. Vemos,

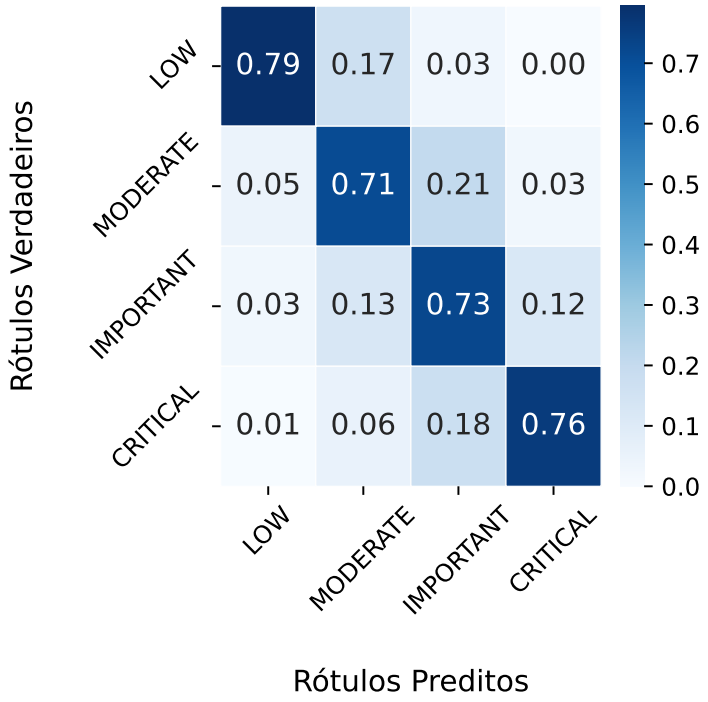
Tabela 11 – A tabela mostra como cada um dos analistas rotulou as vulnerabilidades, em comparação ao *dataset* formado a partir da votação majoritária. Podemos perceber que os analistas divergiram bastante na hora de classificar o risco das vulnerabilidades.

		LOW	**MODERATE**	**IMPORTANT**	**CRITICAL**
Analista #1	**LOW**	54	6		
	MODERATE	1	44		
	IMPORTANT	3	6	34	2
	CRITICAL	1	8	24	25
Analista #2	**LOW**	55	2	3	
	MODERATE	4	16	21	4
	IMPORTANT	1	2	29	13
	CRITICAL				58
Analista #3	**LOW**	34	23	3	
	MODERATE	2	36	7	
	IMPORTANT		9	35	1
	CRITICAL		2	7	49

também, que os analistas obtiveram uma acurácia média de 75%, enquanto a acurácia média obtida pelo AL em conjunto com o aprendizado supervisionado e o algoritmo GB foi de 74%. Vale destacar que a acurácia que o aprendizado ativo e supervisionado obteve para as classes LOW e CRITICAL, 90% e 87% respectivamente, foram maiores que as obtidas pelos especialistas, que foram 79% e 76% respectivamente. No entanto, a acurácia para as classes MODERATE e IMPORTANT piorou. Atribuímos a isso, a quantidade baixa de vulnerabilidades pertencentes a essas classes no *dataset* de treinamento (como apresentado na Subseção 7.2.2).

Portanto, concluímos que o uso da técnica de aprendizado ativo, em conjunto com o aprendizado supervisionado e o algoritmo GB, é a metodologia mais indicada para o processo de classificação de risco. Visto que, ela consegue emular eficientemente a experiência dos analistas e por ser uma solução de ML, traz consigo inúmeros benefícios, como, por exemplo, a velocidade e precisão na classificação de novas instâncias. Assim, será a metodologia utilizada a partir de agora nos demais experimentos realizados nesta tese.

Figura 39 – Matriz de confusão média dos analistas em comparação ao *dataset* formado a partir da votação majoritária. O uso de um grupo de especialistas no processo de rotulação diminuiu a quantidade de instâncias rotuladas erroneamente.



Fonte: elaborado pelo autor.

7.5 Avaliação da Influência da Calibração das Probabilidades no Desempenho do Classificador

Nesta seção, discutiremos a influência da calibração das probabilidades no desempenho final do modelo de ML. Em especial, analisamos o desempenho da regressão sigmoide e isotônica, técnicas discutidas em detalhes na Seção 3.3.

7.5.1 Descrição dos Experimentos

Como dito na Subseção 5.2.3, utilizamos a probabilidade de certeza em relação ao rótulo aplicado pelo classificador como forma de priorizar as vulnerabilidades para correção. Fazemos isso, pois, essa probabilidade é um valor de confiança em relação à classificação feita. De tal forma que, corrigindo as vulnerabilidades com riscos e valores de confiança maiores primeiro, garantimos que a organização estará mais segura. Assim, é importante verificar se a calibração das probabilidades oferece algum galho no desempenho do classificador.

Assim, treinamos três classificadores diferentes utilizando a estratégia *one-vs-all*, sendo: um classificador não calibrado, um calibrado usando a técnica sigmoide e um usando

a técnica isotônica. Dada as conclusões obtidas nas Seções 7.3 e 7.4, nessa rodada de testes, nos limitaremos a testar o aprendizado ativo, em conjunto com o aprendizado supervisionado e o GB, utilizando a técnica de consulta de amostragem por Menor Confiança. O *dataset* de treinamento e teste utilizado pelos modelos de ML, foi o conjunto de vulnerabilidades descrito na Subseção 7.2.1. Já para simular a interação do analista com o modelo de AL, utilizamos o Oráculo Simulado, descrito na Subseção 7.2.5. Por fim, cada teste foi repetido 100 vezes, para se calcular as estatísticas de interesse mostradas na discussão a seguir.

7.5.2 Discussão dos Resultados

A primeira estatística analisada foi a acurácia do modelo após o aprendizado ativo ter realizado 100 consultas ao especialista. O modelo não calibrado apresentou uma acurácia média de 0,73 e um desvio padrão de $\pm$ 0,05. Já os modelos calibrados utilizando a técnica sigmoide e isotônica apresentaram uma acurácia média de 0,72 $\pm$ 0,06 e 0,75 $\pm$ 0,05 respectivamente. Assim, podemos perceber um ganho de 0,02 pontos na acurácia ao calibrarmos o modelo utilizando uma função isotônica.

Em seguida, medimos o Erro de Calibração Esperado – *Expected Calibration Error* (ECE), que calcula a diferença entre os valores de confiança e acurácia dados pelo modelo (NIXON *et al.*, 2019). Portanto, essa métrica pode ser utilizado para quantificar o quão bem calibrado está um modelo. Para se calcular o ECE, primeiro dividimos o intervalo de probabilidade [0, 1] em M compartimentos igualmente espaçados (chamados de *bins*), em seguida calculamos a média ponderada da diferença entre as acurácias dos compartimentos, como mostrado em

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{n} \Big| acc(B_m) - conf(B_m) \Big|, \tag{7.5}$$

onde n representa o número de amostras e acc e $conf$ representam a acurácia e a confiança respectivamente. Valores mais altos de ECE indicam um maior erro na calibração, consequentemente, valores menores de ECE indicam menor erro de calibração. O modelo não calibrado apresentou um valor de ECE de 0,17 $\pm$ 0,05. Já os modelos calibrados utilizando a função sigmoide e isotônica apresentaram valores de ECE de 0,22 $\pm$ 0,06 e 0,13 $\pm$ 0,04. Assim, podemos perceber que a técnica isotônica apresentou o melhor resultado, com o menor valor de ECE. O que corrobora os valores de acurácia média obtidos anteriormente, uma vez que o modelo com a melhor calibração (isotônica) apresentou a melhor acurácia (0,75). Além de melhorar a acurácia, uma melhor calibração é fundamental na etapa de priorização das vulnerabilidades (descrita na

Subseção 5.2.3), uma vez que essa ordenação guiará os analistas no processo de correção das falhas.

Calculamos também o valor da Pontuação de Perda de Brier – *Brier Score Loss* –, que mede a diferença média quadrada entre a confiança prevista pelo classificador e o resultado real da amostra (BRIER *et al.*, 1950). A fórmula da Pontuação de Perda de Brier é dada por

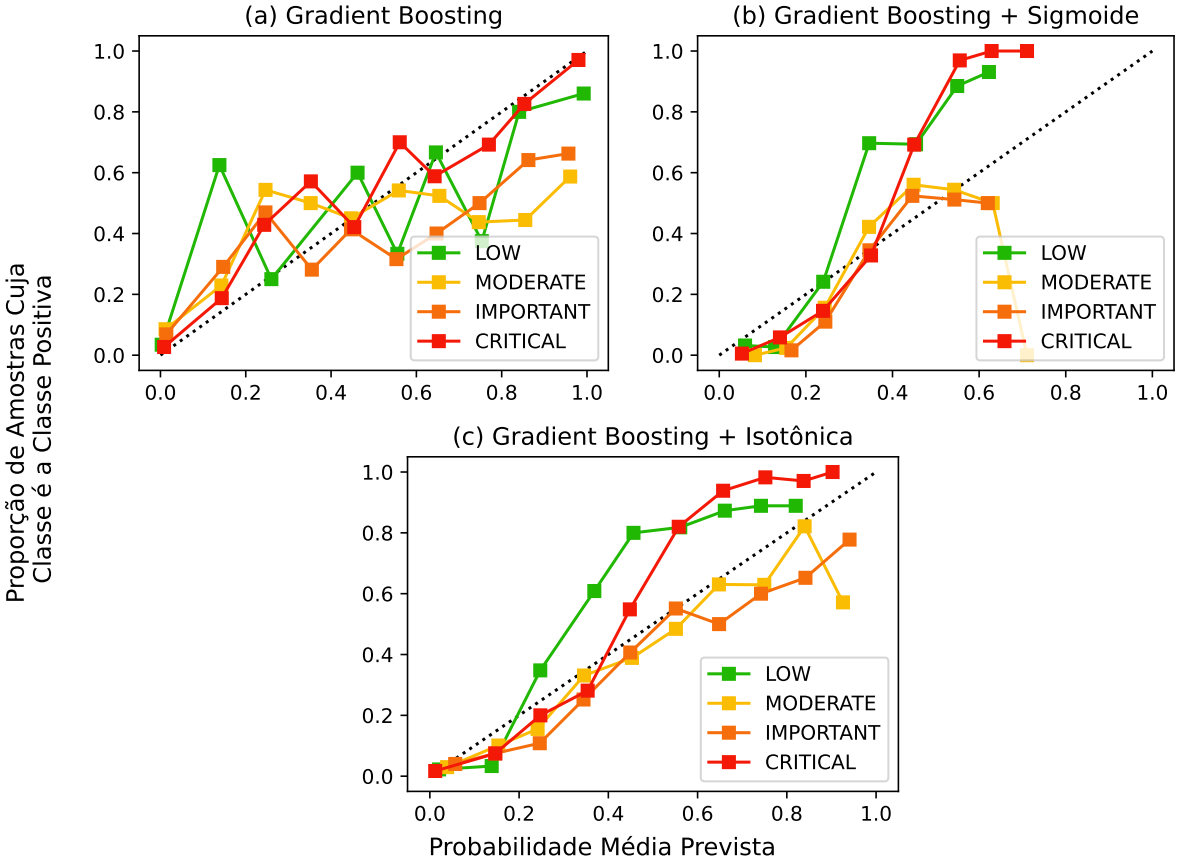
$$BS = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{f}_i)^2, \tag{7.6}$$

onde $\hat{f}_i$ é a confiança prevista, y_i o resultado real da instância i e N é o número de amostras. A Pontuação de Perda de Brier assume valores entre [0, 1] e assim como o valor de ECE, quanto menor for, mais calibrado estará o modelo. O modelo não calibrado apresentou um valor de Brier de $0{,}42 \pm 0{,}09$. Já os modelos calibrados utilizando a função sigmoide e isotônica apresentaram valores de Brier de $0{,}46 \pm 0{,}03$ e $0{,}40 \pm 0{,}06$. Assim, mais uma vez, a calibração utilizando a técnica de regressão isotônica apresentou os melhores resultados.

Na Figura 40, temos as curvas de calibração para os três cenários testados. As curvas de calibração (também conhecidas como diagramas de confiabilidade) comparam o quão calibradas estão as previsões probabilísticas de um classificador. Nesse tipo de gráfico, plotamos a frequência real do rótulo positivo em relação à sua confiança prevista, agrupada em compartimentos (*bins*). O eixo x representa a confiança média prevista em cada *bin*. O eixo y é a fração de positivos, ou seja, a proporção de amostras cuja classe é a classe positiva. A linha tracejada na diagonal do gráfico representa um classificador perfeitamente calibrado. Assim, como podemos perceber, a utilização da técnica de regressão isotônica é a que melhor calibra o modelo, visto que as curvas para cada uma das classes se aproximam mais da diagonal tracejada no gráfico.

Por fim, na Figura 41 temos as curvas das Característica de Operação do Receptor – *Receiver Operating Characteristic* (ROC) –. Os gráficos de curvas ROC normalmente representam a Taxa de Verdadeiro Positivo – *True Positive Rate* (TPR) – no eixo y e a Taxa de Falso Positivo – *False Positive Rate* (FPR) – no eixo x. Isso significa que o canto superior esquerdo do gráfico é o ponto "ideal", visto que o valor de FPR será zero e de TPR um. Alcançar esses valores não é nada realista, mas podemos concluir que quanto maior o valor de AUC melhor. Como podemos observar na Figura 41, as curvas ROC para os três cenários obtiveram valores similares de AUC, que nesse caso podem ser interpretados como uma medida de acurácia. Dessa forma, podemos concluir que, apesar da técnica de regressão isotônica gerar um modelo mais calibrado, o desempenho dos três classificadores se equipara.

Figura 40 – As curvas de calibração relacionam a frequência real dos rótulos positivos e os valores de confiança previstos. Na figura, podemos perceber que a regressão isotônica é a técnica que melhor calibra o classificador, visto que as curvas são as que mais se aproximam da diagonal tracejada no gráfico.



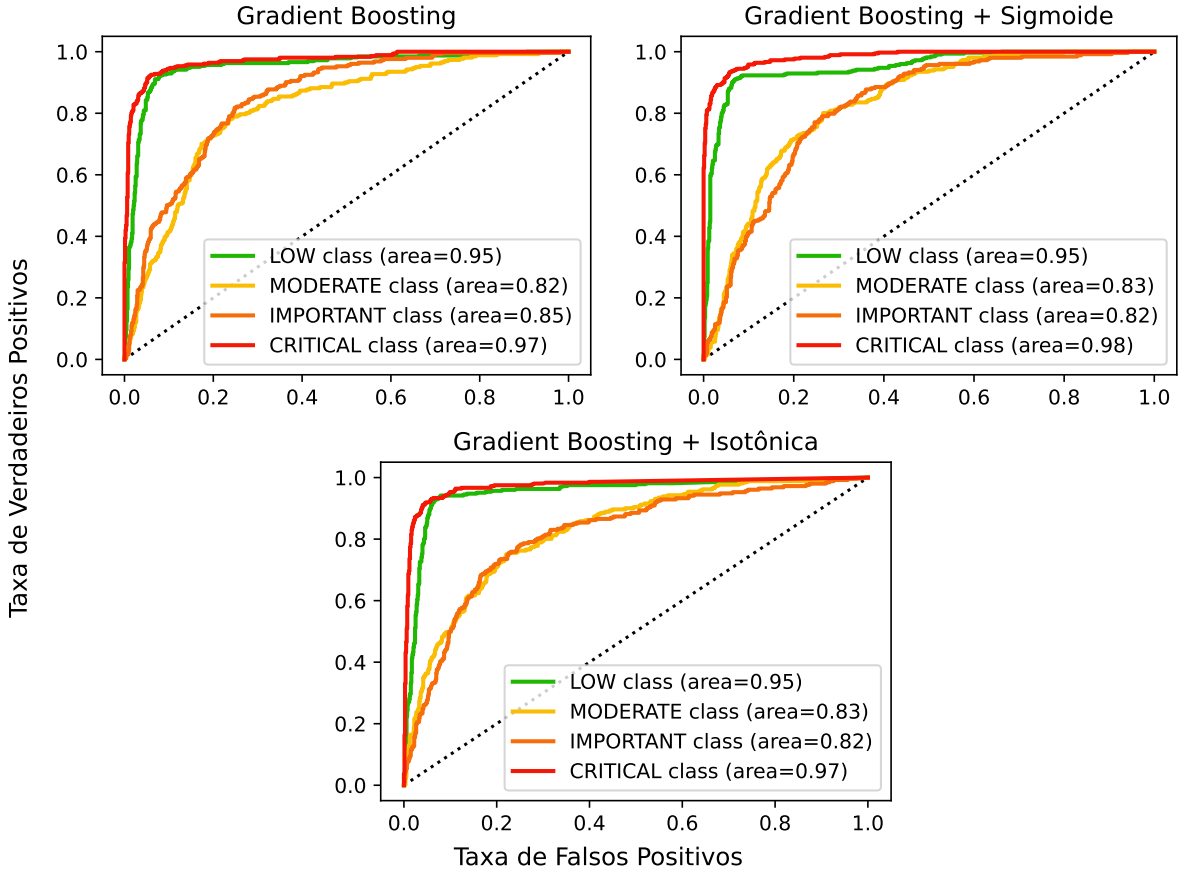
Fonte: elaborado pelo autor.

Assim, como o uso da técnica calibração isotônica melhorou a acurácia média do classificador, apresentou os menores valores de ECE e na Pontuação de Perda de Brier, bem como conseguiu aproximar melhor os valores de confiança previstos, com os reais (como visto na Figura 40), concluímos que sua utilização traz ganhos e deve ser utilizada tanto no treinamento do modelo de AL, quando do modelo final de ML.

7.6 Experimento de Rotulação Online

Por fim, para avaliar o desempenho do aprendizado ativo na prática, realizamos um teste de rotulação *online* no qual o analista interagiu diretamente com o modelo de AL, dispensando a necessidade de utilização do Oráculo Simulado (Subseção 7.2.5). Nesse último experimento, configuramos e calibramos o modelo de AL utilizando os resultados obtidos nas seções 7.4 e 7.5. Além disso, diferentemente do experimento realizado na Seção 7.4 que

Figura 41 – Os gráficos de curva ROC relacionam as taxas de verdadeiro positivo (eixo y) e falso positivo (eixo x). Idealmente, queremos alcançar um valor alto de TPR e baixo de FPR. Na figura, podemos notar que as três técnicas obtiveram um comportamento semelhante e por isso, podemos dizer que seu desempenho se equipara.



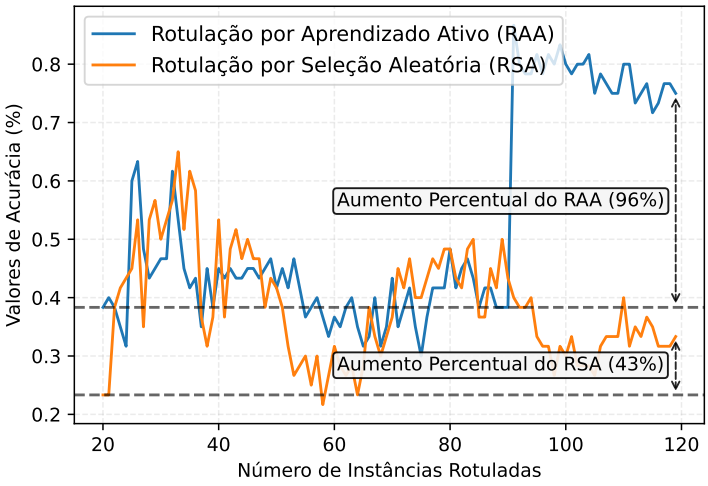
Fonte: elaborado pelo autor.

utilizou um *pool* com aproximadamente 200 instâncias, nesse experimento, utilizamos um *pool* com cerca de 200.000 vulnerabilidades. Nesse cenário, a diferença no desempenho do AL em comparação a rotulação por seleção aleatória é bem maior. Visto que, com um universo de milhares de vulnerabilidades, é muito mais difícil selecionar ao acaso, falhas de segurança com características significativas. Os Apêndices A, B e C descrevem o guia de rotulação e execução, bem como o programa utilizado pelo especialista no teste.

A Figura 42 mostra a evolução dos valores de acurácia para o AL e a rotulação por seleção aleatória. Podemos perceber, que com o aumento do *pool* de vulnerabilidades, o desempenho da rotulação por seleção aleatória cai, alcançando um valor máximo de acurácia de 33% (um aumento percentual de apenas 43% em relação à acurácia inicial). Por outro lado, o valor máximo de acurácia alcançado pelo AL foi de 75% (um aumento percentual de 96% em relação ao valor inicial). O crescimento acentuado observado no AL se deu devido à rotulação de uma vulnerabilidade com características significativas. Esses atributos estavam presentes em

muitas instâncias do conjunto de teste. Assim, a rotulação dessa vulnerabilidade, selecionada pelo AL, auxiliou o classificador a identificar o limite entre as classes com maior precisão, melhorando assim seu desempenho.

Figura 42 – A figura mostra o desempenho das estratégias de rotulação por AL e por seleção aleatória em um cenário contendo milhares de vulnerabilidades. Como podemos notar, o desempenho do AL foi melhor do que a metodologia alternativa.



Fonte: elaborado pelo autor.

Esse resultado é muito importante, pois demonstra na prática que o uso do AL melhora o processo de classificação do risco das vulnerabilidades de segurança. Visto que seu emprego garante que o classificador obtenha uma boa acurácia com poucas instâncias rotuladas.

