Descoberta de *Exploits* Usando Dados da Rede Social *Twitter*

Daniel Alves de Sousa

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação

Uberlândia
2020

Daniel Alves de Sousa

Descoberta de *Exploits* Usando Dados da Rede Social *Twitter*

Dissertação de mestrado apresentada ao Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Ciência da Computação

Orientador: Rodrigo Sanches Miani
Coorientador: Elaine Ribeiro de Faria Paiva

Uberlândia
2020

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Ciência da Computação
Av. João Naves de Ávila, nº 2121, Bloco 1A, Sala 243 - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4470 - www.ppgco.facom.ufu.br - cpgfacom@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Ciência da Computação				
Defesa de:	Mestrado Acadêmico, 27/2020, PPGCO				
Data:	28 de agosto de 2020	Hora de início:	**09h01min**	Hora de encerramento:	**11h45min**
Matrícula do Discente:	11812CCP012				
Nome do Discente:	Daniel Alves de Sousa				
Título do Trabalho:	Descoberta de *exploits* usando dados da rede social *Twitter*				
Área de concentração:	Ciência da Computação				
Linha de pesquisa:	Sistemas de Computação				
Projeto de Pesquisa de vinculação:	-				

Reuniu-se, por videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Ciência da Computação, assim composta: Professores Doutores; Rafael Pasquini - FACOM/UFU; Sylvio Barbon Júnior - DC/UEL; Elaine Ribeiro de Faria Paiva - FACOM/UFU (coorientadora) e Rodrigo Sanches Miani - FACOM/UFU, orientador do candidato.

Os examinadores participaram desde as seguintes localidades: Sylvio Barbon Júnior - Londrina - PR; Rafael Pasquini, Elaine Ribeiro de Faria Paiva e Rodrigo Sanches Miani - Uberlândia-MG. O discente participou da cidade de Uberlândia-MG.

Iniciando os trabalhos o presidente da mesa, Prof. Dr. Rodrigo Sanches Miani, apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao Discente a palavra para a exposição do seu trabalho. A duração da apresentação do Discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

Aprovado

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre.

O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do

Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Elaine Ribeiro de Faria Paiva**, **Professor(a) do Magistério Superior**, em 01/09/2020, às 10:08, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rodrigo Sanches Miani**, **Professor(a) do Magistério Superior**, em 01/09/2020, às 10:52, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Rafael Pasquini**, **Professor(a) do Magistério Superior**, em 01/09/2020, às 17:58, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Sylvio Barbon Junior**, **Usuário Externo**, em 11/09/2020, às 08:29, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **2224109** e o código CRC **DF8C3AE1**.

Referência: Processo nº 23117.050317/2020-37 SEI nº 2224109

Ficha Catalográfica Online do Sistema de Bibliotecas da UFU
com dados informados pelo(a) próprio(a) autor(a).

S725 Sousa, Daniel Alves de, 1990-
2020 Descoberta de Exploits Usando Dados da Rede Social Twitter [recurso eletrônico] / Daniel Alves de Sousa. - 2020.

Orientador: Rodrigo Sanches Miani.
Coorientadora: Elaine Ribeiro de Faria Paiva.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Pós-graduação em Ciência da Computação.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2020.657
Inclui bibliografia.
Inclui ilustrações.

1. Computação. I. Miani, Rodrigo Sanches,1983-, (Orient.). II. Paiva, Elaine Ribeiro de Faria,1980-, (Coorient.). III. Universidade Federal de Uberlândia. Pós-graduação em Ciência da Computação. IV. Título.

CDU: 681.3

Bibliotecários responsáveis pela estrutura de acordo com o AACR2:

Gizele Cristine Nunes do Couto - CRB6/2091

Agradecimentos

Agradeço a Deus, autor da minha vida, a quem devo tudo o que sou.

Agradeço à minha esposa, Camila, por todo o amor e compreensão. Sua companhia torna meus dias melhores e me ajuda a superar desafios. O apoio e carinho demonstrados a todo tempo foram fundamentais para o sucesso deste trabalho.

Agradeço aos meus pais, por serem exemplos de caráter e humanidade. O amor, cuidado e sacrifício empenhados no ensino, meu e de minhas irmãs, nos deram valores e educação que o mundo jamais poderá tirar. A vocês eu devo toda a minha formação.

Agradeço ao professor Rodrigo Sanches Miani, meu orientador, e a professora Elaine Ribeiro de Faria Paiva, minha coorientadora. O apoio e dedicação exemplar demonstrados na minha orientação aumentam ainda mais o respeito e admiração que tenho pelo trabalho de vocês.

"O maior obstáculo a novas descobertas não é a ignorância; é a ilusão do saber"
(Daniel J. Boorstin)

Resumo

No gerenciamento de segurança da informação, um aspecto crucial é a instalação de correções para vulnerabilidades de softwares. O crescente número dessas vulnerabilidades, associado à necessidade de análise dos impactos de cada atualização, podem fazer com que administradores adiem atualizações e deixem seus sistemas vulneráveis por muito tempo. Além disso, estudos relacionados apontam que muitas vulnerabilidades são exploradas apenas em provas de conceito, tornando a identificação de ameaças reais ainda mais difícil. Uma técnica que ajude a detectar quais vulnerabilidades possuem *exploits* no mundo real pode ser uma ferramenta poderosa para ajudar administradores de sistemas. Para agilizar essas detecções, o uso de aprendizado de máquina aplicado a discussões em redes sociais tem se mostrado promissor. Nesta dissertação são aplicadas técnicas de aprendizado de máquina a dados de discussões no Twitter e bases de dados públicas para determinar se uma vulnerabilidade foi ou não explorada. O trabalho também analisa o comportamento de diferentes algoritmos de classificação, investiga a influência do uso de rótulos verdadeiros extraídos de diferentes empresas de antivírus e experimenta com treino em vários tamanhos de janelas temporais. As descobertas deste trabalho sugerem que o uso do *ensemble Light Gradient Boosting Machine* (*LightGBM*) e do algoritmo de balanceamento de classes *All k-Nearest-Neighbor* (*AllKNN*) pode beneficiar os resultados em termos de *F-score* e precisão. O trabalho ainda demonstra como o uso de rótulos extraídos de uma única empresa de antivírus pode enviesar o modelo.

Palavras-chave: Segurança da Informação. Aprendizado de máquina. Vulnerabilidades de software. Ameaças a computadores. Exploits. Redes sociais. Antivírus.

Abstract

One crucial aspect of information systems security is the deployment of security patches. The growing number of software vulnerabilities, together with the need for impact analysis in each update, can cause administrators to postpone software patching and leave their systems vulnerable for a long time. Furthermore, studies have shown that many software vulnerabilities have only proof-of-concept exploits, making the identification of real threads even harder. In this scenario, knowledge of which vulnerabilities were exploited in the wild is a powerful tool to help systems administrators prioritize patches. Social media analysis for this specific application can enhance the results and bring more agility by collecting data from online discussions and applying machine learning techniques to detect real-world exploits. In this dissertation, we use a technique that combines Twitter data with public database information to classify vulnerabilities as exploited or not-exploited. We analyze the behavior of different classifying algorithms, investigate the influence of different antivirus data as ground truth, and experiment with various time window sizes. Our findings suggest that using a Light Gradient Boosting Machine (LightGBM) can benefit the results, and for most cases, the statistics related to a tweet and the users who tweeted are more meaningful than the text tweeted. We also demonstrate the importance of using ground-truth data from security companies not mentioned in previous works.

Keywords: Computer security. Machine learning. Software vulnerability. Computer threats. Exploits. Antivirus.

Lista de ilustrações

Lista de tabelas

Lista de siglas

ADASYN *Adaptive Synthetic*

AllKNN *All k-Nearest-Neighbor*

API *Application Programming Interface*

ASCII *American Standard Code for Information Interchange*

BoW *Bag-of-Words*

CVE *Common Vulnerabilities and Exposures*

CVSS *Common Vulnerability Scoring System*

CNA *CVE Numbering Authorities*

DoS *Denial Of Service*

DHS *U.S. Department of Homeland Security*

EDB *Exploit Database*

GOSS *Gradient-based One-Side Sampling*

IDS *Intrusion Detection System*

IoT *Internet of Things*

IPS *Intrusion Prevention System*

JSON JavaScript Object Notation

KNN *K-Nearest Neighbors*

LightGBM *Light Gradient Boosting Machine*

NB *Naive Bayes*

NER *Named Entity Recognize*

NIST *National Institute of Standards and Technology*

NVD *National Vulnerability Database*

OSVDB *Open Source Vulnerability Database*

PLN Processamento de Linguagem Natural

PoC *Proof of Concept*

POS *part-of-speech*

RF *Random Forest*

RL Regressão Logística

ROC *Receiver Operating Characteristic Curve*

RUS *Random Under Sampler*

SMOTE *Synthetic Minority Over-sampling Technique*

SVM *Support Vector Machine*

XML *Extensible Markup Language*

XGBoost *eXtreme Gradient Boosting*

ZDI *Zero Day Initiative*

Sumário

Capítulo 1

Introdução

No contexto da segurança de computadores, vulnerabilidades são falhas ou fraquezas de softwares que permitem a agentes maliciosos subverterem o uso originalmente desejado de algum sistema computacional e violarem politicas de segurança (SHIREY, 2007). Para explorar uma vulnerabilidade, é necessário que o atacante utilize um método ou ferramenta que aproveite-se da falha para alcançar um objetivo (como acesso não autorizado, execução de código malicioso, etc). A tal ferramenta dá-se o nome de *exploit* (WHITMAN; MATTORD, 2011).

De maneira geral, uma vulnerabilidade possui ciclo de vida conforme a Figura 1, onde a partir da sua descoberta inicia-se uma condição de corrida entre desenvolvedores/usuários e atacantes. Por um lado, os atacantes desejam realizar ataques antes que os usuários tenham a chance de instalarem correções, por outro, usuários precisam que desenvolvedores criem *patches* de correção para que possam instalá-los e não mais estarem vulneráveis.

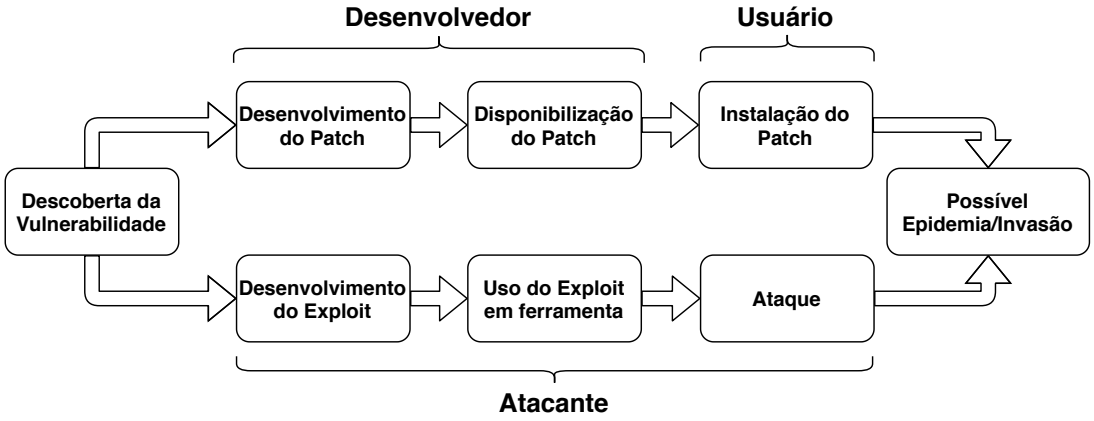


Figura 1: Ciclo de vida de uma vulnerabilidade

Fonte: Adaptado de Xiao et al. (2018, p. 3)

Existem diversas frentes nas quais desenvolvedores e a comunidade de segurança colaboram a fim de protegerem usuários de ataques relacionados a vulnerabilidades. A maior parte dessas frentes tem como objetivo encontrar formas de acelerar etapas do

lado desenvolvedor/usuário no ciclo de vida mostrado na Figura 1. Contudo, é importante questionar o que acontece quando o volume de vulnerabilidades descobertas se torna grande demais para desenvolvedores e usuários tratarem todas elas rapidamente. Nesse cenário, é preciso priorizar algumas vulnerabilidades em detrimento de outras se valendo do fato de que elas apresentam diferentes níveis de riscos.

Um fator que dificulta a descoberta das vulnerabilidades exploradas é que elas formam um pequeno grupo que precisa ser encontrado em meio a um grupo muito maior de vulnerabilidades não exploradas. Bozorgi et al. (2010) mostram que pouco mais de 65% das falhas de segurança em softwares conhecidas entre 1991 e 2007 possuem *exploits* reportados e Nayak et al. (2014) mostram que essa proporção vem diminuindo com o tempo. Além disso, Sabottke, Suciu e Dumitras (2015) apontam que existem *exploits* que são utilizados apenas como prova de conceito (PoC, do inglês *Proof of Concept*) mas não oferecem risco provável ao usuário, uma vez que seu uso é impraticável a atacantes por algum motivo. Levando isso em conta, esse último trabalho encontrou evidências de que apenas 1,3% das vulnerabilidades catalogadas em 2014 foram exploradas no mundo real.

O objetivo desta dissertação é utilizar métodos de aprendizado de máquina para obter informações sobre vulnerabilidades já divulgadas, procurando identificar quais delas possuem maior probabilidade de serem exploradas. Mais especificamente, deseja-se extrair conhecimento de discussões realizadas no *Twitter* sobre o assunto a fim de superar o desempenho dos classificadores que apoiam-se apenas em informações sobre a severidade de uma vulnerabilidade.

O *Twitter*[1] é utilizado por diversas comunidades para a discussão de diferentes tipos de assuntos e segurança de computadores não é uma exceção. Desenvolvedores, analistas de segurança e *hackers* compartilham informações que podem ser úteis para classificar uma vulnerabilidade como explorada ou não explorada. Dois exemplos de vulnerabilidades cujas formas de se explorar foram divulgadas em redes sociais estão na Figura 2, extraída de matérias do site BleepingComputer.com. No primeiro exemplo, a empresa *Zerodium*[2] utiliza seu perfil no Twitter para demonstrar como explorar uma vulnerabilidade, até então desconhecida, poderia ser explorada. No segundo exemplo, um pesquisador da *McAfee Labs*[3] demonstra, em sua conta pessoal do *Twitter*, o funcionamento de um *exploit* para uma vulnerabilidade do *Windows*.

Tendo em vista a existência de conteúdo relevante para segurança de computadores no *Twitter*, diversos outros trabalhos abordam a análise destes conteúdos por meio do aprendizado de máquina. Muitos trabalhos, porém, abordam aspectos preditivos de eventos relacionados à segurança, como o vazamento de informações de uma grande empresa ou surgimento de alguma epidemia de *malwares*. Exemplos dessa abordagem são: Khandpur et al. (2017), Sceller et al. (2017) e Ritter et al. (2015).

[1] https://twitter.com/
[2] https://zerodium.com/
[3] https://www.mcafee.com/enterprise/pt-br/threat-center/mcafee-labs.html

Exploit Affecting Tor Browser Burned In A Tweet

By Ionut Ilascu — September 10, 2018 — 06:35 PM — 0

Zerodium
@Zerodium

Advisory: Tor Browser 7.x has a serious vuln/bugdoor leading to full bypass of Tor / NoScript 'Safest' security level (supposed to block all JS).
PoC: Set the Content-Type of your html/js page to "text/html;/json" and enjoy full JS pwnage. Newly released Tor 8.x is Not affected.
9:23 AM - Sep 10, 2018
1,128 997 people are talking about this

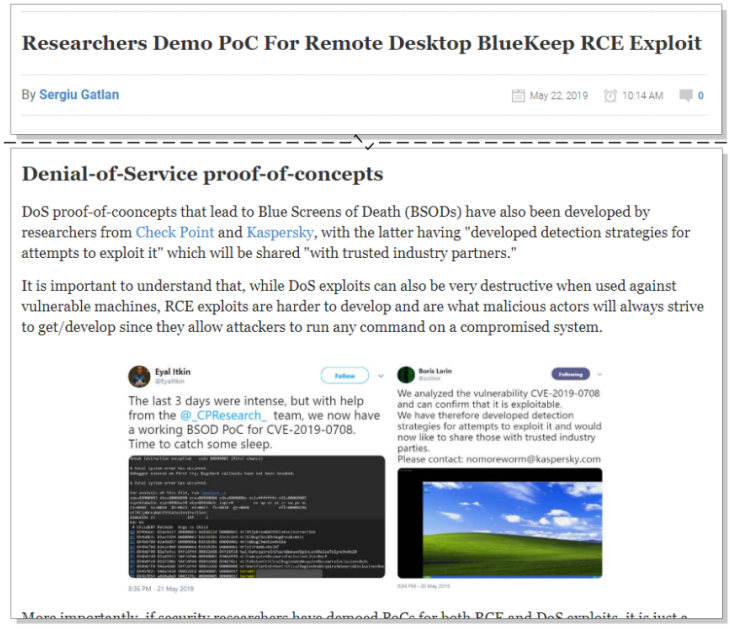
An exploit for a vulnerability in Tor Browser was delivered today in a tweet that left sufficient room for comments. A security vulnerabilities broker disclosed the details because it no longer served its purpose.

The exploit was part of Zerodium's portfolio and worked for Tor Browser 7.x. It existed in the NoScript component, which is a browser add-on that stops web pages from executing JavaScript, Flash, Java or Silverlight.

An exploit that one can only assume Zerodium paid good money for, is just a matter of setting the Content-Type of the attacker's HTML/JS page, or a hidden service in the Tor network, to "text/html/json," to suppress any reaction from NoScript and permit all JavaScript code through.

The bug worked when the user configured NoScript to block out all JavaScript by selecting the add-on's "Safest" security level.

(a) Divulgação de uma nova vulnerabilidade no *Twitter*
Fonte: Bleeping Computer (2018)



Researchers Demo PoC For Remote Desktop BlueKeep RCE Exploit

By Sergiu Gatlan — May 22, 2019 — 10:14 AM — 0

Denial-of-Service proof-of-concepts

DoS proof-of-cooncepts that lead to Blue Screens of Death (BSODs) have also been developed by researchers from Check Point and Kaspersky, with the latter having "developed detection strategies for attempts to exploit it" which will be shared "with trusted industry partners."

It is important to understand that, while DoS exploits can also be very destructive when used against vulnerable machines, RCE exploits are harder to develop and are what malicious actors will always strive to get/develop since they allow attackers to run any command on a compromised system.

Eyal Itkin
The last 3 days were intense, but with help from the @_CPResearch_ team, we now have a working BSOD PoC for CVE-2019-0708. Time to catch some sleep.

Boris Larin
We analyzed the vulnerability CVE-2019-0708 and can confirm that it is exploitable. We have therefore developed detection strategies for attempts to exploit it and would now like to share those with trusted industry parties.
Please contact: nomoreworm@kaspersky.com

(b) Divulgação de um novo *exploit* no *Twitter*
Fonte: Bleeping Computer (2019)

Figura 2: Notícias sobre a divulgação de vulnerabilidade e *exploit* no *Twitter*

Outros trabalhos, apesar de demonstrarem o potencial de uso do *Twitter* para a detecção de *exploits*, ainda possuem algumas limitações:

- Bullough et al. (2017) e Queiroz, Keegan e Mtenzi (2017) não realizam nenhuma discussão quanto à escolha do algoritmo de classificação. Chen et al. (2019) resolve parcialmente esse problema, utilizando vários algoritmos, no entanto não fornecem comparação de desempenho com o conjunto de dados original de Sabottke, Suciu e Dumitras (2015);

- Enquanto Sabottke, Suciu e Dumitras (2015) encontra apenas 1,3% de vulnerabilidades exploradas no mundo real, Bullough et al. (2017) realiza comparações com esse estudo utilizando um conjunto de dados sintético, criado por meio da manipulação de uma base com 17% de vulnerabilidades exploradas em provas de conceito;

- Almukaynizi et al. (2017),Chen et al. (2019) e Khandpur et al. (2017) abordam a detecção de exploits no mundo real, porém todos utilizam apenas informações extraídas das assinaturas de antivírus e *Intrusion Prevention System* (IPS) da *Symantec* para criar rótulos. Como discutido em Sabottke, Suciu e Dumitras (2015), essa prática impõem limitações, já que os produtos da *Symantec* não cobrem todas as plataformas e produtos.

Dessa forma, este trabalho visa contribuir com a detecção de vulnerabilidades exploradas a partir de dados do *Twitter* ao analisar um número superior de bases de dados para obter rótulos corretos, comparar o comportamento de diferentes algoritmos nesta aplicação e utilizar diferentes tamanhos e tipos de janelas de tempo para treino do classificador. As contribuições são detalhadas na Seção 1.4.

1.1 Motivação

O número de vulnerabilidades de softwares descobertas cresceu significativamente desde 2016 (CVE Details, 2020). Só em 2019, foram divulgadas 12.174 falhas de segurança em software. É razoável presumir que este crescimento se deve ao crescente número de processos e dispositivos informatizados, especialmente com o advento do *Internet of Things* (IoT). Independentemente da razão, percebe-se que o problema de detectar quais vulnerabilidades têm maior potencial de serem exploradas tem se tornado, por consequência, cada vez mais difícil.

Uma forma a qual os fabricantes de softwares recorrem para proteger seus produtos de *exploits* é lançar mão de um processo coordenado de procura e divulgação de vulnerabilidades. Empresas como a Microsoft e Google mantêm programas de busca de vulnerabilidades, tanto por equipes próprias como por equipes motivadas por recompensas (Microsoft Security Response Center, 2020)(Google Application Security, 2020), para que

tomem conhecimento de falhas em seus produtos antes que agentes maliciosos o façam. As vulnerabilidades encontradas são corrigidas por meio de *pathces* e, posteriormente, divulgadas ao público.

Mesmo utilizando essa técnica, a divulgação de vulnerabilidades por meios não oficiais ainda é bem frequente e muitas vezes é feita antes que se possa implementar uma correção. Além disso, programas coordenados introduzem um novo problema: o acúmulo de vulnerabilidades a serem divulgadas, uma vez que pacotes de atualizações costumam corrigir diversas falhas de uma só vez. Para entender por que isso pode ser um problema, é necessário ter em mente que, apesar da divulgação acontecer juntamente com as correções, só estará protegido aquele que instalar e utilizar tais atualizações, o que cria uma janela de oportunidade para atacantes. Para piorar, muitas vezes os próprios pacotes de correções são utilizados para descobrir como uma vulnerabilidade pode ser explorada por meio de engenharia reversa (COPPENS; SUTTER; BOSSCHERE, 2013).

Diante desse cenário, fica claro que a comunidade de segurança necessita de métodos para priorizar aquelas vulnerabilidades que devem ser corrigidas ou tratadas. Tais métodos interessa aos desenvolvedores, que muitas vezes se deparam com vulnerabilidades de diferentes severidades reveladas fora de seus programas coordenados. E interessa também aos consumidores, que precisam ponderar qual a real urgência de se instalar um pacote de correções que, apesar de corrigir falhas, pode introduzir um comportamento inadequado aos seus sistemas.

