UNIHACKER.CLUB APRESENTA

SÉRIES

UniHacker: Tecnologias e Desafios em Cibersegurança I

DIEGO KREUTZ

ROBEN LUNARDI

EWERTON ANDRADE

ÉRICO AMARAL

RODRIGO MANSILHA

UNIHACKER.CLUB

PROGRAMA UNIVERSIDADE HACKER

UNIHACKER.CLUB
PROGRAMA UNIVERSIDADE HACKER

Séries Ebook

UniHacker: Tecnologias e Desafios em Cibersegurança I

1ª Edição

Organizadores

Diego Kreutz

Pós doutorado na Universidade de Monash e Doutor em Informática pela Universidade de Luxemburgo (UNI). Atua na área de Cibersegurança, é vice-coordenador Comissão Especial de Cibersegurança (CESeg) da Sociedade Brasileira de Computação (SBC) e coordenador do Programa Clube Universidade Hacker (UniHacker). Atualmente, professor da Universidade Federal do Pampa (UNIPAMPA).

Roben Castagna Lunardi

Doutor em Ciência da Computação pela Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS). Realizou doutorado sanduíche na Newcastle University, Reino Unido. Atualmente é professor do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul (IFRS).

Ewerton Andrade

Doutor em Engenharia de Computação pela Universidade de São Paulo (USP). Pesquisador e entusiasta da área de Criptografia e Segurança da Informação, com premiações nesta área. Atualmente é colaborador do Programa UniHacker, pesquisador no Sidia e professor na Universidade Federal de Rondônia (UNIR).

Érico Marcelo Hoff do Amaral

Doutor em Informática na Educação pela Universidade Federal do Rio Grande do Sul (UFRGS). Entusiasta da área de Cibersegurança, co-coordenador do Programa UniHacker e professor do Curso de Engenharia de Computação e do Mestrado em Computação Aplicada da Universidade Federal do Pampa (UNIPAMPA).

Rodrigo Brandão Mansilha

Doutor em Ciência da Computação pela Universidade Federal do Rio Grande do Sul (UFRGS). Entusiasta da área de Inteligência Artificial e Cibersegurança, atualmente professor dos cursos de Ciência da Computação, Engenharia de Software e Mestrado em Engenharia de Software da Universidade Federal do Pampa (UNIPAMPA).

Editora
EDIURCAMP
Bagé - RS, Brasil
2024

EDIURCAMP

Capa: Brandow Buenos Aires Madeira e Diego Kreutz
Supervisão Gráfica: Diego Kreutz e Rodrigo Brandão Mansilha

Dados Internacionais de Catalogação na Publicação (CIP)

U58

UniHacker: tecnologias e desafios em cibersegurança. / Organizado por Diego Kreutz... [et al.]. - Bagé: Ediurcamp, 2024.
172p.

Livro digital
ISBN 978-65-86471-47-2

1.Tecnologia da informação. I. Kreutz, Diego (Org.). II. Lunardi, Roben C. (Org.). III. Andrade, Ewerton (Org.). IV. Amaral, Érico (Org.). V. Mansilha, Rodrigo (Org.) II. Título.

CDD: 001.6

Catalogação elaborada pelo Sustema de Bibliotecas FAT / URCAMP
Bibliotecária Responsável: Maritza Silveira Martins CRB10/1741

DOI: 10.29327/5463083
DOI URL: https://doi.org/10.29327/5463083

Prefácio

Este é o segundo volume da série de livros do programa **UniHacker.Club**. O conteúdo apresentado resulta do esforço coletivo de membros do programa e colaboradores externos, reunindo especialistas da academia e da indústria para abordar conceitos e tecnologias essenciais no cenário contemporâneo da Cibersegurança. Cada capítulo oferece uma introdução a temas relevantes da área, acompanhada de atividades práticas e reflexivas que permitem ao leitor consolidar e avaliar os conhecimentos adquiridos.

O **Capítulo 1** introduz o Aprendizado de Máquina Adversarial (*Adversarial Machine Learning* – AML), abordando as implicações dessa tecnologia no contexto da sociedade impulsionada pela Inteligência Artificial. Em seguida, no **Capítulo 2**, são apresentadas as Redes Generativas Adversarias (*Generative Adversarial Networks* – GANs), destacando suas aplicações no campo da Cibersegurança. O **Capítulo 3** oferece uma visão introdutória sobre *blockchain*, explorando conceitos básicos, casos de uso práticos e ferramentas como o Hyperledger. No **Capítulo 4**, as vulnerabilidades e ataques relacionados a *blockchain* são analisados, estabelecendo as bases para novas pesquisas e avanços necessários na área. Por fim, o **Capítulo 5** trata da detecção de anomalias em redes de computadores, detalhando técnicas e tecnologias empregadas nesse domínio estratégico.

Um agradecimento especial aos autores deste volume, que se dedicaram e se esforçaram para produzir um material de qualidade. Sua dedicação e compromisso foram fundamentais para que este projeto se concretizasse e fosse entregue à comunidade com o alto padrão.

Gostaríamos de expressar também nossa sincera gratidão aos revisores que contribuíram com seu tempo e expertise para aprimorar este trabalho. Suas valiosas contribuições foram essenciais para garantir a qualidade e a relevância deste material.

Sobre o Programa UniHacker.Club

O Programa Clube Universidade Hacker tem como objetivo acompanhar a evolução tecnológica e aprofundar conhecimentos em segurança da informação por meio de diversas iniciativas, incluindo momentos culturais, workshops com especialistas da indústria, oficinas práticas, treinamentos online e presenciais, promoção de eventos técnico-científicos, interação com grupos externos de tecnologia e segurança, desenvolvimento de soluções tecnológicas para demandas locais e regionais, campanhas de conscientização sobre tecnologia, segurança e privacidade, além de competições de *hack* e outras atividades de integração.

Diego, Roben, Ewerton, Érico e Rodrigo
Dezembro de 2024.

UniHacker 2024

UniHacker: Tecnologias e Desafios em Cibersegurança I

1ª Edição

Editores
Diego Kreutz (UNIPAMPA)
Roben Castagna Lunardi (IFRS)
Ewerton Andrade (UNIR e SIDIA)
Erico Marcelo Hoff do Amaral (UNIPAMPA)
Rodrigo Brandão Mansilha (UNIPAMPA)

Revisores dos Capítulos
Alessandro Bof de Oliveira (UNIPAMPA)
Charles Christian Miers (UDESC)
Claudio Schepke (UNIPAMPA)
Érico Marcelo Hoff do Amaral (UNIPAMPA)
Ewerton Andrade (UNIR e SIDIA)
Gustavo Cardozo Rodrigues (Datum)
Roben Castagna Lunardi (IFRS)
Rodrigo da Rosa Righi (UNISINOS)
Tiago Antônio Rizzetti (UFSM)
Walter Priesnitz Filho (UFSM)

Revisão Geral
Rodrigo Brandão Mansilha (UNIPAMPA)
Diego Kreutz (UNIPAMPA)
Ewerton Andrade (UNIR e SIDIA)

Autores
Alex Dias Camargo (Urcamp)
Angelo Gaspar Diniz Nogueira (UNIPAMPA)
Anna Luiza Gomes da Silva (UNIPAMPA)
Avelino Francisco Zorzo (PUCRS)
Claudio Schepke (UNIPAMPA)
Diego Kreutz (UNIPAMPA)
Eduardo Feitosa (UFAM)
Ewerton Andrade (UNIR e SIDIA)
Hendrio Luis de Souza Bragança (UFAM)
Igor Ferrazza Capeletti (UNIPAMPA)
Jonas Pontes (UFAM)
Karina Casola Fernandes (UNIPAMPA)
Kayuã Oleques Paim (UFRGS)
Laura Caroline Tschiedel (UNIPAMPA)
Luciano Vargas (UNIPAMPA)
Marcia Henke (UFSM)
Raul Ceretta Nunes (UFSM)
Regio Antonio Michelin (UNSW)
Roben Castagna Lunardi (IFRS)
Rodrigo Brandão Mansilha (UNIPAMPA)
Vinicius Nuñez Lopes (UNIPAMPA)

Sumário

Capítulo

1

Introdução ao Aprendizado de Máquina Adversarial: ataques, defesas e consequências

Diego Kreutz (UNIPAMPA), Jonas Pontes (UFAM), Eduardo Feitosa (UFAM), Karina Casola Fernandes (UNIPAMPA), Alex Dias Camargo (Urcamp), Kayuã Oleques Paim (UFRGS)

Resumo. *Neste capítulo, apresentamos uma visão geral sobre aprendizado de máquina adversarial (Adversarial Machine Learning – AML), destacando suas características em ambientes intrinsecamente adversariais, como o da segurança cibernética. Baseamo-nos na taxonomia proposta pelo NIST para explorar as três principais categorias que estruturam o tema: ataques, defesas e consequências. Abordamos os alvos dos ataques, como etapas do pipeline de aprendizado de máquina (e.g., coleta de dados, modelos preditivos), bem como as técnicas empregadas (e.g., envenenamento, evasão e inferência) e os diferentes níveis de conhecimento do adversário. Para ilustrar, discutimos exemplos como ataques de evasão, que manipulam entradas para enganar classificadores, e ataques de envenenamento, que comprometem o treinamento de modelos por meio de dados corrompidos. Também discutimos estratégias de defesa, como treinamento adversarial, sanitização de dados e privacidade diferencial, que buscam mitigar as ações adversariais e fortalecer a robustez dos sistemas. Por fim, refletimos sobre as consequências desses ataques, que incluem violações de integridade (e.g., falsos negativos), disponibilidade (e.g., negação de serviço) e confidencialidade (e.g., extração de dados sensíveis). Nosso objetivo foi proporcionar uma compreensão clara e objetiva do tema, enfatizando tanto os fundamentos quanto os desafios contemporâneos. Concluímos com exemplos práticos que demonstram a relevância do aprendizado de máquina adversarial no contexto da segurança cibernética, além de estimular discussões sobre futuras direções de pesquisa e desenvolvimento.*

1.1. Introdução

Técnicas de aprendizado de máquina fazem parte de soluções tecnológicas para os mais variados domínios e casos de aplicação. Por exemplo, compreensão de linguagem natural [Maulud et al. 2021]; *marketing* [Ngai and Wu 2022];

agroindústria [Diaz-Gonzalez et al. 2022]; medicina [Garg and Mago 2021]; detecção de fraudes [Ashtiani and Raahemi 2021]; visão computacional [Kakani et al. 2020]; detecção de notícias falsas [Collins et al. 2021]. Outrossim, detecção de *malwares* [Gaurav et al. 2022]; segurança em redes veiculares [Dibaei et al. 2021]; controle de acesso a dispositivos e sistemas [Ahmad and Alsmadi 2021] e; segurança em infraestruturas de redes de comunicação [Wang et al. 2021].

Na área de segurança cibernética, é notória a crescente utilização de técnicas de aprendizado de máquina [Geetha and Thilagam 2021, Dasgupta et al. 2022, Siqueira et al. 2021, Kotenko et al. 2022, Siqueira et al. 2022, Vilanova et al. 2021, Miao et al. 2021, Sarker 2021] para resolver problemas que são adversariais. Nesse contexto, existem dois tipos clássicos de atores: (a) os defensores (*e.g.*, usuários, analistas de segurança, empresas de antivírus e fabricantes de soluções de defesa como *firewalls*, e (b) os atacantes (*i.e.*, usuários com intenções maliciosas diversas). Enquanto os defensores buscam alternativas para evitar ataques e vazamento de informação, os atacantes estão continuamente se reinventando para evadir os sistemas de detecção e controle. Tipicamente, um atacante utiliza abordagens de engenharia social (*e.g.*, envia mensagens de *phishing*), espalha *malwares* (*e.g.*, infecta arquivos ou aplicativos existentes, cria um aplicativo gratuito infectado), explora vulnerabilidades dos sistemas e executa ataques de negação de serviço.

O papel natural das técnicas de aprendizado de máquina no contexto de segurança cibernética é a detecção, ou seja, a criação de modelos preditivos para identificar automaticamente mensagens de engenharia social (como *spam* e *phishing*); arquivos comuns infectados (*e.g.*, arquivo PDF[1] contendo código malicioso); aplicativos infectados. Também, tentativas de intrusão em sistemas; intrusões em andamento; comportamento anômalo no tráfego da rede de dados e; comportamento do usuário no controle de acesso a sistemas. A base fundamental dessas técnicas de aprendizado de máquina é, geralmente, um *dataset* [Soares et al. 2021a, Soares et al. 2021b], o qual é um conjunto de dados estruturados de um domínio específico que serve de amostra para o treinamento dos algoritmos. Para o exemplo detecção de *malwares* Android, um *dataset* potencialmente útil ou relevante pode ser obtido a partir da extração de características de aplicativos benignos e maliciosos. Esse processo, quando objetiva obter somente as características estáticas do aplicativo, é realizado a partir do seu pacote instalador, o **A**ndroid ***P**ac**K**age* (APK)[2].