7.7 Conclusão

Neste capítulo, realizamos uma extensa análise e discussão das diversas técnicas de aprendizado de máquina que podem ser utilizadas pelo *framework* FRAPE na classificação dos riscos das vulnerabilidades. Nosso objetivo foi descobrir a configuração que produzisse um classificador que melhor emulasse a experiência dos analistas de segurança no processo de gestão do risco das vulnerabilidades, i.e., o modelo de ML com a melhor acurácia e o mais confiável (melhor calibrado).

A primeira análise que fizemos foi das técnicas de consulta que podem ser utilizadas pelo modelo de AL para seleção de instâncias. Ao todo, testamos quatro técnicas diferentes, sendo, duas que utilizam valores de incerteza e duas que utilizam comitês de modelos. Nossos experimentos demonstraram que apesar das técnicas de consulta possuírem um desempenho semelhante, a amostragem por Menor Confiança foi a que conferiu ao modelo de AL o maior

valor médio de acurácia para todos os algoritmos de ML testados. Essa é uma técnica que utiliza a incerteza em relação aos possíveis rótulos dos itens presentes no *pool* de vulnerabilidades, para selecionar a instância, com as características mais significativas, que deve ser rotulada pelos especialistas.

A segunda análise que fizemos foi o uso do aprendizado ativo em comparação com à seleção aleatória das vulnerabilidades para rotulação, bem como o emprego das técnicas de aprendizado supervisionado e semissupervisionado. Nossos experimentos demonstraram que mesmo em um cenário desfavorável, o AL apresentou um melhor resultado, que a seleção de instâncias de forma aleatória para rotulação. Além disso, com o emprego do AL em conjunto com o aprendizado supervisionado e o algoritmo GB, conseguimos treinar um classificador que possui uma acurácia média de 74%, que é comparável a dos especialistas de segurança que foi de 75%. Ademais, ao analisarmos essas acurácias separadamente para cada um dos rótulos, percebemos que o desempenho do FRAPE se sobressai a dos especialistas, ao identificar as vulnerabilidades críticas com uma acurácia de 87% em comparação a 76% pelos especialistas.

A terceira análise que fizemos foi para avaliar a influência da calibração de probabilidades no desempenho dos modelos treinados. Nossos experimentos envolveram três modelos, que são: um não calibrado, um calibrado utilizando a regressão sigmoide e um calibrado utilizando a regressão isotônica. A regressão isotônica apresentou os menores valores para o ECE e o *Brier Score Loss*. Além disso, como demostrou o gráfico das curvas de calibração, foi a técnica cujas curvas mais se aproximaram da de um classificador perfeitamente calibrado. Finalmente, a técnica de regressão isotônica conferiu, um aumento de 2% na acurácia média do modelo.

Portanto, concluímos que a técnica de consulta a ser utilizada pelo algoritmo de AL é a amostragem por Menor Confiança. O uso do aprendizado ativo e supervisionado consegue criar um classificador capaz de emular com eficiência a experiência dos profissionais de segurança. Por fim, o emprego de uma técnica de calibração, em especial da regressão da isotônica, melhora a confiança do modelo e deve ser utilizando tanto no processo de aprendizado ativo, quanto do aprendizado supervisionado.

8 SIMULADOR DE CLASSIFICAÇÃO, PRIORIZAÇÃO E CORREÇÃO DE VULNERABILIDADES DE SEGURANÇA BASEADO NO SEU RISCO DE EXPLORAÇÃO

Para avaliar o desempenho da proposta apresentada no Capítulo 5 em relação ao uso exclusivo do CVSS (*baseline*) no processo de VM, desenvolvemos um simulador capaz de gerar redes de computadores fictícias com vulnerabilidades de segurança que podem ser classificadas e corrigidas segundo o seu risco de exploração. Assim, neste capítulo descreveremos: as funcionalidades do simulador (Seção 8.1); os materiais e métodos utilizados no seu desenvolvimento (Seção 8.2); seu funcionamento (Seção 8.3); e por fim, mostraremos como ele pode ser usado na avaliação de estratégias de gestão de vulnerabilidades (Seção 8.4).

8.1 Funcionalidades

O principal objetivo do simulador é ser uma ferramenta que oferece ao usuário a capacidade de comparar diferentes estratégias de priorização e correção de vulnerabilidades de segurança. Assim, elencamos estas como as principais funcionalidades do simulador:

- **Geração de uma rede vulnerável:** o simulador gera uma rede de computadores fictícia, de acordo com parâmetros de contexto previamente estabelecidos pelos usuários, e com vulnerabilidades de segurança selecionadas aleatoriamente da base do NIST;
- **Classificação das vulnerabilidades:** o risco de exploração das vulnerabilidades de segurança é obtido utilizando modelos de aprendizado de máquina que podem ser treinados, conforme os critérios estabelecidos pelo usuário;
- **Priorização das vulnerabilidades:** listas ordenadas de vulnerabilidades, as quais seguem a classificação de risco e critérios definidos pelos usuários, são criadas e utilizadas no processo de correção e/ou mitigação das falhas de segurança;
- **Correção das vulnerabilidades:** as vulnerabilidades são corrigidas, seguindo as listas de priorização, até que um determinado número de interações (i.e., classificação, priorização e correção) tenha sido alcançado. Ao final, gráficos que mostram a superfície de ataque da rede ao longo do processo são apresentados ao usuário, de tal forma que possa-se identificar a melhor estratégia de gestão de vulnerabilidades.

8.2 Materiais e Métodos

Os materiais utilizados no desenvolvimento do simulador são:

- ***Visual Studio Code***[1]**:** também conhecido como VSCode, é um editor de código-fonte desenvolvido pela empresa Microsoft, para Windows, Linux e macOS. O editor é bastante utilizado por desenvolvedores para criação de programas de *software* e entre os seus principais recursos estão: realce de sintaxe, criação de *snippets* de código, ferramentas de controle de versão e suporte para depuração e refatoração de código;
- **Git**[2]**:** é um sistema de controle de versão distribuído gratuito e de código aberto, capaz de rastrear alterações em qualquer tipo de arquivos. Foi projetado para permitir que desenvolvedores trabalhem colaborativamente, enquanto garante a integridade dos dados e oferece suporte a fluxos de trabalho não lineares (ou seja, com múltiplas ramificações paralelas executadas em diferentes sistemas);
- **Python**[3]**:** é uma linguagem de programação de alto nível e propósito geral, desenvolvida com uma filosofia que dá enfase a legibilidade de código. Python é dinamicamente tipada e possui um mecanismo de gerenciamento automático de memória chamado de coletor de lixo – *garbage collector* –. A linguagem oferece suporte a vários paradigmas de programação, incluindo programação estruturada, orientada a objetos e funcional;
- **Bibliotecas:** diversas bibliotecas Python foram utilizadas no desenvolvimento do simulador, entre as principais, podemos citar: Numpy[4] e Pandas[5], bibliotecas que permitem a manipulação e análise de dados numéricos; Scikit-learn[6], biblioteca de aprendizado de máquina que inclui diversos algoritmos de regressão, classificação e clusterização; SHAP[7] biblioteca que utiliza uma abordagem da teoria dos jogos para explicar a saída de modelos de aprendizado de máquina; e finalmente, Matplotlib[8], biblioteca utilizada para criação de gráficos estáticos e visualizações de dados em geral.

Para o desenvolvimento do simulador, adotamos uma metodologia ágil chamada de Programação Extrema – *Extreme Programming* (XP) –. XP é uma metodologia criada para melhorar a qualidade de *softwares*, cujos requisitos são vagos e mudam frequentemente. Seu

[1] Editor de código-fonte VSCode: https://code.visualstudio.com/.
[2] Sistema de controle de versão Git: https://git-scm.com/.
[3] Linguagem de programação de propósito geral Python: https://www.python.org/.
[4] Biblioteca matemática utilizada para trabalhas com *arrays* Numpy: https://pandas.pydata.org/.
[5] Biblioteca de análise de dados Pandas: https://pandas.pydata.org/.
[6] Biblioteca de aprendizado de máquina Scikit-learn: https://scikit-learn.org/.
[7] Biblioteca que explica a saída de modelos de aprendizado de máquina SHAP: https://github.com/slundberg/shap.
[8] Biblioteca para criação de gráficos Matplotlib: https://matplotlib.org/.

principal foco é na codificação e elaboração de testes, dando menor ênfase nos processos formais de desenvolvimento. Assim, os métodos utilizados no desenvolvimento do simulador são:

- **Planejamento:** levantamos os requisitos necessários para o desenvolvimento do simulador. Tomamos como base as especificações apresentadas no Capítulo 5 e análises de *softwares* populares de redes, como, por exemplo, OMNeT++[9] e NS-3[10]. Assim, planejamos as funcionalidades do simulador, de modo que ele pudesse facilitar o teste e a comparação de diferentes estratégias de gestão de vulnerabilidades;
- **Esboço da arquitetura:** elaboramos um esboço da arquitetura, utilizando diagramas da Linguagem de Modelagem Unificada – *Unified Modeling Language* (UML). Desse modo, foi possível visualizar todos os componentes do simulador, mapear suas interações e os valores de entrada e saída dos diferentes módulos;
- **Codificação:** utilizamos o editor VSCode e a linguagem Python para codificar o simulador. A estratégia adotada foi a de desenvolver um Produto Viável Mínimo – *Minimum Viable Product* (MVP) –, i.e., uma versão do produto com recursos suficientes para ser usado rapidamente, e ao decorrer das entregas (*releases*), adicionar todas as funcionalidades planejadas;
- **Testes:** desenvolvemos vários testes unitários ao longo do processo de codificação do simulador. Esses testes garantiram a eliminação de *bugs* e permitiram o desenvolvimento de uma solução robusta.

8.3 Descrição do Funcionamento do Simulador

A Figura 43 representa o diagrama de sequência do simulador, que nada mais é que a sequência de processos (mensagens passadas entre objetos) tomadas pelo *software* desde sua inicialização até sua finalização. A seguir, descreveremos em detalhes cada uma dessas fases.

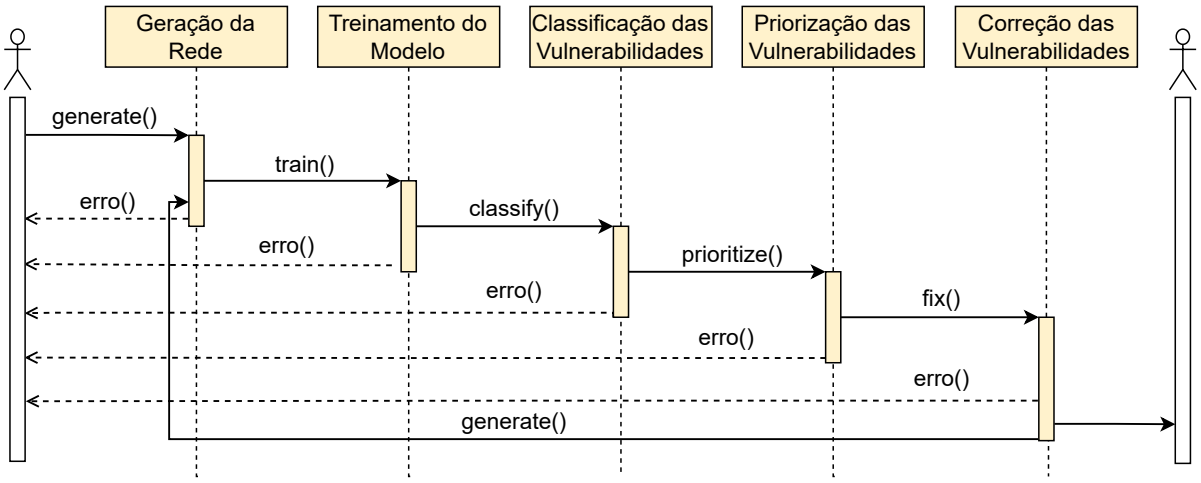
A primeira fase, que corresponde a Geração da Rede, inicia-se a partir da leitura dos parâmetros de configuração fornecidos pelo usuário. O simulador permite configurar vários aspectos da simulação, entre eles: a quantidade de ativos na rede, bem como suas características de contexto; a quantidade de vulnerabilidades por ativo e a distribuição de severidade (CVSS) delas; os parâmetros do modelo de aprendizado de máquina utilizado na classificação de risco das vulnerabilidades; o número de interações do simulador, ou seja, quantos ciclos de análise,

[9] Simulador de redes de computadores, escrito em C++ OMNeT++: https://omnetpp.org/.
[10] Simulador discreto de eventos de redes NS-3: https://www.nsnam.org/.

priorização e correção devem ser executados; e por fim, a quantidade de novas vulnerabilidades que irão surgir em cada iteração e quantas vulnerabilidades devem ser corrigidas ao final de cada ciclo.

Após a leitura dos valores de configuração, a rede é gerada e a segunda fase, que corresponde ao Treinamento do Modelo, se inicia. Nessa fase, um modelo de aprendizado de máquina é treinado conforme os parâmetros presentes no arquivo de configuração. Como um dos intuitos do simulador é analisar o desempenho da proposta, é possível utilizar tanto o aprendizado ativo (descrito na Seção 3.2), quanto as diferentes técnicas de calibração (descritas na Seção 3.3) e validação cruzada (*cross-validation*) no treinamento do modelo. Os algoritmos de aprendizado de máquina suportados são: RF, GB, LR, SVC e MLP. É possível ainda determinar a quantidade de amostras utilizadas no treinamento e teste do modelo de ML.

Figura 43 – O diagrama de sequência mostra o passo-a-passo da execução do simulador. Na figura podemos observar os 5 módulos existentes, responsáveis por: gerar uma rede fictícia contendo vulnerabilidades de segurança, classificá-las e priorizá-las em relação ao risco e finalmente, corrigi-las. Ao fim da execução, é possível identificar a melhor estratégia de VM.



Fonte: elaborado pelo autor.

A terceira fase, que se inicia após o treinamento do modelo de aprendizado de máquina, é a fase de Classificação das Vulnerabilidades. Nessa fase, o simulador utilizará o modelo, treinado na fase anterior, para classificar o risco das vulnerabilidades selecionadas para compor a rede gerada. A classificação pode ser feita considerando uma série de atributos pertencentes a vulnerabilidade, que são: as características da vulnerabilidade, as informações de inteligência de ameaças e de contexto, uma vez que cada vulnerabilidade está atrelada a um ou mais ativo da rede gerada.

Na quarta fase, será realizada a Priorização das Vulnerabilidades. Nessa etapa são

geradas listas de vulnerabilidades ordenadas seguindo os critérios estabelecidos pelo usuário. Essas listas funcionam como um guia de correção a ser seguido pelos analistas de segurança. Por padrão, duas listas serão geradas: a primeira delas é ordenada considerando a nota de CVSS (que corresponde ao nosso *baseline* de comparação com o FRAPE); já a segunda, é ordenada considerando o risco (LOW, MODERATE, IMPORTANT e CRITICAL) atribuído a vulnerabilidade pelo FRAPE e a probabilidade de certeza do modelo em relação ao rótulo aplicado. Assim, primeiro apareceram todas as vulnerabilidades consideradas CRITICAL, depois as IMPORTANT e assim por diante.

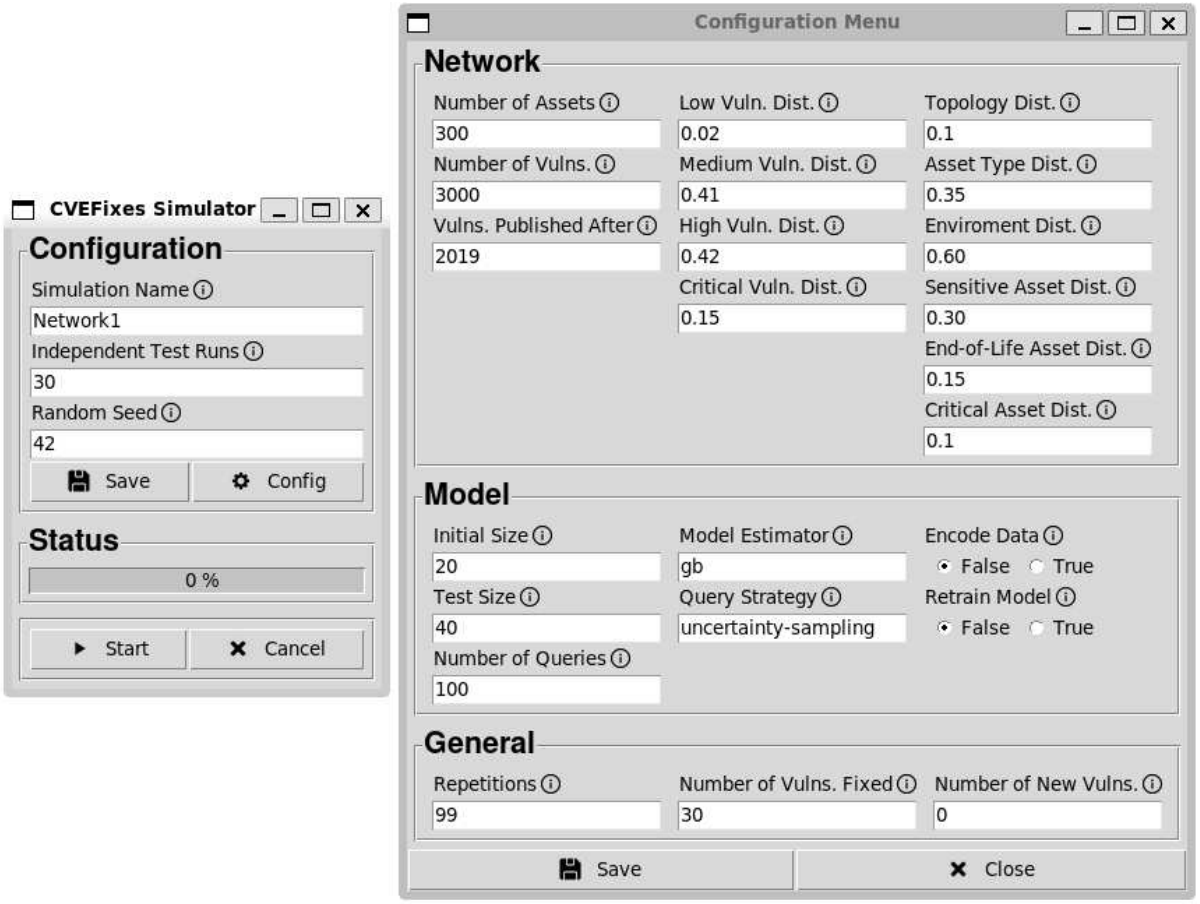
Na quinta e última fase, será realizada a Correção das Vulnerabilidades. Nessa etapa, será corrigido um número fixo de vulnerabilidades (definido nas configurações) a cada iteração. Esse processo tenta capturar a capacidade (i.e., *budget*) que uma equipe de segurança teria para corrigir uma quantidade específica de vulnerabilidades, em um dos ciclos de VM (descrito na Seção 2.4), no mundo real. Vale destacar que o processo de correção equivale a eliminar a ocorrência das vulnerabilidades das listas geradas na fase de Priorização das Vulnerabilidades. Como as listas estão ordenadas seguindo critérios diferentes, as vulnerabilidades corrigidas de cada lista serão diferentes. Após realizar as correções, o simulador terá completado uma iteração e retornara para a fase de Geração da Rede. Nesse ponto, verifica-se a necessidade de gerar novas vulnerabilidades (caráter estático ou dinâmico da simulação). Caso seja necessário, novas vulnerabilidades serão geradas e o processo de treinamento, priorização e correção continuará, até que o número máximo de iterações (definido nas configurações) seja alcançado.

A cada nova iteração, o simulador guarda o estado da rede, isto é, as vulnerabilidades presentes, antes e depois da correção. Além disso, ele também salva um conjunto de métricas tais como a distribuição das vulnerabilidades por severidade do CVSS (*base_severity*) e criticidade do FRAPE (*risk_criticality*). Assim, ao final da execução, é possível acompanhar a influência de cada uma das estratégias no processo de VM. Caso algum erro ocorra em alguma das fases, a execução do simulador é interrompida e uma mensagem de erro detalhando o ocorrido é relatada. Dessa forma, o usuário pode tomar alguma medida para solucionar o erro e reexecutar o simulador.

Por fim, na Figura 44, temos a interface gráfica do simulador, onde é possível perceber o grande número de parâmetros que podem ser ajustados para a execução de simulações. Além disso, caso o usuário prefira, também é possível executar o simulador por meio de linha de comando. Assim, fica claro como o simulador desenvolvido é uma ferramenta poderosa e versátil,

por permitir analisar o desempenho de estratégias de RBVM em vários cenários distintos, com a simples alteração de parâmetros de configuração e reexecução dos testes.

Figura 44 – O simulador pode ser executado tanto a partir de linha de comando, como através de uma interface gráfica desenvolvida especialmente para ele. Na figura, podemos ver as telas de execução e configuração da simulação, com os diversos parâmetros que podem ser ajustados para a realização dos testes.



Fonte: elaborado pelo autor.

8.4 Avaliação de Desempenho

Nesta seção, analisaremos dois cenários de execução do simulador com a mesma configuração inicial de rede, mas variando a natureza do teste. A primeira simulação é de um cenário estático, onde não surgem novas vulnerabilidades na rede. Já a segunda simulação é de um cenário dinâmico, onde novas vulnerabilidades surgem a cada iteração do simulador. Assim, mostramos como o FRAPE se sai em relação à técnica tradicional de gestão de vulnerabilidades que utiliza apenas a nota de CVSS na priorização das falhas de segurança. Para cada um dos cenários, realizamos 30 rodadas independentes (*seeds* diferentes) de teste, a fim de se calcular as

estatísticas de interesse mostradas na discussão a seguir.