Soluções que adotam aprendizado de máquina para classificar vulnerabilidades entre "exploradas" e "não exploradas" podem ser um bom caminho para a automação dessa tarefa. Mas, se por um lado já existem indicadores de severidade que tentam nortear o tratamento de vulnerabilidades, trabalhos anteriores mostraram que tais indicadores são imprecisos ao tentarem detectar *exploits* do mundo real. Sobretudo, apontam que é possível ter um aumento de precisão ao aplicar técnicas de mineração de texto em discussões de redes sociais e fóruns (SABOTTKE; SUCIU; DUMITRAS, 2015) (ALMUKAYNIZI et al., 2017). Tais trabalhos, porém, falham ao utilizar bases de dados, ora muito restritas (de fabricantes e antivírus específicos) e ora muito abrangentes (trabalhando com possíveis *exploits* de vulnerabilidades ainda não catalogadas).

Portanto, a literatura aponta a existência de espaço para avanços nas análises realizadas em cima de vulnerabilidades já divulgadas, levando-se em conta informações de múltiplos fabricantes de software e banco de dados de diferentes antivírus ou *firewalls*. A intenção deste trabalho é exatamente colaborar para tais avanços.

1.2 Objetivos da Pesquisa

O objetivo geral deste trabalho é propor melhorias ao método de classificação que utiliza dados do *Twitter* para classificar vulnerabilidades de *software* em "exploradas"

ou "não exploradas". Tais melhorias envolvem a escolha de algoritmos de classificação, o balanceamento de classes e a inclusão de novas fontes de rótulos corretos para o problema. Espera-se que as alterações se traduzam em aumento de *F-score* e precisão do classificador em comparação à trabalhos anteriores. Para verificar a eficácia do método, foi construída uma solução que classifica vulnerabilidades conforme a existência de *exploits*, de mundo real e prova de conceito. Essa solução utiliza mineração de texto e outros métodos de aprendizado de máquina para analisar informações de bancos de dados de vulnerabilidades e extrair informações de textos no *Twitter*.

1.2.1 Objetivos específicos

- ❏ Criar um conjunto de dados que contemple um número maior de exploits de mundo real por meio do uso de múltiplos sites de antivírus como fontes de informação.
- ❏ Desenvolver uma solução que permita analisar informações de bases de dados de vulnerabilidades e discussões sobre o assunto no *Twitter* para classificar vulnerabilidades como "exploradas" ou "não exploradas" no mundo real. Tal solução é pensada para ser modular e permitir testes com diferentes algoritmos, bases de dados e conjuntos de rótulos corretos;
- ❏ Avaliar o desempenho de diferentes algoritmos de classificação quando executados com diferentes representações de vulnerabilidades. A partir destas análises, determinar o algoritmo mais adequado e as representações mais relevantes para a aplicação;
- ❏ Analisar como o desempenho do classificador é afetado ao longo de diferentes anos;
- ❏ Disponibilizar o conjunto de dados produzido por este trabalho, incluindo dados brutos e pré-processados de rótulos e do *Twitter*, a comunidade para futuros trabalhos.

1.3 Hipótese

A detecção de vulnerabilidades exploradas com o uso de dados do *Twitter* pode ser melhorada em *F-score* e precisão com o uso de novas fontes de rótulos e um modelo de classificação, diferente daqueles usados em trabalhos relacionados.

1.4 Contribuições

Este trabalho possui quatro contribuições principais:

- ❏ A identificação de algoritmos que melhoram o desempenho na detecção de *exploits* usando dados do *Twitter*;

- O desenvolvimento de uma solução que utiliza como rótulos, para *exploits* do mundo real, informações de outros fabricantes de antivírus além da *Symantec*;
- Demonstrar que o uso de rótulos extraídos de um único fabricante de antivírus pode enviesar os resultados e afetar negativamente a performance do classificador em um cenário real;
- Examinar como os resultados do classificador se modificam ao se considerar diferentes janelas temporais.

1.5 Organização da Dissertação

Esta dissertação está organizada da seguinte forma: o Capítulo 2 trata da fundamentação teórica, o Capítulo 3 apresenta os principais trabalhos relacionados, o Capítulo 4 traz a descrição da proposta, o Capítulo 5 apresenta os experimentos, resultados e suas análises, e por fim o Capítulo 6 conclui esta monografia com um resumo do trabalho realizado e sugestões de trabalhos futuros.

Capítulo 2

Fundamentação Teórica

Neste capítulo serão apresentados os conceitos que fundamentam esta dissertação. Eles podem ser divididos em três áreas, segurança de computadores, mineração de texto e processamento de linguagem natural, e aprendizado de máquina. Para cada uma das áreas foi dedicada uma seção.

2.1 Segurança de computadores

O manual do *National Institute of Standards and Technology* (NIST) define o termo "segurança de computadores" como: *"a proteção oferecida para um sistema de informação automatizado a fim de se alcançar os objetivos de preservar a integridade, a disponibilidade e a confidencialidade dos recursos do sistema de informação"* (GUTTMAN; ROBACK, 1995, p.5, tradução do autor). Neste contexto, existem três conceitos de fundamental importância:

- **Integridade** – Ainda de acordo com a definição do NIST, em segurança de computadores, integridade pode ter duas facetas: a *integridade dos dados*, que garante que informações e programas só serão alterados de maneira especificamente autorizada, e a *integridade dos sistemas*, que garante que sistemas trabalhem de maneira intacta, livres de manipulações (intencionais ou não).
- **Disponibilidade** – É a característica que garante que sistemas respondam prontamente e não neguem acesso a usuários autorizados.
- **Confidencialidade** – Característica que garante que dados sigilosos ou privados não serão acessados por pessoas não autorizadas.

Outros dois conceitos frequentemente associados ao tema Segurança de Computadores são: a *autenticidade*, que seria a característica de ser genuíno e verificável (como em uma troca de mensagens onde deve-se garantir que locutor e interlocutor são realmente quem dizem ser); e a *responsabilização*, que garante que as ações de quaisquer entidades dentro

de um sistema possam ser atribuídas aos seus autores, tornando possível rastrear a causa de determinados comportamentos (implementado, geralmente, na forma de registros de ações dos usuários).

2.1.1 Ameaças, vulnerabilidades e *exploits*

Considera-se como **ameaça** tudo que tem o potencial de violar a segurança em um sistema. Pode existir na forma de uma entidade, circunstância, capacidade, ação ou evento que possa causar danos (SHIREY, 2007). Considera-se como **vulnerabilidades** as falhas ou fraquezas de softwares que permitem a agentes maliciosos subverterem o uso originalmente desejado de algum sistema computacional e violarem políticas de segurança (SHIREY, 2007). Aos códigos maliciosos e técnicas que se aproveitam de vulnerabilidades para criar ameaças, dá-se o nome de ***exploits***.

Quanto a esses três conceitos, é importante perceber que ao explorar uma vulnerabilidade cria-se uma ameaça, porém nem toda ameaça é criada por uma vulnerabilidade em *software*. O roubo de senhas por meio de técnicas de *engenharia social*, por exemplo, apesar de constituir uma ameaça à segurança, não é causado por vulnerabilidades. Até mesmo softwares maliciosos podem não ser baseados em vulnerabilidades, sendo instalados por meios legítimos (mas geralmente não intencionais) e se comportando de maneira espúria.

A Figura 1, apresentada no Capítulo 1, ilustra o ciclo de vida de uma vulnerabilidade. Um conceito importante relacionado a esse ciclo é o de **vulnerabilidades de dia zero** ou apenas ***zero days***. Tal conceito se refere a falhas descobertas (por desenvolvedores ou hackers) mas ainda não divulgadas, e, portanto, sem correções implementadas (BILGE; DUMITRAS, 2012). As *zero days* podem ser muito valiosas (estratégica e financeiramente falando) por garantirem que qualquer usuário do sistema afetado estará vulnerável a um ataque que a explore.

Na comunidade de segurança, existem pesquisadores autônomos, empresas e até agências governamentais dedicadas a buscar *zero-days*. Suas descobertas podem ser vendidas para os fabricantes dos softwares afetados (HUANG et al., 2016) ou para empresas como a *Zero Day Initiative* (ZDI) (Zero Day Initiative, 2018) ou Zerodium (ZERODIUM, 2018), podem ser mantidas como reservas estratégicas (no caso de governos) (ABLON; BOGART, 2017) ou até mesmo negociados na *DeepWeb*(MARIN et al., 2018).

Para catalogar e acompanhar o histórico de cada vulnerabilidade, a *MITRE Corporation*, uma organização norte-americana de pesquisa que presta serviços ao governo, fundou em 1999 o *Common Vulnerabilities and Exposures* (CVE), um dicionário que tem o objetivo de trazer informações básicas de todas as vulnerabilidades de softwares divulgadas publicamente. O CVE é patrocinado por agências internas ao *U.S. Department of Homeland Security* (DHS) e administrado pela própria *MITRE*, que mantém o site, preside a mesa diretora (composta por diversas empresas, universidades e organizações de pes-

quisa) além de supervisionar as autoridades de numeração (*CVE Numbering Authorities* – CNA), das quais praticamente qualquer organização pode se candidatar a fazer parte (The MITRE Corporation, 2020).

O *National Vulnerability Database* (NVD), por outro lado, é o repositório do governo norte-americano para o gerenciamento de vulnerabilidades mantido pelo NIST, uma agência governamental que administra, entre outras coisas, padrões de engenharia e tecnologia. O NVD traz informações específicas relacionadas a cada vulnerabilidade como *checklists* de segurança, descrições de falhas e erros de configuração além de métricas de impacto (chamadas de *Common Vulnerability Scoring System* – CVSS) e histórico de correções. Por fazer uso do CVE e trazer algumas informações contidas no site dessa base, NVD e CVE são frequentemente confundidas em suas funções, que são na verdade complementares (NIST, 2020). A Figura 3 mostra a página, no NVD, da vulnerabilidade *BlueKeep* (CVE-2019-0708), uma falha grave que afeta o protocolo de área de trabalho remota em diversas versões do *Windows*. A página mostra informações como a descrição, links para referências e soluções, data de publicação/modificação e o vetor *Common Vulnerability Scoring System* (CVSS).

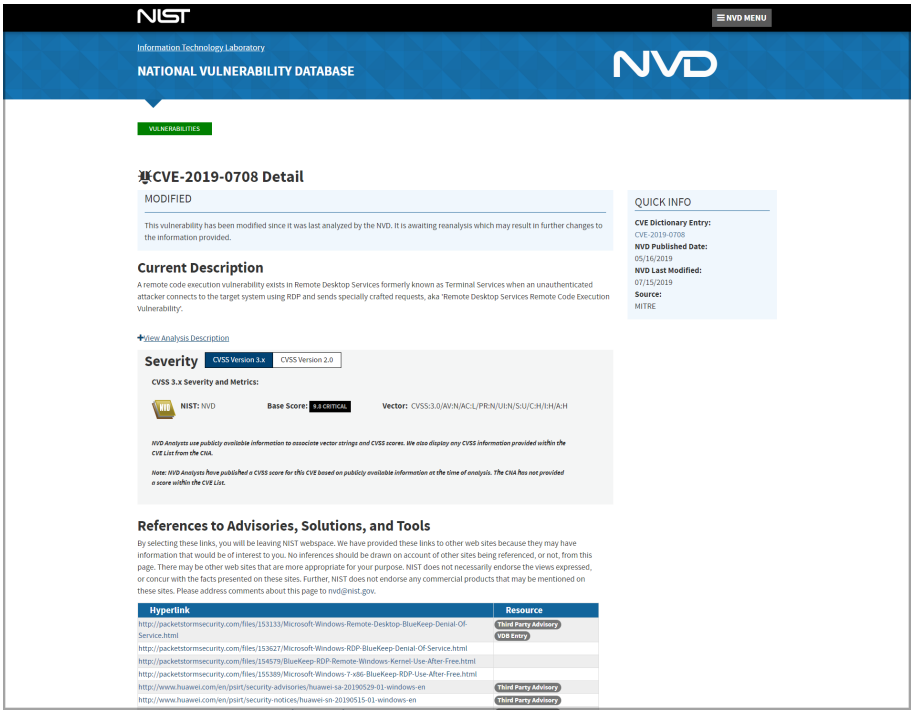


Figura 3: Página da vulnerabilidade BlueKeep no NVD

Fonte: NIST (2019)

A intenção desta dissertação é colaborar com a prevenção de ameaças detectando *exploits*, ou seja, apenas ameaças relacionadas a vulnerabilidades de *softwares* e não aquelas que envolvam engenharia social ou instalação involuntária, por exemplo. Também é objetivo trabalhar apenas com vulnerabilidades catalogadas (que já tenham um código CVE

atribuído) e não com *zero-days*. Por essa razão, ao longo desta dissertação é comum o uso do termo "código CVE" como sinônimo de uma vulnerabilidade. No contexto deste trabalho, o termo "detecção de *exploits*" refere-se somente à capacidade de detectar a existência de algum *exploit* para uma dada vulnerabilidade, e não a detecção do código malicioso ou ataque em si.

2.2 Mineração de texto e processamento de linguagem natural

A mineração de texto pode ser definida como o processo de descoberta e extração de conhecimento não trivial de textos livres e não estruturados, podendo ter objetivos variados como a obtenção de informações específicas, análise de sentimentos, classificação ou agrupamento de documentos (KAO; POTEET, 2007). Ao contrário da mineração de dados tradicional, que parte do princípio de que as informações estão armazenadas de forma estruturada, a mineração de texto possui um grande esforço de pré-processamento focado na extração de características pelo uso do Processamento de Linguagem Natural (PLN).

O PLN é o processo de se extrair representações estruturadas significativas de textos livres e não estruturados por meio da análise de linguagem natural. Tipicamente, o PLN utiliza conceitos linguísticos como classes gramaticais (ou POS, do inglês *part-of-speech*), estrutura gramatical e conjunto lexical para lidar com conceitos de difícil abstração para máquinas, como anáforas e ambiguidades (KAO; POTEET, 2007).

Dois conceitos chaves na mineração de texto são **documentos** e **coleções de documentos**. Um documento é uma unidade básica de informação textual que se deseja analisar. Uma página da web, um artigo científico ou um *tweet* (mensagem do *Twitter*) podem ser exemplos de documentos. Coleções de documentos (algumas vezes chamadas de corpus), como o termo sugere, é um conjunto de documentos dos quais se deseja extrair algum padrão ou relação. Ela pode ser estática, quando não são incluídos nem excluídos elementos ao longo da análise, ou podem ser dinâmicas caso contrário (FELDMAN; SANGER et al., 2007). A mineração de texto em redes sociais, quando feita em tempo real, trabalha com coleções dinâmicas de documentos, com postagens sendo geradas ao longo do tempo.

Para que se obtenha características a partir de documentos, é necessário a estruturação do texto utilizando alguns passos. A Figura 4 ilustra uma possível sequência desses passos para a classificação a partir de textos extraídos de redes sociais. É preciso destacar que o uso de cada procedimento depende da aplicação específica, em muitos casos também pode ser necessário passos adicionais (WEISS; INDURKHYA; ZHANG, 2015).

O processo se inicia com a *coleta de documentos*. Essa etapa é altamente dependente da natureza do documento utilizado. Páginas web, por exemplo, são frequentemente co-

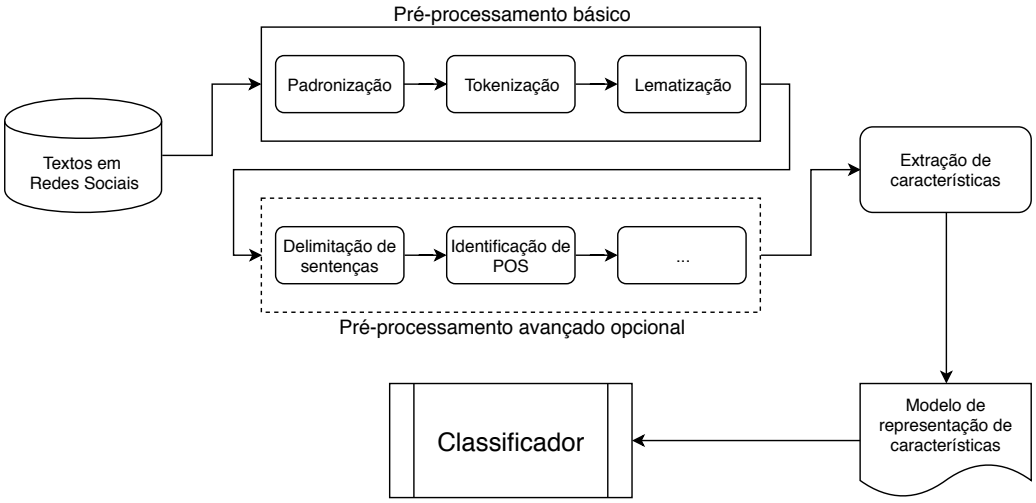


Figura 4: Sequência de passos para a aplicação de classificadores em textos

Fonte: O autor

letadas por algoritmos de *web scraping*, que simula a navegação de um usuário, filtrando e coletando os textos de interesse, outros documentos podem ser extraídos de banco de dados, arquivos de texto ou por meio de uma *Application Programming Interface* (API), como a utilizada para coletar mensagens do *Twitter*. A seguir, executa-se a *padronização de documentos*. Nessa etapa, documentos de diferentes tipos são convertidos para um único formato, como *Extensible Markup Language* (XML) ou texto puro no formato *American Standard Code for Information Interchange* (ASCII). Em seguida realiza-se a *tokenização* do documento que consiste em quebrar a cadeia de caracteres em diferentes palavras (ou *tokens*). A Lematização, por sua vez, substitui flexões de determinados *tokens* pelo seu lema (por exemplo tiver, tenho, tinha são flexões do lema "ter") (BERRY; KOGAN, 2010).

Existem diversos outros pré-processamentos que podem ser aplicados, porém os mais utilizados nos trabalhos relacionados a este são os mencionados. Quanto à representação dos documentos, a mais utilizada é também uma das mais simples, conhecida como *Bag-of-Words* (BoW). Neste modelo, documentos são representados por vetores onde cada posição corresponde a frequência (ou apenas presença/ausência) de uma determinada palavra do corpus. Para o algoritmo de aprendizado, esse vetor será também o vetor de características.

Um exemplo desse processo pode ser retirado do *tweet* mostrado na Figura 5. Após ser coletado por uma ferramenta específica para o Twitter (abordada no Capítulo 4), ele foi padronizado em formato JavaScript Object Notation (JSON). O processo de tokenização obteve os seguintes *tokens*: `"Just"`, `"saw"`, `"this"`, `"interesting"`, `"CVE-2017-8759"`, `"sample"`, `"using"`, `"xlsx"`, `"Nice"`, `"reflection"`, `"trick"`, `"to"`, `"stay"`, `"in"`, `"process"`, `"This"`, `"is"`, `"more"`, `"like"` e `"it"`. Em seguida, o processo

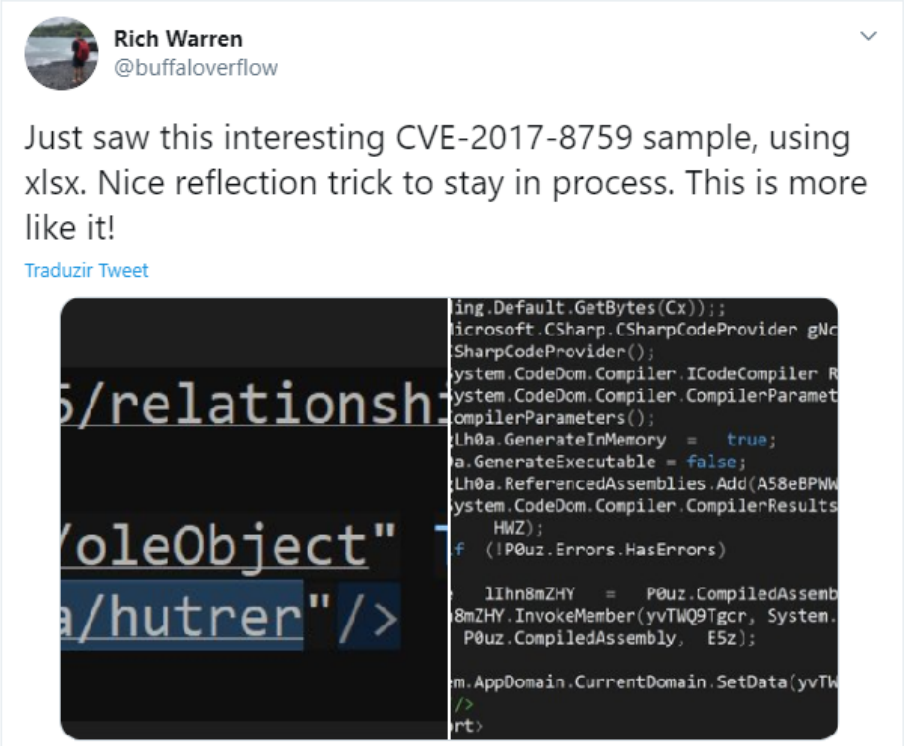


Figura 5: Exemplo de um *tweet* coletado neste trabalho

Fonte: Twitter (2017)

de lematização obteve os seguintes lemas: `"just"`, `"see"`, `"this"`, `"interesting"`, `"CVE-2017-8759"`, `"sample"`, `"use"`, `"xlsx"`, `"nice"`, `"reflection"`, `"trick"`, `"to"`, `"stay"`, `"in"`, `"process"`, `"this"`, `"be"`, `"more"`, `"like"`, e `"PRON"` se referindo a um pronome. Como a palavra "sample" é a única dentre os lemas que está presente na BoW usada, a representação deste texto é um vetor com o número 1 na posição referente a "sample" e 0 nas demais posições.

2.3 Aprendizado de máquina

O aprendizado de máquina é a técnica que permite a programas de computadores "aprenderem" utilizando somente a experiência acumulada (SILVA; ZHAO, 2016, p.71). De acordo com Mitchell (1997, p.2), é possível dizer que um programa de computador aprende com sua experiência se, para determinada tarefa, o desempenho desse programa melhora, de maneira mensurável, com sua execução (ou experiência). O aprendizado de máquina pode ser dividido em três tipos principais: **aprendizado não-supervisionado**, **aprendizado supervisionado** e **aprendizado semi-supervisionado** (SILVA; ZHAO, 2016, p.71).

O aprendizado não-supervisionado tem como objetivo principal procurar por estruturas de relacionamentos não explícitos entre dados. Dessa forma, esse tipo de algoritmo

não utiliza rótulos e encontra padrões tão somente com os dados estudados. Três tarefas comuns no aprendizado não-supervisionado são: agrupamento; detecção de *outliers*; e redução de dimensionalidade.

O aprendizado supervisionado tem por objetivo inferir conceitos a respeito de um conjunto de dados. Essa inferência é possível pelo uso de rótulos em um subgrupo de dados conhecidos como **conjunto de treino**. Em última análise, no aprendizado supervisionado, deseja-se obter um modelo capaz de mapear dados da entrada a rótulos. Para avaliar o desempenho do modelo encontrado, utiliza-se um subgrupo de dados, distinto daquele usado em treino, conhecido como **conjunto de teste**. Duas são as tarefas no aprendizado supervisionado: a **classificação**, quando os rótulos são valores discretos, e a **regressão**, quando os valores buscados são contínuos.

Por último, no aprendizado semi-supervisionado, os objetivos são basicamente os mesmos do supervisionado, porém apenas uma pequena parte dos dados possui rótulos. Portanto, o método utilizado para o modelo deve ser adaptado e frequentemente envolve a propagação de rótulos para a parte não rotulada dos dados, criando um conjunto de treino mais robusto. Esse tipo de abordagem é muito comum em cenários onde o volume de dados coletados é grande e a tarefa de rotular dados de treino pode não ser impossível.