Visando construir um *dataset* para o treinamento e a validação dos modelos de aprendizado de máquina, o usuário deve primeiro extrair as características, como permissões e chamadas de API (*Application Programming Interface*)[3], de um conjunto de aplicativos. Geralmente, as características são identificadas como presentes (1) ou ausentes (0) em cada aplicativo, ou seja, é fornecido um valor numérico para o treinamento dos modelos. Além das características, também é necessário rotular os aplicativos como benigno ou maligno, dado que a existência de rótulo é uma condição necessária para o uso de modelos de aprendizado supervisionado. Essa tarefa pode ser realizada por serviços

[1] *Portable Document Format* (PDF). Definição: https://dictionary.archivists.org/entry/portable-document-format.html

[2] https://www.pcmag.com/encyclopedia/term/apk

[3] https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

online, como o VirusTotal[4], que oferece dezenas de *scanners* que classificam o aplicativo como benigno ou maligno. Uma vez que o *dataset* está pronto, contendo amostras rotuladas e características em formato numérico, podemos então treinar e validar os modelos de classificação de aplicativos. Para um determinado conjunto de dados (*e.g.*, 1 milhão de amostras e mais de 1 milhão de características [Roy et al. 2015, Daoudi et al. 2022]), é possível construir modelos preditivos bastante efetivos para a detecção de aplicativos Android maliciosos. Entretanto, é importante observar que estamos utilizando características e dados com significado potencialmente atrelado a um sistema específico e a um determinado instante de tempo (*e.g.*, rotulação baseada no sistema S e na data DD/MM/AAAA) para o teste e a validação dos modelos.

O que torna esse cenário adversarial é o fato de códigos e aplicativos maliciosos serem gerados por atores maliciosos e com metas específicas em mente. Por exemplo, para atingir o seu objetivo, o atacante tentará criar códigos maliciosos capazes de, simultaneamente, evadir os sistemas de defesa, como os serviços de detecção de *malwares* (*e.g.*, VirusTotal), e ainda atingir a mesma finalidade do ataque original. Esse é um típico cenário de uso para aprendizado de máquina adversarial, ou *adversarial machine learning* [Huang et al. 2011], em que o objetivo principal do adversário é projetar técnicas, algoritmos e soluções específicas para burlar os classificadores baseados em aprendizado de máquina, como os detectores de ameaça.

Apesar dos ataques de evasão serem os mais naturais em cenários de utilização adversarial de aprendizado de máquina, existem também outros ataques, como o de envenenamento. Nesse tipo de ataque, o agente malicioso manipula os dados rotulados de treinamento. Enquanto ataques na rotulagem das amostras (*flipping attacks*) podem afetar algoritmos como o *Naive Bayes* [Zhang et al. 2021a], os ataques de envenenamento mais sofisticados podem se concentrar em um número pequeno de pontos distintos para evadir defesas de sanitização de dados, como o ataque de influência, ou na modelagem de parâmetros com altas taxas de erro nos testes, mas baixas taxas de erro no treinamento, como no caso dos ataques KKT [Koh et al. 2022]. Esses ataques geralmente são difíceis de detectar, pois os dados contaminados podem parecer normais e os pontos de contaminação não necessitam apresentar altas perdas no modelo contaminado. Dessa forma, o defensor não pode simplesmente remover pontos com altas perdas, fato que dificulta a defesa.

Neste capítulo, o nosso objetivo é abordar uma introdução ao aprendizado de máquina adversarial, apresentando exemplos de *ataques*, *defesas* e *consequências* (*e.g.*, violações de integridade, disponibilidade, confidencialidade ou privacidade), consideradas às três grandes categorias da taxonomia do NIST (*National Institute of Standards and Technology*)[5] [Tabassi et al. 2019]. Na categoria de *ataques*, existem os *alvos*, as *técnicas* e o *conhecimento*. Na categoria de *defesas* há mecanismos de *contra-ataque* em tempos de *treinamento* e *teste*. Por fim, fazem parte da categoria de *consequências* as *violações de integridade*, a *disponibilidade* e a *confidencialidade*.

O restante deste capítulo está organizado conforme a Figura 1.1. Na Seção 1.2 apresentamos a taxonomia e os conceitos-base de *adversarial machine learning*. Nas

[4] https://www.virustotal.com
[5] https://www.nist.gov

Seções 1.3, 1.4 e 1.5 introduzimos os ataques, as defesas e as consequências. Na Seção 1.6 abordamos algumas técnicas e desafios que afetam a detecção de *malwares* em sistemas de aprendizado de máquina. Finalmente, apresentamos o resumo do capítulo na Seção 1.7, seguido de uma lista de exercícios na Seção 1.8.

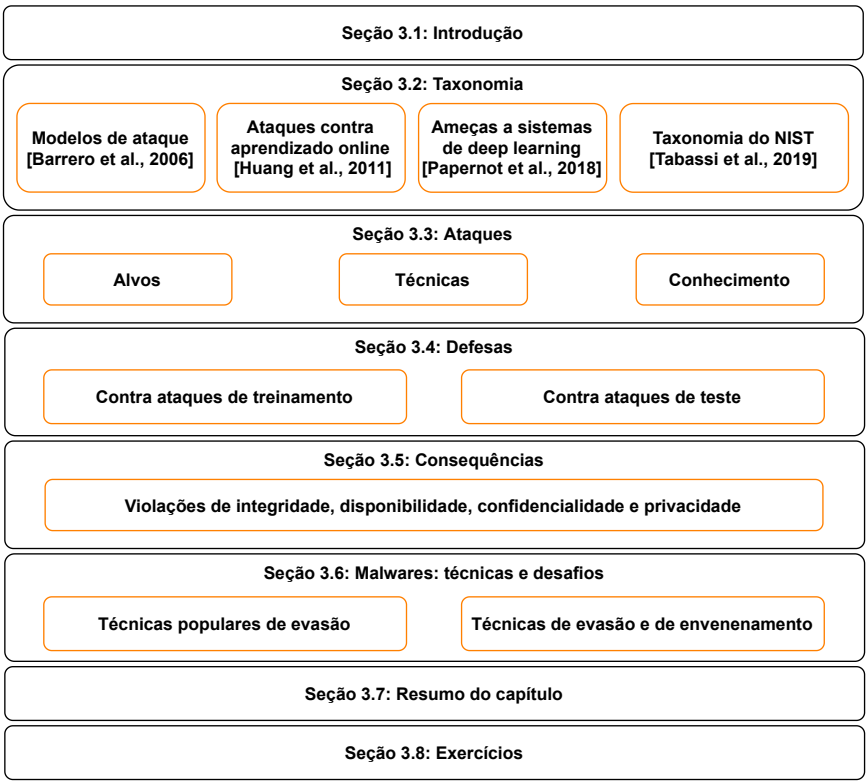

Figura 1.1. Seções deste capítulo

1.2. Taxonomia

O aprendizado de máquina adversarial estuda técnicas efetivas de aprendizado de máquina contra oponentes adversariais [Huang et al. 2011, Tabassi et al. 2019, Rosenberg et al. 2021]. Em um ambiente adversarial, como no caso de segurança cibernética, devemos antecipar que o adversário irá, invariavelmente, trabalhar para causar falhas em sistemas. Nesse contexto, a segurança é similar a um jogo de xadrez entre dois adversários: o atacante, considerado um oponente, e o defensor, tipicamente representado pelo usuário ou administrador de sistemas. Para ganhar, é necessário não apenas ter uma boa estratégia, mas também antecipar as respostas do adversário às jogadas previstas na estratégia.

O estudo das técnicas utilizadas contra oponentes adversariais engloba conhecer os ataques, as defesas e as consequências, como veremos nas próximas seções. Antes disso, apresentaremos as iniciativas de categorização das atividades de adversários (ataques) e as soluções (defesas) elaboradas para tentar impedi-las. A discussão das taxonomias, além de ser importante e relevante para o entendimento da área de aprendizado de máquina adversarial, fundamentará também a construção das seções seguintes.

A primeira taxonomia de aprendizado de máquina adversarial [Barreno et al. 2006a] categoriza os diferentes ataques (modelos de ataque) aplicados aos sistemas de aprendizado de máquina, bem como as defesas existentes contra esses ataques. Os ataques são classificados em três eixos, a *influência*, a *especificidade* e as *violações de segurança*.

1. O *eixo da influência* é dividido em ataques *causativos*, que alteram o processo de treinamento através da influência sobre os dados de treinamento, e *exploratórios*, que não alteram o processo de treinamento, mas utilizam outras técnicas, como sondagem ou análise *offline*, para descobrir informações.
2. O *eixo da especificidade* divide os ataques em *direcionados*, onde o foco do ataque é em um ponto específico ou em um pequeno conjunto de pontos, ou *indiscriminados*, que engloba uma variedade muito maior de pontos, como "qualquer falso negativo".
3. O *eixo das violações de segurança* é dividido em *integridade*, resultante de pontos de intrusão classificados como normais (falsos negativos), ou *disponibilidade*, resultando em muitos erros de classificação, tanto falsos negativos quanto falsos positivos, tornando o sistema inutilizável.

No eixo da influência é identificado se o adversário consegue influenciar os dados de treinamento utilizados para construir o classificador. Caso positivo, temos um *ataque causativo*. Do contrário, teremos um *ataque exploratório*, uma vez que o atacante apenas envia novas instâncias para o classificador e observa suas decisões nessas instâncias. No eixo da especificidade é determinado o tipo de violação de segurança que o adversário causa ao: (a) permitir que instâncias prejudiciais passem pelo filtro como falsos negativos, caracterizando uma *violação de integridade*; ou (b) criar um evento de negação de serviço em que instâncias benignas são filtradas incorretamente como falsos positivos, levando a uma *violação de disponibilidade*. Finalmente, no eixo das violações de segurança é determinada a intenção do invasor: se (a) o ataque é *direcionado* para degradar o desempenho do classificador em uma instância específica ou (b) o ataque visa provocar a falha *indiscriminada* do classificador em uma ampla classe de instâncias.

Trabalhos subsequentes, como [Huang et al. 2011], utilizam a taxonomia proposta por [Barreno et al. 2006a] para classificar ataques contra algoritmos de aprendizado de máquina *online* e discutir fatores específicos do aplicativo que limitam as capacidades de um adversário. Por exemplo, foi descoberto que os adversários exploram as respostas de um filtro de dados para inferir informações confidenciais utilizadas no processo de aprendizagem, levando a uma nova violação de privacidade [Huang et al. 2011]. Para preservar as informações confidenciais, uma solução de aprendizado de máquina deve ser da mesma forma robusta a ataques exploratórios ou causativos que visam violar a privacidade.

Mais recentemente, trabalhos como [Papernot et al. 2018] categorizam os modelos de ameaças em sistemas de aprendizado profundo (*deep learning*) no que diz respeito à resistência do modelo contra os objetivos do adversário e suas capacidades. Com relação aos objetivos, podemos ter quatro distintos, que afetam a integridade da saída do classificador:

1. redução da confiança da saída da classificação,
2. erro de classificação,
3. erro de classificação direcionado, e
4. classificação incorreta de origem/destino.

Com relação às capacidades do adversário, precisamos considerar o acesso aos dados de treinamento, à arquitetura da rede neural, ao "oráculo" (o adversário utiliza a rede neural como um "oráculo"[6]) e às amostras. Sumariamente, temos agora a identificação de abordagens de ataque relacionadas a elementos estruturais de algoritmos de aprendizado de máquina e os dados utilizados para treiná-los.

Logo após, em 2019, o NIST propôs uma taxonomia [Tabassi et al. 2019] baseada na literatura de aprendizado de máquina adversarial [Liu et al. 2018, Biggio and Roli 2018, Chakraborty et al. 2018, Kuznetsov et al. 2019]. A nova taxonomia consegue classificar os ataques, as defesas e também as consequências de ambos. As consequências, sejam elas reais ou potenciais, geradas por ataques e defesas, podem ou não ser consistentes com a intenção do adversário.

A taxonomia proposta pelo NIST é ilustrada resumidamente na Figura 1.2. Como pode ser observado, existem três grandes categorias: *ataques*, *defesas* e *consequências*. Para cada categoria, há algumas subcategorias. Na categoria de ataques, temos os *ativos*, as *técnicas* e o *conhecimento* dos adversários. Na segunda categoria, das defesas, são incluídos *contra-ataques* às etapas de *treinamento* e *testes e inferência* do aprendizado de máquina. Finalmente, a categoria das consequências engloba *violações de integridade*, *disponibilidade* e *confidencialidade*. As *violações de privacidade* representam um subconjunto das violações de confidencialidade. Nas próximas seções iremos detalhar às três grandes categorias da taxonomia do NIST para aprendizado de máquina adversarial.