8.4.1 Cenário Estático

Neste primeiro cenário, configuramos uma rede de computadores com 300 ativos e 3000 vulnerabilidades, publicadas no NIST entre 2019 e 2022. Cada um dos 300 ativos recebeu um conjunto de características de contexto, as mesmas descritas na Subseção 7.2.1, cuja distribuição pode ser vista na Tabela 12. Consideramos que os valores escolhidos representam uma rede de médio porte. Idealmente, esses valores deveriam ter sido obtidos de um cenário real. No entanto, dado a natureza sensível desses dados, não conseguimos obter essas informações de nenhuma organização. Ademais, as vulnerabilidades foram selecionadas de forma aleatória e seguem a seguinte distribuição considerando a nota do CVSS: 2% são LOW, 41% MEDIUM, 42% HIGH e 15% CRITICAL. Essa distribuição foi a mesma encontrada na análise exploratória realizada no Capítulo 6, onde foi examinado o subconjunto de vulnerabilidades publicadas no NIST entre 2019 e 2022.

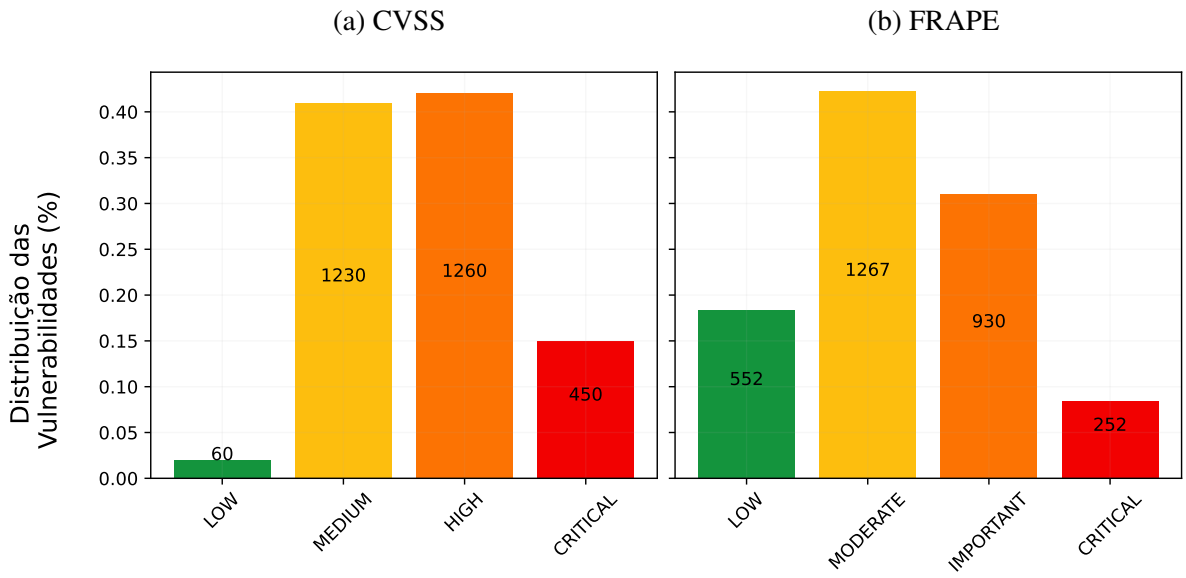
Tabela 12 – Para nossa proposta escolhemos seis características de contexto para categorizar os ativos presentes na rede, que são: topologia, tipo do ativo, ambiente, dados sensíveis, fim da vida e ativos críticos. A tabela a seguir mostra a distribuição dos ativos entre essas características.

Características de Contexto	Opções	Distribuições (%)
Topologia	LOCAL	0,90
	DMZ	0,10
Tipo do Ativo	WORKSTATION	0,65
	SERVER	0,35
Ambiente	DEVELOPMENT	0,40
	PRODUCTION	0,60
Dados Sensíveis	NÃO	0,70
	SIM	0,30
Fim da Vida	NÃO	0,85
	SIM	0,15
Ativos Críticos	NÃO	0,90
	SIM	0,10

Após a fase de Geração da Rede, vem a fase de Treinamento do Modelo de ML. Nessa etapa, para construir o modelo de ML, consideramos os resultados obtidos no Capítulo 7. Dessa forma, para treinar o modelo utilizamos o algoritmo GB em conjunto com a técnica de

consulta do AL, e a amostragem por Menor Confiança. Além disso, calibramos as probabilidades do classificador usando uma função de regressão isotônica. O modelo treinado é utilizado pelo simulador na fase de Classificação das Vulnerabilidades, para classificar o risco das falhas de segurança. A Figura 45 mostra a distribuição das vulnerabilidades segundo o CVSS (*baseline*) e o FRAPE. Como podemos perceber, o FRAPE possui uma distribuição diferente do CVSS. Isso ocorre, pois o FRAPE utiliza informações além do CVSS para classificar o risco das vulnerabilidades. De tal forma que o FRAPE consegue identificar as falhas de segurança que apresentam as maiores chances de exploração e impacto na infraestrutura da organização.

Figura 45 – No gráfico a esquerda, podemos notar a distribuição das vulnerabilidades considerando a severidade de sua nota de CVSS, o chamado *base_severity*. Já no gráfico a direita, temos a distribuição das vulnerabilidades considerando a classificação de risco dada pelo FRAPE, que chamamos de *risk_criticality*.



Fonte: elaborado pelo autor.

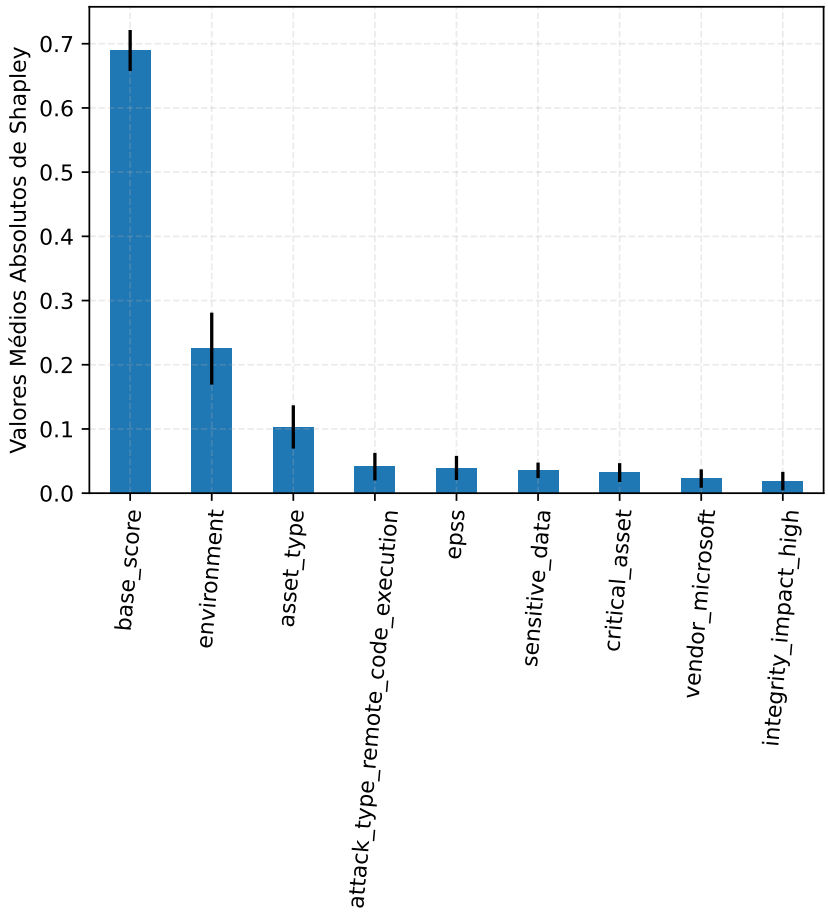
Em seguida, temos as fases de Priorização e Correção das Vulnerabilidades, cujos funcionamentos foram descritos em detalhes na Seção 8.3. Neste momento, ao passarmos por todos os módulos, concluímos uma iteração do simulador. Cada iteração do simulador corresponde a um ciclo no processo de VM (descrito na Seção 2.4), que envolve a análise, priorização e correção das vulnerabilidades. Neste cenário de teste, configuramos o simulador para executar 100 iterações. Em cada iteração do simulador foram corrigidas 30 vulnerabilidades e como esse cenário é estático, não surgem novas vulnerabilidades na rede no decorrer das iterações. Ao final da execução do simulador, podemos analisar como as diferentes estratégias se comportaram.

A Figura 46 mostra os 9 atributos considerados mais importantes pelo modelo, isto é, aqueles que tiveram um maior peso na hora da aplicação do rótulo a vulnerabilidade. Esses valores foram calculados utilizando-se a ferramenta de explicabilidade SHAP (descrita na Subseção 3.4.1). Vale destacar que esse modelo foi treinado utilizando o *dataset* apresentado na Seção 7.2, o qual foi rotulado por três especialistas em cibersegurança. Além disso, como treinamos o modelo utilizando o aprendizado ativo, seu uso funcionará semelhantemente ao processo de RBVM realizado por especialistas experientes. Analisando a Figura 46, percebemos que a nota de CVSS (*base_score*) possui o maior peso entre os atributos considerados. Isso acontece, pois o CVSS é atualmente o padrão mais utilizado pelos especialistas no processo de VM, assim, é de se esperar que os mesmos deem uma maior importância a ele na classificação do risco das vulnerabilidades. No entanto, além do CVSS, também estão presentes o tipo de ataque (*attack_type_remote_code_execution*) e o EPSS (*epss*), que são características de inteligência de ameaças. Assim como, 4 das 6 características de contexto utilizadas, que são: ambiente (*environment*), tipo do ativo (*asset_type*), dados sensíveis (*sensitive_data*) e ativo crítico (*critical_asset*). Isso demonstra como as informações de inteligência e contexto são consideradas importantes para os analistas no processo de RBVM.

Para comparar o desempenho das diferentes estratégias de priorização (CVSS e FRAPE), utilizamos um gráfico de radar, com as arestas sendo os atributos considerados mais importantes pelo modelo (como mostrado na Figura 46). Consideramos que a área desse polígono representa a superfície de ataque da organização, visto que os valores representados nas arestas equivalem ao número absoluto de vulnerabilidades presentes na rede com tais características. Vale destacar que para facilitar a visualização dos dados, esses valores foram normalizados, pelos seus máximos, antes de serem apresentados. Assim, a princípio, o gráfico de radar estará completamente preenchido, mas a medida que as vulnerabilidades forem sendo corrigidas pelo simulador, essa área irá diminuir. Dessa forma, quanto maior a área do polígono, maior a superfície de ataque (isto é, mais vulnerabilidades com essas características para serem exploradas) e mais vulnerável estará a organização. Consequentemente, quanto menor a área, menor a superfície de ataque e menos vulnerável estará a organização.

A Figura 47 mostra à evolução da superfície de ataque ao longo das iterações do simulador. No primeiro gráfico de radar, podemos perceber que a área dos polígonos são iguais e estão sobrepostas, uma vez que esse gráfico representa o momento anterior a inicialização do processo de correção. Os demais gráficos mostram à evolução da rede a cada 10

Figura 46 – Utilizando as técnicas de XAI, conseguimos identificar o impacto de cada um dos atributos pertencentes as vulnerabilidades na classificação de risco dada pelo modelo de ML. Como podemos perceber, o CVSS tem uma forte influência na classificação de risco. Junto dele, vemos duas características de inteligência de ameaças e quatro de contexto.

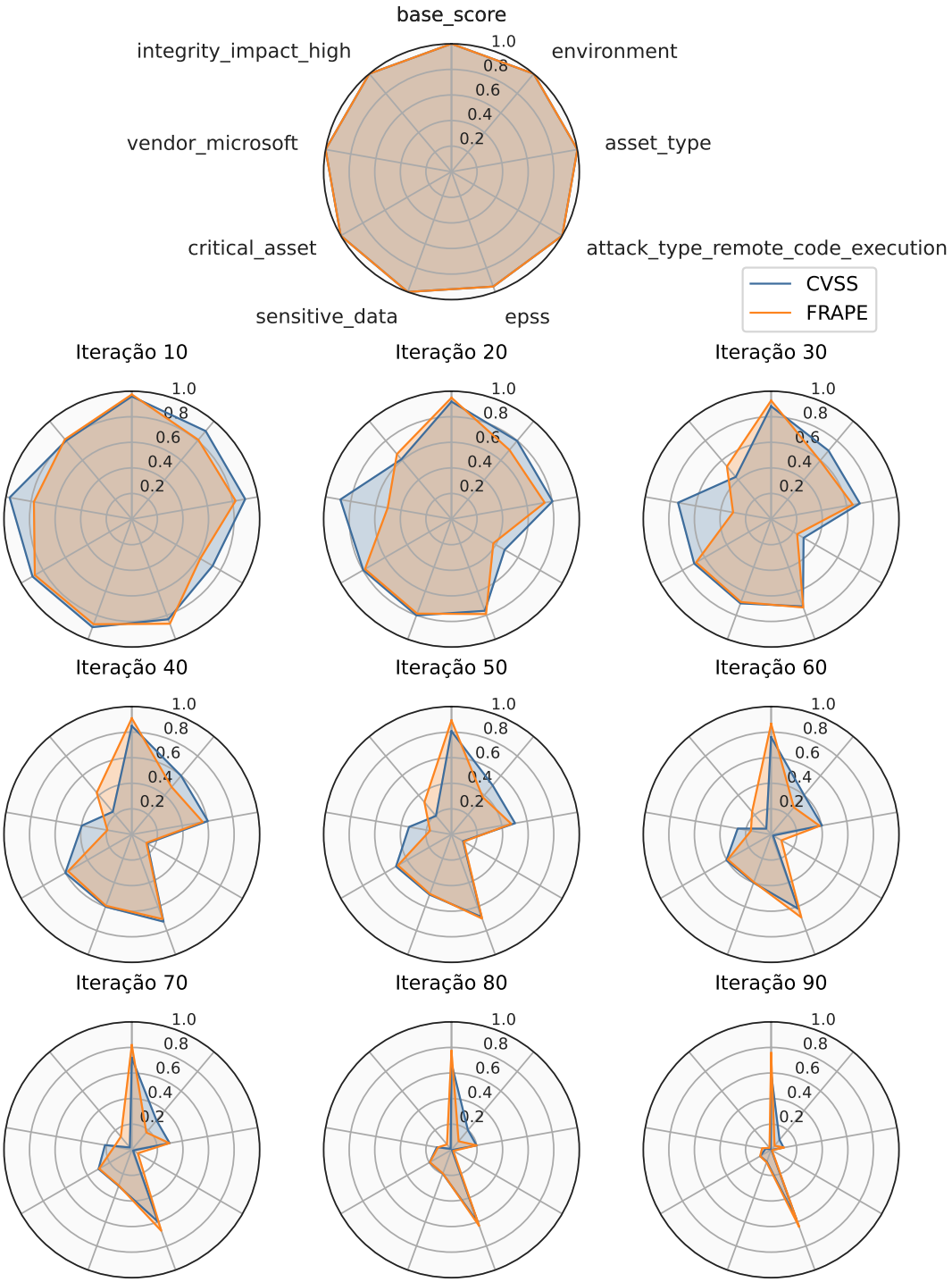


Fonte: elaborado pelo autor.

iterações. Vale destacar que, como nossa rede possui inicialmente 3000 vulnerabilidades e a cada iteração corrigimos 30 vulnerabilidades, podemos interpretar cada iteração dessa simulação como equivalente à correção de 1% das vulnerabilidades da rede. Assim, na 10ª iteração corrigimos 10% das vulnerabilidades e assim sucessivamente. Percebemos pela Figura 47, que a superfície de ataque vai diminuindo e que a área é menor quando se prioriza a correção das vulnerabilidades seguindo a classificação dada pelo FRAPE. Como a metodologia que utiliza o CVSS prioriza as vulnerabilidades considerando apenas a nota de CVSS (*base_score*) e o impacto na integridade (*integrity_impact_high*) está diretamente associada a essa nota, o FRAPE nunca conseguirá ganhar nessas arestas.

A Figura 48 apresenta a área sob a curva, normalizada, do CVSS e do FRAPE. As duas estratégias possuem os mesmos valores de área tanto no início, quanto no fim da simulação. No entanto, podemos perceber pela Figura 48, que o FRAPE ganha do CVSS até por volta da

Figura 47 – Os gráficos de radar representam à evolução da superfície de ataque ao corrigir as vulnerabilidades considerando o CVSS e o FRAPE no cenário estático. Como podemos perceber, ao seguir a estratégia dada pelo FRAPE, teremos uma superfície de ataque menor e consequentemente, a rede estará mais segura.

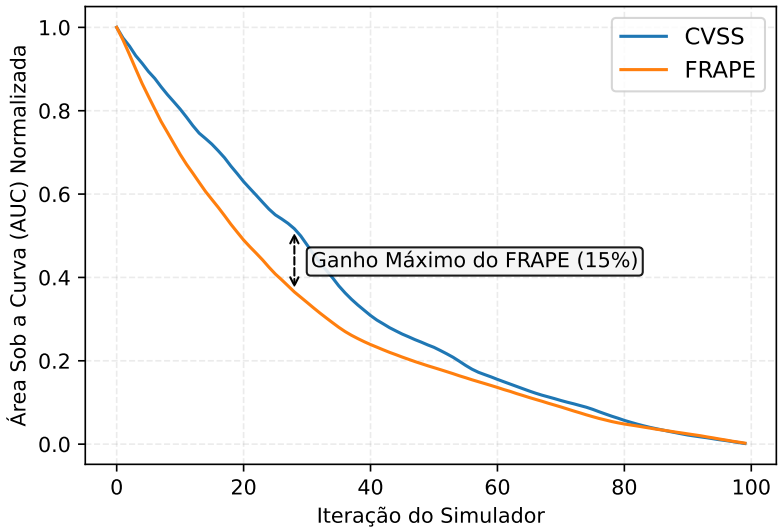


Fonte: elaborado pelo autor.

80ª iteração. A partir desse ponto, a diferença entre as áreas são pequenas e as curvas acabam se sobrepondo. A análise da distribuição das vulnerabilidades pelo risco (mostrado na Figura 45) explica o porquê desse comportamento. Aproximadamente 40% das vulnerabilidades presentes

na rede foram classificadas pelo FRAPE como CRITICAL ou IMPORTANT. Assim, ao corrigir essas vulnerabilidades, o FRAPE estará eliminando as vulnerabilidades mais críticas da rede e consequentemente, diminuindo mais a superfície de ataque, ganhando assim nesse primeiro momento. Após essas vulnerabilidades terem sido corrigidas, a diferença entre as áreas vai diminuindo até que elas acabam se equiparando. Apesar disso, o FRAPE já cumpriu seu papel, que foi de identificar e priorizar a correção das vulnerabilidades mais críticas.

Figura 48 – As curvas apresentadas no gráfico correspondem à evolução da superfície de ataque para o CVSS e o FRAPE na simulação estática. Podemos perceber que o ganho máximo do FRAPE em relação ao CVSS ocorre no intervalo entre [0, 40], que corresponde ao espaço no qual as vulnerabilidades CRITICAL e IMPORTANT estão sendo corrigidas na rede.

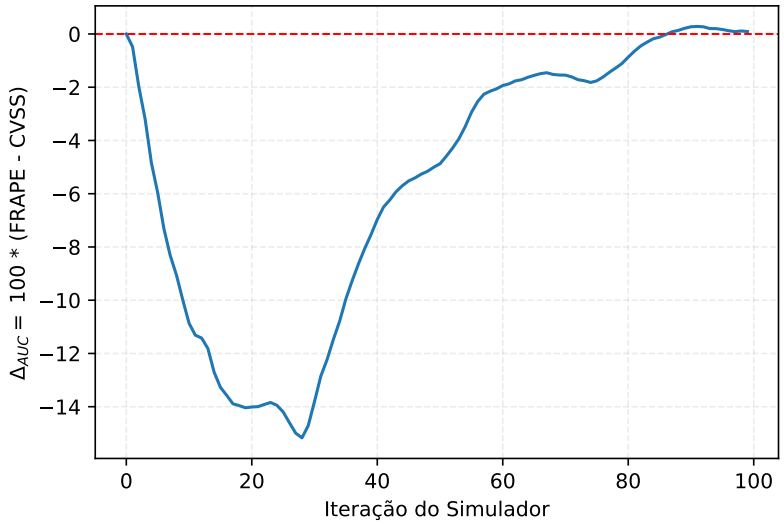


Fonte: elaborado pelo autor.

Alternativamente, na Figura 49, podemos observar a diferença percentual entre as áreas sob as curvas. Valores negativos indicam que o FRAPE ganhou do CVSS e valores positivos que perdeu. Pela análise da figura, podemos observar que até aproximadamente a 10ª iteração, a diferença percentual entre as duas metodologias aumentou negativamente. Esse intervalo corresponde a correção de 300 vulnerabilidades, sendo todas as CRITICAL (252) e uma pequena porção das IMPORTANT (48). A partir desse ponto, quando as vulnerabilidades mais críticas foram eliminadas da rede, a diferença percentual começou a diminuir e a partir da 85ª iteração, o FRAPE eventualmente acabou empatando com o CVSS, pois a diferença entre as duas áreas ficou bem próxima de zero. Ainda na Figura 49, podemos observar que o FRAPE obteve um ganho máximo de 15% em relação ao CVSS. Isso indica que o FRAPE conseguiu deixar a rede 15% menos vulnerável a ataques que possam ser realizados por usuários mal-intencionados. Por fim, vale destacar que a partir da 85ª iteração, o FRAPE já corrigiu todas as vulnerabilidades

CRITICAL, IMPORTANT e MODERATE, de forma que as únicas vulnerabilidades restantes são as LOW. Essas falhas de segurança, como apresentado na Subseção 5.2.2.1, são vulnerabilidades que afetam dispositivos comuns, difíceis de serem exploradas e que não causam grande impacto para os sistemas da empresa, além de não possibilitarem o acesso a informações sensíveis.