Nesta dissertação, a tarefa realizada é a classificação de vulnerabilidades de *softwares* entre um dos dois rótulos: "exploradas" ou "não exploradas". Mais especificamente, deseja-se extrair do *Twitter* conteúdos sobre vulnerabilidades que, juntamente com informações do NVD, permitam criar representações mais completas dessas vulnerabilidades que serão submetidas ao classificador. A seguir, será detalhada a tarefa de classificação e serão apresentados os algoritmos utilizados nessa dissertação.

2.3.1 Classificação e Aprendizado Supervisionado

No aprendizado supervisionado, o conjunto de dados é composto por um conjunto de elementos $X = \{x_1, x_2, ..., x_n\}$ e de rótulos $Y = \{y_1, y_2, ..., y_n\}$, onde para qualquer valor i tal que $1 \leq i \leq n$, a tupla (x_i, y_i) corresponde a uma instância corretamente rotulada. A representação de um elemento x_i é chamada de *vetor de característica* e cada uma de suas m dimensões contêm um valor que descreve o elemento de alguma forma. Exemplo: em um problema cujos dados de entrada sejam pessoas, x_i representa uma pessoa e $x_i^{(1)}$ poderia conter sua altura, $x_i^{(2)}$ o seu peso, $x_i^{(3)}$ sua idade e assim por diante. Considerando o fato de que os vetores de características possuem todos o mesmo número de dimensões m, o conjunto X é frequentemente chamado de *matriz de características* e definido como $X \in \mathbb{R}^{n \times m}$. O rótulo y_i pode ser um elemento de um conjunto finito de classes $C = \{c_1, c_2, ..., c_k\}$, para a tarefa de classificação, ou, para regressões, valores contínuos ou estruturas mais complexa como vetores e matrizes (BURKOV, 2019, p. 3). Considerando o conjunto de classes C de tamanho k, diz-se que o problema é binário, caso k seja igual a 2 e que o problema é multiclasse, caso k seja maior que 2 (RASCHKA;

MIRJALILI, 2019, p. 42). No exemplo da classificação de pessoas, y_i poderia ser algo como o sexo ou a nacionalidade, enquanto para a tarefa de regressão, os rótulos poderiam indicar a probabilidade de uma pessoa ter determinada doença ou ainda a tupla de valores esperados para a pressão sanguínea daquela pessoa (sistólica,diastólica).

O objetivo da classificação é utilizar o conjunto de dados para produzir um modelo que mapeie um vetor de entrada x a um rótulo y. Para criar e avaliar tal modelo, os algoritmos de classificação dividem o conjunto de dados em dois subconjuntos, um de treino e outro de teste. O conjunto de treino é composto por X_{treino} e Y_{treino}, sobre os quais o algoritmo operará para encontrar uma regra de generalização do problema. O conjunto de teste é composto de X_{teste}, tal que $X_{treino} \cap X_{teste} = \varnothing$, e Y_{teste}, a princípio, desconhecido pelo algoritmo. Uma vez treinado sobre o conjunto de treino, o modelo é aplicado aos elementos do conjunto de teste e produzirá um conjunto de rótulos. A comparação desses resultados e os rótulos corretos Y_{teste} permitirão conferir o desempenho do modelo de generalização criado (SILVA; ZHAO, 2016, p.74).

2.3.2 Métodos de validação cruzada

Um aspecto crucial na utilização de modelos de aprendizado de máquina é ser capaz de estimar o desempenho do método escolhido. Para isso, frequentemente usa-se apenas parte do conjunto de dados para treinar o modelo e reserva-se uma outra parte para ser um conjunto de testes, ou seja, um conjunto de instâncias que serão submetidas ao classificador para avaliar qual a capacidade preditiva dele. No aprendizado supervisionado, os rótulos das instâncias de teste são conhecidos, mas não considerados durante o treino. Ao invés disso, eles só são utilizados para a aferição dos erros e acertos do modelo. Às diversas estratégias de divisão do conjunto de dados entre conjunto de treino e conjunto de teste, dá-se o nome de validação cruzada (RASCHKA; MIRJALILI, 2019). Nesta dissertação foram aplicados dois métodos de validação cruzada: o *k-fold cross-validation* e o *holdout*.

2.3.2.1 *Holdout*

O *holdout* é um método clássico e simples para se estimar o desempenho de um modelo de aprendizado de máquina. Nele o conjunto de dados é particionado uma única vez entre conjunto de treino e conjunto de testes, um único modelo é treinado e seu desempenho é obtido em uma única medição. Essa divisão pode ser feita aleatoriamente ou respeitando algum critério como a divisão temporal, dependendo da natureza dos dados. Os tamanhos das partições podem ser escolhidos conforme a disponibilidade dos dados, porém é comum que o conjunto de treino seja maior que o de testes, algo como 2/3 do total, para se maximizar o potencial de aprendizado dos modelos. Para conjuntos de dados

desbalanceados, é comum que se faça divisões estratificadas, na qual a proporção entre as classes do conjunto completo é mantida nas partições.

O método *holdout* possui, porém, algumas desvantagens. A primeira delas é o fato de ser muito sensível à forma como o conjunto de dados é particionado. Diferentes particionamentos podem levar a avaliações muito diferentes do mesmo classificador. O uso do *holdout* para seleção de modelos ou ajuste de parâmetros, também pode gerar modelos sobreajustados ao conjunto de teste (RASCHKA; MIRJALILI, 2019). Apesar disso, nesta dissertação, o método foi escolhido para avaliar os experimentos que envolvem janelas temporais, uma vez que a divisão temporal e o alto desbalanceamento de classes impossibilitariam outras divisões úteis ao aprendizado do classificador.

2.3.2.2 *K-fold cross-validation*

Neste método, o conjunto de dados é dividido aleatoriamente em k pastas independentes, onde $k-1$ pastas serão utilizadas para treino e uma será utilizada para a avaliação de desempenho. O processo se repete por k vezes, produzindo k modelos e k resultados. O desempenho total é obtido pela média dos desempenhos de cada modelo. Um exemplo de funcionamento do método para $k = 5$ pode ser dado a seguir (BURKOV, 2019):

1. Primeiro são escolhidos os valores para os hiperparâmetros do classificador;
2. Em seguida, divide-se o conjunto de dados em 5 subconjuntos (ou pastas): $F = \{f_1, f_2, f_3, f_4, f_5\}$;
3. Para cada pasta f_i, tal que $1 \leq i \leq 5$ treina-se um modelo M_i utilizando os hiperparâmetros definidos e o conjunto de instâncias $F - \{f_i\}$;
4. Mede-se o desempenho de M_i utilizando o subconjunto f_i;
5. Repete-se os passos de 3 e 4 para todo i tal que $1 \leq i \leq 5$;
6. Calcula-se a média dos desempenhos dos modelos M_1, M_2, M_3, M_4, M_5.

Ao avaliar a escolha dos hiperparâmetros em diferentes conjuntos de testes, o *k-fold cross-validation* garante uma melhor generalização dos dados. Assim como no *holdout*, as partições também podem ser estratificadas, mantendo as proporções de classes em todas as pastas. Neste trabalho foi utilizado o *10-fold cross-validation* com pastas estratificadas.

2.3.3 Medidas de desempenho para classificadores binários

Como mencionado, nesta dissertação, o objetivo principal é a classificação de vulnerabilidades em duas classes: vulnerabilidades exploradas e vulnerabilidades não exploradas. Trata-se, portanto, de um problema de classificação binário. Por essa razão, as medidas de desempenho de algoritmos tratadas aqui serão específicas a essa tarefa.

Antes de abordar as medidas de desempenho, é importante apresentar uma ferramenta utilizada para facilitar a visualização dos resultados, a **matriz de confusão**. Trata-se de uma tabela que relaciona os rótulos previstos pelo classificador com os rótulos corretos, mostrando erros e acertos para cada uma das classes (RASCHKA; MIRJALILI, 2019). Ao tratar de problemas binários, a matriz de confusão é geralmente criada com os rótulos *positivo* e *negativo*. É importante ressaltar que o princípio da tabela se aplica a qualquer par de rótulos, porém algumas medidas de desempenho apresentarão valores diferentes a depender da classe escolhida como instância *positiva* (vide Equações 1, 2 e 3). No caso desta dissertação o rótulo da vulnerabilidade "explorada" é a classe positiva, já que ela é a classe minoritária e de maior interesse.

A Figura 6 mostra a estrutura geral da matriz de confusão. Nela, as colunas representam as classes previstas e as linhas representam as classes reais. Nas intersecções entre linhas e colunas tem-se os quatro quadrantes:

- **Positivos verdadeiros ou *True positive* (TP)** – número de instâncias classificadas como positivas que realmente são positivas;
- **Falso-negativo ou *False negative* (FN)** – número de instâncias classificadas como negativas que na realidade são positivas.
- **Falso-positivo ou *False positive* (FP)** – números de instâncias classificadas como positiva que na realidade são negativas.
- **Negativo verdadeiro ou *True negative* (TN)** – número de instâncias classificadas como negativas que realmente são negativas.

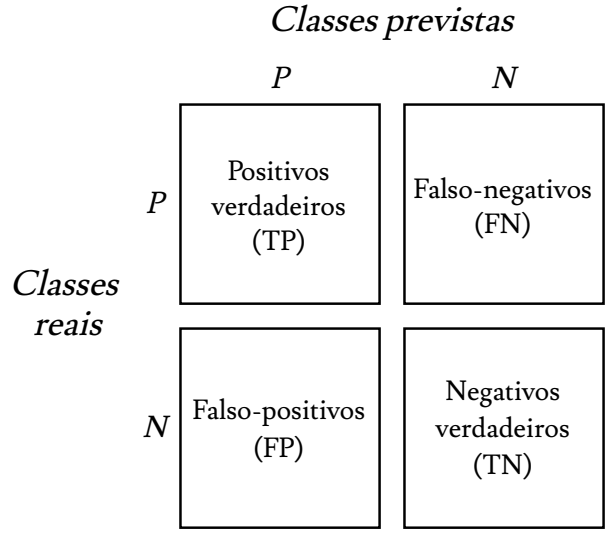


Figura 6: A estrutura de uma matriz de confusão

Fonte: O autor

A partir desta definição é possível estabelecer algumas medidas de desempenho do classificador.

2.3.3.1 Precisão, Revocação e *F-score*

Essas medidas fornecem informações sobre a qualidade das previsões positivas feitas pelo modelo. A *precisão* é a proporção de acertos nas previsões positivas e a *revocação* é a proporção de instâncias positivas previstas corretamente pelo classificador (ou seja, a mesma medida TPR). O *F-score*, também conhecido com *F-measure*, F_1-*score* ou apenas F_1, é a média harmônica entre a precisão e a revocação (RASCHKA; MIRJALILI, 2019, p.323).

$$precisão = \frac{TP}{TP + FP} \tag{1}$$

$$revocação = TPR = \frac{TP}{FN + TP} \tag{2}$$

$$F_1 = 2 \times \frac{precisão \times revocação}{precisão + revocação} \tag{3}$$

2.3.4 Medidas de desempenho e problemas com alto desbalanceamento de classes

Como será mostrado no Capítulo 5, o problema de detecção de vulnerabilidades exploradas apresenta classes de tamanhos muito distintos. Em média, menos de 5% das vulnerabilidades consideradas são rotuladas como "exploradas". As 95% de vulnerabilidades restantes são rotuladas como "não exploradas". Essa diferença na quantidade de instâncias de um problema é chamada de **desbalanceamento de classes** e a sua existência em grande proporção traz desafios para a criação do modelo e a sua avaliação de seu desempenho.

Medidas de desempenho como a acurácia e a taxa de erros, apesar de serem úteis em um problema de classes balanceadas, podem levar a conclusões erradas para classes desbalanceadas. Isso ocorre por dois fatores principais, o primeiro deles é que é fácil obter alta acurácia simplesmente rotulando todas as entradas com o rótulo predominante no conjunto de dados. Exemplo: em uma base com 99% de instâncias negativas e 1% de instâncias positivas, rotular todas as instâncias como negativas garantiria 99% de acurácia. O segundo fator é que essas medidas assumem que os erros das duas classes têm o mesmo custo, o que geralmente não é verdade. Em problemas com classes desbalanceadas, erros na classe majoritária costumam ser menos importantes que na classe minoritária (FERNÁNDEZ et al., 2018). Exemplo: em um exame que detecta uma doença, um resultado falso-negativo pode custar a vida do paciente, enquanto um falso-positivo é aceitável e pode ser revisado.

Para problemas com classes desbalanceadas, o ideal é que se faça a análise do desempenho por classes uma vez que o desafio é rotular corretamente a classe minoritária sem produzir muitos erros na classe majoritária. Essa análise é possível com medidas como a taxa de positivos verdadeiros e a taxa de falso-positivo. No entanto, ela dificilmente poderá ser feita isoladamente, pelo contrário, será necessária a análise conjunta desses valores (FERNÁNDEZ et al., 2018).

Como visto anteriormente, o *F-score* condensa a análise de duas medidas relacionadas a classe positiva (precisão e revocação), o que o torna adequado para realizar comparações entre algoritmos (FERNÁNDEZ et al., 2018). No entanto é importante entender a forma como esse indicador se comporta para classes desbalanceadas. A Tabela 1 mostra diversos exemplos de matrizes de confusão para um conjunto de dados fictício, contendo 100 instâncias e com desbalanceamento entre as classes próximo ao do conjunto abordado nesta dissertação (6% de instâncias positivas e 94% de instâncias negativas). Observe como um classificador que acerta 50% das instâncias em ambas as classes possui baixo *F-score*, isso ocorre devido ao grande número de falsos positivos. Observe também como a alteração no valor previsto de poucas instâncias positivas impactam fortemente a revocação e, por consequência, o *F-score*. Por último, observe como existe uma relação proporcionalmente inversa entre os valores de precisão e revocação. Classificadores que apresentam muitos positivos, tendem a ter melhor revocação e pior precisão, enquanto classificadores que rotulem mais como negativo tendem a ter comportamento oposto.

Tabela 1: Exemplo de matrizes de confusão e medidas de desempenho para conjunto de dados desbalanceado

Matriz	**Precisão**	**Revocação**	F_1	**Descrição**
$\begin{pmatrix} 3 & 47 \\ 3 & 47 \end{pmatrix}$	0,06	0,50	0,10	50% de acertos em cada classe
$\begin{pmatrix} 5 & 47 \\ 1 & 47 \end{pmatrix}$	0,09	0,83	0,17	50% + 2 acertos na classe positiva
$\begin{pmatrix} 6 & 94 \\ 0 & 0 \end{pmatrix}$	0,06	1,00	0,11	100% de rótulos positivos
$\begin{pmatrix} 1 & 0 \\ 5 & 94 \end{pmatrix}$	1,00	0,16	0,28	100% - 1 rótulos negativos

Nesta dissertação, pelos motivos mencionados nesta seção, a principal medida de desempenho utilizada é o *F-score*. A precisão também é utilizada, como um critério de desempate, pelo fato de que já existem indicadores de severidade muito conservadores (com alto número de falso-positivo), como o CVSS. Este trabalho se propõe a ajudar na priorização das vulnerabilidades mais graves e priorizar muitas instâncias falsos-positivas para melhorar a revocação seria, em essência, não priorizar nenhuma instância.

2.4 Algoritmos de classificação

Esta seção aborda os algoritmos de classificação utilizados nesta dissertação e outros que, apesar de não terem sido utilizados, são citados nos trabalhos relacionados. As descrições feitas aqui são direcionadas para problemas binários, muito embora a maior parte dos algoritmos também possuam versões adaptadas a problemas multiclasse.

2.4.1 Máquinas de Vetores de Suporte

Máquinas de vetores de suporte (do inglês, *Support Vector Machine* – SVM) constituem um grupo de algoritmos de classificação baseados no princípio da minimização de risco estrutural. Neles, as instâncias do problema são representadas como pontos no espaço Euclidiano cujo número de dimensões m é o mesmo do vetor de características usado para o problema. O objetivo do SVM é definir um hiperplano ótimo que separe as instâncias em duas classes com a maior margem possível. A Equação 4 descreve o hiperplano, nela existem dois parâmetros: o vetor w de tamanho m igual ao número de dimensões do vetor de característica e uma constante b (BURKOV, 2019, p.6).

$$wx - b = 0 \tag{4}$$

O algoritmo rotula suas instâncias utilizando a Equação 5, onde a função *sign* retorna 1 caso receba um número positivo ou -1 caso receba um número negativo. O objetivo do algoritmo é encontrar valores w e b que maximizem as distâncias entre os pontos (instâncias) e o hiperplano (BURKOV, 2019, p.6).

$$y = \text{sign}(wx - b) \tag{5}$$

2.4.2 Regressão Logística

A regressão logística é um algoritmo de classificação muito utilizado para problemas binários e leva esse nome por ser baseado na função logística padrão, demonstrada na Equação 6. O modelo é definido conforme a Equação 7, onde x é o vetor de características e os valores w e b são, respectivamente, um vetor de parâmetros e uma constante que, juntos, darão forma à curva logística.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{6}$$

$$f_{w,b}(x) = \frac{1}{1 + e^{-(wx+b)}} \tag{7}$$

Uma característica da função logística é que sua imagem é tal que $0 \leq f(x) \leq 1$. Isso a torna ideal para problemas binários pois, ao se estabelecer um limiar t, valores de $f(x)$

menores que t podem ser rotulados como 0 e valores superiores ou iguais a t, como 1 (o valor de t pode ser ajustado conforme o problema, mas é comum que ele seja $0,5$). Para se obter os valores (w, b) ótimos, o algoritmo procura maximizar a probabilidade total do conjunto de treino (BURKOV, 2019, p.25). A probabilidade $P(c|x)$ de uma instância x representada pelo vetor de características $x = (x^{(1)}, x^{(2)}, ..., x^{(m)})$ ser classificada com a classe c é dada pela Equação 8:

$$P(c|x) = \frac{1}{1 + \exp(-w^{(1)}x^{(1)} - w^{(2)}x^{(2)} - ... - b)} \tag{8}$$

O objetivo do algoritmo de regressão logística é encontrar w e b que maximizem a combinação L de todas as probabilidades aplicadas ao conjunto de treino, conforme a Equação 9.

$$L_{w,b} = \prod_{i=1}^{n} P(y_i|x_i) \tag{9}$$

2.4.3 Árvore de Decisão

Uma árvore de decisão é um grafo acíclico que pode ser utilizado para fazer classificações. Em cada nó interno do grafo, uma característica $x^{(j)}$ do vetor de características é analisada e, caso a condição do nó seja satisfeita, o processo decisório seguirá por um lado da árvore, caso contrário, o processo decisório seguirá pelo lado oposto. O algoritmo se repete até que seja encontrado um nó folha, onde um rótulo será aplicado (BURKOV, 2019, p.27). É importante mencionar que: nem todas as características precisam ser analisadas pela árvore; uma mesma característica pode ser analisada em mais de um nó, caso seja necessário e; os rótulos podem se repetir em diferentes nós folhas. Existem diversos métodos para construir uma árvore de decisão. O *Classification And Regression Tree* (CART), por exemplo, é um algoritmo de abordagem gulosa que escolhe as características analisadas em cada nó e transforma folhas em nós intermediários com base em uma medida de pureza, o índice Gini, das instâncias classificadas por aquela folha. O critério de parada depende da implementação, mas pode, por exemplo, levar em conta o número máximo de elementos em uma folha ou a profundidade máxima desejada para árvore.

2.4.4 eXtreme Gradient Boosting e Light Gradient Boosting Machine

eXtreme Gradient Boosting (XGBoost) e *Light Gradient Boosting Machine* (LightGBM) são *ensembles* de classificação que utilizam árvores de decisão. Ao invés de se usar um único modelo, os *ensembles* utilizam múltiplos modelos de decisão, cada um dando o seu voto sobre qual deveria ser o rótulo para a instância do problema. Mais especificamente, o XGBoost e o LightGBM são algoritmos do tipo *boosting*, uma técnica que utiliza uma

coleção de modelos mais simples para compor um modelo mais complexo. O *boosting* parte de um único modelo de decisão fraco (geralmente aleatório ou arbitrário) e treina modelos de maneira consecutiva, corrigindo os erros dos modelos anteriores (RASCHKA; MIRJALILI, 2019, p.368). Nesta técnica o treino é sempre feito com um subconjunto do conjunto de treino, escolhido de maneira aleatória e com repetições. A cada etapa, a escolha dos elementos de treino leva em conta os erros cometidos pelos classificadores das etapas anteriores. A ideia é dar preferência para instâncias que ainda são classificadas de maneira errada. No *boosting*, geralmente é usado um limite para o tamanho das árvores, mas a forma como cada árvore é montada e o sistema de votação varia de acordo como algoritmo.

O XGBoost é uma variação do algoritmo *Gradient Boost*, um método de *boosting* usado para classificação e regressão em que, a cada iteração, um modelo de previsão fraco é construído para prever não os rótulos, mas os resíduos produzidos pela iteração anterior. O XGBoost oferece diversos parâmetros de customização no algoritmo e mudanças específicas na forma como as árvores são criadas para aumentar o desempenho do *ensemble*. Na construção das árvores, é utilizado um parâmetro de regularização que pune modelos complexos evitando sobreajustes. Além disso, o algoritmo utiliza um modelo de poda tardia, onde a árvore só é podada após atingir seu tamanho máximo `max_depth`. As árvores do XGBoost também são capazes de detectar valores faltantes em vetores de características, inferindo qual ramificação é melhor para a instância (CHEN; GUESTRIN, 2016). Outra importante modificação no algoritmo é a forma como a melhor divisão para um nó é computada. Para superar a natureza sequencial e não paralelizável do processo, o XGBoost calcula antecipadamente as melhores divisões para todas as características, o que possibilita uso de paralelismo e reduz consideravelmente o custo da parte sequencial do algoritmo.

O LightGBM, por outro lado, aborda de maneira diferente a construção das árvores, possibilitando melhorar significativamente o tempo do treinamento. A primeira modificação é na seleção da característica para os nós da árvore. O LightGBM introduz o uso de uma técnica chamada *Gradient-based One-Side Sampling* (GOSS), que seleciona apenas as instâncias de maior gradiente e uma pequena parcela das demais, escolhidas aleatoriamente, para calcular qual a melhor característica e o melhor valor para divisão do nó. Essa modificação faz com que o LightGBM lide bem com grandes conjuntos de dados. Outra modificação é na forma como o algoritmo expande suas árvores, ao contrário da maior parte dos algoritmos, que expansão em largura ou em profundidade, o LightGBM usa o método "baseado em folhas". Esse tipo de expansão escolhe sempre a folha que melhor divide o conjunto de dados para ser expandida e é mais eficiente ao lidar com limites de expansão e podas (KE et al., 2017).

2.5 Algoritmos de balanceamento de classes

Como mencionado na Seção 2.3.4, a detecção de vulnerabilidades exploradas é um problema de classes altamente desbalanceadas. Este trabalho investiga se, para tal problema, algoritmos de balanceamento de classes são capazes de melhorar significativamente o desempenho do classificador. A Seção 5.4 abordará um experimento conduzido para responder esse questionamento. Os algoritmos a seguir foram aqueles que apresentaram melhor desempenho no experimento.