1.3. Ataques

Como pôde ser observado na Figura 1.2, os ataques em aprendizado de máquina adversarial tem relação com os alvos, as técnicas empregadas e o conhecimento do adversário sobre o seu alvo. Por exemplo, um ataque direcionado pode ter como alvo o estágio de pré-processamento do *pipeline* de aprendizado de máquina e um único método de aprendizado supervisionado. Adicionalmente, o ataque pode utilizar diferentes técnicas, como evasão e inferência, e utilizar o modo "caixa preta" (*black box*), isto é, o atacante não possui conhecimento sobre o modelo, exceto amostras de entrada e saída de dados de treinamento.

[6]Por exemplo, no contexto de redes generativas adversariais, ou *Generative Adversarial Networks* (GANs), "oráculos" geradores e discriminadores são frequentemente criados e utilizados para treinar GANs [Aung et al. 2022].

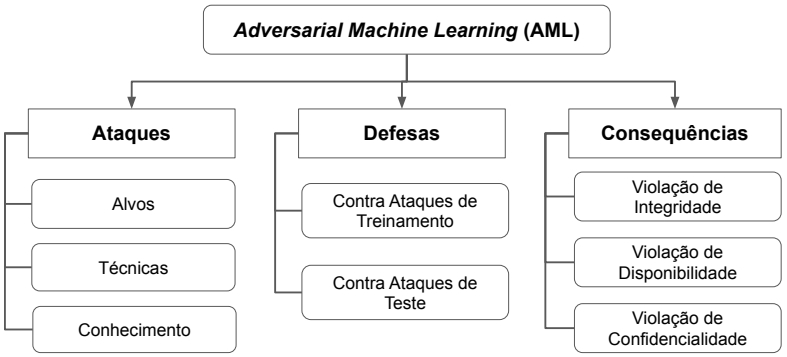

Figura 1.2. Taxonomia (adaptada de [Tabassi et al. 2019])

1.3.1. Alvos

Os alvos de ataques de aprendizado de máquina adversarial estão relacionados com as etapas do aprendizado de máquina, incluindo o domínio físico, a representação digital do pré-processamento e o próprio modelo preditivo [Tabassi et al. 2019]. No domínio físico, o alvo pode ser o ferramental ou os sensores que geram os dados que serão utilizados no treinamento e validação dos modelos preditivos. Um sensor comprometido, gerando dados contaminados, pode enviesar a predição dos modelos, isto é, favorecer o atacante. Por exemplo, o atacante pode comprometer ferramentas, como o AndroGuard[7] [Pontes et al. 2021], utilizadas na extração de características de aplicativos Android para a construção de *datasets*, prejudicando assim o treinamento e a validação dos modelos de classificação. O mesmo se aplica quando um adversário consegue comprometer serviços de rotulação das amostras, como o VirusTotal[4], um dos mais utilizados no domínio de detecção de *malwares*.

Outros exemplos de alvos do *pipeline* de *machine learning* são a representação digital para o pré-processamento, como o *dataset* contendo as características extraídas dos aplicativos Android, e o modelo de aprendizado de máquina propriamente dito. Em termos de modelos, os alvos podem conter os diferentes métodos de aprendizado existentes, como o aprendizado supervisionado (*supervised learning*) [Caruana and Niculescu-Mizil 2006], o aprendizado não supervisionado (*unsupervised learning*) [Ghahramani 2003], o aprendizado profundo (*deep learning*) [LeCun et al. 2015], ou ainda o aprendizado por reforço (*reinforcement learning*) [Kaelbling et al. 1996]. Embora a maioria das pesquisas em *adversarial machine learning* esteja concentrada em sistemas de aprendizado supervisionado, existem algoritmos projetados para atacar classificação baseada em aprendizado por reforço [Papernot et al. 2018], por exemplo.

1.3.2. Técnicas

As técnicas adversariais utilizadas em ataques contra alvos almejam atingir tanto as etapas de treinamento quanto de testes do *pipeline* de *machine learning*, conforme mostrado

[7] https://github.com/androguard/androguard

na Figura 1.3. Os ataques na etapa de treinamento buscam descobrir ou influenciar os dados de treinamento, ou o modelo propriamente dito. Por exemplo, se o atacante possuir acesso a alguns ou todos os dados de treinamento, ele poderá criar ou substituir o modelo. O modelo substituído pode, então, ser utilizado para testar a efetividade de potenciais entradas antes de submetê-las como um ataque à etapa de testes ou inferência.

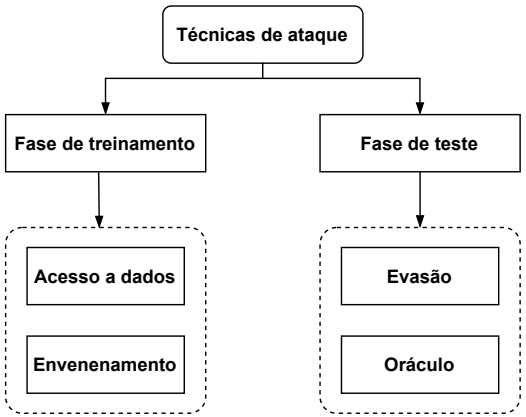

Figura 1.3. Técnicas de ataque em *adversarial machine learning* (adaptada de [Tabassi et al. 2019])

No caso de envenenamento, que pode ser direto ou indireto, os dados ou o modelo são alterados. Em um envenenamento indireto, o atacante precisa contaminar os dados antes da etapa de pré-processamento, o que pode ocorrer no domínio físico, alterando os dados dos sensores ou ferramentas, por exemplo. O atacante consegue alterar os dados por meio de técnicas de injeção ou manipulação de dados, ou ainda alterar o modelo diretamente a fim de corromper o processo de aprendizagem, ou degradar o desempenho do algoritmo [Taheri et al. 2020].

No envenenamento direto, o mais comum é um adversário tentar envenenar os dados de treinamento injetando amostras cuidadosamente projetadas para comprometer o processo de aprendizado [Chakraborty et al. 2018]. Os ataques de envenenamento desse tipo são considerados uma das principais ameaças aos modelos de aprendizado de máquina em produção [Siva Kumar et al. 2020]. Frequentemente, as amostras injetadas pelos atacantes possuem características semelhantes às originais, mas com rótulos incorretos, induzindo a mudança na distribuição dos dados de treinamento [Liu et al. 2018]. É importante destacar que as amostras adversariais injetadas podem ser otimizadas por métodos de programação linear, que mudam a fronteira de decisão do modelo no aprendizado não supervisionado, ou de gradiente ascendente no teste de erro do modelo, para degradar a acurácia da classificação no aprendizado supervisionado. Discutimos esse tipo de ataque com maior profundidade na Seção 1.3.4.

Os ataques na etapa de testes, também conhecidos como ataques exploratórios, não comprometem o modelo e nem os dados utilizados no treinamento. Considerando os ataques de evasão, o atacante gera exemplos adversariais como entradas que conseguem

evadir a classificação de saída do modelo. Já no caso de ataques de "oráculo" (*e.g.*, criação de um modelo substituto), o atacante coleta e infere informações sobre o modelo ou os dados de treinamento.

Sinteticamente, num ataque de evasão, o adversário resolve problemas de otimização de restrições dos dados ou dos modelos para encontrar pequenas perturbações de entrada que conseguem gerar grandes modificações nas funções de perda para resultar em saídas de classificação equivocadas. Esses ataques envolvem algoritmos de busca baseados em gradiente para minimizar ou maximizar as restrições dos dados ou dos modelos, como L-BFGS, ILLC, FGSM, JSMA, DeepFool, CPPN EA Fool, PGD, BIM, Carlini-Wagner L2, MI-FGSM, EOT e GASM [Lee et al. 2022, Qi et al. 2022, Akhtar and Mian 2018, Chakraborty et al. 2018, Yuan et al. 2019, Manoj et al. 2021, Mani et al. 2021, Zhang et al. 2022b]. A utilização de ataques baseados em gradiente ainda é efetiva em diferentes cenários e algoritmos de aprendizado de máquina, como aprendizado profundo em Internet das coisas [Mani et al. 2021], aprendizado profundo aplicado para alocação de energia em *links* de *download* em redes ma-MIMO (*massive Multiple-Input-Multiple-Output* [Manoj et al. 2021], redes neurais randomizadas [Lee et al. 2022], aprendizado profundo por reforço baseado em *clusters* heterogêneos e múltiplos agentes em sistemas de transmissão de dados temporizada [Elhami Fard and Selmic 2022] e redes neurais de uma forma geral através de "modelos-sombra" (*shadow models*) [Zhang et al. 2022b].

Nos ataques de "oráculo", os atacantes utilizam uma API[3] para apresentar entradas e observar as respectivas saídas do modelo. Mesmo quando o adversário não possui conhecimento sobre o modelo, os pares de entrada e saída podem ser utilizados pelo atacante para treinar um modelo substituto que opera de forma muito similar ao modelo alvo [Tabassi et al. 2019]. Por exemplo, é possível criar um ataque adversário de "caixa preta" tolerante à detecção (*detection tolerant black-box adversarial-attack* – DTBA) [Qi et al. 2022]. Para que o ataque seja viável, o pressuposto fundamental é o adversário obter um conjunto limitado de informações sobre os dados utilizados no treinamento do modelo [Qi et al. 2022]. Neste ataque é possível e viável devido a propriedades como transferibilidade, presentes em muitas arquiteturas de modelos. O modelo substituto, ou "modelo sombra", permite ao atacante gerar exemplos adversariais [Zhang et al. 2022b] para, subsequentemente, utilizar ataques de evasão contra o modelo. Na prática, os ataques de "oráculo" incorporam sub-categorias de ataques especializados, como ataques de extração, inversão e inferência de associação. Por exemplo, por um ataque de inversão, o adversário consegue inferir características que permitem reconstruir dados utilizados no treinamento do modelo, incluindo informações pessoais que violam a privacidade do indivíduo [Fredrikson et al. 2015, Veale et al. 2018].

1.3.3. Conhecimento

Além das técnicas utilizadas para lançar ataques sobre os alvos, as ameaças às etapas do *pipeline* de aprendizado de máquina também dependem do conhecimento do adversário sobre o modelo alvo. Similarmente ao que ocorre nos testes de *software*, existem ataques "caixa preta" (*black box*), "caixa cinza" (*gray box*) e "caixa branca" (*white box*).

- Ataque "caixa branca": o adversário possui conhecimento sobre o modelo, isto é, arquitetura, parâmetros, métodos e dados [Huang et al. 2019].

- Ataque "caixa cinza": o adversário possui apenas informação parcial acerca do modelo, como função de perda do método de treinamento ou valores de parâmetros. Com isso, o invasor pode treinar um modelo substituto para criar exemplos adversários e implantá-los no modelo de destino para evitar a detecção [Huang et al. 2019].
- Ataque "caixa preta": o adversário conhece apenas as amostras de entrada e saída do modelo. Como o atacante não possui conhecimento prévio do modelo, ele pode inserir amostras (exemplos adversários) no sistema de detecção e verificar o resultado [Zhang et al. 2021c]. Um ataque bem-sucedido pode ser caracterizado como uma amostra modificada que consegue evadir o sistema de detecção [Huang et al. 2019].

1.3.4. Ataques de envenenamento

Nesta seção exploraremos alguns detalhes de ataques de envenenamento (*poisoning attacks*), incluindo o subtipo de inversão (*flipping attacks*). Os ataques de envenenamento são considerados uma das ameaças de segurança mais críticas e emergentes para modelos de aprendizado de máquina, pois pode ser difícil verificar a confiabilidade dos dados de entrada [Paudice et al. 2019]. Esse cenário é ainda mais crítico quando os dados são coletados de forma distribuída, potencializando o problema de confiabilidade [Zhao et al. 2022b, Zhang et al. 2022a, Zhang et al. 2021b, Shi et al. 2021]. Para ter acesso ao *dataset* do sistema de aprendizado de máquina e poder contaminá-lo, atacantes podem utilizar técnicas de engenharia social, como o envio de mensagens de *phishing*.

Os ataques de envenenamento são considerados um desafio em diferentes aplicações de aprendizado de máquina, como sistemas de recomendação [Chen et al. 2021], filtragem de *spam* [Zhang et al. 2021a], detecção de *malwares* [Bala et al. 2021, Taheri et al. 2020], aprendizado federado e *Internet* das coisas [Zhang et al. 2022a, Zhang et al. 2022c]. Uma das preocupações recorrentes é o acesso ao *dataset* por parte de adversários, que pode envenená-los com rótulos ou valores de características incorretos, o que pode comprometer todo o processo de aprendizado. De modo a mitigar a influência do possível envenenamento do *dataset* por parte do adversário, é necessário empregar técnicas de defesa como sanitização dos dados [Paudice et al. 2019], rotulação de probabilidade [Papernot et al. 2016] e adição de ruídos aleatórios nos dados iniciais [Dwork 2008, Abadi et al. 2016].