Figura 49 – O gráfico indica a diferença percentual entre as áreas sob as curvas do FRAPE e CVSS para o cenário estático. Valores negativos indicam que o FRAPE ganhou naquele intervalo e positivos que perdeu. Como podemos perceber, o FRAPE ganhou por praticamente toda a duração da simulação. No entanto, por volta da 85ª iteração, a diferença entre as áreas do FRAPE e do CVSS são tão próxima de zero que elas acabam empatando. Mas nesse ponto, as vulnerabilidades mais críticas da rede já foram corrigidas pelo FRAPE.



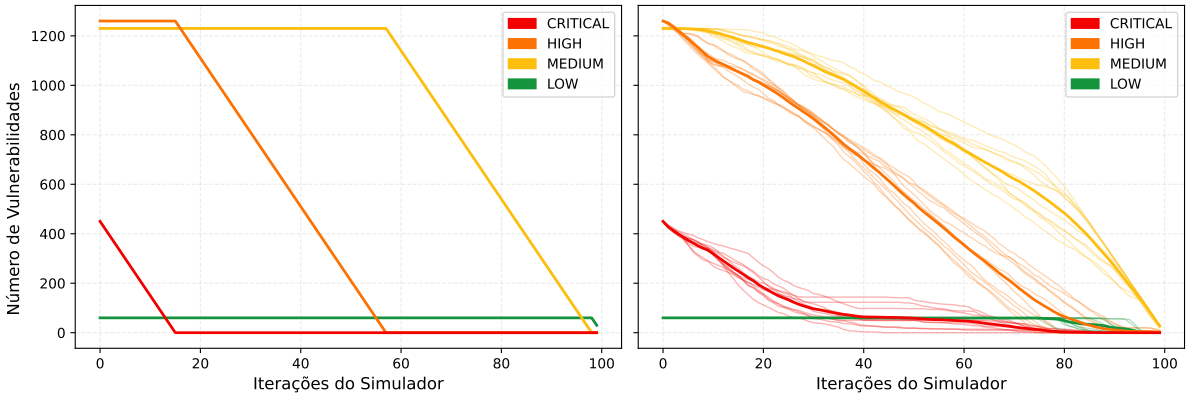
Fonte: elaborado pelo autor.

Para finalizarmos a discussão sobre o cenário estático, analisaremos a distribuição das vulnerabilidades por rótulo ao longo da simulação. Na Figura 50 temos a distribuição das vulnerabilidades considerando a severidade do CVSS (Subseção 2.3.1), que iremos nos referir como *base_severity*, para ambas as metodologias de priorização analisadas. Na Figura 50, plotamos 10 simulações diferentes e as curvas em negrito representam a média desses resultados. A primeira coisa que podemos percebemos é o caráter analítico do FRAPE. Ao seguir a priorização dada pelo CVSS, Figura 50 (a), as curvas são as mesmas independentemente da rodada de teste. Já ao se utilizar o FRAPE, Figura 50 (b), podemos ver que para diferentes rodadas, temos curvas diferentes. Isso faz sentido, pois como as vulnerabilidades são selecionadas e atribuídas de forma aleatória entre os ativos, entre as rodadas de teste, temos redes levemente diferentes. Assim, os riscos das vulnerabilidades também devem ser diferentes.

Ainda sobre a Figura 50, podemos ver que ao seguirmos a priorização do CVSS, Figura 50 (a), as vulnerabilidades são corrigidas em sequência de acordo com seu *base_severity*.

Figura 50 – As curvas mostram a distribuição das vulnerabilidades de acordo com o seu *base_severity* ao longo da simulação estática. Podemos notar o caráter analítico do FRAPE, que utiliza critérios diferentes para correção das vulnerabilidades entre as diferentes rodadas. Algo que o CVSS não consegue fazer, visto que ele corrige as falhas desconsiderando qualquer informação além do *base_score* da vulnerabilidade.

(a) metodologia #1: CVSS (b) metodologia #2: FRAPE



Fonte: elaborado pelo autor.

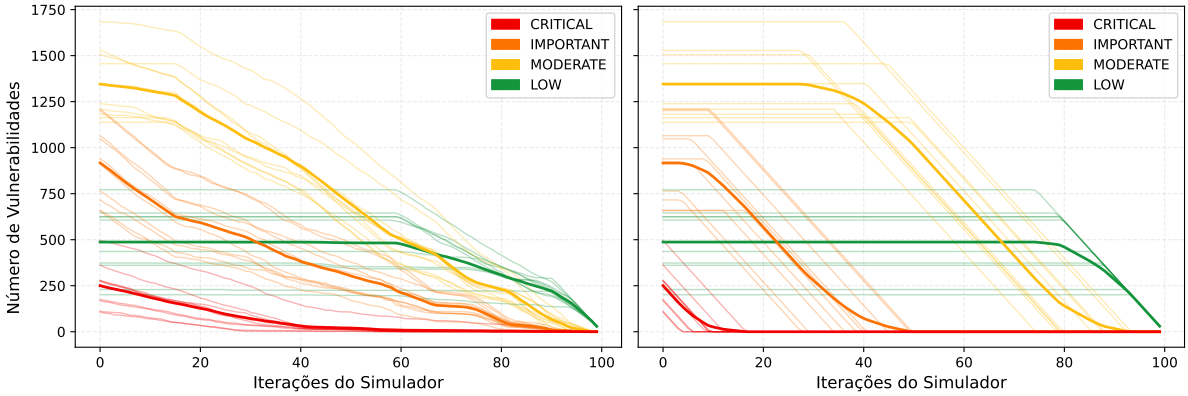
Assim, primeiro corrige-se todas as vulnerabilidades CRITICAL, depois as HIGH e assim por diante. Já na Figura 50 (b), ao priorizarmos a correção segundo o FRAPE, podemos observar que vulnerabilidades com diferentes rótulos de *base_severity* são corrigidas na mesma iteração. Isso evidencia a característica do FRAPE de priorizar as vulnerabilidades considerando informações além do CVSS. Fato importante, pois como demonstrado anteriormente na Seção 5.1, dependendo das circunstâncias, uma vulnerabilidade com *base_severity* HIGH pode, por exemplo, apresentar um risco maior para empresa que uma vulnerabilidade com *base_severity* CRITICAL. Vale destacar que o FRAPE é inteligente o suficiente para não priorizar a correção de vulnerabilidades menos críticas no lugar das mais críticas. Podemos notar na Figura 50 (b), que só após todas as vulnerabilidades com *base_severity* CRITICAL e HIGH terem sido corrigidas, é que o FRAPE começa a dar maior atenção às vulnerabilidades MEDIUM e LOW, como evidenciado pelo declínio acentuado nessas curvas a partir da iteração 80 do simulador.

Por fim, na Figura 51, temos a distribuição das vulnerabilidades considerando o risco aplicado pelo FRAPE (Subseção 5.2.2.1), que iremos nos referir como *risk_criticality*, para ambas as metodologias de priorização analisadas. Podemos notar que ao consideramos o CVSS na correção das falhas, Figura 51 (a), demoramos muito mais para eliminar as vulnerabilidades com *risk_criticality* crítico da rede. Visto que, só após a 40ª iteração, é que eliminamos completamente as vulnerabilidades CRITICAL. Diferentemente do observado no FRAPE, Figura 51 (b), que nesse mesmo ponto, já eliminamos todas as vulnerabilidades CRITICAL e a maioria

das IMPORTANT. A Figura 51 também deixa evidente o porquê do FRAPE perder após a 80ª iteração. Como podemos notar na Figura 51 (a), após a 80ª iteração, existem algumas vulnerabilidades IMPORTANT para serem corrigidas pela metodologia que prioriza utilizando o CVSS. Essas vulnerabilidades IMPORTANT contribuem muito para diminuição da superfície de ataque da rede. No entanto, nesse mesmo intervalo, na Figura 51 (b), o FRAPE está corrigindo só vulnerabilidades com o *risk_criticality* MODERATE e LOW, que possuem um risco menor que as IMPORTANT e assim, contribuem menos para diminuição da superfície de ataque.

Figura 51 – As curvas mostram a distribuição das vulnerabilidades de acordo com o seu *risk_criticality* ao longo da simulação estática. Podemos notar que o FRAPE consegue corrigir muito mais rápido as vulnerabilidades com risco CRITICAL e IMPORTANT. Diferentemente do CVSS, que só consegue eliminar as mesmas vulnerabilidades após a 80ª iteração.

(a) metodologia #1: CVSS (b) metodologia #2: FRAPE



Fonte: elaborado pelo autor.

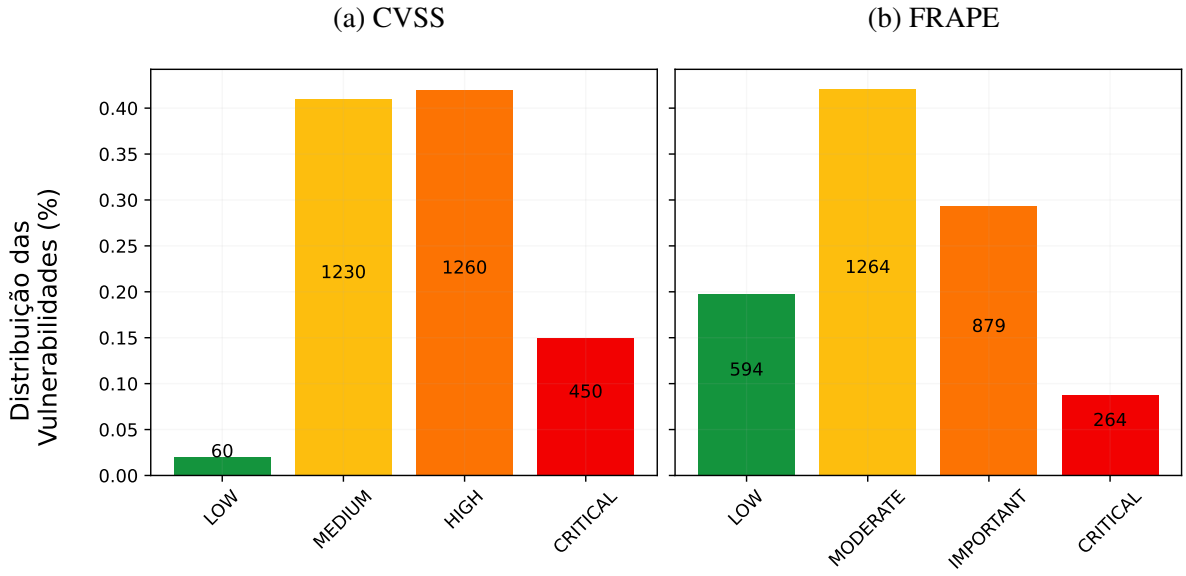
8.4.2 Cenário Dinâmico

O cenário de teste dinâmico foi configurado com os mesmos parâmetros utilizados no cenário estático. A única diferença é que, por ser dinâmico, nesse cenário, novas vulnerabilidades surgem a cada iteração do simulador. No total, 30 novas vulnerabilidades são selecionadas e atribuídas aleatoriamente entre os ativos da rede a cada iteração. A proporção dessas vulnerabilidades por CVSS é a mesma da rede inicial, ou seja: 2% são LOW, 41% MEDIUM, 42% HIGH e 15% CRITICAL. O cenário dinâmico está mais próximo de um cenário real, visto que novas vulnerabilidades são descobertas quase que diariamente e seria impossível eliminar todas as falhas de uma rede de computadores. Além disso, vale destacar que, assim como no cenário estático, o número de iterações do simulador foi configurado como 100.

As Figuras 52 e 53 mostram respectivamente a distribuição das vulnerabilidades

entre as duas metodologias analisadas e os atributos considerados mais importantes pelo modelo de ML. Na Figura 52 (a), vemos que a distribuição das vulnerabilidades por severidade do CVSS são as mesmas do cenário estático, visto que os parâmetros de configuração da rede foram iguais. No entanto, dado o caráter aleatório (distribuição das vulnerabilidades e atribuição de informações de contexto) da simulação, podemos perceber que a classificação de risco feita pelo FRAPE, na Figura 52 (b), foi diferente. Já na Figura 53, notamos que os atributos considerados importantes pelo modelo, foram os mesmos da simulação estática, com exceção do atributo de contexto topologia (*topology*). Esse atributo entrou no lugar do impacto na integridade (*integrity_impact_high*), considerando-se, assim, cinco dos seis atributos de contexto na comparação entre as superfícies de ataque.

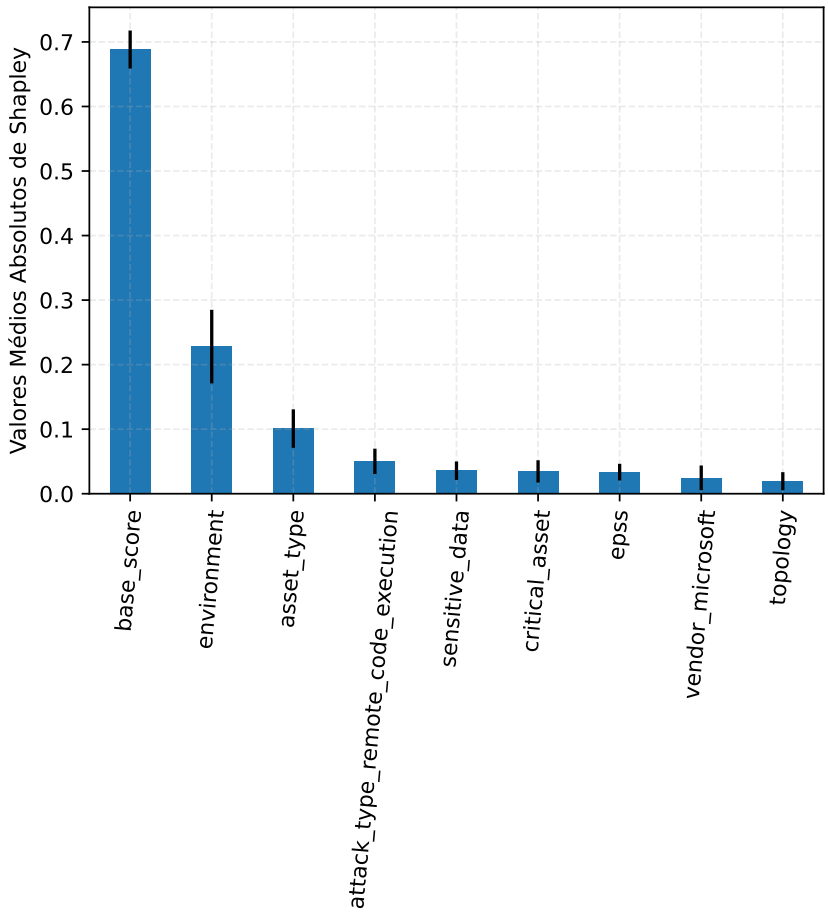
Figura 52 – Nos gráficos a seguir, podemos notar a distribuição das vulnerabilidades no cenário dinâmico considerando: a severidade da nota de CVSS (*base_severity*) e a classificação de risco dada pelo FRAPE (*risk_criticality*). Percebemos que a distribuição do *risk_criticality* foi diferente do cenário estático, visto que as vulnerabilidades e informações de contexto são atribuídas de forma aleatória na criação da rede, o que influencia na classificação de risco.



Fonte: elaborado pelo autor.

A Figura 54 mostra o gráfico de radar para a simulação dinâmica. Assim como no cenário estático, ao priorizarmos a correção das vulnerabilidades utilizando o FRAPE, conseguimos diminuir mais a superfície de ataque da rede. Outro ponto importante de se destacar é que, dado o caráter dinâmico da simulação, as redes não começam com os valores máximos de área. Isso ocorre, pois o ponto de máxima, para algumas arestas, ocorreu no meio da simulação. Esse é o caso das arestas de topologia (*topology*) e fabricante Microsoft (*vendor_microsoft*).

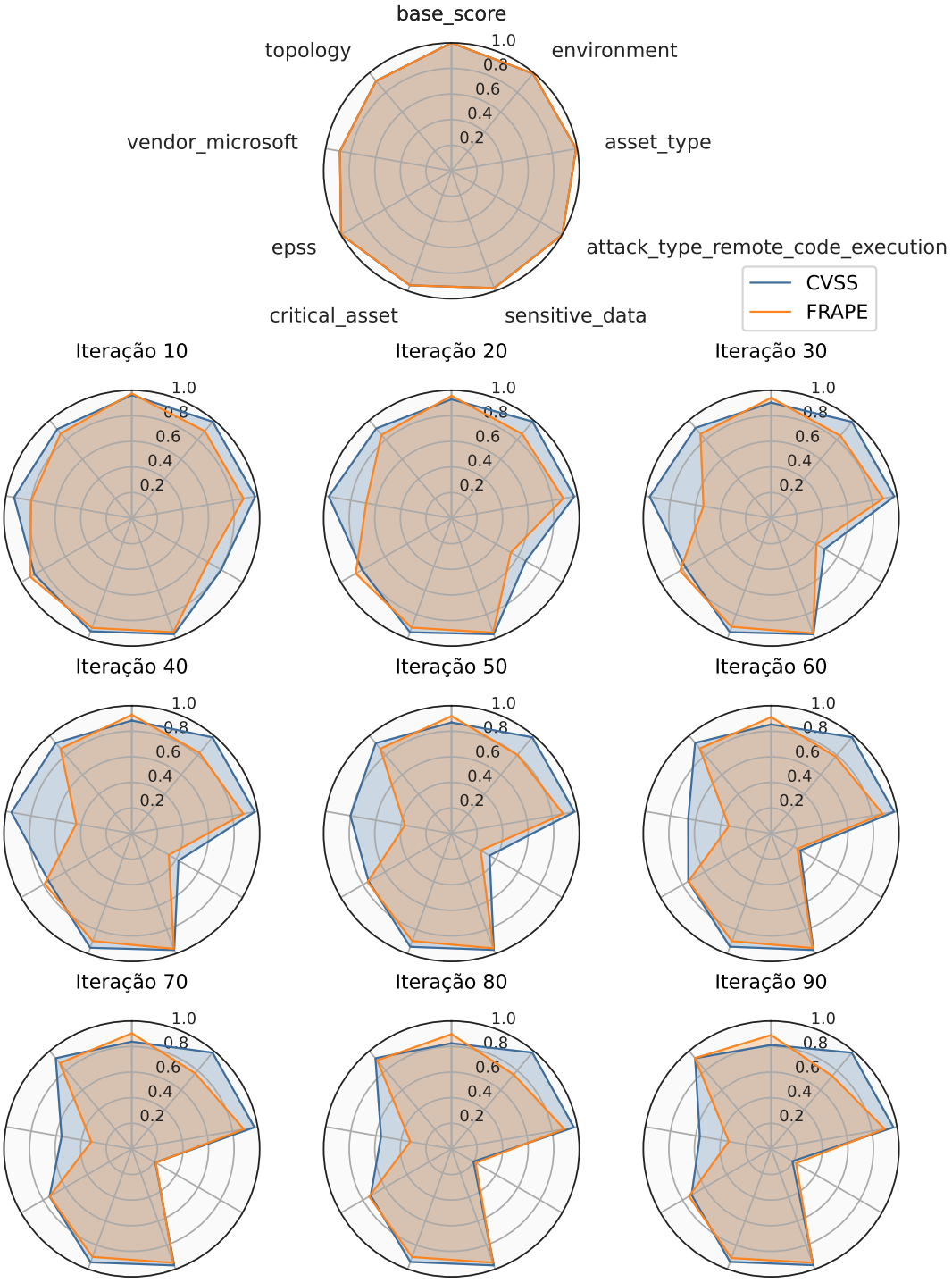
Figura 53 – Os atributos considerados importantes pelo modelo de ML no cenário dinâmico, foram os mesmo do cenário estático, com exceção da característica de contexto topologia (*topology*). Esse atributo substituiu o impacto na integridade (*integrity_impact_high*). Dessa forma, consideramos cinco dos seis atributos de contexto na comparação das superfícies de ataque entre as metodologias no cenário dinâmico.



Fonte: elaborado pelo autor.

Ainda sobre a Figura 54, podemos perceber que o FRAPE consegue de forma mais eficiente lidar com a correção das vulnerabilidades, mesmo com a entrada e saída de novas falhas de segurança a cada iteração do simulador. Esse comportamento fica evidente quando analisamos a aresta fabricante Microsoft (*vendor_microsoft*). No início da simulação, ambas as metodologias começam com os mesmos valores para essa aresta. No entanto, ao longo das iterações, podemos notar que o FRAPE consegue consistentemente diminuir a quantidade de vulnerabilidades com esse atributo na rede. Diferentemente do CVSS, que a princípio negligencia a correção dessas vulnerabilidades (o valor da aresta aumenta), mas posteriormente começa a corrigi-las (seu valor diminui). Isso ocorre porque o FRAPE analisará e classificará o risco de cada nova vulnerabilidade que entra na rede, considerando diversas informações nesse processo. O que não ocorre com o CVSS, que só considera a nota do *base_score*.

Figura 54 – Os gráficos de radar representam à evolução da superfície de ataque ao corrigir as vulnerabilidades considerando o CVSS e o FRAPE no cenário dinâmico.