2.5.1 *Synthetic Minority Over-sampling Technique*

O *Synthetic Minority Over-sampling Technique* (SMOTE) é algoritmo de balanceamento de classes que utiliza a sobreamostragem da classe minoritária pela criação de instâncias sintéticas. O algoritmo opera no espaço das características, traçando linhas entre cada instância real da classe minoritária e seus k vizinhos mais próximos e, em seguida, escolhendo um ponto aleatório pertencente a essa linha. As coordenadas da instância criada são obtidas da seguinte maneira (CHAWLA et al., 2002):

1. Calcule o vetor d, que representa a diferença entre os vetores de características de uma instância real x_a e um de seus k vizinhos mais próximos, x_b, escolhido aleatoriamente;

2. Escolha um número aleatório g, entre 0 e 1, e multiplique-o ao vetor d;

3. Some o vetor obtido ao vetor original x_a.

A razão com que a classe minoritária aumenta é determinada por um número $n \leq k$ que corresponde a quantidade de vezes em que os k vizinhos mais próximos poderão ser escolhidos aleatoriamente para o procedimento. Por exemplo: se n for igual a 1, a classe minoritária aumentará em 100%, se n for igual a 2, ela será aumentada em 200% e assim por diante. O valor padrão para k é 5 e n é calculado de maneira a equalizar as classes.

Além de diminuir o desbalanceamento entre as classes, essa abordagem permite que a região da classe minoritária se torne menos específica, evitando sobreajuste.

2.5.2 *Adaptive Synthetic*

O *Adaptive Synthetic* (ADASYN) é outro algoritmo de balanceamento de classes que utiliza a sobreamostragem da classe minoritária pela criação de instâncias sintéticas. O algoritmo compartilha algumas semelhanças com o SMOTE, uma vez que também traça linhas entre as instâncias da classe minoritária e seus k vizinhos mais próximos e produz as novas instâncias dentro dessas linhas. Porém, o ADASYN é otimizado para produzir mais instâncias sintéticas em regiões da classe minoritária cujo aprendizado seria mais difícil

com poucas instâncias. Para conseguir esse objetivo, o algoritmo funciona da seguinte forma (Haibo He et al., 2008):

1. Calcule o grau de desbalanceamento $d = m_s/m_l$, onde m_s é o tamanho da classe minoritária e m_l o tamanho da classe majoritária;

2. Calcule o número de instâncias sintéticas a serem gerados $G = (m_l - m_s) \times \beta$, onde β é um parâmetro entre 0 e 1 que indica a proporção entre as classes após o balanceamento;

3. Para cada instância x_a da classe minoritária, encontre os k vizinhos mais próximos e calcule a proporção $r_a = \Delta_a/k$, onde Δ_a é o número de instâncias da classe majoritária entre os k vizinhos mais próximos de x_a;

4. Normalize r_a de acordo com $\hat{r}_a = r_a / \sum_{a=1}^{m_s}$ tal que $\hat{r}_a$ represente uma distribuição de densidade;

5. Calcule o número de instâncias sintéticas $g_a = \hat{r}_a \times G$ gerados a partir da instância minoritária x_a;

6. Para cada instância a da classe minoritária, gere g_a instâncias sintéticas seguindo o mesmo procedimento utilizado pelo algoritmo SMOTE.

O objetivo do ADASYN é utilizar a distribuição de densidade para priorizar a criação de novas instâncias próximos a instâncias da classe minoritária que estejam em meio a muitas instâncias da classe majoritária. Dessa forma, o classificador terá maior facilidade de detectar a classe de interesse naquela região.

2.5.3 *Random Under Sampler* e *All k-Nearest-Neighbor*

Ao contrário dos algoritmos anteriores, o *Random Under Sampler* (RUS) e o *All k-Nearest-Neighbor* (AllKNN) são algoritmos de balanceamento de classes que fazem a subamostragem da classe majoritária utilizando métodos muito simples. Como o nome sugere, o RUS escolhe, aleatoriamente, instâncias da classe majoritária para retirar do treinamento do classificador. Já o AllKNN examina as k instâncias mais próximas de cada instância x_a da classe majoritária e, caso x_a não tenha a mesma classe da maior parte dos seus vizinhos, o exclui do conjunto de treino (WILSON, 1972).

Capítulo 3

Trabalhos Relacionados

Neste capítulo, são abordados os principais trabalhos com proposta relacionada ao tema desta dissertação. O objetivo é fornecer uma visão geral do estado da arte em detecção/predição de *exploits* utilizando aprendizado de máquina e redes sociais. Muitos trabalhos que relacionam segurança de computadores e mineração de dados e redes sociais podem ser enquadrados em duas categorias: detecção/predição de eventos relacionados à segurança e; detecção/predição de *exploits*. Este trabalho se enquadra na segunda categoria, mas é importante conhecer os dois tipos de trabalhos para compreender melhor a diferença entre eles.

3.1 Detecção/predição de eventos relacionados à segurança

Nesta abordagem, o objetivo é encontrar ou prever eventos como ataques cibernéticos a empresas ou personalidades, vazamento de dados e roubo de contas em redes sociais usando informação coletada em redes sociais. Os trabalhos que seguem essa linha partem do princípio de que os incidentes estudados são frequentemente compartilhados em redes sociais e, quanto mais relevantes, mais discutidos eles serão. Essa ideia não é exclusiva da área de segurança de computadores. (MARTINO et al., 2017) utiliza mensagens coletadas do *Twitter* para detectar surtos epidêmicos enquanto (POBLETE et al., 2018) utiliza a mesma rede social para a detecção de terremotos. Ambos os trabalhos utilizam o aumento repentino no volume de *tweets* (postagens no *Twitter*) sobre um assunto como sensor que indica a ocorrência de algum fenômeno e o agrupamento desses *tweets* para descrever suas características.

Analogamente, os trabalhos relacionados a segurança de computadores propõem formas de agrupar mensagens que tratam de um mesmo evento. Eles geralmente partem de buscas por termos genéricos relacionados a ataques cibernéticos, agrupam mensagens cujo conteúdo é semelhante e definem algum critério relacionado ao volume de mensagens

para caracterizar um evento.

Khandpur et al. (2017) utilizam um método baseado em expansão de consultas para categorizar e agrupar ocorrências de ataques de negação de serviço (em inglês, *Denial Of Service* – DoS), vazamento de dados sensíveis ou sequestro de contas. A estratégia adotada pelos autores foi monitorar o fluxo de mensagens do *Twitter*, buscando inicialmente por palavras-chave arbitrárias (como *"ddos attack"*, *"hacked account"* ou *"data leak"*) e, com base nos resultados obtidos, elencar os termos mais comuns para formular palavras-chave derivadas. O processo é iterativo e se repete até que o conjunto de palavras derivadas converge e possa ser utilizado como forma de descrever o próprio evento. Em exemplo mostrado pelos autores, o conjunto de palavras *"CENTCOM twitter account hack"* foi obtido no dia em que a conta do Comando Central dos Estados Unidos no *Twitter* foi hackeada em 2015. O método utilizado pelo trabalho apresenta a limitação de só conseguir reconhecer um único evento por dia, já que a expansão de busca acontece diariamente. Além disso, apenas os três tipos de eventos mencionados podem ser detectados.

Ritter et al. (2015) utilizam a busca de eventos já conhecidos para treinar classificadores utilizando um modelo de aprendizado semi-supervisionado. Nesse trabalho, o treinamento é realizado apenas com exemplos positivos e por isso necessita de algum outro método para inferir sobre resultados negativos. Os autores propõem o uso de uma técnica baseada em regularização de expectativa, onde são utilizados poucos exemplos de instâncias rotuladas e a distribuição provável de rótulos para os demais casos, fornecida por um especialista da área.

No trabalho, os *tweets* foram coletados utilizando palavras chaves genéricas (como "hacked", "ddos" e "breach") e, posteriormente, foram submetidos a métodos de PLN para a criação de eventos candidatos. Cada evento foi representado por uma tupla contendo uma entidade e uma data, como "Zuckerberg, 18/08/2013", representando o episódio em que a conta de Mark Zuckerberg no Facebook foi invadida. Em seguida, os eventos candidatos são submetidos ao classificador e agrupados de maneira indireta, a medida que surgem tuplas idênticas. Para avaliar o trabalho, os pesquisadores fizeram a conferência manual dos eventos encontrados. Foi possível comprovar a superioridade em precisão e revocação do método de treino proposto sobre outros três métodos bases.

Sceller et al. (2017) faz uso de técnicas de agrupamento baseadas em similaridade de cossenos e *locality-sensitive hashing* aplicado em *tweets* coletados em uma janela de tempo deslizante. A busca também é feita por palavras-chave, mas nesse trabalho ela é baseada em uma taxonomia criada pelos autores e que pode se alterar de acordo com sugestões do próprio algoritmo. Após formados, os grupos de mensagens são filtrados por entropia e tamanho e, caso não sejam eliminados, poderão ser combinados com grupos de janelas temporais próximas. Ao final, cada grupo indicará a existência de um evento e será representado por sua primeira mensagem.

Mittal et al. (2016) propõem um *framework* chamado "*CyberTwitter*", que utiliza o

Twitter como fonte de inteligência para criar alertas de segurança relativos à vulnerabilidades e ameaças. Esse trabalho pode ser considerado uma ponte entre as duas abordagens (detecção/predição de eventos e detecção/predição de *exploits* e vulnerabilidades) uma vez que os autores mencionam ataques, vulnerabilidades e ameaças. O *framework* proposto aplica técnicas de PLN em *tweets* para extrair meios, consequências e *softwares* afetados por ataques. As mensagens são obtidas buscando por palavras-chave derivadas de um perfil de sistema do usuário e um pequeno grupo de termos relacionados a problemas de segurança.

O trabalho faz uso de ontologias de segurança e bases de conhecimento públicas para filtrar mensagens e mapear termos por meio de um *Named Entity Recognize* (NER) personalizado. O sistema proposto cria sua própria base de conhecimento, onde as ameaças são definidas por tríplices semânticas sujeito-predicado-objeto associadas a informações temporais. A cada novo *tweet*, os dados temporais de um evento podem ser atualizados caso estejam dentro de uma janela de tempo configurável T. Por fim, o *framework* utiliza heurísticas pré-determinadas para decidir se o evento deve ou não ser alertado ao usuário.

Para avaliar o trabalho, os autores coletaram 143.701 *tweets* durante dez dias para um perfil de usuário fictício. Após a filtragem e mapeamento dos termos, 10.004 *tweets* foram considerados relevantes e 158 ameaças foram listadas. A conferência dos resultados foi feita manualmente pelos pesquisadores e foi possível verificar que apenas 37 das ameaças listadas faziam parte do perfil procurado e, dessas, apenas 15 produziram alerta para o usuário. Os pesquisadores também testaram, por meio de amostragem aleatória, a eficácia da categorização automática de *tweets*. Nesse teste, a solução foi capaz de atribuir tríplices semânticas corretas para 57% dos *tweets* avaliados. Os autores não trazem muitos detalhes sobre os tipos de ameaças encontradas pelo *framework*, mas pela descrição de como é feito a busca, entende-se que eles incluam eventos, vulnerabilidades e ameaças. Por essa razão, o trabalho pode ser considerado menos específico e não focado em vulnerabilidades e *exploits* diferentemente desta dissertação.

3.2 Detecção/predição de *exploits*

Apesar do bom desempenho apresentado em alguns dos trabalhos citados, a detecção de eventos traz informações que podem ser pouco úteis para um pesquisador ou analista de segurança, como é o caso dos roubos de contas de celebridades ou o vazamento de dados de algum comércio eletrônico. Informações como essas dizem pouco sobre suas causas e frequentemente estão associadas a vulnerabilidades antigas e facilmente corrigidas. Por outro lado, existem diversos trabalhos que procuram identificar o surgimento de novos *exploits* por meio de redes sociais ou fóruns. Nessa abordagem, uma vulnerabilidade é a instância do problema, as características são compostas por dados que sintetizam as mensagens referentes a ela e deseja-se descobrir se tal vulnerabilidade foi ou será explo-

rada. Essa foi a abordagem escolhida para esta dissertação. A Tabela 2 traz algumas características das principais publicações nessa área.

Tabela 2: Trabalhos sobre detecção/predição de *exploits*

Trab.	**Período**	**% Exploits**	**Algoritmos**	**Características**	**Rótulos**
#1	fev/2014 a jan/2015	6,5%(PoC), 1,3% (real)	SVM	Twitter, OSVDB, NVD	EDB(PoC), Microsoft (PoC), Symantec (real)
#2	jan/2016 a jun/2016	17% (PoC)	SVM	NVD, Twitter*	EDB (PoC)
#3	jan/2010 a mar/2017	15% (real)	Randon Forest	DeepWeb, NVD	Symantec (real)
#4	jan/2015 a dez/2016	1,2% (real)	Randon Forest	DeepWeb, NVD, EDB, ZDI	Symantec (real)
#5	jul/2016 a maio/2018	5,5% (PoC), 0,5% (real)	RF, SVM, RL, NB, XGBoost	Twitter	EDB (PoC), Symantec (real)

#1 – Sabottke, Suciu e Dumitras (2015)
#2 – Bullough et al. (2017)
#3 – Almukaynizi et al. (2017)
#4 – Almukaynizi E. Nunes (2017)
#5 – Chen et al. (2019)

*Somente para comparação com trabalho base

Sabottke, Suciu e Dumitras (2015) criaram um classificador baseado em *Support Vector Machine* (SVM) para detectar, com a ajuda de dados do *Twitter*, se vulnerabilidades foram exploradas. O trabalho compara os resultados obtidos para *exploits* prova de conceito e *exploits* de mundo real. Para rótulos desejados, foi utilizado a menção do código CVE na Exploit Database, para provas de conceito, e a presença do código CVE na descrição de assinaturas da *Symantec* ou *Microsoft Security Advisory*, para *exploits* de mundo real. Para extração de características, os autores utilizam postagens e metadados do *Twitter* combinados com dados do NVD e da *Open Source Vulnerability Database* (OSVDB).

O período estudado foi de fevereiro de 2014 até janeiro de 2015. Foram coletadas 287.717 mensagens utilizando a busca pela palavra-chave "CVE" na API do *stream* do *Twitter*. Nas mensagens, 5.865 códigos CVEs distintos são mencionados e, desses, 77 (1,3%) foram rotulados como explorados em mundo real e 387 (6,5%) como explorados em prova de conceito. Os experimentos principais foram conduzidos utilizando validação cruzada de 10 pastas. Os autores também simularam uma execução online do classificador, onde 10 modelos eram treinados separadamente com porções estratificadas do conjunto de dados completo. Em seguida, os testes foram realizados adicionando gradualmente as mensagens em uma janela deslizante de 1.000 *tweets* e agregando os resultados dos 10 modelos. Além disso, o trabalho investigou o comportamento do classificador na presença de adversários que pudessem produzir informações falsas no *Twitter* para afetar negativamente o sistema.

Como resultado, os autores disponibilizaram principalmente gráficos de precisão e revocação. Um dos poucos números mencionados foi a precisão do classificador caso fosse utilizado apenas o CVSS, menos que 9%. Também é demonstrado como *exploits* prova de conceito são mais facilmente detectáveis e como o classificador tem desempenho superior

ao utilizar apenas vulnerabilidades de produtos suportados pela *Symantec*. Na simulação de testes *online*, foi possível detectar a existência de alguns *exploits* horas depois do primeiro *tweet* a respeito e horas antes da publicação de uma assinatura. Por fim, os experimentos com manipulações adversárias em *tweets* revelou que um possível atacante precisaria comprar um grande número de contas e ter conhecimento sobre as características utilizadas pelo classificador para criar mensagens que conseguissem prejudicar significativamente o modelo.

Bullough et al. (2017) fazem uma crítica a diversos trabalhos anteriores quanto as divisões de treino e teste utilizadas. Os autores discutem como o uso de validação cruzada com pastas definidas aleatoriamente poderiam causar o "vazamento de informações futuras" aos conjuntos de treino. Para comprovar sua hipótese, os autores tentam reproduzir o experimento principal feito por Sabottke, Suciu e Dumitras (2015) e fazem comparações com o uso de grupos de treino e teste separados por tempo.

O trabalho apresenta, porém, limitações como: o período em que os dados foram coletados do *Twitter*, apenas seis meses; o uso de fontes de rótulos apenas para provas de conceito; proporção de vulnerabilidades exploradas muito superior a estudos semelhantes (em torno de 17%) e; o aumento artificial do desbalanceamento para comparação de resultados.

Almukaynizi et al. (2017) utilizam um classificador *Random Forest* para responder se uma vulnerabilidade foi explorada no mundo real com base em informações coletadas em fóruns da *DeepWeb*. O trabalho propõe a montagem de um grafo de relacionamento entre usuários que atuam nessas comunidades para a extração de características juntamente com informações coletadas do NVD. Os autores utilizam menções de vulnerabilidades em assinaturas da *Symantec* como rótulos corretos.

Nesse trabalho, o período estudado foi de janeiro de 2010 a março de 2017. O conjunto de dados usado continha 2.290.000 postagens em 151 fóruns diferentes, porém apenas 3.082 dessas postagens faziam referência explícita a identificadores CVE. Ao todo, 502 vulnerabilidades do período de interesse foram mencionadas por 365 usuários diferentes. Para estudar esses usuários e extrair informações da rede de contato formada por eles, construiu-se um grafo direcional ponderado de relacionamento. Nele cada vértice representa um usuário e cada aresta valorada representa o número de vezes em que um par de usuários postaram em um mesmo tópico. O grafo produzido também foi utilizado para extração de características para o modelo.

Para testar o desempenho do classificador, foram elaborados dois experimentos: um experimento onde o modelo foi treinado com vulnerabilidades mencionadas antes de junho de 2016 e testado com aquelas mencionadas depois; e outro sem divisão temporal das vulnerabilidades e utilizando validação cruzada de 5 pastas. Os resultados foram 0,67 e 0,72 de F-score, para o experimento um e dois, respectivamente. Apesar dos bons resultados, ao utilizar apenas vulnerabilidades citadas na *DeepWeb*, o trabalho apresentou um

desbalanceamento muito menor entre as classes se comparado a trabalhos relacionados. Das vulnerabilidades testadas, 15% eram exploradas. Outros trabalhos apontam proporções bem menores, entre 1,5% a 5% apenas. Os autores ainda reconhecem que o uso da *Symantec* como única fonte de rótulos pode ter enviesado os resultados e diminuído a proporção de casos positivos.

Almukaynizi E. Nunes (2017) propõem o uso da *Exploit Database* (EDB) não para a extração de rótulos, mas para compor uma das características fornecidas ao classificador. O trabalho também extrai características do NVD, de fóruns da *DeepWeb* e do site da ZDI, site privado que pesquisa e cataloga vulnerabilidades. Para obtenção de rótulos, apenas assinaturas da *Symantec* são utilizadas. O algoritmo de classificação utilizado foi o *Random Forest* e as técnicas de validação escolhidas foram a *holdout*, com divisão baseada na data de publicação da vulnerabilidade, e a validação cruzada de 10 pastas, dependendo do experimento.

O trabalho levou em conta todas as vulnerabilidades divulgadas entre 2015 e 2016 e encontrou uma taxa de vulnerabilidades exploradas igual a 1,2%. Os pesquisadores conseguiram obter *F-score* de 0,4 utilizando o método *holdout* e 0,48 ao utilizar validação cruzada. Ao comparar fontes de informações para descobrir a que mais trouxe ganhos, descobriu-se que a combinação de dados da *DeepWeb* e do NVD pôde trazer o melhor resultado.

Ao contrário dos demais trabalhos, Chen et al. (2019) não procuram responder se uma vulnerabilidade será explorada, mas sim quando isso ocorrerá. Os autores fazem uma crítica ao uso do CVSS como característica, alegando que cerca de 49% dos *exploits* ocorrem antes da nota ser atribuída, e propõem um modelo que extrai características apenas do *Twitter*. O período observado foi de julho de 2016 a maio de 2018, foram coletadas 632.873 mensagens a partir da palavra-chave "CVE". A representação dos dados é feita utilizando um grafo multicamadas que interliga CVEs, *tweets* e autores, partindo do princípio de que quanto mais popular é uma vulnerabilidade, maior as chances de ela ser explorada.

Para prever quando uma vulnerabilidade pode ser explorada, os autores utilizaram dois métodos complementares, o primeiro é um conjunto de 5 classificadores binários que preveem se a vulnerabilidade será ou não explorada em 1, 3, 6, 9 e 12 meses. O segundo é aplicado caso a vulnerabilidade seja marcada como explorada em algum dos períodos e consiste em um modelo de regressão que tenta prever o exato dia em que o *exploit* será publicado. O trabalho usa janelas deslizantes com tamanhos variados, dependendo da tarefa, aplicadas a diversos algoritmos de classificação e regressão e a combinações destes algoritmos por meio de fusão tardia. Dentre os métodos usados estão o SVM, Regressão Logística, *Random Forest* e o XGBoost.

Os resultados do trabalho são apresentados separados por tarefa e por meses, um valor para cada período da classificação. É também mostrado um resultado do erro médio na

previsão para o dia exato. No geral, a combinação dos algoritmos com fusão tardia superou, em termos de *F-score*, o uso individual deles. O melhor foi para a previsão de *exploit* em 3 meses, com *F-score* igual a 0,34. Quanto à previsão de dia, foi obtido um erro médio de 11,9 dias.

A abordagem escolhida para esta dissertação de mestrado se aproxima mais daquela usada por Sabottke, Suciu e Dumitras (2015), porém compara diferentes algoritmos de classificação e balanceamento e corrige questões relacionadas ao uso da *Symantec* como única fonte de rótulos para *exploits* do mundo real, falha comum a todos os demais trabalhos. Este trabalho também considera informações de um período de 5 anos, tempo maior do que vários dos trabalhos relacionados. A proporção de vulnerabilidades exploradas dentro das vulnerabilidades totais consideradas é muito mais próxima dos principais trabalhos e fica em torno de 5%, tornando os resultados mais confiáveis. Foi utilizada a validação cruzada, com particionamento aleatório, criticada em alguns dos trabalhos. Mas, para compensar essa limitação, foram realizados experimentos considerando particionamento temporal para treino e teste. Além disso, as contribuições deste trabalho são facilmente aplicadas em soluções que utilizem o segundo método.

Capítulo 4

Proposta

Neste capítulo será abordado o método utilizado no desenvolvimento do projeto. O seu objetivo é descrever o trabalho realizado, as hipóteses formuladas e os testes para verificá-las. O assunto é inicialmente contextualizado na primeira seção e, em seguida, dividido em seções ordenadas de acordo com a sequência em que foram executadas as tarefas nelas descritas. Os conjuntos de dados que serão descritos aqui estão disponíveis no repositório da Faculdade de Computação da Universidade Federal de Uberlândia[1].

4.1 Contextualização

Como visto no Capítulo 1, este trabalho tem como objetivo propor melhorias ao método de classificação que utiliza dados do *Twitter* para classificar vulnerabilidades de *software* como "exploradas" ou "não exploradas". Como mencionado no Capítulo 3, existem trabalhos anteriores que abordaram essa tarefa extraindo características apenas de bases de dados públicas, outros utilizaram o *Twitter* e outros incluíram dados de fóruns e sites da *Deepweb*. Quanto aos rótulos ideais, trabalhos mais antigos utilizavam apenas informações extraídas da EDB enquanto trabalhos mais recentes consideram essa fonte de dados como útil apenas para *exploits* prova de conceito (em inglês, *Proof of Concept* – PoC). Para *exploits* de mundo real, esses trabalhos recorrem a dados da *Symantec*.

A abordagem escolhida para esta dissertação de mestrado utiliza, para a extração de características, dados públicos provenientes do NVD e informações de textos e metadados do *Twitter*. Quanto aos rótulos ideais, foi escolhido comparar os resultados para *exploits* prova de conceito, a partir de dados da EDB, e *exploits* de mundo real, com o uso de informações disponibilizadas por fabricantes de antivírus.