A sanitização de dados e o aprimoramento da robustez do algoritmo de aprendizado de máquina são algumas das técnicas de defesas tipicamente utilizadas contra estes ataques. A Figura 1.4 ilustra ataques de envenenamento na etapa de treinamento do *pipeline* de aprendizado de máquina seguido de dois exemplos de técnicas de defesa. A sanitização de dados é uma técnica de defesa para garantir a pureza dos dados de treinamento, separando e removendo as amostras adversárias das normais. À vista disso, a sanitização de dados pode utilizar, por exemplo, detectores de anomalias para filtrar pontos de treinamento que parecem suspeitos [Paudice et al. 2019]. É importante ressaltar que a sanitização ocorre antes de os dados serem utilizados como entrada para a geração de modelos.

Ataques de inversão

Os ataques de inversão onde o adversário pode controlar e manipular uma parcela dos dados de treinamento têm um impacto significativo no resultado de sistemas de aprendizado de máquina. O objetivo desses ataques é treinar o classificador com um conjunto de dados corrompido [Rosenfeld et al. 2020]. Podemos destacar dois ataques básicos de inversão: os de modificação de dados (*data modification attacks*); e os ataques de modificação de rótulos (*label modification attacks*). No ataque de modificação de dados, o adversário modifica dados que representam as características ou rótulos para um subconjunto dos dados de treinamento. Já no ataque de modificação de rótulos, o adversário consegue modificar os rótulos em *datasets* de aprendizado supervisionado, principalmente aqueles de classificação binária [Vorobeychik and Kantarcioglu 2018]. Tais ataques em geral reduzem significativamente o desempenho do sistema de aprendizado de máquina alvo, mesmo sem o controle total por parte do invasor [Taheri et al. 2020].

Se destaca que ataques de envenenamento, em particular os de inversão, podem ter um impacto significativo em domínios como detecção de *malwares* Android. Atualmente, existem centenas de *datasets* disponíveis, em locais diversos e potencialmente não confiáveis da *Internet* (um levantamento de *datasets* publicamente disponíveis pode ser visto em [Soares et al. 2021a, Soares et al. 2021b]), utilizados indiscriminadamente para o treinamento e validação de modelos de aprendizado de máquina. Entretanto, há muitos ataques conhecidos que comprovadamente comprometem a qualidade dos resultados dos modelos no contexto de detecção de *malwares*, como FGSM, JSMA, Deepfool e C&W [Martins et al. 2020].

1.4. Defesas

Em geral, as técnicas de defesa se dividem dois grupos, conforme ilustrado na Figura 1.4. O primeiro grupo é formado por técnicas de sanitização de dados e o aumento da robustez dos algoritmos para contra-ataques de envenenamento na etapa de treinamento. No segundo grupo aparecem as técnicas de preservação da privacidade e os mecanismos de avaliação de segurança, que objetivam mitigar ataques de evasão, personificação e inversão nas etapas de testes e inferência. Algumas das técnicas específicas de defesa, como treinamento adversarial e privacidade diferencial, são apresentadas na Tabela 1.2.

1.4.1. Contra-ataques de treinamento

A etapa de treinamento em *pipelines* de aprendizado de máquina é fundamental, pois ela irá determinar o desempenho do modelo gerado. Tipicamente, os adversários investem energia na manipulação de dados utilizados no treinamento para reduzir o desempenho geral do modelo. Esse objetivo é tipicamente atingido por amostras contaminadas que afetam a maioria dos algoritmos [Meenakshi and Maragatham 2020]. De fato, as técnicas de contra-ataques que envolvem acesso aos dados requerem medidas clássicas, como a encriptação dos dados. Essencialmente, o conjunto de dados de treinamento é cifrado antes do treinamento. Porém, essa técnica pode gerar um custo adicional de recursos computacionais [Khalid et al. 2018].

É importante mencionar que os contra-ataques de envenenamento podem também incluir técnicas como sanitização de dados [Cretu et al. 2008] e estatísticas robustas [Ronchetti and Huber 2009, Jagielski et al. 2018]. O objetivo da sanitização de da-

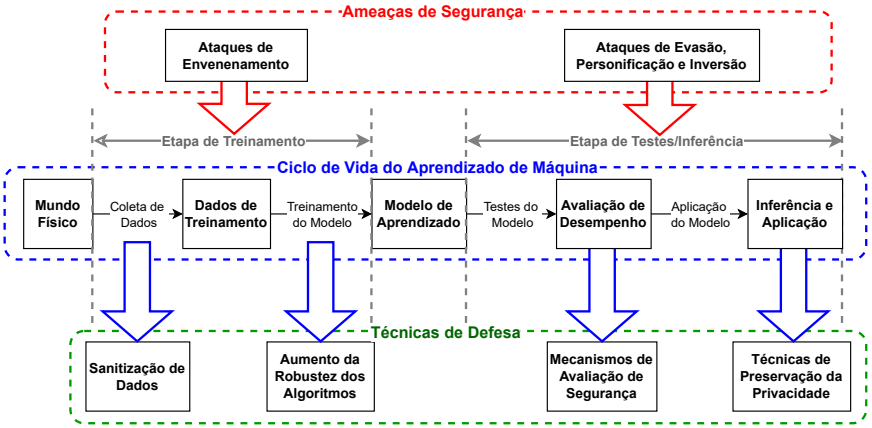

Figura 1.4. Técnicas de defesa (adaptada de [Liu et al. 2018])

dos é identificar e excluir amostras adversariais por testes que buscam identificar o impacto das amostras no desempenho da classificação. Ao identificar e descartar as amostras anômalas, o mecanismo de defensa consegue proteger o sistema contra os ataques de envenenamento que introduzem amostras contaminadas no conjunto de dados de entrada [Koh et al. 2018].

Uma das técnicas para identificar as amostras adversariais é a rejeição por impacto negativo [Barreno et al. 2010]. Essa técnica mede o efeito empírico de cada instância de treinamento e elimina do processo aqueles pontos (amostras) com um impacto negativo considerável na precisão da classificação. A ideia da estratégia é conseguir determinar se uma instância é maliciosa ou não a partir do treinamento de um classificador, utilizando um conjunto de treinamento básico. As amostras candidatas são, então, adicionadas ao conjunto de treinamento com o qual um segundo classificador é treinado. Finalmente, ambos os classificadores são aplicados a um conjunto de instâncias com rótulos conhecidos, sendo mensurada a diferença de precisão entre ambos. Se a amostra candidata adicionada ao conjunto de treinamento levar a um classificador com mais erros, a amostra será rejeitada.

Diferentemente da sanitização de dados, a técnica de estatísticas robustas [Ronchetti and Huber 2009] procura reduzir possíveis distorções do modelo de aprendizado causadas pelos dados envenenados [Tabassi et al. 2019]. As estatísticas robustas têm como lema que qualquer modelo, especialmente um paramétrico, é apenas aproximadamente válido, e que qualquer estimador projetado para uma distribuição particular, utilizado na prática, também seja estável na presença de especificação incorreta do modelo, como dados infectados.

Em resumo, as estatísticas robustas objetivam estimadores que sejam estáveis mesmo sob pertubações dos dados. Por padrão, é assumido que as amostras recebidas pelo modelo são de uma boa distribuição, mas que um adversário consegue corromper arbitrariamente uma fração constante dos dados observados. Atual-

mente, há estimadores comprovadamente robustos, isto é, capazes de tolerar uma fração constante de corrupções independente da dimensão da amostra, como os baseados em covariância [Diakonikolas et al. 2017]. Uma medida de robustez de um estimador é chamada de ponto de ruptura, a fração mínima de ruído (o que o adversário precisa controlar) que pode tornar o estimador arbitrariamente ruim [Lai et al. 2016, Diakonikolas et al. 2019].

Na prática, mecanismos de defesa contra os ataques de envenenamento *backdoor* [Li et al. 2022] podem ser criados se o ataque deixar um rastro detectável, denominado de assinatura espectral (*spectral signatures*) [Tran et al. 2018]. Uma assinatura espectral pode ser identificada através da covariância de uma representação de características compreendidas por redes neurais [Tran et al. 2018].

1.4.2. Contra-ataques de teste

Contra-ataques de testes e inferência incluem melhoramentos na robustez dos modelos, considerando técnicas de treinamento adversário (*adversarial training*), mascaramento de gradiente (*gradient masking*), destilação defensiva (*defensive distillation*), método de conjunto (*ensemble method*), recurso de compressão (*feature squeezing*) e reformadores ou codificadores automáticos (*reformers/autoencoders*) [Tabassi et al. 2019]. Essas técnicas levam a características como defesa ativa, classificadores planos, aumento de robustez, robustez por randomização e processamento de dados cifrados [Liu et al. 2018]. Se ressalta que essas técnicas de proteção, apesar de defenderem a etapa de testes, também são implantadas na etapa de treinamento do *pipeline* de aprendizado de máquina [Tabassi et al. 2019].

A técnica de treinamento adversarial [Goodfellow et al. 2014] utiliza amostras adversariais no treinamento, mas com rótulos de saída corretos, com objetivo de minimizar erros de classificação causados por exemplos adversários. Essencialmente, a técnica utiliza força bruta para gerar muitos exemplos adversariais, gerando pertubações durante a etapa de treinamento, visando aumentar robustez do modelo alvo [Chakraborty et al. 2018].

O conceito de treinamento adversário envolve treinar o classificador para generalizar as amostras adversariais e as amostras limpas. No esquema de treinamento convencional, os dados de treinamento passam pelo modelo e a perda de previsão é retro-propagada para melhorar os resultados da classificação. Como consequência, o modelo generalizará a distribuição dos dados de treinamento para produzir uma previsão precisa de seus rótulos.

O treinamento adversarial expande os métodos convencionais de treinamento ao adicionar uma etapa extra ao procedimento de treinamento para uma rede neural, conforme ilustrado na Figura 1.5. O objetivo é fazer com que o modelo possa generalizar tanto os dados limpos quanto os dados adversariais gerados pelos métodos de ataque utilizados no treinamento adversarial, aprimorando a robustez do modelo contra este ataque [Zhao et al. 2022a].

Um segundo exemplo de mecanismo de defesa é o mascaramento de gradiente, cuja função é reduzir a sensibilidade do modelo a pequenas perturbações nas entradas. O objetivo é atingido através do cálculo das derivadas de primeira ordem do modelo em

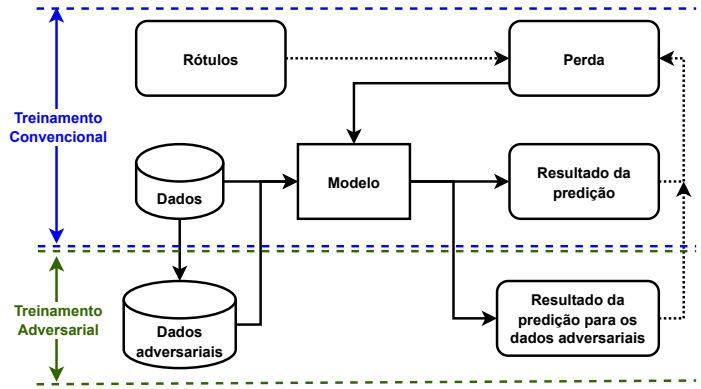

Figura 1.5. Treinamento adversarial (adaptada de [Zhao et al. 2022a])

relação às suas entradas e da minimização dessas derivadas durante a etapa de aprendizagem.

O mascaramento de gradiente é baseado na ideia de que, se o modelo não for diferenciável ou o gradiente do modelo for zero nos pontos de dados, os ataques baseados em gradiente serão ineficazes [Duddu 2018]. A ideia dessa abordagem é não fornecer gradientes úteis ao gerar os exemplos de gradiente e enganar o adversário, fazendo-o utilizar ataques fáceis de mitigar [Lee et al. 2021].

1.5. Consequências

As consequências de um ataque resultam em violações de integridade, disponibilidade, confidencialidade ou privacidade [Barreno et al. 2006b, Huang et al. 2011]. A Tabela 1.1 apresenta um exemplo de modelo de ataque que considera ataques causativos e exploratórios com consequências de integridade e disponibilidade. Um ataque causativo altera o processo de treinamento através de influências sobre os dados de treinamento. Já um ataque exploratório não altera o processo de treinamento, mas utiliza técnicas como análise *offline* ou testes do aprendizado para descobrir informações sobre o modelo. Esses ataques podem ser direcionados ou indiscriminados. Enquanto um ataque direcionado possui um alvo e um ponto, ou um conjunto pequeno de pontos, um ataque indiscriminado possui um objetivo mais flexível, isto é, que pode envolver alvos e classes mais gerais, como "qualquer falso positivo".