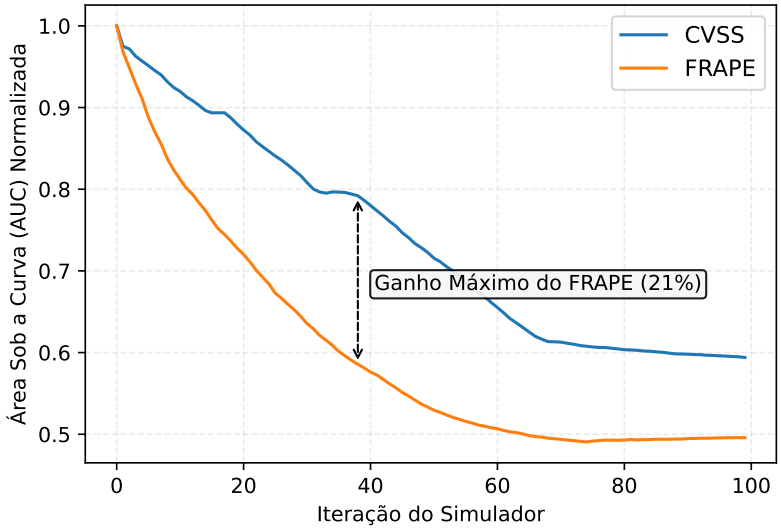


Fonte: elaborado pelo autor.

Na Figura 55, podemos ver à evolução da superfície de ataque para o cenário dinâmico. Nessa figura percebemos que, por um breve número de iterações, as áreas entre as metodologias são similares. No entanto, logo a diferença entre o CVSS e o FRAPE se torna acentuada, atingindo uma estabilidade após a 70ª iteração. No ponto de maior distância,

o FRAPE apresentou um ganho máximo de 21% em relação ao CVSS. Lembrando que isso significa que o FRAPE conseguiu deixar a rede 21% menos vulnerável a ataques. No ponto de equilíbrio entre as metodologias, a diferença foi pouco mais de 10%.

Figura 55 – As curvas apresentadas no gráfico correspondem à evolução da superfície de ataque para o CVSS e o FRAPE na simulação dinâmica.



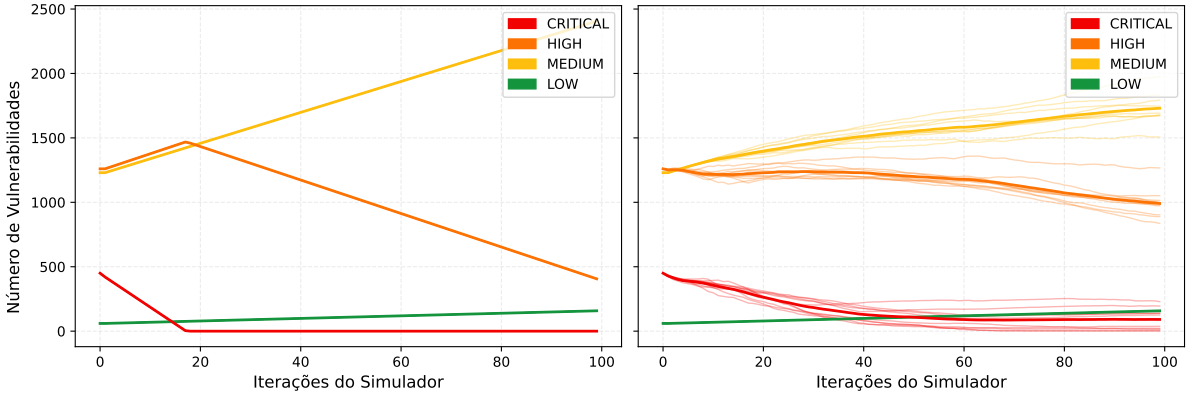
Fonte: elaborado pelo autor.

A Figura 56 mostra a distribuição das vulnerabilidades no cenário dinâmico, considerando o *base_severity* das notas de CVSS, para ambas as metodologias de priorização analisadas. Nessa figura, podemos observar o mesmo comportamento visto no cenário estático (Figura 50), onde o CVSS se limitou a corrigir as vulnerabilidades pelo *base_severity*, na Figura 56 (a), e o FRAPE analisou cada vulnerabilidade considerando seus atributos intrínsecos, na Figura 56 (b). No entanto, diferentemente do cenário estático, a cada nova iteração surgem novas falhas de segurança na rede. De tal forma, a distribuição das vulnerabilidades segundo o *base_severity* não chega a zero, com exceção das CRITICAL.

Já na Figura 57, temos a distribuição das vulnerabilidades no cenário dinâmico, considerando o *risk_criticality*, para ambas as metodologias de priorização analisadas. Assim como no cenário estático, o FRAPE conseguiu eliminar rapidamente as vulnerabilidades CRITICAL da rede, o que ocorreu antes da 20ª iteração. O CVSS teve um pior desempenho, eliminando as vulnerabilidades CRITICAL por volta da 60ª iteração. Além disso, o CVSS não conseguiu eliminar as vulnerabilidades IMPORTANT da rede. Diferentemente do FRAPE, que conseguiu eliminar praticamente todas as vulnerabilidades IMPORTANT, por volta da 60ª iteração. Nesse mesmo ponto, o CVSS ainda estava eliminando as vulnerabilidades CRITICAL.

Figura 56 – As curvas mostram a distribuição das vulnerabilidades de acordo com o seu *base_severity* ao longo da simulação dinâmica.

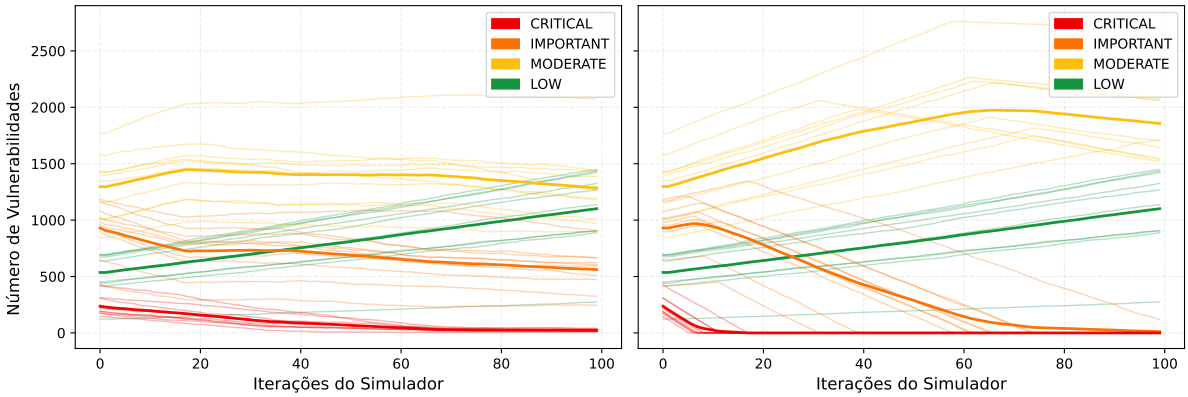
(a) metodologia #1: CVSS (b) metodologia #2: FRAPE



Fonte: elaborado pelo autor.

Figura 57 – As curvas mostram a distribuição das vulnerabilidades de acordo com o seu *risk_criticality* ao longo da simulação dinâmico.

(a) metodologia #1: CVSS (b) metodologia #2: FRAPE



Fonte: elaborado pelo autor.

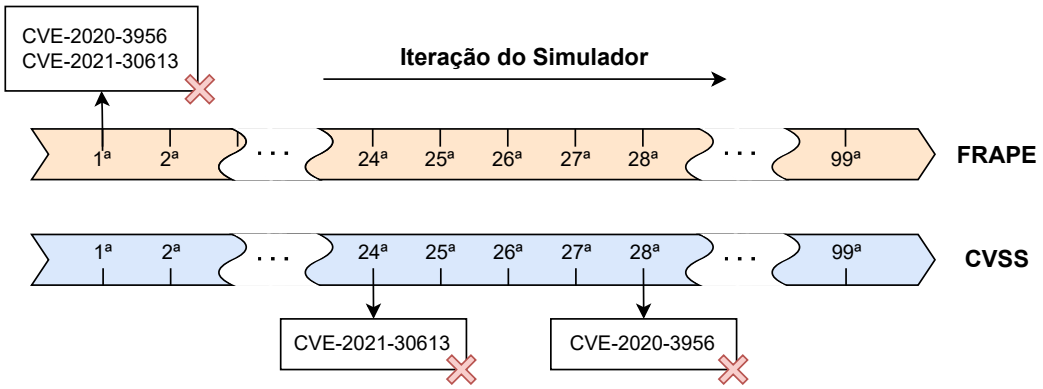
Por fim, realizamos uma análise qualitativa das vulnerabilidades priorizadas e corrigidas por cada uma das metodologias aplicadas pelo simulador, isto é, pelo CVSS e FRAPE. A Figura 58 mostra duas vulnerabilidades com atributos críticos pertencentes ao grupo de informações de inteligência de ameaças (CVE-2020-3956) e de características de contexto (CVE-2021-30613). Como podemos perceber, ambas as falhas de segurança possuem nota de CVSS igual a 8,8 cuja severidade é HIGH e por isso, só serão corrigidas pelo CVSS quando todas as vulnerabilidades CRITICAL tiverem sido eliminadas. No entanto, dado suas características adicionais, o FRAPE considerou que seus riscos e probabilidades de exploração eram altas e, portanto, as classificou como CRITICAL. Assim, as vulnerabilidades identificadas pelo CVE-2020-3956 e CVE-2021-30613 são eliminadas na primeira iteração do simulador pelo FRAPE

e somente após a 20ª iteração pelo CVSS. Se considerarmos que cada iteração do simulador corresponde a um ciclo de VM que ocorre entre 15 e 30 dias (CIS, 2021), essas vulnerabilidades levariam entre 10 e 20 meses para serem eliminadas da rede. Tempo no qual elas poderiam ser exploradas por atores maliciosos para causar sérios danos a organização.

Figura 58 – A figura apresenta duas vulnerabilidades classificadas como CRITICAL pelo FRAPE por possuírem características críticas de inteligência de ameças e contexto, e que portanto, foram corrigidas na primeira iteração do simulador. No entanto, as mesmas duas falhas foram corrigidas pela metodologia que prioriza a correção das vulnerabilidades partir do CVSS, apenas após a 20º iteração do simulador.

CVE ID	Nota de CVSS	Fabricante	EPSS	Nº Exploits	Publicação do Exploit	Tipo de Ataque	OWASP
2020-3956	8.8	Linux	0.52	1	02/06/2020	RCE	True

CVE ID	Nota de CVSS	Fabricante	Topologia	Tipo do Ativo	Ambiente	Dados Sensíveis	Fim da Vida
2021-30613	8.8	Microsoft	DMZ	SERVER	PROD.	True	True



Fonte: elaborado pelo autor.

Além do cenário descrito e analisado acima, modelamos e testamos outros dois cenários dinâmicos. Os parâmetros de configuração utilizados nessas simulações foram os mesmos utilizados anteriormente, com exceção das quantidades de vulnerabilidades novas e corrigidas a cada iteração do simulador. No primeiro cenário, configuramos a simulação para que a quantidade de vulnerabilidades corrigidas fosse maior que a quantidade de vulnerabilidades que surgem na mesma iteração. Já no segundo cenário, configuramos a simulação para que a quantidade de vulnerabilidades corrigidas fosse menor que a quantidade de vulnerabilidades que surgem. A Figura 59 apresenta à evolução da superfície de ataque para esses dois cenários.

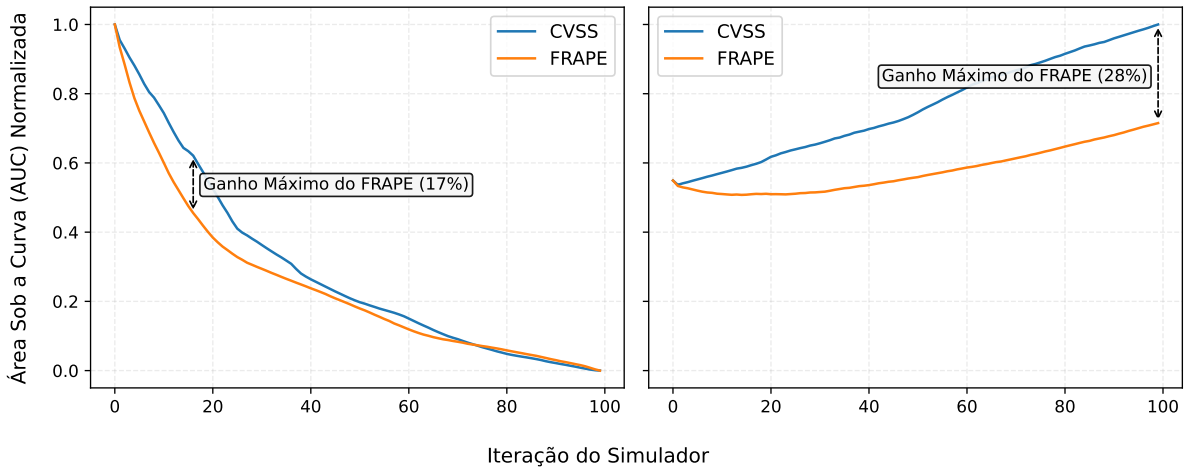
Na Figura 59 (a), onde corrigimos mais vulnerabilidades do que surgem no ambiente, podemos perceber que a superfície de ataque tende a zero. A curva se assemelha a apresentada

no cenário estático (visto na Figura 48), com o *FRAPE* ganhando do CVSS durante a maioria do teste. No ponto de maior distância, o FRAPE conseguiu tornar a rede 17% mais segura que o CVSS. Já na Figura 59 (b), onde a quantidade de vulnerabilidades corrigidas é menor que a quantidade de vulnerabilidades que surgem, podemos notar que a superfície de ataque tende a aumentar. No entanto, a utilização do FRAPE propiciou que esse crescimento fosse menor que a alternativa que seria utilizar o CVSS. O ponto de maior distância entre as duas metodologias foi na última iteração da simulação, onde o FRAPE tornou a rede 28% mais segura que o CVSS.

Figura 59 – As curvas correspondem à evolução da superfície de ataque para o CVSS e o FRAPE em dois cenários dinâmicos onde a quantidade de vulnerabilidades corrigidas é diferente da quantidade de vulnerabilidades que surgem durante uma iteração do simulador.

(a) cenário onde se corrigem 60 vulnerabilidades, mas surgem 30

(b) cenário onde se corrigem 30 vulnerabilidades, mas surgem 60



Fonte: elaborado pelo autor.

8.5 Conclusão

Neste capítulo, apresentamos o simulador de classificação, priorização e correção de vulnerabilidades. Esse programa foi desenvolvido para comparar diferentes estratégias de gestão de vulnerabilidades e fornecer aos especialistas insumos que os ajudem a selecionar aquela capaz de melhorar a segurança da organização. O programa consegue gerar uma rede fictícia com vulnerabilidades reais publicadas no NIST e simular a correção dessas falhas de segurança utilizando duas estratégias de correções, que são: (*i*) priorizar a correção das vulnerabilidades pela nota de CVSS; e (*ii*) priorizar a correção pela classificação de risco dada pelo FRAPE.

Experimentos mostraram que com a utilização do FRAPE foi possível diminuir a

superfície de ataque da rede de forma mais consistente, quando comparado com a estratégia alternativa que foi a utilização unicamente do CVSS. Para o cenário estático, essa diferença foi mais acentuada no intervalo contendo vulnerabilidades críticas (aproximadamente 15%), visto que elas são as que apresentam maiores riscos para a rede. Já para o cenário dinâmico, o FRAPE ganhou por praticamente toda a simulação, alcançando um valor máximo de ganho de aproximadamente 21% em relação ao CVSS. Isso foi um importante resultado, pois o FRAPE, ao contrário do CVSS, utiliza, além das características das vulnerabilidades, informações de inteligência de ameaças e contexto, dados importantes para se avaliar corretamente o risco e o impacto de exploração das vulnerabilidades.

9 CONCLUSÕES E TRABALHOS FUTUROS

Para avaliar corretamente o risco de exploração de falhas de segurança é necessário correlacionar informações sobre vulnerabilidades, inteligência de ameaças e contexto. Na maioria das vezes, os especialistas fazem isso manualmente e sem o auxílio de ferramentas especializadas, o que torna essa tarefa complexa e sujeita a erros. Assim, propomos o FRAPE, um *framework* que visa facilitar e automatizar o trabalho do analista de segurança no processo de gestão de vulnerabilidades.

O FRAPE é capaz de auxiliar os analistas na coleta de informações e classificação dos riscos das vulnerabilidades, bem como guiá-los no processo de correção das falhas de segurança. Para isso, o FRAPE combina a técnica de aprendizado ativo com algoritmos de aprendizado supervisionado para criar um modelo de ML capaz de emular a experiência de especialistas na tarefa de avaliação de risco.

Experimentos demonstraram que o emprego do FRAPE em substituição à abordagem clássica de priorização das vulnerabilidades utilizando apenas o CVSS é benéfico. Visto que o FRAPE ajuda a diminuir a superfície de ataque da empresa, deixando-a menos vulnerável a ataques que possam ser realizados por usuários mal-intencionados.

9.1 Contribuições do Trabalho

As principais contribuições desta tese, detalhadas no Capítulo 5 e avaliadas nos Capítulos 6, 7 e 8 são descritas a seguir:

- **FRAPE:** *framework* de segurança proposto para auxiliar os analistas de IS na tarefa de classificação de risco das vulnerabilidades. O FRAPE é composto por 4 módulos, que são: *i*) Coleta de Dados; *ii*) Rotulação das Vulnerabilidades; *iii*) Classificação e Priorização das Vulnerabilidades; e *iv*) Interpretação dos Resultados. O *framework* tem como intuito auxiliar os analistas no processo de coleta de informação e classificação das vulnerabilidades e, além disso, guiá-los no processo de correção das falhas de segurança. Isso é possível graças ao uso de múltiplas técnicas de aprendizado de máquina;
- **Conjunto de Dados de Vulnerabilidades de Segurança:** para avaliar o desempenho da proposta, criamos um conjunto de dados, de propósito geral, contendo informações acerca das características das vulnerabilidades, de informações de inteligência de ameaças e contexto. Para isso, desenvolvemos uma ferramenta capaz de coletar e cruzar informações

sobre as vulnerabilidades de mais de 10 fontes de segurança;

- **Técnica de Classificação de Risco:** utilizamos os três conjuntos de informações coletados sobre as vulnerabilidades e uma técnica chamada de aprendizado ativo para treinar um classificador de risco. Aprendizado ativo é uma técnica de aprendizado de máquina que iterativamente consulta os especialistas acerca do rótulo que deve ser aplicado às instâncias analisadas. Assim, é possível treinar um modelo mais eficiente, que emule a experiência dos especialistas na tarefa de classificação de risco as vulnerabilidades;
- **Simulador de Avaliação de Estratégias de VM:** desenvolvemos um simulador de redes de computadores, capaz de aplicar a técnica de classificação de risco proposta nesta tese e comparar sua eficiência com a técnica de VM que utiliza apenas o CVSS como forma de priorização.

9.2 Publicações e Submissões da Tese

Durante o Doutorado, foram publicados dois artigos diretamente relacionados à tese e um *workshop* que contribuiu para o amadurecimento do aluno na área de aprendizado de máquina. Os trabalhos estão listados a seguir, com o nome do congresso e o seu QUALIS[1], índice científico oficial brasileiro utilizado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES). Os artigos diretamente relacionados a tese e fruto dos experimentos demonstrados nos Capítulos 6 e 7, são descritos a seguir:

- O artigo "*CVEjoin: An Information Security Vulnerability and Threat Intelligence Dataset*" (PONTE *et al.*, 2023a) publicado na conferência internacional em *Advanced Information Networking and Applications* (AINA), QUALIS A2, apresenta um *dataset* de vulnerabilidades de segurança da informação de uso geral.
- O artigo "*A Vulnerability Risk Assessment Methodology Using Active Learning*" (PONTE *et al.*, 2023b) publicado na conferência internacional em *Advanced Information Networking and Applications* (AINA), QUALIS A2, apresenta um processo de classificação do risco das vulnerabilidades de segurança utilizando o aprendizado ativo e supervisionado.

Vale ressaltar que submetemos um trabalho que engloba a proposta dessa tese (Capitulo 5), bem como os resultados obtidos a partir do simulador (Capítulo 8), para o *Computer & Security*, um importante periódico da área de cibersegurança. Estamos otimistas com sua aprova-

[1] O índice QUALIS é dividido em 10 estratos, em ordem decrescente de valor: A1, A2, A3, A4, B1, B2, B3, B4, B5, C.

ção e esperamos obter uma resposta em breve. Além disso, fizemos o depósito de uma patente intitulada “Sistema Implementado por Programa de Computador para Avaliação, Classificação, Priorização e Explicabilidade de Riscos de Vulnerabilidades de Segurança Cibernética e Plataforma de Gestão de Vulnerabilidades de Segurança Baseada em Risco” no dia 27 de fevereiro de 2024. O depósito da patente pode ser acessado através do Nº BR 10 2024 003842 8 pelo site do Instituto Nacional de Propriedade Industrial (INPI). Por fim, houve outras publicações que contribuíram para o amadurecimento do autor no doutorado, que são:

– O minicurso “Inteligência Artificial e Função como Serviço: Provisionando Aplicações com o AWS Lambda” (ARAÚJO *et al.*, 2022) publicado no Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD), QUALIS B4, introduz conceitos de aprendizado de máquina, bem como de funções como serviço e AWS Lambda.