Para este trabalho, foi adotado o modelo supervisionado de aprendizado e escolhido a execução *offline*, onde os dados de treino e teste são conhecidos de antemão. É importante ressaltar que, na maioria dos casos, a detecção de um *exploit* pelo método de aprendizado de máquina é mais útil no período anterior a sua ampla divulgação pelo fabricante do

[1] http://gitlab.facom.ufu.br/cybersecurity-text-mining/detecting-exploits-using-tweets

software afetado ou mídia especializada, portanto o método *online* talvez se aproximasse mais do cenário real. Entretanto, as contribuições deste trabalho relativas a balanceamento de classes e escolha das bases de dados podem ser estendidas para métodos *online*. Além disso, uma das hipóteses específicas levantadas aborda a questão temporal e pode ser útil para comparações entre os métodos.

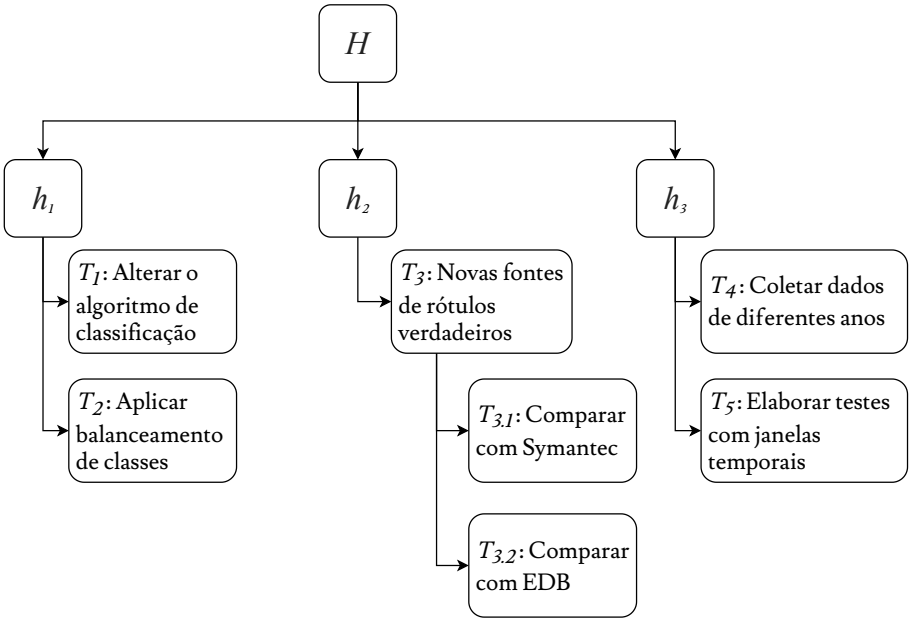


Figura 7: Hipóteses e suas tarefas para validá-las

Fonte: O autor

Pela semelhança na abordagem, os resultados de Sabottke, Suciu e Dumitras (2015) foram escolhidos como linha de base para o presente trabalho. Portanto, a hipótese descrita na Seção 1.3 é aplicada neste contexto e foi dividida em três hipóteses específicas h_i, conforme mostra a Figura 7. Dessa forma, tem-se:

- H: A detecção de vulnerabilidades exploradas com o uso de dados do *Twitter* pode ser melhorada em *F-score* e precisão com o uso de novas fontes de rótulos e um modelo de classificação, diferente daqueles usados em trabalhos relacionados.
- h_1: O uso de algoritmos estado-da-arte em classificação, tais como os ensembles XGBoost e LightGBM, e algoritmos para balanceamento de classes melhora o desempenho da detecção de vulnerabilidades exploradas quanto ao seu *F-score*.
- h_2: O uso de fontes de rótulos alternativas à *Symantec* para de *exploits* de mundo real e o uso dessas fontes pode melhorar o desempenho do classificador quanto ao seu *F-score*.
- h_3: O uso de janelas temporais maiores para treinar o modelo reverte-se em melhores resultados.

Para cada hipótese específica, foram elaboradas tarefas que possibilitaram aceitar ou rejeitar cada uma. As tarefas são explicadas nas subseções a seguir.

4.1.1 T_1: Alterar o algoritmo de classificação

A primeira alternativa para comprovar h_1 é encontrar um algoritmo de classificação que se mostre mais adequado à detecção de *exploits* comparado ao SVM de *kernel* linear, utilizado na linha de base e em diversos trabalhos anteriores. Para este problema, um classificador é mais adequado quando apresenta melhor desempenho em precisão e revocação (evidenciado por um *F-score* mais alto) ou quando exibe valor parecido de *F-score* mas superior em precisão (menor número de falsos positivos). Isso ocorre porque medidas já existentes, como o CVSS, são conservadoras e por isso apresentam alta revocação. Esta tarefa é abordada nas Seções 4.2.3 e 5.2.

4.1.2 T_2: Aplicar balanceamento de classes

Uma segunda forma de se comprovar h_1 é verificar se o uso de algoritmos de balanceamento de classes resulta em melhor desempenho do classificador. Esta tarefa tem como motivação o alto desbalanceamento entre as classes do problema ("explorada" e "não explorada"), sendo a classe de interesse ("explorada") pouco representada e, em alguns casos, podendo corresponder a apenas 1,3% das vulnerabilidades (SABOTTKE; SUCIU; DUMITRAS, 2015). Apesar desse desbalanceamento, até o início deste projeto não haviam trabalhos relacionados que aplicavam esse tipo de algoritmo, reforçando a ideia de que seria possível melhorar os resultados da linha de base. Esta tarefa é abordada nas Seções 4.2.3 e 5.4

4.1.3 T_3: Identificar novas fontes de rótulos

Em trabalhos anteriores que, assim como este, separaram *exploits* entre "mundo real" e "prova de conceito", a única fonte de dados sobre o primeiro era a *Symantec*[2]. Nesse método, a menção de vulnerabilidades na descrição de uma assinatura de antivírus ou assinatura dos sistemas de *Intrusion Detection System* (IDS) e IPS da *Symantec* indicam que elas foram exploradas e detectadas no mundo real (conforme mostrado na Figura 8). A hipótese h_2 é motivada pela suspeita de que outros fabricantes de antivírus também disponibilizem informações similares e que as vulnerabilidades mencionadas por esses fabricantes podem não ser mencionadas pela *Symantec*, o que ampliaria o rol de *exploits* de mundo real e poderia modificar os resultados do classificador.

O levantamento de fabricantes de antivírus que disponibilizem seus dados de assinaturas e que mencionem neles códigos CVEs de vulnerabilidades, pode comprovar h_2. Mas

[2] www.nortonlifelock.com

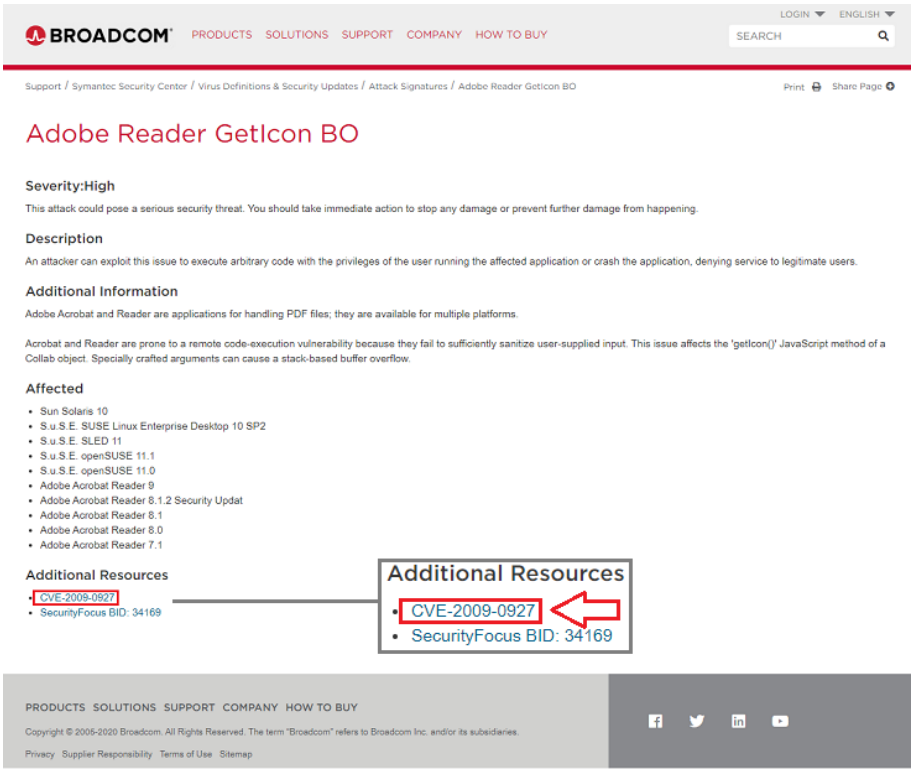


Figura 8: Exemplo da citação de um código CVE no site da *Broadcom*, empresa proprietária da *Symantec*

Fonte: Twitter (2017)

para isso, também é necessário que as vulnerabilidades citadas pelas novas fontes sejam realmente diferentes daquelas citadas pela *Symantec*, garantindo que a informação é relevante, e diferentes daquelas citadas pela EDB, indicando que não se tratam apenas de provas de conceito. Portanto, foram adicionadas a T_3 as subtarefas $T_{3.1}$ e $T_{3.2}$. Esta tarefa é abordada nas Seções 4.3 e 5.3

4.1.4 T_4: Coletar dados de diferentes anos

Para se comprovar h_3, é necessário obter um conjunto de dados que abranja um período maior do que aquele utilizado pelo trabalho linha de base (um ano). Por essa razão decidiu-se que seriam coletados dados dos anos seguintes ao trabalho base, ou seja de 2015 até 2018. Esta tarefa é abordada nas Seções 4.4 e 5.5

4.1.5 T_5: Elaborar testes com janelas temporais

Após a construção de um conjunto de dados relativo a um período maior, a comprovação de h_3 dependerá da elaboração e execução de testes com diferentes janelas temporais. Estes testes devem comparar o desempenho de classificadores treinados com dados de um

único ano (linha de base) com o de classificadores treinados com dados de múltiplos anos. Esta tarefa é abordada na Seção 5.6

4.2 Elaboração de um detector de *exploits* aplicado ao *Twitter*

A Figura 9 representa a arquitetura escolhida para o sistema de detecção de *exploits* proposto. Os números 1 e 2 representam as fontes de informações para a extração de características enquanto 4 e 5 são fontes para rótulos corretos. Os números 3, 6 e 7 indicam processos utilizados para a criação de um modelo de decisão. Nesta seção serão abordados aspectos da implementação dessa ferramenta.

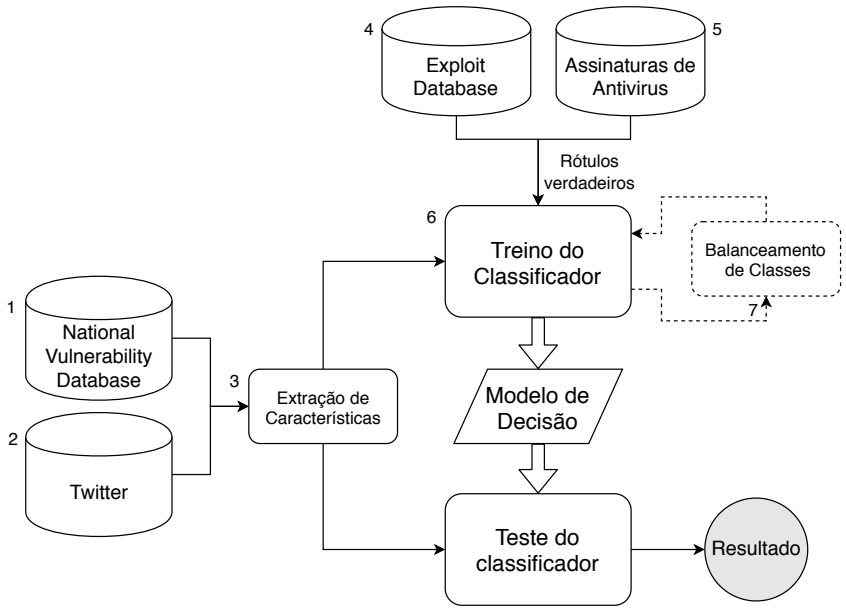


Figura 9: Arquitetura do sistema de detecção de *exploits* proposto

Fonte: O autor

A partir do levantamento bibliográfico realizado, pôde-se perceber que a linguagem *Python*[3] é uma das linguagens mais utilizadas para a construção de ferramentas semelhantes a proposta. A existência de bibliotecas como o *scikit-learn*[4], com centenas de conjuntos de dados e algoritmos de aprendizado, avaliação e visualização, favorece muito o trabalho. A princípio a versão utilizada seria a 3.6, a mais recente no início do projeto, porém uma atualização de parâmetros do classificador SVM dificultaria a comparação com o trabalho escolhido para linha de base. Portanto, as versões utilizadas foram a 2.7 do *Python* e a 0.20.3 do *scikit-learn*.

[3] https://www.python.org/
[4] https://scikit-learn.org/

Para se obter o modelo de decisão indicado na Figura 9, foram criados algoritmos em *Python* para três etapas distintas: coleta dos dados brutos; extração de características e criação do conjunto de rótulos; e treino/teste do modelo. Contudo, para uma primeira fase da pesquisa, foram obtidos os códigos e conjunto de dados usados no trabalho base.

4.2.1 Conjunto de dados

Os dados fornecidos pelos autores de Sabottke, Suciu e Dumitras (2015) e utilizados para os primeiros testes são compostos por *tweets* coletados utilizando a API de *stream* do *Twitter* entre fevereiro de 2014 e janeiro de 2015. Foram buscadas mensagens que continham a palavra-chave "CVE" e consideradas apenas aquelas que faziam referência a um código CVE válido. Ao todo foram coletados 287.717 *tweets* referenciando 5.865 CVEs distintos e cujas datas de divulgação foram dentro do período observado. O conjunto de dados trazia os *tweets* agrupados por vulnerabilidade referenciada.

Para cada vulnerabilidade, o conjunto também continha informações coletadas do NVD e da OSVDB, uma base de dados pública descontinuada em 2016. Os dados incluíam o nome do fabricante do software afetado, o vetor e score CVSS, a descrição da vulnerabilidade, dentre outros. As informações de rótulos corretos foram obtidas dos sites da *Exploit Database*[5], da *Microsoft Secutiry Advisories*[6] e da *Symantec*, sendo os dois primeiros para prova de conceito e o último para *exploits* de mundo real.

Com o conjunto de dados descrito, foi possível realizar os testes relativos à hipótese h_1. Por outro lado, para h_2 e h_3, o conjunto precisou ser primeiro estendido, com o aumento do conjunto de conjunto rótulos corretos, e depois substituído por um novo. A construção do novo conjunto de rótulos envolveu o levantamento de novas fontes de dados, coleta de conteúdo e compilação dos novos rótulos, ela é abordada em detalhes na Seção 4.3.

O novo conjunto de dados, gerado para o estudo de h_3, foi coletado utilizando a busca de texto do *Twitter*, por meio da ferramenta *GetOldTweets*3 [7], filtrando por mensagens que continham a palavra-chave "CVE" no período de janeiro de 2015 a dezembro de 2018. Ao todo, foram coletadas 44.570 mensagens de 4.033 usuários diferentes, mencionando 6.643 vulnerabilidades. Para todas as vulnerabilidades mencionadas, foram coletadas informações do NVD, do site CVE-Details[8] e foram associados rótulos do novo conjunto de rótulos corretos. A construção desse conjunto de dados é abordada em detalhes na Seção 4.4.

[5] https://www.exploit-db.com/
[6] https://www.microsoft.com/en-us/msrc/technical-security-notifications
[7] https://github.com/Jefferson-Henrique/GetOldTweets-python
[8] https://www.cvedetails.com/

4.2.2 Vetor de características

Antes de apresentar o vetor de características utilizado, é preciso explicar como é representada uma instância do problema de detecção de *exploits*. No método adotado, deseja-se rotular uma vulnerabilidade como "explorada" ou "não explorada", portanto cada vulnerabilidade (ou CVE) é uma instância do problema e o vetor que a representa é o conjunto de todas as informações coletadas sobre ela. Os vetores utilizados podem ser divididos em quatro partes: (1) textos do *Twitter*; (2) estatísticas do *Twitter*; (3) dados do CVSS; (4) dados de bases públicas. Textos do *Twitter* são a representação de todas as mensagens (*tweets*) que mencionaram um determinado código CVE, essa representação é feita por meio de uma BoW elaborada por Sabottke, Suciu e Dumitras (2015), contendo 31 palavras com maior valor de informação mútua na separação das classes.

As estatísticas do *Twitter* são dados como a soma do número de *retweets*, a média da idade das contas que enviaram mensagens e a média da quantidade de seguidores dessas contas. Os dados de bases públicas contêm informações coletadas no NVD, CVE-Details e OSVDB (apenas no conjunto fornecido). Por fim, dados do CVSS contém a nota básica do CVSS e seus componentes. Como será explicado na Seção 4.4.2, no conjunto fornecido havia apenas nota e vetor do CVSS 2.0. Durante a atualização dos dados, foram incluídos CVSS 3.0, seus componentes e sub-scores para vulnerabilidades de 2015 ou mais recentes. A Tabela 3 detalha as palavras utilizadas para a BoW. A Tabela 4 detalha cada uma das demais entradas do vetor de características.

Tabela 3: Palavras da representação BoW

#	**Palavras**	#	**Palavras**	#	**Palavras**	#	**Palavras**
1	advisory	9	fix	17	microsoft	25	sample
2	bash	10	go	18	ok	26	ssl
3	beware	11	ie	19	patch	27	tested
4	blog	12	iis	20	pc	28	web
5	bug	13	info	21	poc	29	windows
6	day	14	java	22	post	30	working
7	eset	15	java	23	rce	31	xp
8	exploit	16	mcafee	24	reading		

4.2.3 Algoritmos de Classificação

Diversos trabalhos anteriores utilizaram o algoritmo de classificação SVM de *kernel* linear para detectar a existência de *exploits*, por isso ele foi escolhido como linha de base para os resultados. Conforme a Figura 7 mostra, este trabalho propõe duas formas de melhorar o classificador, escolhendo um algoritmo mais adequado e por meio do balanceamento de classes. Portanto, a construção da ferramenta foi feita de maneira modular para facilitar testes com múltiplos algoritmos de classificação e balanceamento.

Tabela 4: Vetor de características

#	Característica	Descrição
1	BoW	Texto do *Twitter* (31 entradas, vide Tabela 3)
2	tweet_count	Número de *Tweets*
3	accounts	Número de usuários
4	followers	Número de contas com mais de 460 seguidores
5	friends	Número de contas com mais de 400 amigos
6	retweets	Número de de *retweets*
7	replies	Número de de respostas
8	hashtags	Média de *hashtags* p/ *tweet*
9	urls	Média de links/*tweet*
10	mentions	Média de citações/*tweet*
11	verified	Número de contas verificadas
12	account_age	Média de idades das contas (em dias)
13	acc_tweets	Média de *tweet*/conta
14	baseScore	*Base Score* (CVSS 2.0)
15	impactScore	*Impact Subscore* (CVSS 2.0)
16	exploitabilityScore	*Exploitability Subscore* (CVSS 2.0)
17	accessVector	Vetor de ataque (Métrica CVSS 2.0)
18	accessComplexity	Complexidade (Métrica CVSS 2.0)
19	authentication	Autenticação (Métrica CVSS 2.0)
20	confidentialityImpact	Confidencialidade (Métrica CVSS 2.0)
21	integrityImpact	Integridade (Métrica CVSS 2.0)
22	availabilityImpact	Disponibilidade (Métrica CVSS 2.0)
23	baseScore	*Base Score* (CVSS 3.0)
24	impactScore	*Impact Subscore* (CVSS 3.0)
25	exploitabilityScore	*Exploitability Subscore* (CVSS 3.0)
26	attackVector	Vetor de ataque (Métrica CVSS 3.0)
27	attackComplexity	Complexidade (Métrica CVSS 3.0)
28	privilegesRequired	Privilégios necessários (Métrica CVSS 3.0)
29	userInteraction	Interação com usuário (Métrica CVSS 3.0)
30	scope	Escopo (Métrica CVSS 3.0)
31	confidentialityImpact	Confidencialidade (Métrica CVSS 3.0)
32	integrityImpact	Integridade (Métrica CVSS 3.0)
33	availabilityImpact	Disponibilidade (Métrica CVSS 3.0)
34	num_references	Número de referências no NVD
35	num_unique_references	Número de referências únicas no NVD
36	bugtraq_ref	Bugtraq entre as referências
37	secunia_ref	Secunia entre as referências
38	allow_in_summary	Palavra "allow"na descrição
39	last_mod_date_pub_date	Última modificação - Data da publicação (NVD)
40	now_last_mod_date	Data coleta - Data da publicação (NVD)
41	code-Summ	Palavra "code"na descrição
42	num_vendors	Número de fabricantes afetados
43	num_affected_products	Número de produtos afetados

Para os experimentos, foram escolhidos algoritmos de classificação convencionais como o SVM de linear, *K-Nearest Neighbors* (KNN), *Naive Bayes* e Regressão Logística. Também foram escolhidos os *ensembles Random Forest*, LightGBM e XGBoost, algoritmos amplamente utilizados em soluções vencedoras de desafios que envolvem dados estruturados no Kaggle[9]. A maior parte desses algoritmos estão disponíveis na biblioteca *scikit-learn*, porém dois métodos baseados em *ensembles* necessitaram a importação de bibliotecas específicas do *Python*, o XGBoost e o LightGBM.

O balanceamento de classes foi feito usando a biblioteca *Imbalanced-Learn* 0.4.3 [10]. Foram testados todos os algoritmos de subamostragem e sobreamostragem disponibilizados pela biblioteca. Para os experimentos, esses algoritmos foram aplicados sobre os conjuntos escolhidos para treino.

Por fim, para todos os testes que não envolviam janelas temporais, foi utilizada a validação cruzada com 10 pastas estratificadas, implementada pela classe StratifiedKFold do *scikit-learn*. Os detalhes dos algoritmos executados serão apresentados no Capítulo 5.

4.3 Construção de um novo conjunto de rótulos corretos

A hipótese h_2 afirma que existem fontes de rótulos corretos alternativas à *Symantec* para *exploits* de mundo real e o uso dessas fontes pode melhorar o desempenho do classificador. Para comprovar ou refutar essa afirmação, é necessário que um novo conjunto de rótulos corretos seja construído, comparado com os já existentes e avaliado quanto aos seus efeitos no detector de vulnerabilidades exploradas. Esta seção descreve a estratégia adotada para levantamento e compilação dessas informações.

4.3.1 Levantamento de bases de dados

Uma das primeiras atividades associadas a hipótese h_2 foi o levantamento de fabricantes de antivírus que disponibilizassem dados sobre assinaturas de *malwares* ou ameaças em geral. Uma vez encontradas, as informações sobre assinaturas foram analisadas para verificar a existência de citações a CVEs e, para os fabricantes que as continham, foi desenvolvido um método de coleta dos dados. É importe salientar que, para a comprovação de h_2 (Figura 7) não era necessário uma busca exaustiva, mas apenas a comprovação de que essas fontes alternativas existem e podem impactar o resultado do classificador.