Se tratando de violação de segurança, o exemplo de modelo de ataque assume violações de integridade e disponibilidade. Nesse modelo, um ataque de integridade resulta em uma intrusão considerada normal, isto é, um falso negativo. No contexto de detecção de *malwares*, um falso negativo indica que o classificador categoriza um aplicativo como benigno mesmo ele sendo maligno. Já um ataque de disponibilidade é mais amplo, resultando em muitos erros de classificação, tanto falsos negativos quanto falsos positivos (*i.e.*, aplicativo benigno detectado como *malware*), tornando o sistema inutilizável.

Tabela 1.1. Modelo de ataque (adaptado de [Barreno et al. 2006b])

Influência do Ataque		**Integridade**	**Disponibilidade**
Causativo	*direcionado*	Permitir uma intrusão específico	Criar erros o suficiente para tornar o sistema inutilizável
	indiscriminado	Permitir ao menos uma intrusão	Criar erros o suficiente para tornar o aprendizado inutilizável
Exploratório	*direcionado*	Encontrar uma intrusão explorável de poucas possibilidades	Encontrar um conjunto de pontos classificados erroneamente pelo aprendizado
	indiscriminado	Encontrar uma intrusão explorável	Encontrar um conjunto de pontos classificados erroneamente pelo aprendizado

Alguns desses ataques podem ser difíceis de detectar. Por exemplo, os modelos que são continuamente treinados *online* são tipicamente projetados para acomodar mudanças de longo prazo na distribuição dos dados vistos pelo aprendizado. Entretanto, apesar de o treinamento *online* ser mais flexível, esse comportamento também simplifica os ataques causativos. Como o aprendizado *online* muda a função de predição com o passar do tempo, um adversário consegue explorar esse comportamento para intervir nas mudanças dos modelos. Nesse contexto, ataques causativos graduais podem ser difíceis de serem detectados [Barreno et al. 2006b].

Se ressalta que existe uma relação forte entre as consequências e os mecanismos de defesa. Para cada combinação de ataque (alvo, técnica e conhecimento) e defesa(s), há um resultado onde os níveis de severidade podem variar quanto a consequências. Por exemplo, as violações de integridade [Biggio et al. 2013] afetam o processo de inferência e resultam em redução de confidência ou classificação errônea para qualquer classe diferente da classe original. No contexto de classificações errôneas, podem ocorrer aquelas direcionadas para entradas de uma classe alvo específica de saída e outras direcionadas a segmentos de origem de uma entrada específica para uma classe alvo de saída (*e.g.*, o adversário pode levar o aprendizado a considerar instâncias malignas no filtro de falsos negativos). No caso do aprendizado não supervisionado, uma violação de integridade pode produzir uma representação de entrada sem significado em um extrator de características não supervisionado. Já no caso de aprendizado por reforço, uma violação de integridade pode levar o agente de aprendizado a se comportar de maneira não inteligente ou ainda degradar o seu desempenho. Por exemplo, o ataque *red herring* [Newsome et al. 2006] introduz características espúrias para o aprendizado construir uma assinatura válida a partir de dados falhos. Tão logo a assinatura é construída, o adversário remove as características espúrias para evitar a detecção. Resumidamente, este ataque pode levar sistemas de detecção de *malwares* baseados em assinaturas a classificar erroneamente uma assinatura.

As violações de disponibilidade [Huang et al. 2011, Machado et al. 2021] induzem reduções de qualidade, como a velocidade de inferência, ou acesso, como negação de serviço, ao ponto de tornar o componente de aprendizado de máquina indisponível aos usuários. Apesar de violações de disponibilidade envolverem também reduções de confiança e classificações errôneas, diferentemente de violações de integridade, os resultados no âmbito de comportamento, como velocidade inaceitável ou negação de acesso, levam o modelo a ações e saídas inutilizáveis. Por exemplo, o ataque *correla-*

ted outlier [Newsome et al. 2006] adiciona características espúrias em instâncias de treinamento positivas, levando o filtro a bloquear tráfego benigno. Similarmente, o ataque *allergy* [Chung and Mok 2006] envia pacotes que imitam o tráfego alvo do aprendizado, fazendo o modelo criar regras que irão bloquear o tráfego legítimo.

As violações de confidencialidade [Huang et al. 2011, Biggio et al. 2015] ocorrem quando um adversário extraí ou infere informação utilizável sobre o modelo e dados. Exemplos de ataques de confidencialidade, como o de extração [Quiring et al. 2018, Zhang et al. 2021d], revelam a arquitetura ou os parâmetros do modelo. No entanto, um ataque de "oráculo" [Quiring et al. 2018, Rawal et al. 2021, Ibitoye et al. 2019] permite ao atacante construir um modelo substituto. Outros ataques, como o de inversão [He et al. 2019, Madono et al. 2021], permitem relevar informações sobre os dados. Por exemplo, o adversário pode explorar o modelo alvo para recuperar dados faltantes utilizando entradas parcialmente conhecidas. No caso de ataques de inferência de associação [Shokri et al. 2017, Song et al. 2019, Choquette-Choo et al. 2021, Truex et al. 2019], o adversário realiza testes de associação para determinar se um indivíduo foi incluído no *dataset* utilizado para treinar o modelo alvo.

Por fim, as violações de privacidade [Huang et al. 2011, Biggio et al. 2015] podem ser consideradas um tipo específico de violações de confidencialidade. A violação de privacidade ocorre quando o adversário consegue obter informações pessoais sobre um ou mais indivíduos e entradas legítimas do modelo, sejam elas parte do conjunto de dados de treinamento ou não. Um exemplo pode considerar um adversário que consegue adquirir ou extrair os registros médicos de um indivíduo, violando políticas de privacidade.

Na Tabela 1.2 apresentamos alguns exemplos de ataques, defesas e consequências. Para cada ataque ou defesa, apresentamos também a técnica, as características e a ideia básica. Como é possível destacar, existem diversos ataques e mecanismos de defesa, que levam a diferentes consequências. Os exemplos de ataques e defesas atuam sobre as etapas (alvos) de treinamento e testes do *pipeline* de aprendizado de máquina. Também é possível observar na tabela que há consequências que levam a violações de integridade, disponibilidade e privacidade. Ao mesmo tempo, existe uma relação de mecanismos de defesa que podem ser utilizados para mitigar as consequências, como técnicas de sanitização de dados, treinamento adversarial, privacidade diferencial e criptografia homomórfica. Para maiores detalhes sobre os ataques e os mecanismos de defesa apresentados na tabela, um amplo material está disponível na revisão de literatura apresentada em [Liu et al. 2018].

1.6. *Malwares*: técnicas e desafios

A medida que defensores têm desenvolvido diversos mecanismos baseados em aprendizado de máquina para a detecção de aplicativos Android maliciosos, atacantes vêm criando métodos adversariais para evadir os sistemas de detecção. Nesta seção apresentamos técnicas de evasão populares e recentes utilizadas em *malwares* modernos. Além disso, também introduzimos algumas técnicas de evasão específicas, utilizadas por *malwares* Android. Algumas dessas técnicas, como o uso de inteligência artificial na geração e mutação de *malwares*, representam um desafio significativo no contexto de *adversarial machine learning*. Ao descobrir informações sobre os modelos de classificação, os adversários estão cada vez mais criativos e sofisticados no desenvolvimento de técnicas de

Tabela 1.2. Ataques, defesas e consequências (adaptada de [Liu et al. 2018])

Ataque ou Defesa			Consequências	Características	Ideia básica
Tipo	**Técnica**	**Alvo/Etapa**			
Ataque	Envenenamento	Treinamento	Violação de integridade e disponibilidade	Ataques causativos, direcionados ou indiscriminados,	Injetar amostras contraditórias, ou modificar características e rótulos do dataset de treinamento.
Ataque	Evasão	Testes	Violação de integridade e disponibilidade	Ataques exploratórios ou direcionados	Criar amostras contraditórias para evitar a detecção nos sistemas alvo.
Ataque	Personificação	Testes	Violação de integridade e disponibilidade	Ataques exploratórios, direcionados ou indiscriminados	Criar amostras contraditórias para imitar amostras alvo e confundir sistemas alvo.
Ataque	Inversão	Testes	Violação de privacidade	Ataques exploratórios ou direcionados	Roubar informações sensíveis dos classificadores, ou datasets alvos.
Defesa	Sanitização de dados	Treinamento	Proteção de integridade e disponibilidade	Defesa ativa	Sanitizar dados de treinamento e rejeitar amostras que impactam negativamente os classificadores.
Defesa	Treinamento adversarial	Treinamento	Proteção de integridade e disponibilidade	Defesa ativa	Considerar amostras contraditórias no treinamento e realizar a mineração delas entre os datasets.
Defesa	Destilação	Treinamento	Proteção de integridade e disponibilidade	Classificador plano	Utilizar a rotulação de probabilidade gerada por DNN para os dados de treinamento.
Defesa	Método de conjunto	Treinamento	Proteção de integridade e disponibilidade	Aumentar a robustez	Integrar diferentes classificadores ou técnicas de defesa para mitigar amostras contraditórias.
Defesa	Privacidade diferencial	Treinamento & Testes	Proteção de privacidade	Robustez por randomização	Adicionar ruídos aleatórios nos dados iniciais ou métodos de randomização no treinamento.
Defesa	Criptografia homomórfica	Treinamento & Testes	Proteção de privacidade	Processamento de dados cifrados	Viabilizar o processamento de dados cifrados e proteção da privacidade em computação multipartes.

evasão, por exemplo.

1.6.1. Técnicas populares de evasão

A seguir apresentamos algumas das técnicas de evasão consideradas populares no contexto *malwares* [Tavares 2022]:

Suspensão prolongada: o *malware* realiza uma suspensão prolongada (*e.g.*, 15 minutos). Ao interromper a sua execução, o *malware* evade a etapa de análise e posterga a infecção propriamente dita.

Bomba lógica: o *malware* agenda a sua execução em uma data e hora pré-determinas visando evadir sistemas de detecção que atuam apenas no momento de instalação ou na primeira execução do aplicativo contaminado.

Código de travamento: ocorre quando o *malware* utiliza ciclos da CPU (*Central Processing Unit*), através de cargas maliciosas, para atrasar o processo até completar a infecção do sistema.

***Stegosploit*:** é uma maneira de ocultar códigos maliciosos em imagens. Por exemplo, explorações de navegador do tipo *drive-by* podem ser criadas e entregues por meio de um arquivo de imagem simples. Esses tipos de cargas úteis são eficientes porque são furtivos e difíceis de serem detectados.

Ofuscação de código, criptografia ou compactação: são técnicas bastante populares no cenário de *malwares*. Com essas técnicas, adversários ofuscam ou criptografam partes do código de execução do *malware* para evitar uma análise estática, dificultar a compreensão e evadir mecanismos de detecção em tempo real.

Malwares sofisticados, como no caso do DeepLocker [Kirat et al. 2018, Haran 2018], utilizam inteligência artificial para permanecer escondido até atingir o alvo, isto é, uma vítima específica. Uma vez que o modelo de inteligência artificial detecta o alvo (*e.g.*, reconhecimento facial, geolocalização, reconhecimento de voz), o *malware* é ativado. Através de um código cuidadosamente projetado e esse comportamento sorrateiro, o *malware* permanece escondido e evade sistemas de detecção. O principal perigo desta ameaça, que utiliza inteligência artificial, é potencialmente infectar milhares de sistemas sem ser detectada. Uma das particularidades do DeepLocker é que a implementação da inteligência artificial para determinar a ativação faz com que o ataque do *malware* seja muito difícil de ser detectado e evitado, pois o *payload* (carga útil do ataque) será liberado somente no momento do ataque.

1.6.2. *Malwares* Android: técnicas de evasão

No contexto de *malwares* Android, os adversários utilizam cada vez mais técnicas de evasão como criptografia, transformações de pacote e código, ofuscação de código e anti-emulação para evitar sistemas de detecção baseados em assinaturas e aprendizado de máquina [Elsersy et al. 2022, Sihag et al. 2021, Xue et al. 2017, Kim et al. 2016, Jusoh et al. 2021]. Na prática, as transformações triviais não exigem alterações no código ou no nível de *bytecode* e dificultam particularmente a análise baseada em assinaturas [Tam et al. 2017]. Por exemplo, o ato de descompactar e reempacotar arquivos APK é uma forma trivial de ofuscação que não modifica nenhum dado no manifesto do aplicativo. Isso ocorre porque, ao reempacotar o novo aplicativo, ele é assinado com chaves personalizadas ao invés das chaves do desenvolvedor original. Portanto, as assinaturas criadas com as chaves do desenvolvedor, ou a soma de verificação do aplicativo original, se tornam ineficazes, permitindo que um adversário distribua aplicativos aparentemente legítimos com assinaturas diferentes [Tam et al. 2017]. Os arquivos DEX (*Dalvik Executable*) do APK, que representam os arquivos de código compilado do aplicativo, também podem ser descompilados e remontados. Na remontagem, os componentes podem ser reorganizados ou ainda as suas representações podem ser alteradas.