9.3 Trabalhos Futuros

Como trabalhos futuros, planejamos aumentar o número de fontes utilizadas na coleta de informações sobre as vulnerabilidades, adicionando, por exemplo, repositórios públicos de código (Github). Adicionalmente, planejamos analisar o impacto de técnicas de aprendizado de máquina que suponham a independência entre os atributos das instâncias analisadas, como, por exemplo, o método Naive Bayes, na acurácia do modelo. Dessa forma, a solução de RBVM será mais flexível e se beneficiará com a inclusão de novos atributos (e.g., o surgimento de uma nova fonte de dados) no processo de classificação do risco das vulnerabilidades. Por fim, queremos implementar o *framework* FRAPE em um ambiente organizacional real, para avaliar o seu desempenho, na prática. Além disso, existem algumas limitações do nosso trabalho que podem ser investigadas por outros pesquisadores, por exemplo:

– como a área de atuação do profissional pode influenciar na criação do modelo de classificação, visto que, por exemplo, uma vulnerabilidade crítica para o mercado financeiro, pode não ser crítica para a área educacional e vice-versa;
– como estimar o risco de vulnerabilidades *zero-day*, visto que, muitas vezes, pouco se sabe sobre essas falhas de segurança no momento de sua descoberta;
– como a exploração de vulnerabilidades com riscos baixos e médios podem ser equivalentes à exploração de uma vulnerabilidade com risco alto (tática conhecida como *chain attack*);

REFERÊNCIAS

ABDELRAHIM, A. A. A.; ELHADI, A. A. E.; IBRAHIM, H.; ELMISBAH, N. Feature selection and similarity coefficient based method for email spam filtering. In: **2013 INTERNATIONAL CONFERENCE ON COMPUTING, ELECTRICAL AND ELECTRONIC ENGINEERING (ICCEEE)**. [*S. l.*: *s. n.*], 2013. p. 630–633.

ABOWD, G. D.; DEY, A. K.; BROWN, P. J.; DAVIES, N.; SMITH, M.; STEGGLES, P. Towards a better understanding of context and context-awareness. In: SPRINGER. **International symposium on handheld and ubiquitous computing**. [*S. l.*], 1999. p. 304–307.

ABU, M. S.; SELAMAT, S. R.; ARIFFIN, A.; YUSOF, R. Cyber threat intelligence–issue and challenges. **Indonesian Journal of Electrical Engineering and Computer Science**, v. 10, n. 1, p. 371–379, 2018.

ADOBE. **Adobe: Severity ratings**. 2022. Disponível em: https://helpx.adobe.com/security/severity-ratings.html. Acesso em: 15 jul. 2022.

AFEK, Y.; BREMLER-BARR, A.; FEIBISH, S. L. Zero-day signature extraction for high-volume attacks. **IEEE/ACM Transactions on Networking**, IEEE, v. 27, n. 2, p. 691–706, 2019.

AGHAM, V. Unified threat management. **International Research Journal of Engineering and Technology**, v. 3, n. 4, p. 32–36, 2016.

ALEXANDER, J. **Risk, Threat, or Vulnerability? How to Tell the Difference**. 2021. Disponível em: https://www.kennasecurity.com/blog/risk-vs-threat-vs-vulnerability/. Acesso em: 22 dez. 2021.

ALLA, S.; ADARI, S. K. What is mlops? In: **Beginning MLOps with MLFlow**. [*S. l.*]: Springer, 2021. p. 79–124.

ALPAYDIN, E. **Introduction to machine learning**. [*S. l.*]: MIT press, 2020.

ALSMADI, I.; ALHAMI, I. Clustering and classification of email contents. **Journal of King Saud University-Computer and Information Sciences**, Elsevier, v. 27, n. 1, p. 46–57, 2015.

ALVES, H.; FONSECA, B.; ANTUNES, N. Software metrics and security vulnerabilities: dataset and exploratory study. In: IEEE. **2016 12th European Dependable Computing Conference (EDCC)**. [*S. l.*], 2016. p. 37–44.

AMAZON. **Preço da rotulagem de dados do Amazon SageMaker**. 2021. Disponível em: https://aws.amazon.com/pt/sagemaker/data-labeling/pricing/. Acesso em: 25 jun. 2022.

AMOLI, P. V.; HAMALAINEN, T.; DAVID, G.; ZOLOTUKHIN, M.; MIRZAMOHAMMAD, M. Unsupervised network intrusion detection systems for zero-day fast-spreading attacks and botnets. **JDCTA (International Journal of Digital Content Technology and its Applications**, v. 10, n. 2, p. 1–13, 2016.

ANDRESS, J. **Foundations of Information Security: A Straightforward Introduction**. [*S. l.*]: No Starch Press, 2019.

ARAÚJO, S. S.; PONTE, F. R. da; OLIVEIRA, V. T.; SILVA, W. S. da; VIEIRA, D.; CASTRO, M. F. de; RODRIGUES, E. B. Inteligência artificial e função como serviço: Provisionando aplicações com o aws lambda. **Sociedade Brasileira de Computação**, 2022.

AUGUSTEIJN, M.; FOLKERT, B. Neural network classification and novelty detection. **International Journal of Remote Sensing**, Taylor & Francis, v. 23, n. 14, p. 2891–2902, 2002.

BAKER, W. **Metrics That Actually Matter for Vulnerability Management**. [*S. l.*], 2022. Disponível em: https://vtechworks.lib.vt.edu/bitstream/handle/10919/112234/TAGCyber_2022Q4.pdf?sequence=2.

BAUM, E. B.; LANG, K. Query learning can work poorly when a human oracle is used. In: BEIJING CHINA. **International joint conference on neural networks**. [*S. l.*], 1992. v. 8, p. 8.

BEEK, C.; DUNTON, T.; FOKKER, J.; GROBMAN, S.; HUX, T.; POLZER, T.; RIVERO, M.; ROCCIA, T.; SAAVEDRA-MORALES, J.; SAMANI, R. *et al.* **Mcafee labs threats report: August 2019**. 2019. Disponível em: https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-aug-2019.pdf. Acesso em: 18 dez. 2021.

BHAJANKA, P.; SCHNEIDER, M.; LAWSON, C. **A Guide to Choosing a Vulnerability Assessment Solution**. 2019. Disponível em: https://www.tenable.com/analyst-research/gartner-a-guide-to-choosing-a-vulnerability-assessment-solution-2019. Acesso em: 10 jan. 2021.

BHANDARI, G.; NASEER, A.; MOONEN, L. Cvefixes: automated collection of vulnerabilities and their fixes from open-source software. In: **Proceedings of the 17th International Conference on Predictive Models and Data Analytics in Software Engineering**. [*S. l.*: *s. n.*], 2021. p. 30–39.

BHATT, U.; XIANG, A.; SHARMA, S.; WELLER, A.; TALY, A.; JIA, Y.; GHOSH, J.; PURI, R.; MOURA, J. M.; ECKERSLEY, P. Explainable machine learning in deployment. In: **Proceedings of the 2020 conference on fairness, accountability, and transparency**. [*S. l.*: *s. n.*], 2020. p. 648–657.

BONACCORSO, G. **Machine learning algorithms**. [*S. l.*]: Packt Publishing Ltd, 2017.

BRIER, G. W. *et al.* Verification of forecasts expressed in terms of probability. **Monthly weather review**, v. 78, n. 1, p. 1–3, 1950.

BROMANDER, S. **Understanding Cyber Threat Intelligence: Towards Automation**. Dissertação (Mestrado) – University of Oslo, 2021.

BROWNLEE, J. **Master Machine Learning Algorithms: discover how they work and implement them from scratch**. [*S. l.*]: Machine Learning Mastery, 2016.

BURKART, N.; HUBER, M. F. A survey on the explainability of supervised machine learning. **Journal of Artificial Intelligence Research**, v. 70, p. 245–317, 2021.

CASAS, P.; MAZEL, J.; OWEZARSKI, P. Unsupervised network intrusion detection systems: Detecting the unknown without knowledge. **Computer Communications**, v. 35, n. 7, p. 772–783, 2012. ISSN 0140-3664. Disponível em: https://www.sciencedirect.com/science/article/pii/S0140366412000266.

CASTILLO, C.; DAVISON, B. Adversarial web search. **Foundations and Trends in Information Retrieval**, v. 4, p. 377–486, 01 2010.

CEZARINA, C. **What Is Vulnerability Risk Management?** 2021. Disponível em: https://heimdalsecurity.com/blog/what-is-vulnerability-risk-management/. Acesso em: 09 dez. 2021.

CHAKRABORTY, M.; PAL, S.; PRAMANIK, R.; Ravindranath Chowdary, C. Recent developments in social spam detection and combating techniques: A survey. **Information Processing & Management**, v. 52, n. 6, p. 1053–1073, 2016. ISSN 0306-4573. Disponível em: https://www.sciencedirect.com/science/article/pii/S0306457316300966.

CHANDRASEKAR, C.; PRIYATHARSINI, P. Classification techniques using spam filtering email. **International Journal of Advanced Research in Computer Science**, v. 9, n. 2, 2018.

CHANDRASEKHAR, A. M.; RAGHUVEER, K. Intrusion detection technique by using k-means, fuzzy neural network and svm classifiers. In: **2013 International Conference on Computer Communication and Informatics**. [*S. l.*: *s. n.*], 2013. p. 1–7.

CHAWLA, G.; SHARMA, N.; RAWAL, N. K. Ivsev: Improved vulnerability scoring mechanism with environment representative and vulnerability type. **International Journal of Scientific & Technology Research**, v. 8, n. 10, p. 1043–1047, 2019. ISSN 2277-8616.

CHEN, W.-H.; HSU, S.-H.; SHEN, H.-P. Application of svm and ann for intrusion detection. **Computers & Operations Research**, Elsevier, v. 32, n. 10, p. 2617–2634, 2005.

CHEUNG, K.-F.; BELL, M. G. Attacker–defender model against quantal response adversaries for cyber security in logistics management: An introductory study. **European Journal of Operational Research**, Elsevier, v. 291, n. 2, p. 471–481, 2021.

CHOI, H.; KIM, M.; LEE, G.; KIM, W. Unsupervised learning approach for network intrusion detection system using autoencoders. **The Journal of Supercomputing**, Springer, v. 75, p. 5597–5621, 2019.

CHORDIYA, A. R.; MAJUMDER, S.; JAVAID, A. Y. Man-in-the-middle (mitm) attack based hijacking of http traffic using open source tools. In: IEEE. **2018 IEEE International Conference on Electro/Information Technology (EIT)**. [*S. l.*], 2018. p. 0438–0443.

CIS, C. f. I. S. **CIS Critical Security Controls**. 2021. Disponível em: https://www.cisecurity.org/controls/. Acesso em: 09 dez. 2021.

CISCO. **What Is a Cyberattack?** 2018. Disponível em: https://www.cisco.com/c/en_au/products/security/common-cyberattacks.html#~types-of-cyber-attacks. Acesso em: 04 jan. 2022.

CONTI, M.; DARGAHI, T.; DEHGHANTANHA, A. Cyber threat intelligence: challenges and opportunities. In: **Cyber Threat Intelligence**. [*S. l.*]: Springer, 2018. p. 1–6.

DAGAN, I.; ENGELSON, S. P. Committee-based sampling for training probabilistic classifiers. In: **Machine Learning Proceedings 1995**. [*S. l.*]: Elsevier, 1995. p. 150–157.

DARGAHI, T.; DEHGHANTANHA, A.; BAHRAMI, P. N.; CONTI, M.; BIANCHI, G.; BENEDETTO, L. A cyber-kill-chain based taxonomy of crypto-ransomware features. **Journal of Computer Virology and Hacking Techniques**, Springer, v. 15, n. 4, p. 277–305, 2019.

DEY, D.; LAHIRI, A.; ZHANG, G. Optimal policies for security patch management. **INFORMS Journal on Computing**, INFORMS, v. 27, n. 3, p. 462–477, 2015.

DOSSETT, J. **A timeline of the biggest ransomware attacks**. 2021. Disponível em: https://www.cnet.com/personal-finance/crypto/a-timeline-of-the-biggest-ransomware-attacks/. Acesso em: 18 dez. 2021.

ELBAZ, C.; RILLING, L.; MORIN, C. Automated risk analysis of a vulnerability disclosure using active learning. In: **Proceedings of the 28th Computer & Electronics Security Application Rendezvous**. [*S. l.*: *s. n.*], 2021.

ELHADI, A.; MAAROF, M.; OSMAN, A. H. Malware detection based on hybrid signature behaviour application programming interface call graph. **American Journal of Applied Sciences**, v. 9, p. 283–288, 01 2012.

ELHAG, S.; FERNáNDEZ, A.; BAWAKID, A.; ALSHOMRANI, S.; HERRERA, F. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. **Expert Systems with Applications**, v. 42, n. 1, p. 193–202, 2015. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417414004783.

FAN, J.; LI, Y.; WANG, S.; NGUYEN, T. N. Ac/c++ code vulnerability dataset with code changes and cve summaries. In: **Proceedings of the 17th International Conference on Mining Software Repositories**. [*S. l.*: *s. n.*], 2020. p. 508–512.

FENG, S.; ZHOU, H.; DONG, H. Using deep neural network with small dataset to predict material defects. **Materials & Design**, Elsevier, v. 162, p. 300–310, 2019.

FIROIU, M. General considerations on risk management and information system security assessment according to iso/iec 27005: 2011 and iso 31000: 2009 standards. **Quality-Access to Success**, v. 16, n. 149, 2015.

FIRST, I. **Common Vulnerability Scoring System v3.0**. 2019. Disponível em: https://www.first.org/cvss/v3.1/specification-document. Acesso em: 07 dez. 2021.

FOREMAN, P. **Vulnerability management**. [*S. l.*]: CRC Press, 2019.

FREDRIKSON, M.; JHA, S.; RISTENPART, T. Model inversion attacks that exploit confidence information and basic countermeasures. In: **Proceedings of the 22nd ACM SIGSAC conference on computer and communications security**. [*S. l.*: *s. n.*], 2015. p. 1322–1333.

FREITAS, A. A. Comprehensible classification models: a position paper. **ACM SIGKDD explorations newsletter**, ACM New York, NY, USA, v. 15, n. 1, p. 1–10, 2014.

FUDENBERG, D.; TIROLE, J. **Game theory**. [*S. l.*]: MIT press, 1991.

FURNELL, S.; FISCHER, P.; FINCH, A. Can't get the staff? the growing need for cyber-security skills. **Computer Fraud & Security**, v. 2017, p. 5–10, 02 2017.

GANDOTRA, E.; BANSAL, D.; SOFAT, S. Malware analysis and classification: A survey. **Journal of Information Security**, v. 05, p. 56–64, 01 2014.

GARCIA, V. H.; MONROY, R.; QUINTANA, M. Web attack detection using id3. In: SPRINGER. **IFIP World Computer Congress, TC 12**. [*S. l.*], 2006. p. 323–332.

GELUVARAJ, B.; SATWIK, P.; KUMAR, T. A. The future of cybersecurity: Major role of artificial intelligence, machine learning, and deep learning in cyberspace. In: SPRINGER. **International Conference on Computer Networks and Communication Technologies**. [*S. l.*], 2019. p. 739–747.

GEORGIEV, D. **Internet of Things Statistics, Facts & Predictions**. 2021. Disponível em: https://review42.com/resources/internet-of-things-stats/. Acesso em: 22 dez. 2021.

GHARAEE, H.; HOSSEINVAND, H. A new feature selection ids based on genetic algorithm and svm. In: **2016 8th International Symposium on Telecommunications (IST)**. [*S. l.*: *s. n.*], 2016. p. 139–144.

GKORTZIS, A.; MITROPOULOS, D.; SPINELLIS, D. Vulinoss: a dataset of security vulnerabilities in open-source systems. In: **Proceedings of the 15th International conference on mining software repositories**. [*S. l.*: *s. n.*], 2018. p. 18–21.

GLOVER, C. **The Difference Between Threat, Vulnerability, and Risk, and Why You Need to Know**. 2020. Disponível em: https://lifars.com/2020/07/threat-vulnerability-risk-what-is-the-difference/. Acesso em: 22 dez. 2021.

GOLDBLUM, M.; TSIPRAS, D.; XIE, C.; CHEN, X.; SCHWARZSCHILD, A.; SONG, D.; MADRY, A.; LI, B.; GOLDSTEIN, T. Dataset security for machine learning: Data poisoning, backdoor attacks, and defenses. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, IEEE, 2022.

GOLDSTEIN, A. **Vulnerability Management – What You Need To Know**. 2021. Disponível em: https://www.whitesourcesoftware.com/resources/blog/vulnerability-management/. Acesso em: 05 jan. 2021.

GOMES, L. H.; CAZITA, C.; ALMEIDA, J. M.; ALMEIDA, V.; MEIRA, W. Characterizing a spam traffic. In: . New York, NY, USA: Association for Computing Machinery, 2004. (IMC '04), p. 356–369. ISBN 1581138210. Disponível em: https://doi.org/10.1145/1028788.1028837.

GOODMAN, B.; FLAXMAN, S. European union regulations on algorithmic decision-making and a "right to explanation". **AI magazine**, v. 38, n. 3, p. 50–57, 2017.

GRANADILLO, G. G.; DIAZ, R.; VERONI, E.; XENAKIS, C. A multi-factor assessment mechanism to define priorities on vulnerabilities affecting healthcare organizations. **Italian Conference on Cybersecurity**, 2021.

GREENERT, J. W. **Kill Chain Approach**. 2013. Disponível em: https://web.archive.org/web/20130613233413/http://cno.navylive.dodlive.mil/2013/04/23/kill-chain-approach-4/. Acesso em: 23 dez. 2021.

GUERCIO, K. **Best Intrusion Detection and Prevention Systems for 2021: Guide to IDPS**. 2021. Disponível em: https://www.esecurityplanet.com/products/top-intrusion-detection-prevention-systems.html. Acesso em: 17 dez. 2021.

GUO, H.; CHENG, H.; KELLEY, K. Impact of network structure on malware propagation: A growth curve perspective. **Journal of Management Information Systems**, v. 33, p. 296–325, 01 2016.

HANDA, A.; SHARMA, A.; SHUKLA, S. K. Machine learning in cybersecurity: A review. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 9, n. 4, p. e1306, 2019.

HATHAWAY, O. A.; CROOTOF, R.; LEVITZ, P.; NIX, H.; NOWLAN, A.; PERDUE, W.; SPIEGEL, J. The law of cyber-attack. **California Law Review**, JSTOR, p. 817–885, 2012.

HELM, J. M.; SWIERGOSZ, A. M.; HAEBERLE, H. S.; KARNUTA, J. M.; SCHAFFER, J. L.; KREBS, V. E.; SPITZER, A. I.; RAMKUMAR, P. N. Machine learning and artificial intelligence: definitions, applications, and future directions. **Current reviews in musculoskeletal medicine**, Springer, v. 13, p. 69–76, 2020.

HOLZINGER, A.; LANGS, G.; DENK, H.; ZATLOUKAL, K.; MÜLLER, H. Causability and explainability of artificial intelligence in medicine. **Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery**, Wiley Online Library, v. 9, n. 4, p. e1312, 2019.

HUMAYUN, M.; NIAZI, M.; JHANJHI, N.; ALSHAYEB, M.; MAHMOOD, S. Cyber security threats and vulnerabilities: a systematic mapping study. **Arabian Journal for Science and Engineering**, Springer, v. 45, n. 4, p. 3171–3189, 2020.

HUSEINOVIĆ, A.; MRDOVIĆ, S.; BICAKCI, K.; ULUDAG, S. A survey of denial-of-service attacks and solutions in the smart grid. **IEEE Access**, IEEE, v. 8, p. 177447–177470, 2020.

ISO Central Secretary. **Information Security Management Systems**. Geneva, CH, 2022. Disponível em: https://www.iso.org/standard/27001.

JAGOTA, A. Novelty detection on a very large number of memories stored in a hopfield-style network. In: IEEE. **IJCNN-91-seattle international joint conference on neural networks**. [*S. l.*], 1991. v. 2, p. 905–vol.

JASON, R. **What is Risk-Based Vulnerability Management?** 2020. Disponível em: https://www.kennasecurity.com/blog/what-is-risk-based-vulnerability-management/. Acesso em: 07 dez. 2021.

JIMENEZ, M.; TRAON, Y. L.; PAPADAKIS, M. [engineering paper] enabling the continuous analysis of security vulnerabilities with vuldata7. In: IEEE. **2018 IEEE 18th International Working Conference on Source Code Analysis and Manipulation (SCAM)**. [*S. l.*], 2018. p. 56–61.

JOHNSON, J. **Number of e-mail users worldwide from 2017 to 2025**. 2021. Disponível em: https://www.statista.com/topics/8338/malware/#dossierKeyfigures. Acesso em: 18 dez. 2021.

JOSEPH, J. **Number of e-mail users worldwide from 2017 to 2025**. 2021. Disponível em: https://www.statista.com/statistics/270899/global-e-mail-spam-rate/. Acesso em: 18 dez. 2021.

JOSHI, P. **Artificial intelligence with python**. [*S. l.*]: Packt Publishing Ltd, 2017.

KARPERSKY. **What is Cyber Security?** 2020. Disponível em: https://www.kaspersky.com.au/resource-center/definitions/what-is-cyber-security. Acesso em: 22 dez. 2021.

Kenna Security. **Understanding the Kenna Risk Score Prioritizing Vulnerabilities with Data Science**. [*S. l.*], 2020. Disponível em: https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/docs/vmwcb-whitepaper-understanding-the-kenna-security-vulnerability-risk-score.pdf.

KENNA SECURITY. **How to Implement Risk-Based Vulnerability Management Now: A Pratical Guide**. 2020. Disponível em: https://website.kennasecurity.com/wp-content/uploads/2021/03/Kenna_Implementing_RBVM_Guide-2.pdf. Acesso em: 10 jan. 2021.

KENNA SECURITY; CYENTIA INSTITUTE. **Winning the Remediation Race**. [*S. l.*], 2019. Disponível em: https://website.kennasecurity.com/wp-content/uploads/2020/09/Kenna_Prioritization_to_Prediction_Vol3.pdf. Acesso em: 07 dez. 2021.