O ponto de partida para o levantamento foi a listagem dos fabricantes a serem investigados. para isso foram escolhidos 20 fabricantes citados em diferentes listas de líderes de mercado em 2018 (OPSWAT, 2018), (Statista, 2018) e (AntivirusGuide, 2018). Foi

[9] https://www.kaggle.com/
[10] https://github.com/scikit-learn-contrib/imbalanced-learn

possível descobrir que, dentre esses, 8 disponibilizavam informações sobre assinaturas de *malwares* em seus sites. Deste grupo, foram escolhidos 5 fabricantes para terem dados coletados seguindo critério de participação de mercado e facilidade da coleta, são eles: *Avast*[11], *ESET*[12], *Kaspersky*[13], *Symantec*[14] e *Trend Micro*[15]. A Tabela 5 mostra a relação de todos os fabricantes examinados e o resultado do levantamento.

Tabela 5: Fabricantes/Produtos de antivírus examinados para a construção do novo conjunto de rótulos corretos

Fabricante/Produto	**Disponibiliza Assinaturas**	**Assinaturas Coletadas**
Avast (AVAST, 2020)	✓	✓
AVG (AVG, 2020)		
Avira (AVIRA, 2020)	✓	
BitDefender (BITDEFENDER, 2020)		
BullGuard (BULLGUARD, 2020)		
Comodo (COMODO SECURITY, 2020)		
ESET (ESET, 2020)	✓	✓
FSecure (F-SECURE, 2020)	✓	
Intrusta (INTRUSTA ANTIVIRUS, 2020)		
Kaspersky (KASPERSKY, 2020)	✓	✓
Malwarebytes (MALWAREBYTES, 2020)		
McAfee (MCAFEE, 2020)		
ReasonCore Security (REASONCORE, 2020)		
Symantec (BROADCOM, 2020a) (BROADCOM, 2020b)	✓	✓
Trend Micro (TREND MICRO, 2020)	✓	✓
Voodoosoft Voodooshield (Voodoosoft, 2020)		
Webroot iNC (Webroot, 2020)		
Windows Defender (WINDOWS DEFENDER, 2020)	✓	
Zone Alarm (ZONEALARM, 2020)		

4.3.2 Coleta e compilação dos rótulos corretos

Como mencionado na seção anterior, um dos critérios para selecionar um fabricante como fonte de informação foi a facilidade de se coletar os dados. Em geral, os fabricantes escolhidos disponibilizam as listas de assinaturas em seus sites e, para cada assinatura, existe uma página específica informando detalhes da ameaça, como nome, produto afetado, forma de contágio e de remoção. Isso permite que a coleta dos dados seja feita por meio de *web scraping* utilizando o *Python*. Fabricantes que disponibilizavam dados

[11] https://www.avast.com/
[12] https://www.eset.com/
[13] https://www.kaspersky.com/
[14] https://www.nortonlifelock.com/
[15] https://www.trendmicro.com/

apenas por meio de busca ou por páginas geradas por *JavaScript*, por exemplo, foram deixados de lado por adicionarem complexidade extra ao processo.

Utilizando a biblioteca *BeautifulSoup* 4[16] do *Python*, foram construídos *scripts* específicos para coletar todos os dados possíveis dos sites de cada fabricante e salvar em um banco de dados local, criado utilizando *SQLite* 3[17]. Posteriormente, outro *script* foi utilizado para procurar por referências a CVEs e gerar listas de vulnerabilidades exploradas. As listas foram separadas por fabricantes e, de acordo com o teste executado, elas foram combinadas para formar conjuntos de rótulos específicos. Dos fabricantes analisados, a *Symantec* é realmente a que disponibiliza o maior volume de dados, tanto sobre antivírus quanto de IDS e IPS, porém as informações de outros fabricantes combinadas chegam a 30% do volume total. Além disso, o site da *Avast* possui uma lista dedicada a assinaturas de *exploits*, já associadas com CVEs. Ao todo, foram coletadas 20.618 assinaturas de ameaças e encontradas 3.207 referências a vulnerabilidades. Excluindo-se as repetições, 2.656 vulnerabilidades foram mencionadas, das quais 1.531 eram do período estudado, de 2014 (linha de base) a 2018. A Tabela 6 detalha o volume de informação encontrado.

Tabela 6: Quantidade de assinaturas coletas e CVEs mencionados

	# Assinaturas	**%**	**# CVEs Mencionados**	**%**
Symantec	8.469	41,08%	1.123	35,02%
Symantec Attack	6.029	29,24%	1.221	38,07%
Avast	734	3,56%	734	22,89%
ESET	3.634	17,63%	44	1,37%
Kaspersky	1.252	6,07%	11	0,34%
Trend Micro	500	2,43%	74	2,31%
Total	**20.618**	—	**3.207**	—

4.4 Construção do novo conjunto de dados

Para a construção de um conjunto de dados que desse suporte à validação das hipóteses propostas neste capítulo, foi necessário desenvolver ferramentas de coleta e processamento dos dados escolhidos como fontes de informações. Como já mencionado, os primeiros testes desta pesquisa puderam ser feitos utilizando um conjunto de dados construído por outros autores, mas para conseguir um controle maior sobre os testes e comprovar algumas das hipóteses levantadas, era necessário um conjunto de dados que abrangesse um período mais longo. Esta seção trata dos dados utilizados para a extração de características para o classificador. Os rótulos corretos incluídos nesse conjunto de dados são os mesmos abordados na Seção 4.3.2, coletados e compilados em uma etapa anterior da pesquisa.

[16] https://www.crummy.com/software/BeautifulSoup/
[17] https://www.sqlite.org/

4.4.1 Ferramentas de coleta

Para coletar dados referentes às características de cada vulnerabilidade, foram desenvolvidas ferramentas específicas de acordo com a fonte da informação. Para dados provenientes do *Twitter* foram utilizadas duas bibliotecas, *GetOldTweets* 3 (HENRIQUE, 2018) e *Python-Twitter* 3.5 (TAYLOR, 2020). A primeira é uma biblioteca não-oficial do *Twitter* que é capaz de coletar mensagens antigas a partir de palavras-chave. Ela foi utilizada para obter informações como texto, data da postagem, nome do usuário que enviou a mensagem, e quantidades de *retweets*, menções e curtidas. Já a biblioteca *Python-Twitter* é parte da API oficial do site e foi utilizada para obter informações dos usuários que não são acessíveis pela ferramenta anterior, informações como quantidades de seguidores, amigos e *tweets*, idade da conta e se a conta é verificada (conta de interesse público autenticada pela plataforma). As mensagens foram obtidas procurando por ocorrências da palavra-chave "CVE" no período de janeiro de 2015 a dezembro de 2018. Dos 59.059 *tweets* coletados, 44.570 faziam referência a algum identificador CVE válido. Foram encontradas referências a 6.643 vulnerabilidades. Todos os dados coletados foram armazenados localmente em formato JSON.

Informações oficiais sobre vulnerabilidades puderam ser coletadas por meio de arquivos JSON separados por ano e disponibilizados no próprio site do NVD (NVD, 2020). Para complementar os dados, foram coletadas informações sobre os fabricantes e produtos afetados por cada vulnerabilidade utilizando *web scraping* do site *CVE-Details* (CVE Details, 2020). Nesse último caso, apenas as vulnerabilidades citadas no *Twitter* foram coletadas.

4.4.2 Compilação dos dados

Conforme discutido na Seção 4.2.2, dois grupos de características foram derivados dos dados do *Twitter*, um representando o texto e outro representando estatísticas das mensagens. Para a representação do texto, foi utilizada a mesma BoW utilizada por Sabottke, Suciu e Dumitras (2015), contendo 31 palavras com maior valor de informação mútua na separação das classes. Os *tweets* foram primeiro agrupados por vulnerabilidades, caso mais de um CVE fosse mencionado na mesma mensagem, ela se repetiria em mais de um grupo. Em seguida, para cada vulnerabilidade, concatenou-se os textos de todos os *tweets*. Não foi aplicado filtro por idioma, mas para montar a BoW foi utilizado a biblioteca SpaCy 2.2 para fazer tokenização e lematização do texto resultante em Inglês, idioma da maior parte das palavras na BoW. Para as demais características extraídas do *Twitter*, foi adotado um procedimento parecido, combinando os diferentes números (*retweets*, curtidas, compartilhamento etc.) de *tweets* agrupados por vulnerabilidades. A descrição das características foi mostrada na Tabela 4.

Os dados do NVD, por outro lado, já estavam compilados por vulnerabilidade e pre-

cisaram apenas serem lidos, filtrados e transformados em vetor de característica. Ao contrário dos dados predominantemente numéricos do *Twitter*, a maior parte dos dados do NVD são categóricos, em especial aqueles relacionados ao CVSS. Esse indicador é composto de métricas categóricas que se agregam para formar pontuações numéricas, sendo a principal delas chamada *Base Score*. Algumas vulnerabilidades trazem outras duas notas, *Temporal Score* e *Environmental Score*, mas elas são menos utilizadas e por isso não foram coletadas. O *Base Score* pode ser dividido em dois subscores: *Exploitability* e *Impact Subscore*. Neste trabalho, foi coletado o *Base Score*, seus dois subscores e as métricas que os compõem.

Capítulo 5

Experimentos e Análise dos Resultados

Neste capítulo, serão descritos os experimentos realizados para validação das hipóteses apresentadas na Seção 4.1. Cada um dos experimentos está diretamente relacionado com uma das tarefas indicadas naquela seção.

5.1 Avaliação e visão geral dos experimentos

Um ponto comum entre as hipóteses levantadas na Seção 4.1 é o intuito de se comprovar a melhora nos resultados do classificador para a tarefa de detecção de vulnerabilidades exploradas. Para isso, foi utilizado um método de avaliação baseado na comparação dos valores de *F-score*, precisão e revocação. Como já mencionado, para o problema abordado neste trabalho, um classificador é considerado mais adequado quando apresenta melhor desempenho em *F-score* ou quando exibe valor parecido de *F-score*, mas superior em precisão (menor número de falsos positivos). Conforme discutido no Capítulo 2, classificadores mais conservadores, e com alto índice de falsos positivos, poderiam ser facilmente substituídos pelo uso do *score* CVSS, por exemplo.

Portanto, escolheu-se o *F-score* como medida primária de desempenho, seguido da precisão. Como, para a maior parte dos testes, foi utilizada a validação cruzada de 10 pastas estratificadas, as medidas de precisão $\bar{p}$ e de revocação $\bar{r}$ mostradas serão relativas à média dos resultados obtidos em cada uma das pastas e a medida de *F-score* F_1 será calculada a partir desses valores. Sendo assim, para cada resultado apresentado considere:

$$F_1 = 2 \times \frac{\bar{p} \times \bar{r}}{\bar{p} + \bar{r}} \tag{10}$$

As comparações entre os resultados são submetidas ao teste t de Student para amostras dependentes (pareadas) e a diferença encontrada é considerada significante se o valor-p for inferior a 5% (0,05). Nos experimentos, as amostras são consideradas dependentes, pois, para cada execução, o conjunto de dez pastas da validação cruzada foi gerado a partir de um mesmo valor semente e, portanto, cada pasta continha sempre as mesmas instâncias do

problema. Os valores linha de base foram obtidos reproduzindo os experimentos realizados por Sabottke, Suciu e Dumitras (2015) com o conjunto de dados fornecido pelos autores.

Como representação visual dos resultados, foi escolhida a curva de precisão e revocação devido ao alto desbalanceamento do conjunto de dados. O uso de curvas *Receiver Operating Characteristic Curve* (ROC) poderiam levar a uma avaliação otimista dos resultados já que a razão de falsos positivos (falsos positivos / negativos totais) tende a ser baixa devido ao grande número de casos negativos totais. Em alguns dos experimentos, a representação gráfica traz linhas que correspondem aos subgrupos de características, ou seja, o desempenho do classificador considerando apenas as características daquele subgrupo. A Seção 4.2.2 contém a descrição do vetor de características e de seus subgrupos.

O restante deste capítulo está dividido de acordo com os experimentos realizados. Cada subseção aborda o experimento relativo a uma das tarefas elencadas no capítulo anterior, apresentando seus detalhes e resultados. Ao final do capítulo é feita uma discussão destes resultados. A Tabela 7 traz um resumo dos experimentos que serão abordados.

Experimento	**Período**	**Rótulos**	**Algoritmos**
Comparação entre algoritmos de classificação	fev/2014 a jan/2015	*Symantec*	SVM (base), Árvore de decisão, KNN, NB, RL, RF, LightGBM e XGBoost.
Comparações entre fontes de rótulos	fev/2014 a jan/2015	Todos*	LightGBM
Comparação entre algoritmos de balanceamento	fev/2014 a jan/2015	Todos*	LightGBM + ADASYN, SMOTE, AllKNN e RUS
Avaliação do desempenho do melhor classificador nos anos de 2015 a 2018	jan/2015 a dez/2018	Todos*	LightGBM + AllKNN
Comparações entre diferentes janelas temporais	jan/2015 a dez/2018	Todos*	LightGBM + AllKNN

**Symantec + Avast + Outros*

Tabela 7: Resumo dos experimentos realizados

5.2 Comparação entre algoritmos de classificação

O objetivo deste experimento é verificar se é possível melhorar o desempenho do detector de vulnerabilidades exploradas utilizando algoritmos de classificação diferentes daqueles encontrados na literatura (vide Seção 3.2). A linha de base é o classificador SVM de *kernel* linear, por isso ele foi executado e analisado primeiro. Os demais algoritmos

testados foram: Árvore de decisão, KNN, *Naive Bayes*, Regressão Logística, e os *ensembles* *Random Forest*, LightGBM e XGBoost. Desses, os algoritmos de Regressão Logística, LightGBM e XGBoost superaram a linha de base e serão analisados aqui. A tabela completa com os resultados de todos os algoritmos pode ser encontrada no Apêndice A.1.

Os testes foram executados com o conjunto de dados fornecidos por Sabottke, Suciu e Dumitras (2015), coletado entre fevereiro de 2014 e janeiro de 2015, conforme descrito na Seção 4.2.1. Os rótulos corretos são provenientes da EDB e da *Symantec*, para *exploits* prova de conceito (em inglês, *Proof of Concept* – PoC) e para *exploits* de mundo real, respectivamente. Foi utilizada a validação cruzada de 10 pastas estratificadas. A Tabela 8 mostra os resultados gerais do experimento. As Seções 5.2.1, 5.2.2 e 5.2.3 abordam detalhes sobre os algoritmos SVM, regressão logística e dos *ensembles* XGBoost e LightGBM, respectivamente. A Seção 5.2.4 faz conclusões sobre o experimento.

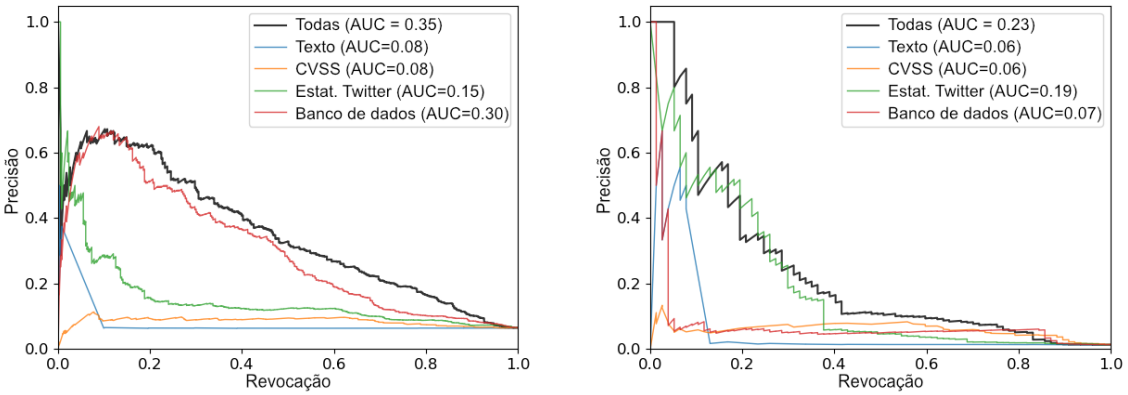
	Precisão	**Revocação**	**F-score**	**valor-p**
Baseline - SVM (PoC)	0,2075	0,7053	0,3207	—
RegLog (PoC)	0,6678	0,2434	0,3568	0,252
XGBoost (PoC)	0,7454	0,2746	0,4014	0,078
LightGBM (PoC)	0,7170	0,3293	0,4513	$< 0{,}001$
Baseline - SVM(real)	0,0632	0,7660	0,1166	—
Reg. Log. (real)	0,7	0,1857	0,2935	0,007
XGBoost (real)	0,4916	0,1535	0,2340	0,080
LightGBM (real)	0,5219	0,2196	0,3091	0,004

Tabela 8: Resultados do experimento 1

5.2.1 SVM (linha de base)

A Figura 10 mostra as curvas de precisão e revocação para a tarefa de detecção de vulnerabilidades exploradas, tanto para *exploits* de mundo real quanto para *exploits* prova de conceito. Nela, é possível perceber a superioridade em desempenho do classificador ao lidar com dados de prova de conceito. Isso pode estar ligado a dois fatores, o primeiro é que alguns dos dados coletados do NVD podem fazer referência a prova de conceito ou ao site da EDB. É possível ver que o subgrupo de bases de dados públicas representa grande parte do desempenho geral. O segundo fator é o desbalanceamento entre as classes, que é mais severo para *exploits* de mundo real. A Tabela 9 mostra a proporção de vulnerabilidades exploradas em todo o período estudado por esta dissertação. A coluna de 2014 traz o desbalanceamento das classes neste experimento.

Ainda considerando a Figura 10, é possível perceber que, para *exploits* de mundo real, as estatísticas do *Twitter* (linha verde do gráfico) compõem grande parte do desempenho geral. Essa característica foi observada em diferentes intensidades nos demais algoritmos,



(a) *Exploits* prova de conceito (b) *Exploits* de mundo real

Figura 10: Curvas de precisão e revocação para o algoritmo SVM

	2014*	**2015**	**2016**	**2017**	**2018**
CVEs mencionadas	5.865	822	776	1.292	3.753
Exploradas (real)	77 (1,3%)	71 (8,6%)	31 (4,0%)	61 (4,7%)	133 (4,2%)
Exploradas (PoC)	383 (6,5%)	115 (14,0%)	90 (11,6%)	220 (17,0%)	257 (11,9%)

*Conjunto de dados de Sabottke, Suciu e Dumitras (2015)

Tabela 9: Número de vulnerabilidades exploradas comparado com o total de CVEs mencionados

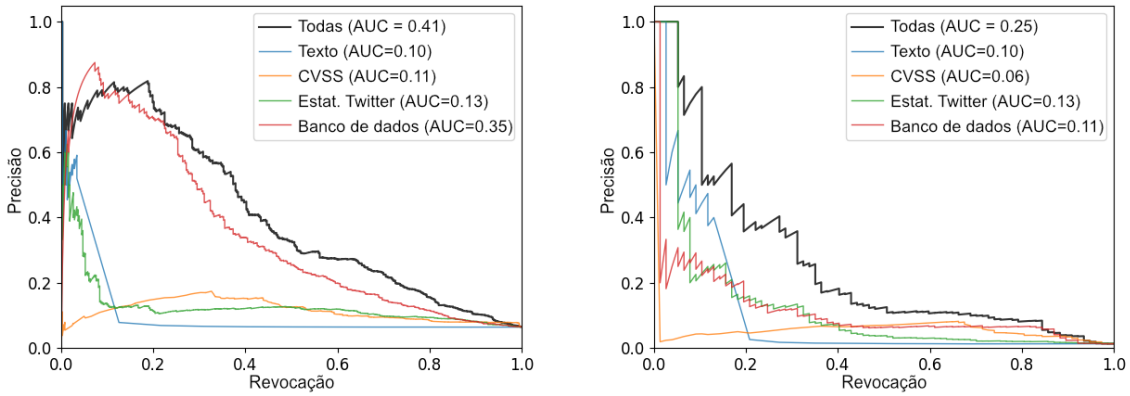
mas parece demonstrar que, para esta aplicação, informações sobre quem posta uma mensagem é mais relevante do que o próprio texto da mensagem.

Outra característica percebida na Figura 10 é a forma como o CVSS tende a ser conservador, apresentando uma curva de baixa precisão e alta revocação. Isso significa que um classificador que utilizasse apenas esse subconjunto teria um alto número de falsos positivos, como demonstrado por Sabottke, Suciu e Dumitras (2015) e Almukaynizi E. Nunes (2017). Por outro lado, o subgrupo de texto tende a ter comportamento oposto, com melhor precisão, porém baixa revocação. Isso ocorre porque a presença de palavras como "*exploit*" ou "*beware*" (cuidado, em inglês), são indicativos fortes de que uma vulnerabilidade foi explorada no mundo real.

5.2.2 Regressão Logística

Ao analisar os dados da Tabela 8, é possível perceber que o algoritmo baseado em regressão logística foi o que apresentou melhor precisão quando aplicado a *exploits* de mundo real. Porém, o seu menor desempenho em revocação fez com que seu *F-score* fosse apenas o segundo melhor. Para *exploits* prova de conceito, o algoritmo também não atingiu a significância estipulada para os testes (valor-p menor ou igual a 0,05). A Figura 11 mostra a curva de precisão e revocação do algoritmo, nela é possível perceber como o subgrupo de texto é melhor aproveitado em comparação com o SVM. É importante

salientar que, apesar de não ter apresentado o melhor *F-score*, o algoritmo de regressão logística foi o de execução mais rápida em todos os testes.

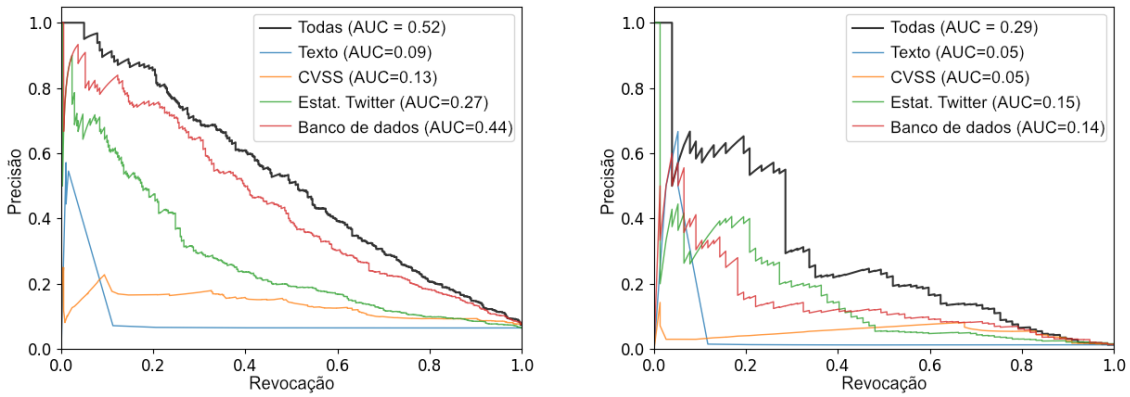


(a) *Exploits* prova de conceito (b) *Exploits* de mundo real

Figura 11: Curvas de precisão e revocação para o algoritmo Regressão Logística

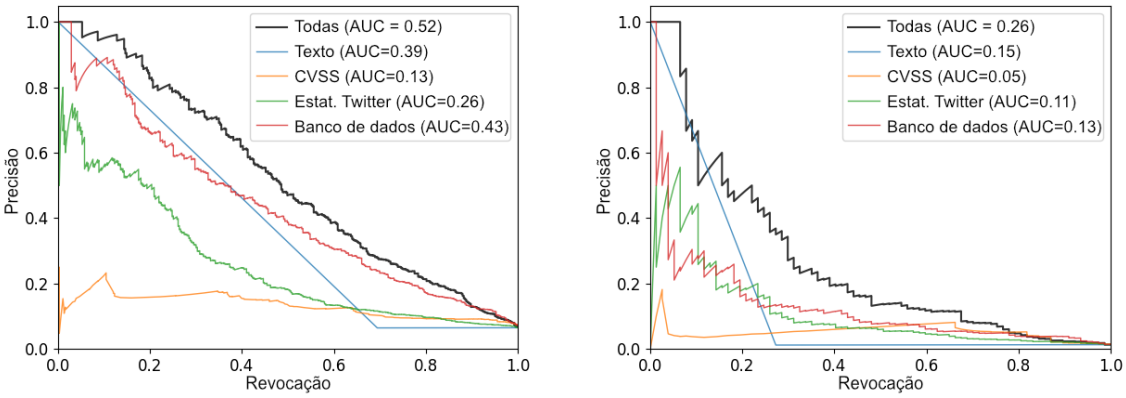
5.2.3 XGBoost e LightGBM

Ambos os *ensembles* testados superaram o desempenho da linha de base, porém o XGBoost não atingiu a significância estipulada para os testes (valor-p menor ou igual a 0,05). Por outro lado, o LightGBM apresentou os melhores resultados em termos de *F-score* tanto para prova de conceito quanto para *exploits* de mundo real. A execução do LightGBM foi também uma das mais rápidas e, ao analisar a sua curva de precisão e revocação, percebe-se que ele foi o algoritmo que mais se beneficiou conteúdo do *tweet*. As Figuras 12 e 13 mostram os gráficos de precisão e revocação para ambos os algoritmos.