1.6.3. *Malwares* Android: técnicas de envenenamento

Os ataques de envenenamento criam padrões de comportamento que provam ser falsos positivos, confundindo a rotulagem precisa dos sistemas de detecção [Shackleford 2022, EBCRG 2019]. O que pode ocorrer na prática é o sistema anti-*malware*, baseado em aprendizado de máquina, classificar incorretamente o aplicativo devido à alteração da marcação do *malware* como benigno. Essa prática é facilitada, pois os sistemas realizam suas análises com assinaturas ou padrões conhecidos para classificar uma amostra como benigna ou maligna [Mahlangu et al. 2019, Van der Walt et al. 2018].

Algumas técnicas de envenenamento de sistemas de aprendizado de máquina utilizam a mutação de *malware*, também conhecida como metamorfismo. A mutação consiste no *malware* se modificar e imitar comportamentos esperados como sendo de amostras

benignas [Gray 2022]. Um *malware* metamórfico altera seu código durante a propagação para que cada instância da mesma família exiba pouca semelhança com outra variante [Bala et al. 2021, Elsersy et al. 2022].

Atualmente, adversários utilizam redes Generativas Adversariais (*Generative Adversarial Networks* - GANs) [Bau et al. 2018] para gerar mutações em *malwares*, o que pode ser considerado bastante efetivo para enganar sistemas de detecção [EBCRG 2019]. As GANs possuem em seu núcleo uma parte geradora e outra discriminadora, ou seja, enquanto a rede gera amostras, dado um conjunto de entrada com semelhanças benignas, a outra parte aprende como classificar e com isso aumenta a precisão e a aceitabilidade das amostras geradas.

1.7. Resumo do Capítulo

Neste capítulo foi apresentado uma introdução ao estado da arte do aprendizado de máquina adversarial (*adversarial machine learning*) na perspectiva de um ambiente naturalmente adversarial, como no caso da segurança cibernética. Para introduzir o tema, foi utilizado como guia a taxonomia do NIST [Tabassi et al. 2019], que define três grandes categorias: *ataques*, *defesas* e *consequências*. Na categoria dos ataques temos os *alvos* (*e.g.*, etapas do aprendizado de máquina), as *técnicas* (*e.g.*, ataques de envenenamento, ataques exploratórios) e o *conhecimento* (*e.g.*, "caixa preta", "caixa branca"). Na categoria das defesas, consideramos duas subcategorias: os *contra-ataques de treinamento* e *contra-ataques de testes e inferência*. Finalmente, na categoria das consequências abordamos as *violações de integridade*, *disponibilidade* e *confidencialidade*. As *violações de privacidade*, cada vez mais comuns no contexto de aprendizado de máquina adversarial, são um sub-conjunto das violações de confidencialidade. Na Seção 1.5, destacamos também um modelo de ataque na Tabela 1.1, que identifica a influência do ataque (*e.g.*, *exploratório direcionado*) e os respectivos objetivos de integridade e disponibilidade. Adicionalmente, na Tabela 1.2, apresentamos exemplos de técnicas de ataques e defesas, alvos, consequências, características e ideias básicas.

1.8. Exercícios

(Q_1) **Marque a alternativa que corresponde aos três grandes eixos da taxonomia do NIST:**

a) Ataques, Defesas, Características
b) Alvos, Defesas, Conhecimento
c) Alvos, Técnicas, Conhecimento
d) Alvos, Defesas, Consequências
e) Ataques, Defesas, Consequências

(Q_2) **Marque a alternativa que contém somente técnicas de ataque:**

a) Envenenamento, Testes, Treinamento
b) Evasão, Personificação, Classificador plano
c) Sanitização de dados, Personificação, Evasão
d) Envenenamento, Evasão, Inversão, Personificação
e) Evasão, Envenenamento, Violação de Privacidade, Inversão

(Q_3) **Leia as frases a seguir sobre os ataques a sistemas baseados em aprendizado de máquina:**

I - O ataque de envenenamento é também conhecido como ataque de inversão e tem como objetivo apagar dados da etapa de treinamento dos modelos de aprendizado de máquina.
II - Ataques de inversão são impossíveis de serem detectados e evitados.
III - Os ataques de inversão podem ser de dois tipos, os de modificação de dados ou os de modificação de rótulos.
Marque a alternativa que apresenta todas as afirmações incorretas:
a) Apenas I
b) Apenas II
c) Apenas I e II
d) Apenas I e III
e) Apenas III

(Q_4) **Marque a alternativa que contém somente técnicas de defesa:**

a) Sanitização de dados, Destilação, Treinamento, Método de conjunto
b) Sanitização de dados, Destilação, Treinamento adversarial, Privacidade diferencial
c) Privacidade diferencial, Criptografia homomórfica, Personificação, Método de conjunto
d) Privacidade diferencial, Criptografia homomórfica, Personificação, Destilação
e) Privacidade diferencial, Criptografia homomórfica, Violação de Privacidade, Treinamento adversarial

(Q_5) **Leia as frases a seguir sobre as técnicas de defesa em *pipelines* de aprendizado de máquina:**

I - A técnica de treinamento adversarial tem como objetivo minimizar erros de classificação através da utilização de amostras adversariais na etapa do treinamento do modelo.

II - O mascaramento de gradiente tem por objetivo reduzir a sensibilidade do modelo a pequenas perturbações nas entradas.
III - O método de conjunto integra diferentes classificadores ou técnicas de defesa para mitigar o efeito de amostras contraditórias.
IV - A privacidade diferencial adiciona ruídos pseudo-aleatórios nos dados iniciais ou utiliza ainda métodos de randomização no treinamento dos modelos.
Marque a alternativa que apresenta todas as afirmações corretas:
a) Apenas I
b) Apenas II
c) Apenas I e II
d) Apenas I, II e III
e) Todas estão corretas

(Q_6) **Marque a alternativa que contém somente consequências válidas de ataques a sistemas de aprendizado de máquina:**

a) Violação de integridade, Violação de disponibilidade, Violação de privacidade, Violação de conformidade
b) Violação de integridade, Violação de autenticação, Violação de privacidade, Violação de conformidade
c) Violação de integridade, Violação de disponibilidade, Violação de privacidade, Violação de defesa
d) Violação de disponibilidade, Violação de integridade, Violação de confidencialidade, Violação de direito
e) Violação de disponibilidade, Violação de integridade, Violação de confidencialidade, Violação de privacidade

(Q_7) **Com relação às técnicas de defesa, é correto afirmar que:**

a) A sanitização de dados ocorre no mundo físico, ou seja, nos sensores que geram os dados (e.g., sensor de temperatura).
b) A sanitização de dados ocorre em todas as etapas do *pipeline* de aprendizado de máquina.
c) O aumento da robustez dos algoritmos ocorre durante a coleta dos dados e treinamento do modelo.
d) Os mecanismos de avaliação de segurança são incorporados na etapa de coleta dos dados.
e) As técnicas de preservação da privacidade são aplicadas no último estágio (inferência) do ciclo de vida do aprendizado de máquina.

(Q_8) **Ataques de evasão utilizam tipicamente algoritmos de busca baseados em gradiente, como L-BFGS, ILLC, FGSM, JSMA, DeepFool e CPPN EA Fool. Sobre o ataque de evasão é correto afirmar que:**

a) É um ataque causativo e exploratório onde o adversário resolve problemas de otimização de restrições para encontrar pequenas perturbações de entrada que conseguem gerar grandes modificações nas funções de perda e resultar em saídas de classificações equivocadas.
b) É um ataque causativo onde o adversário resolve problemas de otimização de restrições para encontrar pequenas perturbações de entrada que conseguem gerar

grandes modificações nas funções de perda e resultar em saídas de classificações equivocadas.
c) É um ataque exploratório onde o adversário resolve problemas de otimização de restrições para encontrar grandes perturbações de entrada que conseguem gerar pequenas modificações nas funções de perda e resultar em saídas de classificações equivocadas.
c) É um ataque causativo onde o adversário resolve problemas de otimização de restrições para encontrar grandes perturbações de entrada que conseguem gerar pequenas modificações nas funções de perda e resultar em saídas de classificações equivocadas.
d) É um ataque causativo onde o adversário resolve problemas de otimização de restrições para encontrar grandes perturbações de entrada que conseguem gerar pequenas modificações nas funções de perda e resultar em saídas de classificações equivocadas.
e) É um ataque exploratório onde o adversário resolve problemas de otimização de restrições para encontrar pequenas perturbações de entrada que conseguem gerar grandes modificações nas funções de perda e resultar em saídas de classificações equivocadas.

(Q_9) **Ameaças de segurança estão tipicamente presentes nas etapas de treinamento e testes do ciclo de vida do aprendizado de máquina. Sobre as ameaças de segurança é correto afirmar que:**

a) Ataques de envenenamento ocorrem na etapa de treinamento do modelo.
b) Ataques de envenenamento ocorrem na etapa de inferência do ciclo de vida do aprendizado de máquina.
c) Ataques de evasão, personificação e inversão ocorrem em todas as etapas do ciclo de vida do aprendizado de máquina.
d) Ataques de evasão, personificação e inversão ocorrem apenas na etapa de treinamento do ciclo de vida do aprendizado de máquina.
e) Ataques de evasão, personificação e inversão são combatidos essencialmente com técnicas de sanitização de dados.

(Q_{10}) **Com relação à definição básica de aprendizado de máquina adversarial (*adversarial machine learning*), é correto afirmar que:)**

a) Aprendizado de máquina adversarial estuda apenas técnicas de defesa contra oponentes em ambientes naturalmente adversariais, como no caso da segurança cibernética. Nesses ambientes é fundamental conseguirmos antecipar que o adversário irá fazer para causar falhas no ciclo de vida do aprendizado de máquina.
b) Aprendizado de máquina adversarial estuda apenas técnicas de ataque contra oponentes em ambientes naturalmente adversariais, como no caso da segurança cibernética. Nesses ambientes é fundamental conseguirmos antecipar que o adversário irá fazer para causar falhas no ciclo de vida do aprendizado de máquina.
c) Aprendizado de máquina adversarial estuda técnicas efetivas contra oponentes em ambientes naturalmente adversariais, como no caso da segurança cibernética. Nesses ambientes é fundamental conseguirmos antecipar que o adversário irá fazer para causar falhas no ciclo de vida do aprendizado de máquina.

d) Aprendizado de máquina adversarial estuda técnicas efetivas contra oponentes em ambientes naturalmente adversariais, como no caso da segurança cibernética. Nesses ambientes é fundamental conseguirmos enganar o adversário antes de ele roubar o código do sistema.
e) Aprendizado de máquina adversarial estuda técnicas efetivas contra oponentes em ambientes naturalmente adversariais, como no caso da segurança cibernética. Nesses ambientes é fundamental conversarmos com e educarmos agentes mal-intencionados.

Gabarito

(Q_1) Resposta: e
(Q_2) Resposta: d
(Q_3) Resposta: c
(Q_4) Resposta: b
(Q_5) Resposta: e
(Q_6) Resposta: e
(Q_7) Resposta: e
(Q_8) Resposta: e
(Q_9) Resposta: a
(Q_{10}) Resposta: c

Referências

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Ahmad, R. and Alsmadi, I. (2021). Machine learning approaches to iot security: A systematic literature review. *Internet of Things*, 14:100365.

Akhtar, N. and Mian, A. (2018). Threat of adversarial attacks on deep learning in computer vision: A survey. *Ieee Access*, 6:14410–14430.

Ashtiani, M. N. and Raahemi, B. (2021). Intelligent fraud detection in financial statements using machine learning and data mining: a systematic literature review. *IEEE Access*, 10:72504–72525.

Aung, A. P. P., Wang, X., Yu, R., An, B., Jayavelu, S., and Li, X. (2022). DO-GAN: A double oracle framework for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11275–11284.

Bala, N., Ahmar, A., Li, W., Tovar, F., Battu, A., and Bambarkar, P. (2021). DroidEnemy: battling adversarial example attacks for android malware detection. *Digital Communications and Networks*.

Barreno, M., Nelson, B., Joseph, A. D., and Tygar, J. D. (2010). The security of machine learning. *Machine Learning*, 81(2):121–148.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. (2006a). Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, ASIACCS '06, page 16–25, New York, NY, USA. Association for Computing Machinery.