KERAMATI, M. New vulnerability scoring system for dynamic security evaluation. In: IEEE. **2016 8th International Symposium on Telecommunications (IST)**. [*S. l.*], 2016. p. 746–751.

KIM, G.; LEE, S.; KIM, S. A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. **Expert Systems with Applications**, v. 41, n. 4, Part 2, p. 1690–1700, 2014. ISSN 0957-4174. Disponível em: https://www.sciencedirect.com/science/article/pii/S0957417413006878.

KINGMA, D. P.; MOHAMED, S.; REZENDE, D. J.; WELLING, M. Semi-supervised learning with deep generative models. **Advances in neural information processing systems**, v. 27, 2014.

KOK, S.; ABDULLAH, A.; JHANJHI, N.; SUPRAMANIAM, M. Ransomware, threat and detection techniques: A review. **International Journal of Computer Science and Network Security**, v. 19, n. 2, p. 136, 2019.

KONDALWAR, M. N.; SHELKE, C. Remote administrative trojan/tool (rat). **Int. J. Comput. Sci. Mob. Comput**, v. 3333, n. 3, p. 482–487, 2014.

KUMAR, M. N.; KOUSHIK, K.; SUNDAR, K. J. Data mining and machine learning techniques for cyber security intrusion detection. **International Journal of Scientific Research in Computer Science, Engineering and Information Technology**, v. 3, n. 3, p. 162–167, 2018.

KUMPULAINEN, P.; HäTöNEN, K.; FINLAND, E. Anomaly detection algorithm test bench for mobile network management. **Tampere University of Technology**, 2008.

KURE, H. I.; ISLAM, S.; GHAZANFAR, M.; RAZA, A.; PASHA, M. Asset criticality and risk prediction for an effective cybersecurity risk management of cyber-physical system. **Neural Computing and Applications**, Springer, v. 34, n. 1, p. 493–514, 2022.

LATCHOUMI, T.; REDDY, M. S.; BALAMURUGAN, K. Applied machine learning predictive analytics to sql injection attack detection and prevention. **European Journal of Molecular & Clinical Medicine**, v. 7, n. 02, p. 2020, 2020.

LAWSON, C. **Gartner's Strategic Vision for Vulnerability Management**. 2020. Disponível em: https://assets-powerstores-com.s3.amazonaws.com/data/org/20033/media/doc/gartner_s_strategic_vision_for_vulnerability_management_15998516437370014obf-374cac7236d6163e8bb34b4faa465965.pdf. Acesso em: 10 jan. 2021.

LAWSON, C.; SCHNEIDER, M.; BHAJANKA, P.; GARDNER, D. Technical Report, **Market Guide for Vulnerability Assessment**. 2019. Disponível em: https://www.gartner.com/en/documents/3975388. Acesso em: 19 may 2022.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. **nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LEE, S. M.; KIM, D. S.; KIM, J. H.; PARK, J. S. Spam detection using feature selection and parameters optimization. In: IEEE. **2010 International Conference on Complex, Intelligent and Software Intensive Systems**. [*S. l.*], 2010. p. 883–888.

LEI, M.; YANG, Y.; MA, N.; HUIZHONG, S.; ZHOU, C.; MA, M. Dynamically enabled defense effectiveness evaluation of a home internet based on vulnerability analysis and attack layer measurement. **Personal and Ubiquitous Computing**, v. 22, 02 2018.

LIFARS. **Threat, vulnerability, risk: What is the difference?** 2020. Disponível em: https://www.travasecurity.com/resources/the-difference-between-threat-vulnerability-and-risk-and-why-you-need-to-know. Acesso em: 22 dez. 2021.

LIN, W.-C.; KE, S.-W.; TSAI, C.-F. Cann: An intrusion detection system based on combining cluster centers and nearest neighbors. **Knowledge-Based Systems**, v. 78, p. 13–21, 2015. ISSN 0950-7051. Disponível em: https://www.sciencedirect.com/science/article/pii/S0950705115000167.

LIPTON, Z. C. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. **Queue**, ACM New York, NY, USA, v. 16, n. 3, p. 31–57, 2018.

LUNDBERG, S. M.; ERION, G.; CHEN, H.; DEGRAVE, A.; PRUTKIN, J. M.; NAIR, B.; KATZ, R.; HIMMELFARB, J.; BANSAL, N.; LEE, S.-I. From local explanations to global understanding with explainable ai for trees. **Nature machine intelligence**, Nature Publishing Group UK London, v. 2, n. 1, p. 56–67, 2020.

LUNDBERG, S. M.; LEE, S.-I. A unified approach to interpreting model predictions. **Advances in neural information processing systems**, v. 30, 2017.

MAALØE, L.; SØNDERBY, C. K.; SØNDERBY, S. K.; WINTHER, O. Auxiliary deep generative models. In: PMLR. **International conference on machine learning**. [*S. l.*], 2016. p. 1445–1453.

MAĆKIEWICZ, A.; RATAJCZAK, W. Principal components analysis (pca). **Computers & Geosciences**, Elsevier, v. 19, n. 3, p. 303–342, 1993.

MADNICK, S. Preparing for the cyberattack that will knock out us power grids. **Harvard Business Review**, v. 10, 2017.

MAHESH, B. Machine learning algorithms-a review. **International Journal of Science and Research (IJSR).[Internet]**, v. 9, p. 381–386, 2020.

MAKINEN, S.; SKOGSTRöM, H.; LAAKSONEN, E.; MIKKONEN, T. Who needs mlops: What data scientists seek to accomplish and how can mlops help? In: **2021 IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN)**. [*S. l.*: *s. n.*], 2021. p. 109–112.

MARTIN, L. **Seven Ways to Apply the Cyber Kill Chain with a Threat Intelligence Platform**. 2015. Disponível em: https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/Seven_Ways_to_Apply_the_Cyber_Kill_Chain_with_a_Threat_Intelligence_Platform.pdf. Acesso em: 23 dez. 2021.

MARTINEZ, D. Neural tree density estimation for novelty detection. **IEEE Transactions on Neural Networks**, IEEE, v. 9, n. 2, p. 330–338, 1998.

MARTINEZ, P. L.; DARRINGTON, E. L.; BOYD, K. D. The journey to vulnerability management? 5 2019. Disponível em: https://www.osti.gov/biblio/1512792.

MCCALLUMZY, A. K.; NIGAMY, K. Employing em and pool-based active learning for text classification. In: CITESEER. **Proc. International Conference on Machine Learning (ICML)**. [*S. l.*], 1998. p. 359–367.

MEHRI, V. A.; ARLOS, P.; CASALICCHIO, E. Normalization framework for vulnerability risk management in cloud. In: IEEE. **IEEE International Conference on Future Internet of Things and Cloud (FiCloud)**. [*S. l.*], 2021.

MELL, P.; BERGERON, T.; HENNING, D. *et al.* Creating a patch and vulnerability management program. **NIST Special Publication**, v. 800, p. 40, 2005.

MELTZER, J. P. Cybersecurity, digital trade, and data flows: Re-thinking a role for international trade rules. **Global Economy & Development WP**, v. 132, 2020.

MICHALSKI, R. S.; CARBONELL, J. G.; MITCHELL, T. M. **Machine learning: An artificial intelligence approach**. [*S. l.*]: Springer Science & Business Media, 2013.

MICROSOFT. **Microsoft: Security Update Severity Rating System**. 2022. Disponível em: https://www.microsoft.com/en-us/msrc/security-update-severity-rating-system. Acesso em: 15 jul. 2022.

MILLER, B.; LINDER, F.; MEBANE, W. R. Active learning approaches for labeling text: review and assessment of the performance of active learning approaches. **Political Analysis**, Cambridge University Press, v. 28, n. 4, p. 532–551, 2020.

MITCHELL, T. Machine learning. McGraw hill Burr Ridge, 1997.

MOHRI, M.; ROSTAMIZADEH, A.; TALWALKAR, A. **Foundations of machine learning**. [*S. l.*]: MIT press, 2018.

MOLNAR, C. **Interpretable Machine Learning: A Guide for Making Black Box Models Explainable**. [*S. l.*]: https://christophm.github.io/interpretable-ml-book/, 2022. 540 p.

MONITOR, C. **The Shocking Truth about How Many Emails Are Sent**. 2019. Disponível em: https://www.campaignmonitor.com/blog/email-marketing/shocking-truth-about-how-many-emails-sent/. Acesso em: 18 dez. 2021.

MORGAN, S. **Cybercrime To Cost The World $10.5 Trillion Annually By 2025**. 2020. Disponível em: https://cybersecurityventures.com/cybercrime-damage-costs-10-trillion-by-2025/. Acesso em: 22 dez. 2021.

MURPHY, K. P. **Probabilistic machine learning: an introduction**. [*S. l.*]: MIT press, 2022.

NAQA, I. E.; MURPHY, M. J. What is machine learning? In: **machine learning in radiation oncology**. [*S. l.*]: Springer, 2015. p. 3–11.

NARAYAN, V.; SHANMUGAPRIYA, D. Big data analytics with machine learning and deep learning methods for detection of anomalies in network traffic. In: **Research Anthology on Big Data Analytics, Architectures, and Applications**. [*S. l.*]: IGI Global, 2022. p. 678–707.

NICULESCU-MIZIL, A.; CARUANA, R. Predicting good probabilities with supervised learning. In: **Proceedings of the 22nd international conference on Machine learning**. [*S. l.*: *s. n.*], 2005. p. 625–632.

NIST, N. **NIST National Vulnerability Database**. 2021. Disponível em: https://nvd.nist.gov/. Acesso em: 07 dez. 2021.

NIST, National Institute of Standards and Technology. **Cyber Attack - Glossary | CSRC**. 2012. Disponível em: https://csrc.nist.gov/glossary/term/cyber_attack. Acesso em: 04 jan. 2022.

NIXON, J.; DUSENBERRY, M. W.; ZHANG, L.; JERFEL, G.; TRAN, D. Measuring calibration in deep learning. In: **CVPR workshops**. [*S. l.*: *s. n.*], 2019. v. 2, n. 7.

PARENTE, R.; BEZERRA, E.; MATTOS, C. CVEjoin: An Information Security Vulnerability and Threat Intelligence Dataset. 11 2022. Disponível em: https://figshare.com/articles/dataset/CVEjoin_A_Security_Dataset_of_Vulnerability_and_Threat_Intelligence_Information/21586923.

PARK, H.-S.; JUN, C.-H. A simple and fast algorithm for k-medoids clustering. **Expert systems with applications**, Elsevier, v. 36, n. 2, p. 3336–3341, 2009.

PATHAK, P. B. Cybercrime: A global threat to cybercommunity. **International Journal of Computer Science & Engineering Technology**, v. 7, n. 3, p. 46–49, 03 2016. ISSN 2229-3345.

PITROPAKIS, N.; PANAOUSIS, E.; GIANNETSOS, T.; ANASTASIADIS, E.; LOUKAS, G. A taxonomy and survey of attacks against machine learning. **Computer Science Review**, Elsevier, v. 34, p. 100199, 2019.

PLATT, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. **Advances in large margin classifiers**, v. 10, p. 61–74, 06 2000.

PONTA, S. E.; PLATE, H.; SABETTA, A.; BEZZI, M.; DANGREMONT, C. A manually-curated dataset of fixes to vulnerabilities of open-source software. In: IEEE. **2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)**. [*S. l.*], 2019. p. 383–387.

PONTE, F. R. da; RODRIGUES, E. B.; MATTOS, C. L. Cvejoin: An information security vulnerability and threat intelligence dataset. In: SPRINGER. **Advanced Information Networking and Applications: Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Volume 1**. [*S. l.*], 2023. p. 380–392.

PONTE, F. R. da; RODRIGUES, E. B.; MATTOS, C. L. A vulnerability risk assessment methodology using active learning. In: SPRINGER. **Advanced Information Networking and Applications: Proceedings of the 37th International Conference on Advanced Information Networking and Applications (AINA-2023), Volume 2**. [*S. l.*], 2023. p. 171–182.

PRASAD, R.; ROHOKALE, V. Artificial intelligence and machine learning in cyber security. In: **Cyber Security: The Lifeline of Information and Communication Technology**. [*S. l.*]: Springer, 2020. p. 231–247.

PRODROMIDIS, A. L.; STOLFO, S. J. Cost complexity-based pruning of ensemble classifiers. **Knowledge and Information Systems**, Springer, v. 3, n. 4, p. 449–469, 2001.

PURUSHOTHAM, S.; MENG, C.; CHE, Z.; LIU, Y. Benchmarking deep learning models on large healthcare datasets. **Journal of biomedical informatics**, Elsevier, v. 83, p. 112–134, 2018.

QUINLAN, J. R. **C4. 5: programs for machine learning**. [*S. l.*]: Elsevier, 2014.

RAPID7. **4 Key Pillars of Modern Vulnerability Risk Management**. [*S. l.*], 2021. Disponível em: https://www.rapid7.com/info/whitepaper-the-four-pillars-of-modern-vulnerability-management/.

RAPID7. **Vulnerability Management Process: Continuously identify and assess risk across your environment**. c2022. Disponível em: https://www.rapid7.com/fundamentals/vulnerability-management-and-scanning/. Acesso em: 05 jan. 2021.

RASTOGI, M.; CHHETRI, A.; SINGH, D. K. *et al.* Survey on detection and prevention of phishing websites using machine learning. In: IEEE. **2021 International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE)**. [*S. l.*], 2021. p. 78–82.

RecordFuture. **Top Exploited Vulnerabilities in 2020 Affect Citrix, Microsoft Products**. 2021. Disponível em: https://go.recordedfuture.com/hubfs/reports/cta-2021-0209.pdf. Acesso em: 28 jan. 2022.

RODRÍGUEZ, G. E.; TORRES, J. G.; FLORES, P.; BENAVIDES, D. E. Cross-site scripting (xss) attacks and mitigation: A survey. **Computer Networks**, Elsevier, v. 166, p. 106960, 2020.

ROSCHER, R.; BOHN, B.; DUARTE, M. F.; GARCKE, J. Explainable machine learning for scientific insights and discoveries. **Ieee Access**, IEEE, v. 8, p. 42200–42216, 2020.

SCHÖLKOPF, B.; WILLIAMSON, R. C.; SMOLA, A. J.; SHAWE-TAYLOR, J.; PLATT, J. C. *et al.* Support vector method for novelty detection. In: CITESEER. **NIPS**. [*S. l.*], 1999. v. 12, p. 582–588.

SELVARAJ, S. **Applying of machine learning for spam classification**. Tese (Doutorado) – Instytut Telekomunikacji, 2019.

SETTLES, B. **Active learning literature survey**. [*S. l.*], 2009. Disponível em: https://minds.wisconsin.edu/handle/1793/60660.

SETTLES, B. From theories to queries: Active learning in practice. In: JMLR WORKSHOP AND CONFERENCE PROCEEDINGS. **Active learning and experimental design workshop in conjunction with AISTATS 2010**. [*S. l.*], 2011. p. 1–18.

SHAH, N. F.; KUMAR, P. A comparative analysis of various spam classifications. In: **Progress in intelligent computing techniques: theory, practice, and applications**. [*S. l.*]: Springer, 2018. p. 265–271.

SHAPOORIFARD, H.; SHAMSINEJAD, P. Intrusion detection using a novel hybrid method incorporating an improved knn. **Int. J. Comput. Appl**, v. 173, n. 1, p. 5–9, 2017.

SHARMA, A.; KALBARCZYK, Z.; BARLOW, J.; IYER, R. Analysis of security data from a large computing organization. In: IEEE. **2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)**. [*S. l.*], 2011. p. 506–517.

SHAUKAT, K.; LUO, S.; VARADHARAJAN, V.; HAMEED, I. A.; XU, M. A survey on machine learning techniques for cyber security in the last decade. **IEEE Access**, IEEE, v. 8, p. 222310–222354, 2020.

SHOBHA, G.; RANGASWAMY, S. Machine learning. In: **Handbook of statistics**. [*S. l.*]: Elsevier, 2018. v. 38, p. 197–228.

SHREAD, P. **Best SIEM Tools & Software for 2021**. 2021. Disponível em: https://www.esecurityplanet.com/products/siem-tools/. Acesso em: 17 dez. 2021.

SKYBOX SECURITY. **Vulnerability and Threat Trends Report 2021: Cybersecurity comes of age**. [*S. l.*], 2021. Disponível em: https://lp.skyboxsecurity.com/rs/440-MPQ-510/images/Skybox-Security-vulnerability-and-threat-trends-report-2021.pdf.

SOROKIN, A.; FORSYTH, D. Utility data annotation with amazon mechanical turk. In: IEEE. **2008 IEEE computer society conference on computer vision and pattern recognition workshops**. [*S. l.*], 2008. p. 1–8.

SPRING, J.; HATLEBACK, E.; HOUSEHOLDER, A.; MANION, A.; SHICK, D. Time to change the cvss? **IEEE Security Privacy**, v. 19, n. 2, p. 74–78, 2021.

STERN, H. A survey of modern spam tools. In: **CEAS**. [*S. l.*: *s. n.*], 2008.

SUN, X.; DAI, J.; LIU, P.; SINGHAL, A.; YEN, J. Using bayesian networks for probabilistic identification of zero-day attack paths. **IEEE Transactions on Information Forensics and Security**, v. 13, n. 10, p. 2506–2521, 2018.

TANHA, J.; SOMEREN, M. V.; AFSARMANESH, H. Semi-supervised self-training for decision tree classifiers. **International Journal of Machine Learning and Cybernetics**, Springer, v. 8, p. 355–370, 2017.

TENABLE. **Predictive Priorization: Data Science Lets You Focus on the 3% of Vulnerabilities Likely to be Exploited**. [*S. l.*], 2020. Disponível em: https://lookbook.tenable.com/predictive-prioritization/technical-whitepaper-predictive-prioritization.

TENNABLE. **Risk-Based Vulnerability Management: Understanding Vulnerability Risk with Threat Context and Business Impact**. 2020. Disponível em: https://tenable.com/risk-based-vulnerability-management. Acesso em: 07 dez. 2021.

TENNABLE. **YOUR ANSWER TO THE VULNERABILITY OVERLOAD PROBLEM: RISK-BASED VULNERABILITY MANAGEMENT**. 2020. Disponível em: https://static.tenable.com/marketing/whitepapers/ComparisonGuide-Risk-Based-Vulnerability-Management.pdf. Acesso em: 10 jan. 2021.

THOMAS, T.; VIJAYARAGHAVAN, A. P.; EMMANUEL, S. Machine learning and cybersecurity. In: **Machine Learning Approaches in Cyber Security Analytics**. [*S. l.*]: Springer, 2020. p. 37–47.

TRAMÈR, F.; ZHANG, F.; JUELS, A.; REITER, M. K.; RISTENPART, T. Stealing machine learning models via prediction {APIs}. In: **25th USENIX security symposium (USENIX Security 16)**. [*S. l.*: *s. n.*], 2016. p. 601–618.

UCCI, D.; ANIELLO, L.; BALDONI, R. Survey of machine learning techniques for malware analysis. **Computers & Security**, Elsevier, v. 81, p. 123–147, 2019.

WAGNER, T. D.; MAHBUB, K.; PALOMAR, E.; ABDALLAH, A. E. Cyber threat intelligence sharing: Survey and research directions. **Computers & Security**, Elsevier, v. 87, p. 101589, 2019.

WALKOWSKI, M.; KRAKOWIAK, M.; JAROSZEWSKI, M.; OKO, J.; SUJECKI, S. Automatic cvss-based vulnerability prioritization and response with context information. In: IEEE. **2021 International Conference on Software, Telecommunications and Computer Networks (SoftCOM)**. [*S. l.*], 2021. p. 1–6.

WALTERMIRE, D.; QUINN, S.; BOOTH, H.; SCARFONE, K.; PRISACA, D. **The Technical Specification for the Security Content Automation Protocol (SCAP): SCAP Version 1.3**. 2018. Disponível em: https://csrc.nist.gov/publications/detail/sp/800-126/rev-3/final. Acesso em: 06 jan. 2021.

WANG, W.; SHI, F.; ZHANG, M.; XU, C.; ZHENG, J. A vulnerability risk assessment method based on heterogeneous information network. **IEEE Access**, IEEE, v. 8, p. 148315–148330, 2020.

WU, C.; WEN, T.; ZHANG, Y. A revised cvss-based system to improve the dispersion of vulnerability risk scores. **Science China Information Sciences**, v. 62, 03 2019.

YADAV, T.; RAO, A. M. Technical aspects of cyber kill chain. In: SPRINGER. **International Symposium on Security in Computing and Communication**. [*S. l.*], 2015. p. 438–452.

YAROWSKY, D. Unsupervised word sense disambiguation rivaling supervised methods. In: **Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics**. USA: Association for Computational Linguistics, 1995. (ACL '95), p. 189–196. Disponível em: https://doi.org/10.3115/981658.981684.

ZADROZNY, B.; ELKAN, C. Transforming classifier scores into accurate multiclass probability estimates. In: **Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining**. [*S. l.*: *s. n.*], 2002. p. 694–699.

ZENISEK, J.; HOLZINGER, F.; AFFENZELLER, M. Machine learning based concept drift detection for predictive maintenance. **Computers & Industrial Engineering**, Elsevier, v. 137, p. 106031, 2019.

ZHANG, X.; YAO, L.; YUAN, F. Adversarial variational embedding for robust semi-supervised learning. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining**. [*S. l.*: *s. n.*], 2019. p. 139–147.

ZHU, X. **Semi-supervised learning with graphs**. [*S. l.*]: Carnegie Mellon University, 2005.