(a) *Exploits* prova de conceito (b) *Exploits* de mundo real

Figura 12: Curvas de precisão e revocação para o XGBoost



(a) *Exploits* prova de conceito (b) *Exploits* de mundo real

Figura 13: Curvas de precisão e revocação para o LightGBM

5.2.4 Conclusão do experimento

Como visto, dois dos algoritmos analisados foram capazes de superar o desempenho do SVM de maneira significativa, sendo que o algoritmo LightGBM foi o algoritmo que apresentou o maior ganho de *F-score*. É importante ressaltar que ao longo dos próximos experimentos, esses algoritmos foram testados novamente para ver como se comportam e, como será mostrado, o LighGBM não só se manteve como melhor, mas também ampliou sua margem em relação à linha de base. Por último, sobre o comportamento dos subgrupos de características, foi possível perceber que, independentemente do algoritmo, o uso do CVSS é pouco relevante para *exploits* de mundo real. Por outro lado, destacaram-se, nesse cenário, os dados estatísticos do *Twitter* e informações do NVD, como número de referências, data da publicação e número de *softwares* afetados.

5.3 Comparações entre fontes de rótulos

Neste experimento, deseja-se investigar o uso de fontes alternativas de rótulos corretos para *exploits* de mundo real e verificar se elas poderiam melhorar o desempenho do classificador. Trabalhos relacionados utilizam apenas dados da *Symantec* para extrair rótulos. Na coleta de dados, abordada na Seção 4.3, foi possível encontrar quatro novas fontes: *Avast*, *ESET*, *Kaspersky* e *Trend Micro*.

A Tabela 10 mostra a quantidade de vulnerabilidades exploradas, de acordo com cada fabricante, e quantas delas foram mencionadas no *Twitter*. Os dados da *ESET*, *Kaspersky* e *Trend Micro* foram combinados e referenciados apenas como “Outros” por representarem menor volume. Ao dividir essa tabela por anos, é possível perceber como a quantidade de *exploits* reportados pela *Avast* vem caindo significativamente ao longo dos últimos anos, isso será discutido no final desta seção.

	2014	2015	2016	2017	2018	Total
Symantec Mencionadas	**77***	**43**	**29**	**58**	**131**	**338**
- Symantec Total	*90**	*261*	*247*	*219*	*364*	*1.181*
Avast Mencionadas	**112**	**33**	**3**	**4**	**4**	**156**
- Avast Total	*123*	*220*	*97*	*21*	*8*	*469*
Outros Mencionadas	**11**	**5**	**2**	**14**	**7**	**39**
- Outros Total	*14*	*19*	*3*	*15*	*7*	*58*
PoC Mencionadas	**383**	**115**	**90**	**220**	**257**	**1.065**
- PoC Total	*823*	*721*	*549*	*1124*	*981*	*4.198*

*Conjunto de dados de Sabottke, Suciu e Dumitras (2015)

Tabela 10: Número de vulnerabilidades exploradas citadas por fabricante

Uma das preocupações relacionadas ao uso de dados de assinaturas fornecidos por outros antivírus era que eles pudessem conter um grande volume de provas de conceito. O produto da *Avast*, por exemplo, é chamado de "*Exploit Protection*" o que, apesar de ser exatamente o que se buscava, poderia sugerir que ele simplesmente criasse assinaturas a partir de provas de conceitos conhecidas. Para averiguar essa possibilidade, os códigos CVEs citados pelas novas fontes foram comparados com aqueles citados pela EDB e pela *Symantec*. A Figura 14 mostra a intersecção desses grupos e aponta que, entre 2014 e 2018, 321 *exploits* foram mencionados apenas pelas novas fontes e não foram citados pela EDB ou pela *Symantec*, indicando assim se tratar de informação nova e útil. Pela figura, também é possível perceber que, de 2014 a 2018, pelo menos 351 vulnerabilidades exploradas em mundo real não foram mencionadas pela *Symantec*. Além disso, ao considerar as vulnerabilidades anteriores à 2014 e não retratadas na Figura 14, 519 delas foram exploradas mas não são citadas pela *Symantec*. É importante salientar que isso não significa que os produtos da empresa não detectem estes *exploits*, mas sim que as vulnerabilidades exploradas por eles não são especificadas nas descrições de suas assinaturas.

Após essa análise, foi construído um novo conjunto de rótulos corretos contendo esses dados e, para verificar como ele afetaria o classificador, foi elaborado um novo experimento. Nele, o modelo de decisão era treinado com rótulos provenientes de um único fabricante, mas era sempre testado com o conjunto combinado de rótulos corretos, onde a menção de um CVE por qualquer fabricante o rotulava como "explorado". Para todas as execuções, os conjuntos de testes e treinos eram sempre os mesmos, permitindo a comparação direta dos resultados. O objetivo foi mostrar como a utilização de dados de uma única fonte por trabalhos anteriores pode ter afetado positivamente os seus resultados, uma vez que ignora parte dos *exploits* reais. Para efeito de comparação com a linha de base, o teste foi executado apenas para o ano de 2014, utilizando as características extraídas por Sabottke, Suciu e Dumitras (2015).

A Tabela 11 exibe os resultados do experimento utilizando o melhor algoritmo testado, o LightGBM. É possível perceber que o classificador treinado com dados da *Symantec* teve

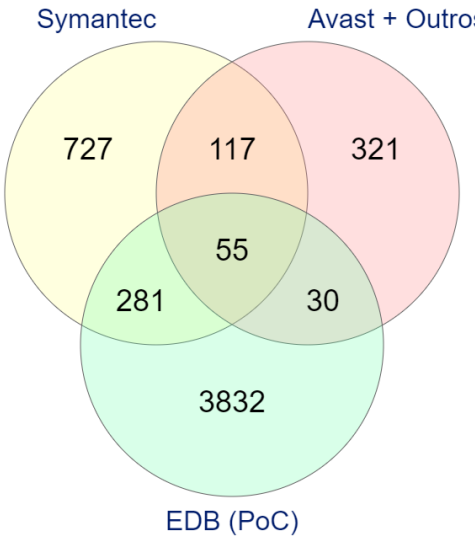


Figura 14: Intersecção entre as listas de *exploits*

Fonte: O autor

	Precision	**Recall**	**F-score**	**Valor-p**
Symantec	0,5605	0,1233	0,2021	–
Avast	0,5242	0,2209	0,3109	0,086
Sym + Avast	0,5238	0,2795	0,3645	0,046
Sym + Avast + Outros	0,5458	0,2844	0,3740	0,032

Tabela 11: Resultados com treino em múltiplas fontes de rótulos corretos

seu desempenho reduzido em relação ao experimento anterior, mostrado na Tabela 8 da Seção 5.2. Isso ocorre porque suas amostras de treino, agora, contêm rótulos falso-negativos quando comparado com o novo conjunto de rótulos. Por outro lado, ao considerar dados de todos os fabricantes para treino e teste, foi possível obter aumento no *F-score*. Esse comportamento se repete ao considerar o uso apenas de dados da *Avast*, o que sugere que vulnerabilidades mencionadas por um determinado fabricante de antivírus podem conter características semelhantes entre si e, ao serem usadas isoladamente para treino, podem enviesar o classificador para essas características. Um trabalho futuro poderia investigar quais características se repetem nos *exploits* reportados por cada fabricante de antivírus.

O experimento permite concluir que o uso de múltiplos fabricantes de antivírus como fonte para rotular vulnerabilidades como exploradas no mundo real é crucial para qualquer aplicação que utilize aprendizado de máquina para detectar *exploits*. Ainda assim, é possível perceber que o número de vulnerabilidades citadas por fontes alternativas à *Symantec* vem caindo drasticamente nos últimos anos, o que pode significar que os resultados de experimentos com dados mais recentes, e até mesmo resultados de trabalhos futuros, podem ficar enviesados a esse fabricante e serem menos precisos.

5.4 Comparação entre algoritmos de balanceamento

Neste experimento, o objetivo é reduzir o efeito negativo do problema evidenciado pela Tabela 9, o alto desbalanceamento entre as classes. Para isso, foram testados diversos algoritmos de balanceamento de classes disponíveis na biblioteca *Imbalanced-Learn*[1], utilizando técnicas de subamostragem e sobreamostragem, dentre as quais as que mais se destacaram foram: ADASYN *Sampling*, SMOTE, AllKNN e RUS. Esses algoritmos foram executados nas instâncias de treino de cada uma das 10 pastas da validação cruzada, enquanto os grupos de testes permaneceram inalterados. Este experimento foi realizado usando o conjunto de dados de 2014 e o novo conjunto de rótulos corretos apresentado no experimento anterior. Como linha de base, foi utilizado o melhor resultado do experimento anterior sem a aplicação de balanceamento, uma vez que o trabalho também não o fez.

***Exploits* PoC**

	SVM			LightGBM		
	Precisão	**Revocação**	**F-score**	**Precisão**	**Revocação**	**F-score**
Linha de base	0,2075	0,7053	0,3207	0,7170	0,3293	0,4513
ADASYN[I]	0,1829	0,7781	0,2961	0,5811	0,4157	Gray0,4846
SMOTE[I]	0,2084	0,7157	0,3228	0,5722	0,4260	0,4884
AllKNN[II]	0,2092	0,7312	0,3253	0,5445	0,4472	Gray0,4911
RUS[II]	0,2273	0,6321	0,3343	0,1924	0,7627	0,3073

***Exploits* reais**

	SVM			LightGBM		
	Precisão	**Revocação**	**F-score**	**Precisão**	**Revocação**	**F-score**
Linha de base	0,2244	0,8278	0,3531	0,5458	0,2844	Gray0,3740
ADASYN[I]	0,2271	0,8285	0,3565	0,4640	0,3774	0,4162
SMOTE[I]	0,2391	0,8044	0,3686	0,4867	0,3785	0,4258
AllKNN[II]	0,2271	0,8285	0,3565	0,4956	0,5267	Gray**0,5107**
RUS[II]	0,2330	0,8303	0,3639	0,2212	0,8642	0,3523

[I]Técnica de sobreamostragem
[II]Técnica de subamostragem

Tabela 12: Resultados para *exploits* PoC e de mundo real para o balanceamento de classes abordado no Experimento 3

A Tabela 12 mostra o desempenho dos classificadores utilizando os algoritmos linha de base e o aquele que apresentou o melhor desempenho no experimento, o LightGBM. Como se pode ver, os algoritmos tiveram comportamentos bem distintos entre si. Enquanto o SVM não demonstrou benefício algum no uso de balanceamento de classes, o LightGBM apresentou considerável melhora (com valor-p de 0,001) ao utilizar o AllKNN e, mais uma vez, foi o melhor dentre os classificadores testados. Outros algoritmos de balanceamento

[1] https://github.com/scikit-learn-contrib/imbalanced-learn

também foram capazes de aumentar o F-score do LightGBM, mas não apresentaram significância estatística nessa melhora.

De maneira geral, o AllKNN foi capaz de aumentar o desempenho de todos os métodos de classificação testados, com exceção do SVM, tanto para *exploits* prova de conceito quanto para *exploits* de mundo real. É importante mencionar que quando testado somente com os rótulos da *Symantec*, estes ganhos foram menores, talvez devido ao maior desbalanceamento.

5.5 Avaliação do desempenho do melhor classificador nos anos de 2015 a 2018

Neste experimento, a intenção foi verificar se as conclusões obtidas a partir dos dados de 2014 podem ser generalizados também para os anos seguintes. Para isso, foi construído um novo conjunto de dados compreendendo o período de 2015 a 2018, conforme explicado na Seção 4.4. As Tabelas 13 e 14 trazem, para cada ano, os resultados obtidos pelo melhor classificador testado, o LightGBM, com a utilização do método AllKNN de balanceamento de classes. A Figura 15 mostra as curvas de precisão e revocação também para os anos testados.

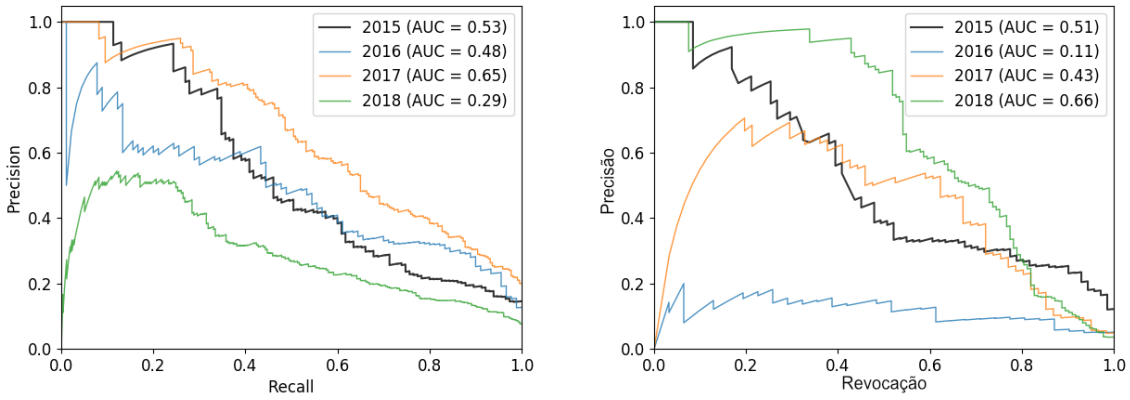
	Precisão	**Revocação**	**F-score**
2015	0,5617	0,5114	0,5353
2016	0,5963	0,5333	0,5630
2017	0,5693	0,6500	0,6070
2018	0,4609	0,2882	0,3546

Tabela 13: Resultados para o novo conjunto de dados com *exploits* prova de conceito

	Precisão	**Revocação**	**F-score**
2015	0,6048	0,6282	0,6163
2016	0,2089	0,2333	0,2204
2017	0,5419	0,5681	0,5547
2018	0,6999	0,5529	0,6178

Tabela 14: Resultados para o novo conjunto de dados com *exploits* de mundo real

Os resultados apresentam alguma variação, mas, com exceção do ano de 2016 para exploits de mundo real, os valores de F-score permanecem entre 0,51 e 0,61. Ao compará-los com os dados da Tabela 9, é possível inferir que exista uma correlação entre o número de vulnerabilidades exploradas citadas no *Twitter* e o desempenho do classificador, sendo 2016 o ano com menor número de menções a *exploits* de mundo real.



(a) *Exploits* prova de conceito (b) *Exploits* de mundo real

Figura 15: Curvas de precisão e revocação para detecção de *exploits* nos anos de 2015 a 2018

5.6 Comparações entre diferentes janelas temporais

O último experimento executado procurava responder se o treino com dados de um período maior que um ano poderia melhorar os resultados. A ideia por trás deste experimento era descobrir se o desempenho de modelos treinados com dados coletados ao longo um ano, período utilizado em diversos trabalhos relacionados, era tão bom quanto o desempenho de modelos treinados com períodos maiores. Para isso, foi elaborado uma série de avaliações onde o classificador foi treinado com dados de anos anteriores a 2018 e testado com dados de 2018, depois treinado com dados de anos anteriores a 2017 e testado com dados de 2017 e por fim treinado com dados anteriores a 2016 e testado com dados de 2016. Os testes foram realizados para *exploits* de mundo real e prova de conceito, utilizando o algoritmo LightGBM para classificação, AllKNN para balanceamento e com o conjunto completo e atualizado de rótulos corretos. Foi utilizada uma execução simples do classificador, sem validação cruzada, com exceção dos valores linha de base, obtidos no experimento anterior. O uso desse tipo de validação é impossibilitado pela divisão temporal do experimento, que não permitiria misturar elementos das janelas de treino e teste originais. Os resultados são mostrados nas Tabelas 16 e 15.

Em comparação com o teste sem divisão temporal aplicado a um único ano, os resultados mostram uma diminuição de desempenho do classificador quando treinado com dados diferentes do ano de teste. Essa diferença é especialmente grande no experimento com *exploits* de mundo real. Isso pode sugerir que exista alguma relação entre CVEs do mesmo ano que facilite o aprendizado do modelo. Quanto ao tamanho da janela temporal, não houve grande diferença entre os valores de precisão, revocação e *F-score*. Em certa medida, alguma diferença de desempenho era prevista ao utilizar critérios temporais para divisão de treino e teste (BULLOUGH et al., 2017) (ALMUKAYNIZI E. NUNES, 2017), porém esperava-se que a diferença não seria tão acentuada.

Treino	Teste	Precisão	Revocação	F-score
2018 (linha de base)	2018	0,4609	0,2882	0,3546
2017	2018	0,3555	0,2918	0,3205
2016 2017	2018	0,3831	0,2296	0,2871
2015 2016 2017	2018	0,3173	0,2568	0,2839
2017 (linha de base)	2017	0,5693	0,6500	0,6070
2016	2017	0,5568	0,2227	0,3182
2015 2016	2017	0,4797	0,2682	0,3440
2016 (linha de base)	2016	0,5963	0,5333	0,5630
2015	2016	0,4133	0,3444	0,3758

Tabela 15: Resultados para treinos com diferentes janelas temporais com *exploits* prova de conceito

Treino	Teste	Precisão	Revocação	F-score
2018 (linha de base)	2018	0,6999	0,5529	0,6178
2017	2018	0,3333	0,1654	0,2211
2016 2017	2018	0,3051	0,1353	0,1875
2015 2016 2017	2018	0,3208	0,1278	0,1828
2017 (linha de base)	2017	0,5419	0,5681	0,5547
2016	2017	0,6667	0,0984	0,1714
2015 2016	2017	0,8000	0,1967	0,3158
2016 (linha de base)	2016	0,2089	0,2333	0,2204
2015	2016	0,1818	0,0645	0,0952

Tabela 16: Resultados para treinos com diferentes janelas temporais com *exploits* de mundo real

Uma das possíveis causas desse fenômeno poderia ser o surgimento de algum conhecimento criado somente no ano do teste. Dois exemplos seriam as palavras "crypto" (de criptografia ou criptomoeda) e "ransom" (resgate, em inglês), que dificilmente estariam associadas juntas a uma vulnerabilidade antes da popularização dos crypto ransomwares, em 2013 (TAILOR; PATEL, 2017). Para averiguar se o fenômeno observado tinha relação com o surgimento de novos termos ou algum tipo de conhecimento criado somente no ano do teste, foram realizados diversos testes com combinações diferentes de janelas temporais para treino e teste. Em todos os casos o desempenho foi inferior aos testes com um único ano.

Ao examinar as vulnerabilidades rotuladas como "exploradas", foi possível concluir que esse fenômeno pode estar ligado ao fato de que vulnerabilidades são, muitas vezes, divulgadas em grupos e podem estar todas relacionadas a um único incidente ou *malware*. No conjunto de rótulos coletados, 731 vulnerabilidades foram divulgadas em grupos com pelo menos uma outra vulnerabilidade. O *ransomware* WannaCry, por exemplo, está relacionado a seis vulnerabilidades diferentes (CVE-2017-0143, CVE-2017-0144, CVE-2017-

0145, CVE-2017-0146, CVE-2017-0147 e CVE-2017-0148), todas com descrições parecidas e afetando os mesmos produtos. Portanto, modelos que utilizem divisão aleatória, como a validação cruzada feita para o resultado base de 2018, talvez se beneficiem desses conjuntos de vulnerabilidades relacionadas. Sendo assim, é importante que, em trabalhos futuros, detectores de *exploits* levem em conta esses grupos de vulnerabilidades ao criar instâncias do problema.

A conclusão deste experimento é que, para o modelo de testes *offline* adotado, não há benefício em utilizar janelas temporais mais longas que um ano, porém a existência de grupos de vulnerabilidades divulgadas dentro de um ano parece interferir nos resultados de experimentos que utilize divisão aleatória entre grupos de treino e teste.

5.7 Discussão

Ao final da etapa de experimentação, foi possível comprovar duas das hipóteses específicas levantadas no Capítulo 4, h_1 e h_2. Os experimentos foram capazes de demonstrar que:

1. O uso do algoritmo de classificação LightGBM e de balanceamento de classes AllKNN são capazes de melhorar o desempenho de detectores de *exploits* aplicados ao *Twitter* e superar o valor de *F-score* SVM linear utilizado por grande parte dos trabalhos anteriores. Por essa razão, aceita-se a hipótese h_1.

2. Existem fontes de rótulos alternativas à *Symantec* para o problema de detecção de *exploits* de mundo real. Essas fontes possuem informações relevantes e são capazes de propiciar aumento de *F-score* nos classificadores que as usem para treino. Por essa razão, aceita-se a hipótese h_2.

3. O uso de janelas temporais maiores que um ano não propicia melhor desempenho de detectores de *exploits* aplicados ao *Twitter*, que utilizem o método *offline* de treinamento. Por essa razão, rejeita-se a hipótese h_3.

Além das descobertas sobre as hipóteses, foi possível constatar as seguintes descobertas secundárias:

1. O uso de estatísticas sobre usuários e mensagens do *Twitter* podem ser mais importantes que o texto das mensagens.

2. O ano de 2018 teve 133 vulnerabilidades exploradas mencionadas no *Twitter*, o maior número absoluto desde 2014.

3. O ano de 2016 teve 31 vulnerabilidades exploradas mencionadas no *Twitter*, o menor número absoluto desde 2014.

4. Com exceção do ano de 2016, o desempenho do método utilizado para a detecção de *exploits* permaneceu constante entre 2014 e 2018.

5. Apesar de fornecerem dados importantes, fontes de rótulos alternativas à *Symantec* vêm divulgando cada vez menos informações sobre assinaturas de *malwares* e as vulnerabilidades afetadas, o que pode prejudicar os resultados de trabalhos futuros.

6. O agrupamento de vulnerabilidades relacionadas talvez seja uma prática aconselhável na implementação de detectores de *exploits* que utilizem aprendizado de máquina a fim de se evitar treino e teste em CVEs de um mesmo incidente.