Barreno, M., Nelson, B., Sears, R., Joseph, A. D., and Tygar, J. D. (2006b). Can machine learning be secure? In *Proceedings of the 2006 ACM Symposium on Information, computer and communications security*, pages 16–25.

Bau, D., Zhu, J.-Y., Strobelt, H., Zhou, B., Tenenbaum, J. B., Freeman, W. T., and Torralba, A. (2018). GAN dissection: Visualizing and understanding generative adversarial networks. https://arxiv.org/pdf/1811.10597.pdf. arXiv preprint arXiv:1811.10597.

Biggio, B., Fumera, G., and Roli, F. (2013). Security evaluation of pattern classifiers under attack. *IEEE transactions on knowledge and data engineering*, 26(4):984–996.

Biggio, B. and Roli, F. (2018). Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition*, 84:317–331.

Biggio, B., Russu, P., Didaci, L., Roli, F., et al. (2015). Adversarial biometric recognition: A review on biometric system security from the adversarial machine-learning perspective. *IEEE Signal Processing Magazine*, 32(5):31–41.

Caruana, R. and Niculescu-Mizil, A. (2006). An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd international conference on Machine learning*, pages 161–168.

Chakraborty, A., Alam, M., Dey, V., Chattopadhyay, A., and Mukhopadhyay, D. (2018). Adversarial attacks and defences: A survey. *arXiv preprint arXiv:1810.00069*.

Chen, L., Xu, Y., Xie, F., Huang, M., and Zheng, Z. (2021). Data poisoning attacks on neighborhood-based recommender systems. *Transactions on Emerging Telecommunications Technologies*, 32(6):e3872.

Choquette-Choo, C. A., Tramer, F., Carlini, N., and Papernot, N. (2021). Label-only membership inference attacks. In *International conference on machine learning*, pages 1964–1974. PMLR.

Chung, S. P. and Mok, A. K. (2006). Allergy attack against automatic signature generation. In *International Workshop on Recent Advances in Intrusion Detection*, pages 61–80. Springer.

Collins, B., Hoang, D. T., Nguyen, N. T., and Hwang, D. (2021). Trends in combating fake news on social media–a survey. *Journal of Information and Telecommunication*, 5(2):247–266.

Cretu, G. F., Stavrou, A., Locasto, M. E., Stolfo, S. J., and Keromytis, A. D. (2008). Casting out demons: Sanitizing training data for anomaly sensors. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 81–95.

Daoudi, N., Allix, K., Bissyandé, T. F., and Klein, J. (2022). A deep dive inside drebin: An explorative analysis beyond android malware detection scores. *ACM Trans. Priv. Secur.*, 25(2).

Dasgupta, D., Akhtar, Z., and Sen, S. (2022). Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation*, 19(1):57–106.

Diakonikolas, I., Kamath, G., Kane, D., Li, J., Moitra, A., and Stewart, A. (2019). Robust estimators in high-dimensions without the computational intractability. *SIAM Journal on Computing*, 48(2):742–864.

Diakonikolas, I., Kamath, G., Kane, D. M., Li, J., Moitra, A., and Stewart, A. (2017). Being robust (in high dimensions) can be practical. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 999–1008. PMLR.

Diaz-Gonzalez, F. A., Vuelvas, J., Correa, C. A., Vallejo, V. E., and Patino, D. (2022). Machine learning and remote sensing techniques applied to estimate soil indicators–review. *Ecological Indicators*, 135:108517.

Dibaei, M., Zheng, X., Xia, Y., Xu, X., Jolfaei, A., Bashir, A. K., Tariq, U., Yu, D., and Vasilakos, A. V. (2021). Investigating the prospect of leveraging blockchain and machine learning to secure vehicular networks: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 23(2):683–700.

Duddu, V. (2018). A survey of adversarial machine learning in cyber warfare. *Defence Science Journal*, 68(4):356.

Dwork, C. (2008). Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer.

EBCRG (2019). How can hackers use machine learning? https://ebcs.gsu.edu/2019/10/24/how-can-hackers-use-machine-learning/.

Elhami Fard, N. and Selmic, R. R. (2022). Adversarial attacks on heterogeneous multi-agent deep reinforcement learning system with time-delayed data transmission. *Journal of Sensor and Actuator Networks*, 11(3):45.

Elsersy, W. F., Feizollah, A., and Anuar, N. B. (2022). The rise of obfuscated android malware and impacts on detection methods. *PeerJ Computer Science*, 8:e907.

Fredrikson, M., Jha, S., and Ristenpart, T. (2015). Model inversion attacks that exploit confidence information and basic countermeasures. In *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*, pages 1322–1333.

Garg, A. and Mago, V. (2021). Role of machine learning in medical research: A survey. *Computer Science Review*, 40:100370.

Gaurav, A., Gupta, B. B., and Panigrahi, P. K. (2022). A comprehensive survey on machine learning approaches for malware detection in iot-based enterprise information system. *Enterprise Information Systems*, pages 1–25.

Geetha, R. and Thilagam, T. (2021). A review on the effectiveness of machine learning and deep learning algorithms for cyber security. *Archives of Computational Methods in Engineering*, 28(4):2861–2879.

Ghahramani, Z. (2003). Unsupervised learning. In *Summer school on machine learning*, pages 72–112. Springer.

Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014). Explaining and harnessing adversarial examples.

Gray, L. (2022). 5 interesting applications of ai in cyber security in 2022. https://devcount.com/ai-on-security/.

Haran, J. M. (2018). DeepLocker: malware que utiliza inteligência artificial e é ativado através do reconhecimento facial. https://www.welivesecurity.com/br/2018/08/13/deeplocker-e-ativado-atraves-do-reconhecimento-facial/.

He, Z., Zhang, T., and Lee, R. B. (2019). Model inversion attacks against collaborative inference. In *Proceedings of the 35th Annual Computer Security Applications Conference*, pages 148–162.

Huang, L., Joseph, A. D., Nelson, B., Rubinstein, B. I., and Tygar, J. D. (2011). Adversarial machine learning. In *Proceedings of the 4th ACM workshop on Security and artificial intelligence*, pages 43–58.

Huang, Y., Verma, U., Fralick, C., Infantec-Lopez, G., Kumar, B., and Woodward, C. (2019). Malware evasion attack and defense. In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, pages 34–38. IEEE.

Ibitoye, O., Abou-Khamis, R., Matrawy, A., and Shafiq, M. O. (2019). The threat of adversarial attacks on machine learning in network security–a survey. *arXiv preprint arXiv:1911.02621*.

Jagielski, M., Oprea, A., Biggio, B., Liu, C., Nita-Rotaru, C., and Li, B. (2018). Manipulating machine learning: Poisoning attacks and countermeasures for regression learning. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 19–35. IEEE.

Jusoh, R., Firdaus, A., Anwar, S., Osman, M. Z., Darmawan, M. F., and Ab Razak, M. F. (2021). Malware detection using static analysis in android: a review of feco (features, classification, and obfuscation). *PeerJ Computer Science*, 7:e522.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

Kakani, V., Nguyen, V. H., Kumar, B. P., Kim, H., and Pasupuleti, V. R. (2020). A critical review on computer vision and artificial intelligence in food industry. *Journal of Agriculture and Food Research*, 2:100033.

Khalid, F., Hanif, M. A., Rehman, S., and Shafique, M. (2018). Security for machine learning-based systems: Attacks and challenges during training and inference. In *2018 International Conference on Frontiers of Information Technology (FIT)*, pages 327–332.

Kim, M., Lee, T. J., Shin, Y., and Youm, H. Y. (2016). A study on behavior-based mobile malware analysis system against evasion techniques. In *2016 international conference on information networking (ICOIN)*, pages 455–457. IEEE.

Kirat, D., Jang, J., and Stoecklin, M. P. (2018). DeepLocker - concealing targeted attacks with AI locksmithing. https://www.blackhat.com/us-18/briefings/schedule/#deeplocker---concealing-targeted-attacks-with-ai-locksmithing-11549.

Koh, P. W., Steinhardt, J., and Liang, P. (2018). Stronger data poisoning attacks break data sanitization defenses.

Koh, P. W., Steinhardt, J., and Liang, P. (2022). Stronger data poisoning attacks break data sanitization defenses. *Machine Learning*, 111(1):1–47.

Kotenko, I., Izrailov, K., and Buinevich, M. (2022). Static analysis of information systems for iot cyber security: A survey of machine learning approaches. *Sensors*, 22(4):1335.

Kuznetsov, P., Edmunds, R., Xiao, T., Iqbal, H., Puri, R., Golmant, N., and Shih, S. (2019). Iadversarial machine learning. In Yampolskiy, R. V., editor, *Artificial Intelligence Safety and Security*, chapter 17, pages 235–248. Chapman and Hall/CRC.

Lai, K. A., Rao, A. B., and Vempala, S. (2016). Agnostic estimation of mean and covariance. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 665–674.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Lee, H., Bae, H., and Yoon, S. (2021). Gradient masking of label smoothing in adversarial robustness. *IEEE Access*, 9:6453–6464.

Lee, S., Kim, H., and Lee, J. (2022). Graddiv: Adversarial robustness of randomized neural networks via gradient diversity regularization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Li, Y., Jiang, Y., Li, Z., and Xia, S.-T. (2022). Backdoor learning: A survey. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–18.

Liu, Q., Li, P., Zhao, W., Cai, W., Yu, S., and Leung, V. C. M. (2018). A survey on security threats and defensive techniques of machine learning: A data driven view. *IEEE Access*, 6:12103–12117.

Machado, G. R., Silva, E., and Goldschmidt, R. R. (2021). Adversarial machine learning in image classification: A survey toward the defender's perspective. *ACM Computing Surveys (CSUR)*, 55(1):1–38.

Madono, K., Tanaka, M., Onishi, M., and Ogawa, T. (2021). Sia-gan: Scrambling inversion attack using generative adversarial network. *IEEE Access*, 9:129385–129393.

Mahlangu, T., January, S., Mashiane, T., Dlamini, M., Ngobeni, S., and Ruxwana, N. (2019). Data poisoning: Achilles heel of cyber threat intelligence systems. In *Proceedings of the ICCWS 2019 14th International Conference on Cyber Warfare and Security: ICCWS*.

Mani, N., Moh, M., and Moh, T.-S. (2021). Defending deep learning models against adversarial attacks. *International Journal of Software Science and Computational Intelligence (IJSSCI)*, 13(1):72–89.

Manoj, B., Sadeghi, M., and Larsson, E. G. (2021). Adversarial attacks on deep learning based power allocation in a massive mimo network. In *ICC 2021-IEEE International Conference on Communications*, pages 1–6. IEEE.

Martins, N., Cruz, J. M., Cruz, T., and Abreu, P. H. (2020). Adversarial machine learning applied to intrusion and malware scenarios: a systematic review. *IEEE Access*, 8:35403–35419.

Maulud, D. H., Zeebaree, S. R., Jacksi, K., Sadeeq, M. A. M., and Sharif, K. H. (2021). State of art for semantic analysis of natural language processing. *Qubahan Academic Journal*, 1(2):21–28.

Meenakshi, K. and Maragatham, G. (2020). A review on security attacks and protective strategies of machine learning. In Hemanth, D. J., Kumar, V. D. A., Malathi, S., Castillo, O., and Patrut, B., editors, *Emerging Trends in Computing and Expert Technology*, pages 1076–1087, Cham. Springer International Publishing.

Miao, Y., Chen, C., Pan, L., Han, Q.-L., Zhang, J., and Xiang, Y. (2021). Machine learning–based cyber attacks targeting on controlled information: A survey. *ACM Computing Surveys (CSUR)*, 54(7):1–36.

Newsome, J., Karp, B., and Song, D. (2006). Paragraph: Thwarting signature learning by training maliciously. In *International Workshop on Recent Advances in Intrusion Detection*, pages 81–105. Springer.

Ngai, E. W. and Wu, Y. (2022). Machine learning in marketing: A literature review, conceptual framework, and research agenda. *Journal of Business Research*, 145:35–48.

Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. P. (2018). Sok: Security and privacy in machine learning. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 399–414.

Papernot, N., McDaniel, P., Wu, X., Jha, S., and Swami, A. (2016). Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE.

Paudice, A., Muñoz-González, L., and Lupu, E. C. (2019). Label sanitization against label flipping poisoning attacks. In Alzate, C., Monreale, A., Assem, H., Bifet, A., Buda, T. S., Caglayan, B., Drury, B., García-Martín, E., Gavaldà, R., Koprinska, I., Kramer, S., Lavesson, N., Madden, M., Molloy, I., Nicolae, M.-I., and Sinn, M., editors, *ECML PKDD 2018 Workshops*, pages 5–15, Cham. Springer International Publishing.