ZOPPI, T.; CECCARELLI, A.; BONDAVALLI, A. Unsupervised algorithms to detect zero-day attacks: Strategy and application. **IEEE Access**, IEEE, v. 9, p. 90603–90615, 2021.

APÊNDICE A – GUIA PARA ROTULAÇÃO ONLINE

A.1 Visão Geral

Dado o surgimento acentuado de vulnerabilidades ao longo dos últimos anos e a falta de profissionais capacitados na área de segurança, fica evidente a necessidade de utilizarmos estratégias que permitam a seleção e priorização das falhas de *software* quanto ao seu risco e probabilidade de exploração. Um padrão bastante utilizado pela comunidade de segurança é a priorização das vulnerabilidades de acordo com sua nota de *Common Vulnerability Scoring System* (CVSS). No entanto, o CVSS não foi idealizado como uma nota de risco e sim como um indicador de severidade. Dessa forma, é impossível caracterizar o risco de uma vulnerabilidade de forma precisa só com o CVSS.

Para superar essa e outras limitações, propomos a utilização de 24 características distintas das vulnerabilidades para classificar o risco e a probabilidade de exploração das falhas de segurança. Esse trabalho consiste na análise de risco, por um especialista da área de segurança, de um conjunto de vulnerabilidades, tendo como base as suas características. Portanto, o analista deve classificar cada vulnerabilidade em uma das seguintes classes de risco: baixo (LOW), moderado (MODERATE), importante (IMPORTANT) e crítico (CRITICAL). A seguir, discutimos em detalhes cada uma das características das vulnerabilidades consideradas na classificação e as diferenças entre as classes de risco.

A.2 Atributos Considerados na Classificação

A seguir, descreveremos os atributos considerados pelos especialistas durante a classificação do risco das vulnerabilidades. No total, consideramos 24 atributos, divididos logicamente em três conjuntos, que são: as características das vulnerabilidades, as informações de inteligência de ameaças e as características de contexto.

A.2.1 Características das Vulnerabilidades

Esse grupo corresponde às informações que são constantes e não mudam entre ambientes ou ao longo do tempo, com exceção da data de publicação. Essas informações auxiliam os analistas a identificar o impacto da exploração das vulnerabilidades nos sistemas organizacionais.

- **Plataforma (*platform*):** identifica qual plataforma a vulnerabilidade afeta. Seus possíveis valores são: *hardware*, *software* ou *operating system*.
- **Fabricante (*vendor*):** refere-se ao nome do fabricante do produto vulnerável. Dado o grande número de fornecedores existentes, selecionamos os dez fabricantes mais significativos, ou seja, aqueles com maior número de vulnerabilidades. Assim, os possíveis valores são: *microsoft*, *google*, *apple*, *oracle*, *debian*, *IBM*, *cisco*, *adobe*, *redhat*, *fedora project* e *others*, para os demais fabricantes;
- **Nota de CVSS (*base_score*):** nota que considera as características intrínsecas da vulnerabilidade para definir sua severidade. Pode assumir valores numéricos entre 0 e 10, com 0 indicando uma vulnerabilidade com severidade baixa e 10 com severidade alta;
- **Impacto na Confidencialidade (*confidentiality_impact*):** este atributo mede o impacto que a exploração da vulnerabilidade terá na confidencialidade do sistema vulnerável. O possível valor é: NONE, quando não existe impacto; LOW, quando o impacto é baixo; ou HIGH, quando existe um alto impacto.
- **Impacto na Integridade (*integrity_impact*):** este atributo mede o impacto que a exploração da vulnerabilidade terá na integridade do sistema vulnerável. O possível valor é: NONE, quando não existe impacto; LOW, quando o impacto é baixo; ou HIGH, quando existe um alto impacto.
- **Impacto na Disponibilidade (*availability_impact*):** este atributo mede o impacto que a exploração da vulnerabilidade terá na disponibilidade do sistema vulnerável. O possível valor é: NONE, quando não existe impacto; LOW, quando o impacto é baixo; ou HIGH, quando existe um alto impacto.
- **Data de Publicação da Vulnerabilidade (*cve_published_date*):** data de publicação da vulnerabilidade na base de dados do *National Vulnerability Database* (NVD). Seu formato é DD/MM/YYYY;
- **Atualização Disponível (*update_available*):** indica se a vulnerabilidade em questão possui um *patch* de segurança que corrige a falha. Seu possível valor é um (1), quando existe um *patch* e zero (0), que não necessariamente indica que não existe um *patch*, mas que não conseguimos encontrar um.

A.2.2 Informações de Inteligência de Ameaças (Threat Intelligence)

O conjunto de informações de inteligência que variam ao longo do tempo, fornecendo os dados necessários para o analista conhecer o cenário atual de cibersegurança. Possibilitando assim, que a classificação de risco das vulnerabilidades seja feita de forma mais assertiva.

- **Mitre Top 25 (*mitre_top_25*) e OWASP Top 10 (*owasp_top_10*):** identifica se a vulnerabilidade em questão está presente ou não na lista de falhas de *software* considerada perigosa pelo Mitre, ou OWASP. Assume o valor de um (1) caso esteja presente na lista e zero (0) caso contrário;
- **Quantidade de Exploits Públicos (*exploit_count*):** número total de exploits que afetam a vulnerabilidade. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será zero (0).
- **Data de Publicação do Exploit (*exploit_published_date*):** data de publicação do *exploit* mais recente, no formato DD/MM/YYYY. Caso a vulnerabilidade não possua nenhum *exploit*, o valor do campo será nulo.
- ***Exploit Prediction Scoring System* (*epss*):** sistema de pontuação que visa definir a probabilidade de uma vulnerabilidade ser explorada nos próximos 30 dias. A nota assume valores decimais entre zero (0) e um (1), com valores próximos a zero indicando que a probabilidade de exploração é baixa e próximos a um, alta.
- **Tipo de Ataque (*attack_type*):** tipo de ataque utilizado para explorar a vulnerabilidade. Seus possíveis valores são: *remote code execution*, *arbitrary code execution*, *tampering*, *denial of service*, *spoofing, defense in depth*, *elevation of privilege*, *security feature bypass*, *information disclosure*, *xss*, *memory leak*, *sql-injection*, *zero-day* e *proof-of-concepts*.
- **Audiência no Twitter (*audience*):** estimativa de quantos usuários do Twitter falaram sobre uma determinada vulnerabilidade nas últimas 24h. Esse valor é calculado como a soma de todos os seguidores de cada usuário que tweetou ou retweetou sobre uma dada vulnerabilidade. Os valores são normalizados entre zero (0) e um (1), com valores próximos a zero indicando que houveram poucas menções e próximos a um indicando muitas menções;
- **Tendência no Google (*google_trend*):** esse campo indica se existe uma tendência de que as buscas por determinada vulnerabilidade vão diminuir, permanecer como estão ou crescer nos próximos dias. Os possíveis valores são: *decreasing*, *steady* ou *increasing*.
- **Interesse no Google (*google_interest*):** indica o interesse dos usuários do Google acerca

de uma determinada vulnerabilidade. Esse valor é calculado a partir do número de buscas pela vulnerabilidade realizadas nos últimos 7 dias. Assume valores entre zero (0) e um (1), com zero indicando que no período não houve nenhuma busca e um indicando que nesse período houveram muitas buscas;

– **Data de publicação do Boletim de Segurança (*advisory_published_date*):** data de publicação do boletim de segurança sobre a vulnerabilidade, no formato DD/MM/YYYY. Um boletim de segurança é um alerta emitido pelo fabricante do produto vulnerável informando que tal vulnerabilidade é crítica e que ela deve ser corrigida quanto antes.

A.2.3 Características de Contexto

Esse grupo fornece informações específicas sobre o ativo vulnerável. Essas informações consideram a rede e o ativo no qual a vulnerabilidade foi detectada. Portanto, elas mudam dependendo da organização e do ambiente.

– **Topologia (*topology*):** indica onde na rede está o ativo, que pode ser na rede local ou na *Demilitarized Zone* (DMZ). As redes locais são mais seguras porque são protegidas por *firewalls* robustos. Além disso, a comunicação com ativos dessas redes só é possível a partir da mesma rede ou por meio de *Virtual Private Network* (VPN). Os ativos na DMZ são mais críticos porque geralmente são visíveis a partir da internet, o que permite que todos os usuários, inclusive os mal-intencionados, tenham acesso direto a eles. Os possíveis valores são: LOCAL ou DMZ;

– **Tipo do Ativo (*asset_type*):** indica a finalidade do ativo, que pode ser uma estação de trabalho ou um servidor. Os servidores geralmente armazenam sistemas e bancos de dados críticos, que possuem informações confidenciais, tornando-os alvos potenciais de ataques cibernéticos. Os possíveis valores são: WORKSTATION ou SERVER ;

– **Ambiente (*environment*):** indica o ambiente onde o ativo está localizado e pode ser de desenvolvimento ou produção. Os ativos de desenvolvimento geralmente estão em redes segregadas, atrás de *firewalls* e não possuem dados confidenciais. Assim, eles são menos críticos do que os ativos de produção, que normalmente contêm informações confidenciais e estão acessíveis a mais pessoas. Os possíveis valores são: DEVELOPMENT ou PRODUCTION;

– **Dados Sensíveis (*sensitive_data*):** indica se o tipo de dado armazenado no ativo é sensível ou não. Servidores que armazenam informações confidenciais de usuários, como nome

completo e números de cartão de crédito, são mais críticos. Vale destacar que estações de trabalho também podem armazenar informações sensíveis. Os possíveis valores são: zero (0) quando o ativo não armazena dados sensíveis ou um (1) caso contrário;

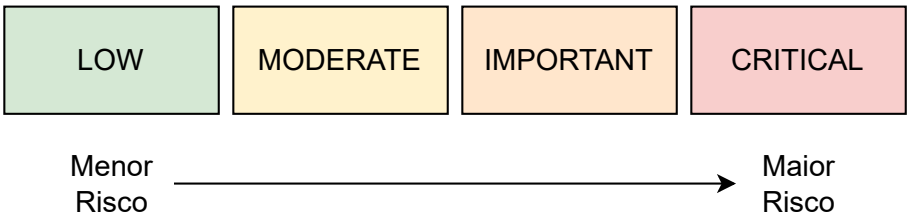
- **Fim da vida (*end_of_life*):** indica se o ativo em questão chegou ao fim de sua vida útil, ou seja, se continuará recebendo ou não atualizações de segurança. Esse atributo é importante, por ser comum existirem ativos legados com *softwares* ou sistemas operacionais que não recebem mais atualizações. Esses ativos são críticos e devem receber atenção especial por suas vulnerabilidades serem mais difíceis de corrigir. Os possíveis valores são: zero (0) quando o ativo não está no fim de sua vida útil ou um (1) caso contrário;
- **Ativo crítico (*critical_asset*):** indica se o ativo é crítico para a organização e pode ser um alvo potencial de invasores. *Active Directories* (ferramenta que permite o controle de *softwares*, arquivos e contas de usuários na rede das organizações), cofres de senhas ou computadores pessoais de executivos são exemplos de ativos críticos que devem receber atenção especial. Os possíveis valores são: zero (0) quando não for um ativo sensível e um (1) caso contrário.

OBS: O valor nulo (representado no teste como *nan*) indica que não foi possível recuperar informação a respeito do atributo. Por exemplo, se uma vulnerabilidade apresenta um valor de *nan* para o atributo "quantidade de *exploits*", não podemos afirmar que a vulnerabilidade não possui exploits. Podemos dizer apenas que, nenhum *exploit* foi descoberto e tornado público. Isso vale para qualquer um dos atributos. Dessa forma, o analista deve avaliar o risco considerando apenas os valores existentes.

A.3 Rótulos

As vulnerabilidades devem ser classificadas em uma das quatros diferentes classes de risco, que são: baixo (LOW), moderado (MODERATE), importante (IMPORTANT) e crítico (CRITICAL). O analista deve considerar que uma vulnerabilidade classificada como crítica possui um risco maior que uma vulnerabilidade importante, que por sua vez possui um risco maior que uma vulnerabilidade moderada e assim por diante. A Figura 1, a seguir, mostra a distribuição dos rótulos de acordo com seu risco.

Figura 1 – As vulnerabilidades, podem ser classificadas em 4 classes, que são: LOW, MODERATE, IMPORTANT e CRITICAL. Sendo a classe LOW atribuída as vulnerabilidades com os menores riscos e a CRITICAL aquelas com os maiores riscos.



Fonte: elaborado pelo autor.

OBS: o risco, no contexto dessa pesquisa, indica a probabilidade que uma vulnerabilidade tem de ser explorada e o dano que sua exploração pode causar para a organização. Dessa forma, uma vulnerabilidade crítica terá uma probabilidade alta de ser explorada (e.g., dada a existência de campanhas de malware em andamento ou a presença de exploits públicos) e trará sérios riscos para empresa (e.g., tal vulnerabilidade aparece em um ativo no "fim da vida", que armazena informações sensíveis de usuários). Enquanto que uma vulnerabilidade baixa será difícil de ser explorada (e.g., ela pode está presente em um ativo presente na rede local protegido atrás de um *firewall*) e caso seja explorada, não trará grandes risco para organização (e.g., o ativo em questão é uma estação de trabalho que não armazena dados sensíveis).

OBS: Como esse modelo considera informações além do CVSS, não é impossível que uma vulnerabilidade que possua severidade alta, segundo o CVSS, possa ser considerada crítica em relação ao seu risco. Basta que ela possua informações de inteligência de ameaças e contexto, que a torne crítica.

A seguir um breve direcionamento sobre como as vulnerabilidades podem ser classificadas entre os diferentes rótulos:

CRITICAL

Vulnerabilidades que afetam dispositivos críticos, facilmente exploradas (e.g., dada a presença de exploits), e podem causar grande impacto na confiabilidade, integridade e disponibilidade dos ativos vulneráveis, permitindo possivelmente o acesso a informações sensíveis das organizações. A seguir, **sugestões de características** que podem ser utilizadas para classificar uma vulnerabilidade como crítica:

– CVSS 9,0 ou superior;

- Valores HIGH para CIA;
- Aparecem nas listas do top 25 Mitre e/ou top 10 OWASP;
- Possuem *exploits*, EPSS maior que ou igual a 0,9 e/ou receberam destaque nas redes sociais (i.e., Twitter e/ou Google);
- Afetam ativos críticos (e.g., servidores que armazenam informações sensíveis);

IMPORTANT

Vulnerabilidades que podem afetar dispositivos críticos e que causam algum impacto na confiabilidade, integridade e disponibilidade desses ativos, permitindo ou não o acesso a informações sensíveis das organizações. A seguir, **sugestões de características** que podem ser utilizadas para classificar uma vulnerabilidade como importante:

- CVSS 7,0 ou superior;
- Valores LOW ou HIGH para CIA;
- Podem aparecer nas listas do Mitre e/ou OWASP;
- Podem ou não possuir *exploits* e/ou terem recebido algum destaque nas redes sociais;
- Podem ou não afetar ativos críticos;

MODERATE

Vulnerabilidades que afetam dispositivos comuns, difíceis de serem exploradas e que não causam grande impacto para os sistemas da empresa, além de não possibilitarem o acesso a informações sensíveis. A seguir, **sugestões de características** que podem ser utilizadas para classificar uma vulnerabilidade como moderada:

- CVSS abaixo 6,9;
- Valores NONE ou LOW para CIA;
- Não aparecem em listas do Mitre e/ou OWASP;
- Não possuem *exploits* e nem receberam destaque nas redes sociais;
- Não afetam ativos críticos;

LOW

As demais vulnerabilidades que não se encaixam em nenhuma das classes acima.

OBS: É importante destacar que, como queremos capturar a experiência dos analistas de segurança, a rotulação deve ser feita da forma que o especialista achar mais adequada. As características das vulnerabilidades apresentadas acima para cada um dos rótulos é apenas uma sugestão.

APÊNDICE B – GUIA DE EXECUÇÃO

Na pasta existem dois executáveis que podem ser utilizados para realizar a rotulação online das vulnerabilidades. Os executáveis são compatíveis com o Linux e o Windows. Para execução do programa de rotulação no Windows, basta um duplo clique no programa. Já no Linux é necessário, primeiro, tornar o programa executável através do seguinte comando no terminal:

```
$ sudo chmod +x
```

Então, para rodá-lo, basta executar o seguinte comando, também no terminal:

```
$ ./al-online-labelling
```

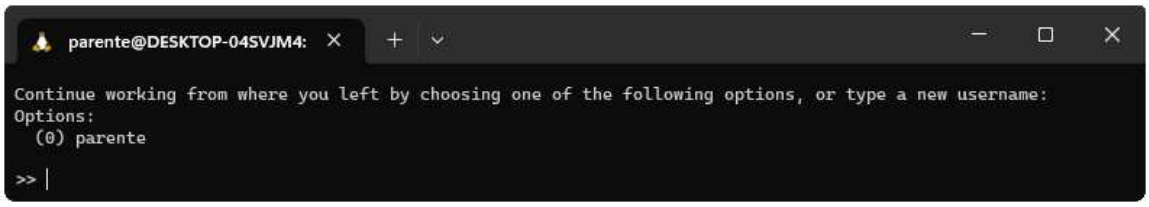
OBS: o programa demora alguns segundos para inicializar e irá criar uma pasta chamada de “AL_ONLINE_LABELLING_RESULTS_RParente” no mesmo local onde estiver sendo executado. Esta pasta conterá as instâncias rotuladas, os modelos e diversas informações estatísticas. Ao final do teste, quando o especialista tiver rotulado todas as vulnerabilidades, por gentileza, comprima a pasta e me enviei.

OBS: caso o programa apresente algum erro durante a execução, entre em contato comigo quanto antes, de modo que eu possa analisar e corrigir o problema.

APÊNDICE C – PROGRAMA DE ROTULAÇÃO ONLINE

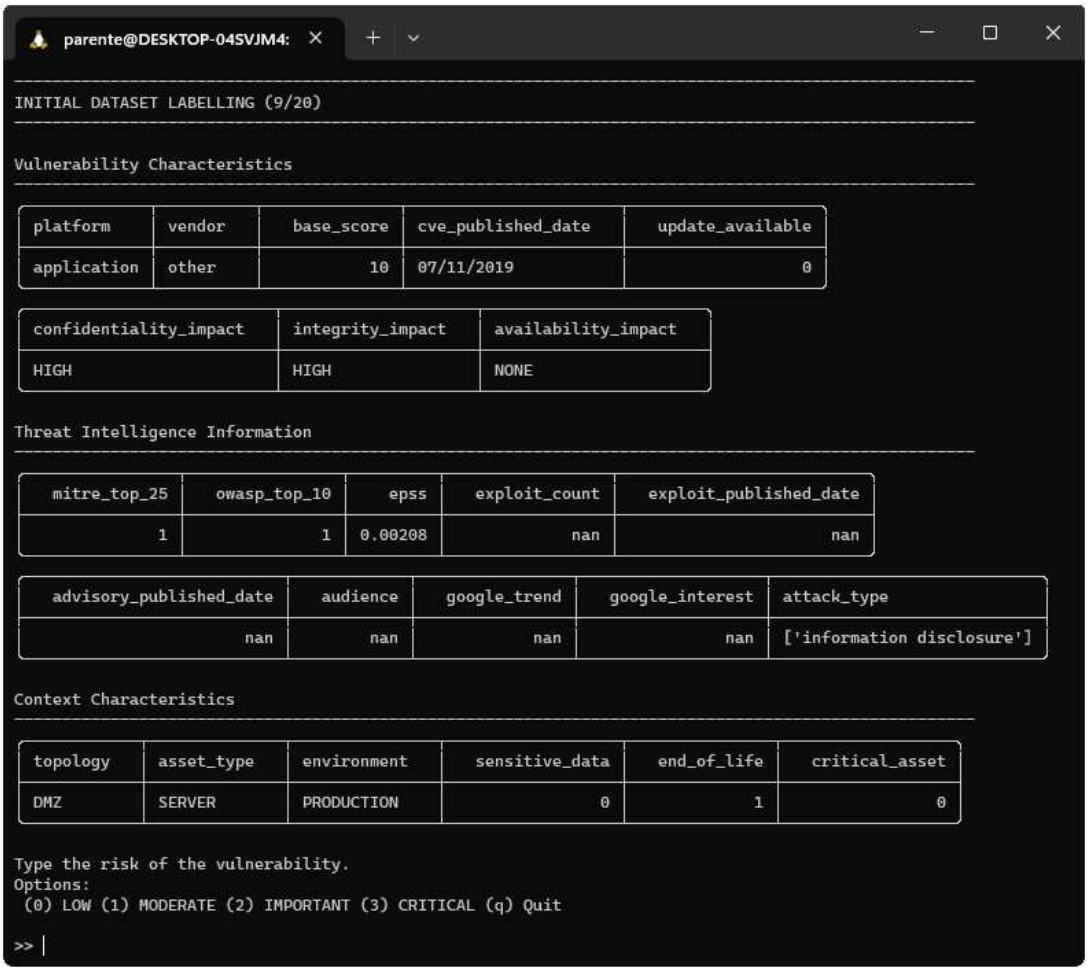
As imagens a seguir são do programa de rotulação online, desenvolvido utilizando a linguagem de programação Python. A Figura 1 mostra a tela de inicial do programa, onde o usuário deve digitar seu nome para começar o processo de rotulação das vulnerabilidades. A Figura 2 mostra um exemplo de instância que deve ser rotulada pelo especialista utilizando uma das quatro classes de risco.

Figura 1 – Tela inicial do programa de rotulação online de vulnerabilidades.



Fonte: elaborado pelo autor.

Figura 2 – Exemplo de instância que deve ser rotulada pelo especialista.



Fonte: elaborado pelo autor.