Capítulo 6

Conclusão

Este trabalho abordou o problema da detecção de *exploits* usando técnicas de aprendizado de máquina aplicadas a dados obtidos no *Twitter* e em bases públicas de vulnerabilidades. Os Capítulos 2 e 3 abordaram a revisão da literatura correlata, a partir da qual foi possível compreender o estado da arte e identificar aspectos ainda pouco explorados no tema. Esses capítulos demonstraram como os principais trabalhos relacionados se dividem em dois grupos: trabalhos que abordam a detecção/previsão de eventos relacionados à segurança e trabalhos que abordam a detecção/predição de *exploits*. Esta dissertação seguiu pela segunda abordagem. O estudo da fundamentação teórica permitiu identificar possíveis contribuições na escolha de algoritmos de classificação, na aplicação de técnicas de balanceamento de classes, no uso de novas fontes para rótulos e no uso de períodos mais longos que os usados em trabalhos relacionados.

O Capítulo 4 tratou da proposta desenvolvida com base no levantamento das contribuições, sobre as quais foi elaborada a hipótese geral. Essa hipótese afirma que os resultados de trabalhos relacionados anteriores poderiam ser superados, em termos de *F-score* e precisão, com o uso de novos algoritmos de classificação e novas fontes para rótulos. A partir da hipótese geral, foram elaboradas três hipóteses específicas que tratam da escolha do modelo de classificação, das novas fontes de rótulos e do uso de janelas temporais maiores para treino. Cada hipótese específica deu origem a tarefas de validação, que puderam ser convertidas em experimentos objetivos para comprovação da hipótese principal. Ainda no Capítulo 4, foram descritos os procedimentos e as ferramentas usadas para a coleta dos dados e desenvolvimento de uma solução capaz de classificar vulnerabilidades como "explorada" ou "não explorada".

O Capítulo 5 apresentou os experimentos objetivos, desenvolvidos com base nas tarefas descritas no capítulo anterior, e seus resultados. O primeiro experimento aplicou diferentes algoritmos de classificação para o problema e constatou que o ensemble LightGBM é significativamente melhor, em termos de *F-score*, que o algoritmo base usado em diversos trabalhos anteriores, o SVM. A seção também discutiu como os classificadores foram capazes de adquirir melhor conhecimento com dados estatísticos do *Twitter* do que com

os textos propriamente ditos.

O segundo experimento comparou, para a detecção de *exploits* de mundo real, o desempenho de modelos treinados com rótulos de apenas uma empresa (*Symantec* ou *Avast*) com o desempenho do classificador treinado com rótulos combinados de várias empresas. Para esse experimento, foi criado um novo conjunto de rótulos corretos, coletados a partir dos sites dos seguintes antivírus: *Avast*, *ESET*, *Kaspersky*, *Symantec* e *Trend Micro*. Constatou-se o aumento no desempenho quando treinado com o conjunto de rótulos combinados. O experimento ainda mostrou que trabalhos anteriores que basearam seus rótulos somente nas assinaturas da *Symantec* podem ter ignorado uma grande quantidade de vulnerabilidades exploradas em mundo real.

O terceiro experimento realizado aplicou algoritmos de balanceamento de classes durante o treino do classificador. Mais uma vez foi possível constatar significante melhora em *F-score* em comparação com a linha de base, sobretudo com o algoritmo AllKNN. O quarto experimento utilizou um conjunto de dados que abrangia os anos de 2015 a 2018. Nele, comparou-se o desempenho do classificador ao longo desses anos e constatou-se que ele permaneceu constante com exceção do ano de 2016, quando houve poucas menções de vulnerabilidades exploradas.

O último experimento também utilizou o conjunto de dados de 2015 a 2018 e tratou de janelas temporais, treinando e testando o classificador em diferentes períodos. Foi possível constatar que o desempenho de classificadores treinados e testados dentro de um mesmo ano, utilizando a validação cruzada, tende a ser maior que do que o desempenho daqueles que utilizam divisão temporal para treino e teste. Neste experimento também foi possível descobrir que diversas ameaças de software relacionam mais de uma vulnerabilidade explorada, o que, se não for corretamente endereçado, pode causar efeito de treino e teste sobre mesmos dados. Por fim, o Capitulo 5 conclui que foi possível comprovar duas das hipóteses específicas e rejeitou-se a terceira. Ainda assim, a hipótese geral do trabalho pode ser comprovada, uma vez que diversos experimentos puderam melhorar o desempenho do classificador na tarefa de detecção de *exploits*, como proposto.

6.1 Principais Contribuições

As principais contribuições deste trabalho foram:

- A identificação de algoritmos que melhoram o desempenho na detecção de *exploits* usando dados do *Twitter*, sendo eles o LightGBM para a classificação e o AllKNN para o balanceamento de classes;

- A comparação inédita da qualidade dos rótulos extraídos de assinaturas da *Symantec* com rótulos extraídos de outros fabricantes de antivírus para a detecção de *exploits* de mundo real;

- ❏ A demonstração de que o uso de rótulos extraídos de um único fabricante de antivírus pode enviesar os resultados e afetar negativamente a performance do classificador em cenário real;

- ❏ A demonstração de que, para este problema, o treino com janelas temporais mais longas não beneficia o classificador.

Além das contribuições principais, o trabalho também proveu algumas descobertas importantes para a área e que podem ajudar trabalhos futuros:

- ❏ A técnica proposta tem desempenho constante quando considerados os anos de 2014 a 2018, com exceção de 2016, ano com menor número absoluto de menções de vulnerabilidades exploradas no *Twitter*;

- ❏ O volume de informação sobre vulnerabilidades exploradas fornecidas por fabricantes de antivírus alternativos à *Symantec*, apesar de representarem 30% do total no período estudado neste trabalho, vem caindo bastante, o que pode comprometer os resultados de trabalhos futuros;

- ❏ Vulnerabilidades exploradas são, frequentemente, divulgadas em grupos de vulnerabilidades muito semelhantes. Por essa razão, é ideal que sistemas de aprendizado de máquina levem esses grupos em consideração ao criar grupos de treino e teste.

6.2 Trabalhos Futuros

Este trabalho se propôs a lidar apenas com a detecção de *exploits* com o uso de dados do *Twitter*, porém a construção dele permitiu a percepção de aspectos ainda não estudados em diversas áreas relacionadas. Alguns desses aspectos podem ser objeto de estudo de trabalhos futuros, são eles:

- ❏ O aprimoramento das características usadas, com a utilização de técnicas de *feature importance*, para melhor compreensão dos resultados obtidos com os subgrupos de características.

- ❏ A criação de um modelo de aprendizado *online* que utilize as contribuições deste trabalho relacionadas a escolha do algoritmo de classificação, algoritmo de balanceamento de classes e o conjunto estendido de rótulos para vulnerabilidades exploradas no mundo real;

- ❏ O monitoramento do NVD para entender a ordem em que as informações se tornam disponíveis no site, o que ajudaria a criar experimentos mais realistas para modelos *online*;

- Melhoria na escolha das palavras-chave para busca dos *tweets* relacionados com, por exemplo, o uso de técnicas de agrupamento;
- Um levantamento extenso, mais completo do que aquele feito por este trabalho, dos fabricantes de antivírus, IDS e IPS que fornecem descrições de suas assinaturas e as relacionem com códigos de vulnerabilidades. Esse levantamento é essencial para melhorar a detecção de vulnerabilidades exploradas em mundo real, independentemente da técnica utilizada;
- Com base no item anterior, seria possível também relacionar que tipo de similaridade possuem as vulnerabilidades reportadas por um mesmo fabricante. Esse trabalho permitiria detectar possíveis vieses nos resultados de estudos que utilizem tais fabricantes.

Referências

ABLON, L.; BOGART, A. **Zero Days, Thousands of Nights: The Life and Times of Zero-Day Vulnerabilities and Their Exploits**. Santa Monica, California, USA: Rand Corporation, 2017.

ALMUKAYNIZI E. NUNES, K. D. M. S. J. S. P. S. M. Proactive identification of exploits in the wild through vulnerability mentions online. In: **2017 International Conference on Cyber Conflict (CyCon U.S.)**. [S.l.: s.n.], 2017. p. 82–88. <https://doi.org/10.1109/CYCONUS.2017.8167501>.

ALMUKAYNIZI, M. et al. Predicting cyber threats through hacker social networks in darkweb and deepweb forums. In: ACM. **Proceedings of the 2017 International Conference of The Computational Social Science Society of the Americas**. [S.l.], 2017. p. 12. <https://doi.org/10.1145/3145574.3145590>.

AntivirusGuide. **Best Antivirus 2018**. 2018. Disponível em: <https://www.antivirusguide.com/best-antivirus/>.

AVAST. **Exploit Protection**. 2020. Disponível em: <https://www.avast.com/exploit-protection.php>.

AVG. **AVG 2020 | Antivírus, VPN e ajustes grátis para todos seus dispositivos**. 2020. Disponível em: <https://www.avg.com/>.

AVIRA. **Avira Virus Lab**. 2020. Disponível em: <https://www.avira.com/en/support-virus-lab>.

BERRY, M. W.; KOGAN, J. **Text mining: applications and theory**. [S.l.]: John Wiley & Sons, 2010.

BILGE, L.; DUMITRAS, T. Before we knew it: an empirical study of zero-day attacks in the real world. In: ACM. **Proceedings of the 2012 ACM conference on Computer and communications security**. Raleigh, North Carolina, USA, 2012. p. 833–844. <https://doi.org/10.1145/2382196.2382284>.

BITDEFENDER. **BitDefender - Global Leader in Cybersecurity**. 2020. Disponível em: <https://www.bitdefender.com/>.

Bleeping Computer. **Exploit Affecting Tor Browser Burned In A Tweet**. 2018. Disponível em: <https://www.bleepingcomputer.com/news/security/exploit-affecting-tor-browser-burned-in-a-tweet/>.

_____. **Researchers Demo PoC For Remote Desktop BlueKeep RCE Exploit**. 2019. Disponível em: <https://www.bleepingcomputer.com/news/security/researchers-demo-poc-for-remote-desktop-bluekeep-rce-exploit/>.

BOZORGI, M. et al. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In: **Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: ACM, 2010. (KDD '10), p. 105–114. ISBN 978-1-4503-0055-1.

BROADCOM. **Attack Signatures**. 2020. Disponível em: <https://www.broadcom.com/support/security-center/attacksignatures>.

_____. **Threats Risks**. 2020. Disponível em: <https://www.broadcom.com/support/security-center/a-z>.

BULLGUARD. **BullGuard 2020 | Antivirus and VPN solutions for Windows, MAC and Android**. 2020. Disponível em: <https://www.avira.com/en/support-virus-lab>.

BULLOUGH, B. L. et al. Predicting exploitation of disclosed software vulnerabilities using open-source data. **CoRR**, 2017. <https://doi.org/10.1145/3041008.3041009>.

BURKOV, A. **The Hundred-Page Machine Learning Book**. [S.l.]: Andriy Burkov, 2019. ISBN 9781999579517.

CHAWLA, N. V. et al. Smote: synthetic minority over-sampling technique. **Journal of artificial intelligence research**, v. 16, p. 321–357, 2002. <https://doi.org/10.1613/jair.953>.

CHEN, H. et al. Using twitter to predict when vulnerabilities will be exploited. In: **Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery Data Mining**. New York, NY, USA: Association for Computing Machinery, 2019. (KDD '19), p. 3143–3152. ISBN 9781450362016. <https://doi.org/10.1145/3292500.3330742>.

CHEN, T.; GUESTRIN, C. Xgboost: A scalable tree boosting system. In: **Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining**. New York, NY, USA: Association for Computing Machinery, 2016. (KDD '16), p. 785–794. ISBN 9781450342322. <https://doi.org/10.1145/2939672.2939785>.

COMODO SECURITY. **Comodo: Cloud Native Cyber Security Platform**. 2020. Disponível em: <https://www.comodo.com/>.

COPPENS, B.; SUTTER, B. D.; BOSSCHERE, K. D. Protecting your software updates. **IEEE Security & Privacy**, IEEE, v. 11, n. 2, p. 47–54, 2013.

CVE Details. **Browse Vulnerabilities By Date**. 2020. Disponível em: <https://www.cvedetails.com/browse-by-date.php>.

ESET. **ESET Virusradar**. 2020. Disponível em: <https://www.virusradar.com/en/threat_encyclopaedia>.

F-SECURE. **F-Secure Labs**. 2020. Disponível em: <https://www.f-secure.com/v-descs/index.shtml>.

FELDMAN, R.; SANGER, J. et al. **The text mining handbook: advanced approaches in analyzing unstructured data**. New York, NY, USA: Cambridge University Press, 2007.

FERNÁNDEZ, A. et al. **Learning from Imbalanced Data Sets**. [S.l.]: Springer International Publishing, 2018. ISBN 9783319980744.

Google Application Security. **Google Vulnerability Reward Program (VRP) Rules**. 2020. Disponível em: <https://www.google.com/about/appsecurity/reward-program/>.

GUTTMAN, B.; ROBACK, E. A. **An introduction to computer security: the NIST handbook**. [S.l.]: DIANE Publishing, 1995.

Haibo He et al. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. In: **2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)**. [S.l.: s.n.], 2008. p. 1322–1328. <https://doi.org/10.1109/IJCNN.2008.4633969>.

HENRIQUE, J. **GitHub – Jefferson Henrique/GetOldTweets-python: A project written in Python to get old tweets, it bypass some limitations of Twitter Official API**. 2018. Disponível em: <https://github.com/Jefferson-Henrique/GetOldTweets-python>.

HUANG, K. et al. Poster: Diversity or concentration? hackers' strategy for working across multiple bug bounty programs. In: **37th IEEE Symposium on Security and Privacy (S&P)**. San Jose, CA, USA: [s.n.], 2016.

INTRUSTA ANTIVIRUS. **Intrusta Antivirus: Simplifying Security**. 2020. Disponível em: <https://intrusta.com/>.

KAO, A.; POTEET, S. R. **Natural language processing and text mining**. [S.l.]: Springer Science & Business Media, 2007.

KASPERSKY. **Kaspersky Threats**. 2020. Disponível em: <https://threats.kaspersky.com/>.

KE, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. In: **Advances in neural information processing systems**. [S.l.: s.n.], 2017. p. 3146–3154.

KHANDPUR, R. P. et al. Crowdsourcing cybersecurity: Cyber attack detection using social media. In: **Proceedings of the 2017 ACM on Conference on Information and Knowledge Management**. New York, NY, USA: Association for Computing Machinery, 2017. p. 1049–1057. <https://doi.org/10.1145/3132847.3132866>.

MALWAREBYTES. **Malwarebytes Cybersecurity for Home and Business**. 2020. Disponível em: <https://malwarebytes.com/>.

MARIN, E. et al. Community finding of malware and exploit vendors on darkweb marketplaces. In: IEEE. **Data Intelligence and Security (ICDIS), 2018 1st International Conference on**. South Padre Island, TX, USA, 2018. p. 81–84. <https://doi.org/10.1109/ICDIS.2018.00019>.

MARTINO, S. D. et al. Towards exploiting social networks for detecting epidemic outbreaks. **Global Journal of Flexible Systems Management**, Springer, v. 18, n. 1, p. 61–71, 2017. <https://doi.org/10.1007/s40171-016-0148-y>.

MCAFEE. **McAfee | Soluções de segurança para nuvem, terminais e antivírus**. 2020. Disponível em: <https://www.mcafee.com>.

Microsoft Security Response Center. **Microsoft Bug Bounty Program**. 2020. Disponível em: <https://www.microsoft.com/en-us/msrc/bounty>.

MITCHELL, T. **Machine Learning**. [S.l.]: McGraw-Hill Education, 1997. (McGraw-Hill international editions - computer science series). ISBN 9780070428072.

MITTAL, S. et al. Cybertwitter: Using twitter to generate alerts for cybersecurity threats and vulnerabilities. In: **2016 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)**. [S.l.: s.n.], 2016. p. 860–867. <https://doi.org/10.1109/ASONAM.2016.7752338>.

NAYAK, K. et al. Some vulnerabilities are different than others. In: **International Workshop on Recent Advances in Intrusion Detection**. Gothenburg, Sweden: Springer, 2014. p. 426–446. <https://doi.org/10.1007/978-3-319-11379-1_21>.

NIST. **NVD - CVE-2019-0708**. 2019. Disponível em: <https://nvd.nist.gov/vuln/detail/CVE-2019-0708>.

_____. **NVD - General Information**. 2020. Disponível em: <https://nvd.nist.gov/general>.

NVD. **NVD Data Feeds**. 2020. Disponível em: <https://nvd.nist.gov/vuln/data-feeds>.

OPSWAT. **Windows Anti-Malware Market Share Report, October 2018**. 2018. Disponível em: <https://www.opswat.com/blog/windows-anti-malware-market-share-report-october-2018>.

POBLETE, B. et al. Robust detection of extreme events using twitter: worldwide earthquake monitoring. **IEEE Transactions on Multimedia**, IEEE, v. 20, n. 10, p. 2551–2561, 2018. <https://doi.org/10.1109/TMM.2018.2855107>.

QUEIROZ, A.; KEEGAN, B.; MTENZI, F. Predicting software vulnerability using security discussion in social media. **European Conference on Information Warfare and Security, ECCWS**, p. 628–634, 2017. ISSN 20488610.

RASCHKA, S.; MIRJALILI, V. **Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2, 3rd Edition**. Quebec City, Can.: Packt Publishing, 2019. ISBN 9781789958294.

REASONCORE. **Reason security antivirus**. 2020. Disponível em: <https://www.reasonsecurity.com/>.

RITTER, A. et al. Weakly supervised extraction of computer security events from twitter. In: **Proceedings of the 24th International Conference on World Wide Web**. Republic and Canton of Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2015. (WWW '15), p. 896–905. ISBN 978-1-4503-3469-3. <https://doi.org/10.1145/2736277.2741083>.

SABOTTKE, C.; SUCIU, O.; DUMITRAS, T. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In: **USENIX Security Symposium**. Washington, D.C.: USENIX Association, 2015. p. 1041–1056.

SCELLER, Q. L. et al. Sonar: Automatic detection of cyber security events over the twitter stream. In: **Proceedings of the 12th International Conference on Availability, Reliability and Security**. New York, NY, USA: ACM, 2017. (ARES '17), p. 23:1–23:11. ISBN 978-1-4503-5257-4. <https://doi.org/10.1145/3098954.3098992>.

SHIREY, R. **Internet Security Glossary, Version 2**. [S.l.], 2007. <http://www.rfc-editor.org/rfc/rfc4949.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc4949.txt>.

SILVA, T.; ZHAO, L. **Machine Learning in Complex Networks**. [S.l.]: Springer International Publishing, 2016. ISBN 9783319172903.

Statista. **Market share held by the leading Windows anti-malware application vendors worldwide**. 2018. Disponível em: <https://www.statista.com/statistics/271048/market-share-held-by-antivirus-vendors-for-windows-systems/>.

TAILOR, J. P.; PATEL, A. D. A comprehensive survey: ransomware attacks prevention, monitoring and damage control. **Int. J. Res. Sci. Innov**, v. 4, p. 2321–2705, 2017.

TAYLOR, M. **GitHub – bear/python-twitter: A Python wrapper around the Twitter API**. 2020. Disponível em: <https://github.com/bear/python-twitter>.

The MITRE Corporation. **CVE - Frequently Asked Questions**. 2020. Disponível em: <https://cve.mitre.org/about/faqs.html>.

TREND MICRO. **Threat Encyclopedia - Trend Micro USA**. 2020. Disponível em: <https://www.trendmicro.com/vinfo/us/threat-encyclopedia/>.

Twitter. **Rich Warren on Twitter**. 2017. Disponível em: <https://twitter.com/buffaloverflow/status/943767508158439425>.

Voodoosoft. **VoodooShield™ - The User-Friendly Toggling Computer Lock.** 2020. Disponível em: <https://voodooshield.com/>.

Webroot. **Serviços de informações sobre ameaças e cibersegurança | Webroot**. 2020. Disponível em: <https://www.webroot.com/pt/>.

WEISS, S. M.; INDURKHYA, N.; ZHANG, T. **Fundamentals of predictive text mining**. London,UK: Springer, 2015. <=>https://doi.org/10.1007/978-1-4471-6750-1.

WHITMAN, M.; MATTORD, H. **Principles of Information Security**. Boston, MA, USA: Cengage Learning, 2011. ISBN 9781111138219.

WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. **IEEE Transactions on Systems, Man, and Cybernetics**, IEEE, n. 3, p. 408–421, 1972. <https://doi.org/10.1109/TSMC.1972.4309137>.

WINDOWS DEFENDER. **Cyberthreats, viruses, and malware - Microsoft Security Intelligence**. 2020. Disponível em: <https://www.microsoft.com/en-us/wdsi/threats>.

XIAO, C. et al. From patching delays to infection symptoms: Using risk profiles for an early discovery of vulnerabilities exploited in the wild. In: **27th USENIX Security Symposium (USENIX Security 18)**. Baltimore, MD, USA: USENIX Association, 2018. p. 903–918. ISBN 978-1-931971-46-1.

Zero Day Initiative. **PROGRAM BENEFITS**. 2018. Disponível em: <https://www.zerodayinitiative.com/about/benefits/>.

ZERODIUM. **How to Sell Your 0day Exploit to ZERODIUM**. 2018. Disponível em: <https://zerodium.com/program.html>.

ZONEALARM. **PC and Mobile Security Software | ZoneAlarm**. 2020. Disponível em: <https://www.zonealarm.com/>.

Apêndices

APÊNDICE **A**

Resultados adicionais de experimentos

Esta seção traz resultados adicionais de alguns dos experimentos realizados nesta dissertação.

A.1 Experimento 1

	Precisão	**Revocação**	**F-score**	**valor-p**
Baseline - SVM (PoC)	0,2075	0,7053	0,3199	—
RegLog (PoC)	0,6678	0,2434	0,3568	0,252
XGBoost (PoC)	0,7454	0,2746	0,4014	0,078
LightGBM (PoC)	0,7170	0,3293	0,4513	$< 0{,}001$
Árvor de decisão (PoC)	0,1703	0,5961	0,2649	0,002
Random Forest (PoC)	0,7530	0,2041	0,3212	0,861
Bernoulli Naive Bayes (PoC)	0,2979	0,3739	0,3316	0,792
Gaussian Naive Bayes (PoC)	0,3366	0,0234	0,0438	$< 0{,}001$
Multinomial Naive Bayes (PoC)	0,6730	0,1101	0,1892	0,001
KNN (PoC)	0,6152	0,2720	0,3772	0,059
Baseline - SVM(real)	0,0632	0,7660	0,1166	—
Reg. Log. (real)	0,7000	0,1857	0,2935	0,007
XGBoost (real)	0,4916	0,1535	0,2340	0,08
LightGBM (real)	0,5219	0,2196	0,3091	0,004
Árvor de decisão (real)	0,0902	0,7410	0,1608	0,001
Random Forest (real)	0,2666	0,1017	0,1472	0,666
Bernoulli Naive Bayes (real)	0,1019	0,5928	0,1739	0,043
Gaussian Naive Bayes (real)	0,2600	0,1160	0,1604	0,580
Multinomial Naive Bayes (real)	0,6833	0,1589	0,2579	0,037
KNN (real)	0,2500	0,0392	0,0678	0,226

Tabela 17: Resultados para todos os algoritmos testados no Experimento 1