Pontes, J., Costa, E., Rocha, V., Neves, N., Feitosa, E., Assolin, J., and Kreutz, D. (2021). Ferramentas de extração de características para análise estática de aplicativos android. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Qi, P., Jiang, T., Wang, L., Yuan, X., and Li, Z. (2022). Detection tolerant black-box adversarial attack against automatic modulation classification with deep learning. *IEEE Transactions on Reliability*.

Quiring, E., Arp, D., and Rieck, K. (2018). Forgotten siblings: Unifying attacks on machine learning and digital watermarking. In *2018 IEEE European symposium on security and privacy (EuroS&P)*, pages 488–502. IEEE.

Rawal, A., Rawat, D., and Sadler, B. M. (2021). Recent advances in adversarial machine learning: status, challenges and perspectives. *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications III*, 11746:701–712.

Ronchetti, E. M. and Huber, P. J. (2009). *Robust statistics*. John Wiley & Sons.

Rosenberg, I., Shabtai, A., Elovici, Y., and Rokach, L. (2021). Adversarial machine learning attacks and defense methods in the cyber security domain. *ACM Comput. Surv.*, 54(5).

Rosenfeld, E., Winston, E., Ravikumar, P., and Kolter, Z. (2020). Certified robustness to label-flipping attacks via randomized smoothing. In III, H. D. and Singh, A., editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8230–8241. PMLR.

Roy, S., DeLoach, J., Li, Y., Herndon, N., Caragea, D., Ou, X., Ranganath, V. P., Li, H., and Guevara, N. (2015). Experimental study with real-world data for Android app security analysis using machine learning. In *31st ACSAC*, page 81–90. ACM.

Sarker, I. H. (2021). Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2(3):1–21.

Shackleford, D. (2022). How hackers use ai and machine learning to target enterprises. https://www.techtarget.com/searchsecurity/tip/How-hackers-use-AI-and-machine-learning-to-target-enterprises.

Shi, Z., Ding, X., Li, F., Chen, Y., and Li, C. (2021). Mitigation of poisoning attack in federated learning by using historical distance detection. In *2021 5th Cyber Security in Networking Conference (CSNet)*, pages 10–17.

Shokri, R., Stronati, M., Song, C., and Shmatikov, V. (2017). Membership inference attacks against machine learning models. In *2017 IEEE symposium on security and privacy (SP)*, pages 3–18. IEEE.

Sihag, V., Vardhan, M., and Singh, P. (2021). A survey of android application and malware hardening. *Computer Science Review*, 39:100365.

Siqueira, G., Kreutz, D., Costa, E., Mello, J., Mansilha, R., Pontes, J., Miers, C., and Feitosa, E. (2022). Avaliação de ferramentas de AutoML em datasets de detecção de malwares android. In *Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais (SBSeg 2022)*, Santa Maria–RS, Brasil.

Siqueira, G., Rodrigues, G., Feitosa, E., and Kreutz, D. (2021). QuickAutoML: Uma ferramenta para treinamento automatizado de modelos de aprendizado de máquina. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Siva Kumar, R. S., Nyström, M., Lambert, J., Marshall, A., Goertzel, M., Comissoneru, A., Swann, M., and Xia, S. (2020). Adversarial machine learning-industry perspectives. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 69–75.

Soares, T., Mello, J., Barcellos, L., Sayyed, R., Siqueira, G., Casola, K., Costa, E., Gustavo, N., Feitosa, E., and Kreutz, D. (2021a). Detecção de Malwares Android: Levantamento empírico da disponibilidade e da atualização das fontes de dados. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Soares, T., Siqueira, G., Barcellos, L., Sayyed, R., Vargas, L., Rodrigues, G., Assolin, J., Pontes, J., Feitosa, E., and Kreutz, D. (2021b). Detecção de Malwares Android: datasets e reprodutibilidade. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Song, L., Shokri, R., and Mittal, P. (2019). Membership inference attacks against adversarially robust deep learning models. In *2019 IEEE Security and Privacy Workshops (SPW)*, pages 50–56. IEEE.

Tabassi, E., Burns, K. J., Hadjimichael, M., Molina-Markham, A. D., and Sexton, J. T. (2019). A taxonomy and terminology of adversarial machine learning. https://doi.org/10.6028/NIST.IR.8269-draft. Draft NISTIR 8269.

Taheri, R., Javidan, R., Shojafar, M., Pooranian, Z., Miri, A., and Conti, M. (2020). On defending against label flipping attacks on malware detection systems. *Neural Computing and Applications*, 32(18):14781–14800.

Tam, K., Feizollah, A., Anuar, N. B., Salleh, R., and Cavallaro, L. (2017). The evolution of android malware and android analysis techniques. *ACM Computing Surveys (CSUR)*, 49(4):1–41.

Tavares, P. (2022). Popular evasion techniques in the malware landscape. https://resources.infosecinstitute.com/topic/popular-evasion-techniques-in-the-malware-landscape/.

Tran, B., Li, J., and Madry, A. (2018). Spectral signatures in backdoor attacks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R.,

editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Truex, S., Liu, L., Gursoy, M. E., Yu, L., and Wei, W. (2019). Demystifying membership inference attacks in machine learning as a service. *IEEE Transactions on Services Computing*.

Van der Walt, E., Eloff, J. H., and Grobler, J. (2018). Cyber-security: Identity deception detection on social media platforms. *Computers & Security*, 78:76–89.

Veale, M., Binns, R., and Edwards, L. (2018). Algorithms that remember: model inversion attacks and data protection law. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 376(2133):20180083.

Vilanova, L., Sayyed, R., Soares, T., Siqueira, G., Rodrigues, G., Feitosa, E., and Kreutz, D. (2021). Análise do impacto de viés nos conjuntos de dados para detecção de malwares android. In *VI Workshop Regional de Segurança da Informação e de Sistemas Computacionais (WRSeg)*, Charqueadas-RS, Brasil.

Vorobeychik, Y. and Kantarcioglu, M. (2018). Adversarial machine learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 12(3):1–169.

Wang, S., Balarezo, J. F., Kandeepan, S., Al-Hourani, A., Chavez, K. G., and Rubinstein, B. (2021). Machine learning in network anomaly detection: A survey. *IEEE Access*, 9:152379–152396.

Xue, Y., Meng, G., Liu, Y., Tan, T. H., Chen, H., Sun, J., and Zhang, J. (2017). Auditing anti-malware tools by evolving android malware and dynamic loading technique. *IEEE Transactions on Information Forensics and Security*, 12(7):1529–1544.

Yuan, X., He, P., Zhu, Q., and Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824.

Zhang, H., Cheng, N., Zhang, Y., and Li, Z. (2021a). Label flipping attacks against Naive Bayes on spam filtering systems. *Applied Intelligence*, 51(7):4503–4514.

Zhang, J., Chen, B., Cheng, X., Binh, H. T. T., and Yu, S. (2021b). PoisonGAN: Generative poisoning attacks against federated learning in edge computing systems. *IEEE Internet of Things Journal*, 8(5):3310–3322.

Zhang, J., Ge, C., Hu, F., and Chen, B. (2022a). Robustfl: Robust federated learning against poisoning attacks in industrial iot systems. *IEEE Transactions on Industrial Informatics*, 18(9):6388–6397.

Zhang, J., Zhang, C., Liu, X., Wang, Y., Diao, W., and Guo, S. (2021c). Shadowdroid: Practical black-box attack against ml-based android malware detection. In *2021 IEEE 27th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 629–636. IEEE.

Zhang, R., Xia, H., Hu, C., Zhang, C., Liu, C., and Xiao, F. (2022b). Generating adversarial examples with shadow model. *IEEE Transactions on Industrial Informatics*.

Zhang, T., Song, A., Dong, X., Shen, Y., and Ma, J. (2022c). Privacy-preserving asynchronous grouped federated learning for iot. *IEEE Internet of Things Journal*, 9(7):5511–5523.

Zhang, Z., Chen, Y., and Wagner, D. (2021d). Seat: Similarity encoder by adversarial training for detecting model extraction attack queries. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, pages 37–48.

Zhao, W., Alwidian, S., and Mahmoud, Q. H. (2022a). Adversarial training methods for deep learning: A systematic review. *Algorithms*, 15(8).

Zhao, Y., Chen, J., Zhang, J., Wu, D., Blumenstein, M., and Yu, S. (2022b). Detecting and mitigating poisoning attacks in federated learning using generative adversarial networks. *Concurrency and Computation: Practice and Experience*, 34(7):e5906.

Capítulo

2

Uma Introdução sobre Redes Adversárias Generativas (GANs) e suas Aplicações na Cibersegurança

Karina Casola Fernandes (UNIPAMPA), Angelo Gaspar Diniz Nogueira (UNIPAMPA), Anna Luiza Gomes da Silva (UNIPAMPA), Kayuã Oleques Paim (UFRGS), Diego Kreutz (UNIPAMPA), Rodrigo Mansilha (UNIPAMPA), Hendrio Luis de Souza Bragança (UFAM)

Resumo. *Neste capítulo, exploramos as Redes Generativas Adversariais (Generative Adversarial Networks – GANs), destacando sua importância no campo da Inteligência Artificial. Iniciamos com uma visão geral do aprendizado de máquina e das Redes Neurais Artificiais (RNAs), abordando os fundamentos necessários para compreender as GANs. Em seguida, explicamos a arquitetura adversarial que define o funcionamento das GANs, composta por duas redes neurais opostas: o gerador e o discriminador. Detalhamos como essa dinâmica adversarial permite que as GANs criem dados sintéticos realistas, discutindo conceitos como o equilíbrio de Nash, desafios no treinamento e as principais métricas de avaliação para mensurar sua eficácia. Na parte final do capítulo, apresentamos as aplicações práticas das GANs na cibersegurança. Essas redes têm se destacado na geração de variantes de malware para treinamento defensivo, na detecção de deepfakes, e no desenvolvimento de técnicas avançadas de criptografia e esteganografia. Discutimos também como GANs podem ser empregadas para criar dados sintéticos em cenários de segurança, fortalecendo sistemas contra ataques avançados.*

2.1. Introdução

Nas últimas décadas, a rápida evolução da tecnologia digital tem gerado um aumento exponencial na quantidade de dados e na complexidade dos sistemas computacionais, o que, por sua vez, trouxeram novos desafios no campo da cibersegurança. A crescente sofisticação dos ataques cibernéticos exige a adoção de estratégias inovadoras e eficazes de defesa. Nesse contexto, técnicas avançadas de inteligência artificial (IA), como as Redes Neurais Artificiais (RNAs) e, mais recentemente, as Redes Adversárias Generativas (GANs), têm se destacado por sua capacidade de aprender padrões complexos e gerar dados realistas. Esses modelos, que inicialmente foram desenvolvidos para criar imagens,

vídeos e dados sintéticos, hoje encontram aplicações promissoras na detecção e prevenção de ameaças cibernéticas.

Apresentamos uma visão introdutória sobre conceitos fundamentais de inteligência artificial (IA), aprendizado de máquina (do inglês, *Machine Learning* - ML) e aprendizado profundo (do inglês, *Deep Learning* - DL), fornece os conceitos principais para uma compreensão mais detalhada de Redes Adversárias Generativas (do inglês, *Generative Adversarial Networks - GANs*) e suas aplicações na cibersegurança. Começamos, na Seção 2.2, abordando as diferentes escolas de aprendizado de máquina, os tipos de treinamento e como as Redes Neurais Artificiais (RNAs) se encaixam nesse contexto. Em seguida, na Seção 2.3, exploramos o funcionamento interno das RNAs, discutindo a arquitetura de redes *feedforward*, funções de ativação, *retropropagação* e o papel do aprendizado profundo na evolução desses modelos. A partir da Seção 2.4, o foco se volta para as GANs, com explicações sobre sua arquitetura adversarial, processo de treinamento, e desafios, além de suas diversas aplicações gerais. Então, na Seção 2.5, destacamos como as GANs podem ser utilizadas na cibersegurança, oferecendo um novo enfoque para a detecção de ameaças e defesa contra ataques avançados. Por fim, na Seção 2.6, apresentamos as considerações finais.

2.2. Aprendizado de máquina

Nesta seção são apresentados os conceitos fundamentais de aprendizado de máquina, com ênfase nas principais abordagens e métodos de treinamento. Esses fundamentos estabelecem a base teórica necessária para a compreensão das Redes Neurais Artificiais (RNAs), seguida pela introdução a Redes Adversárias Generativas (GANs). Por fim, é explorada a aplicação de GANs no contexto da cibersegurança.

O ML, ou aprendizado de máquina, em português, refere-se à construção de modelos computacionais que têm a capacidade de aprender a partir de dados e realizar tarefas como diagnóstico, predição e reconhecimento de padrões. Por meio de abstrações de algoritmos inteligentes, alimentados por dados amostrais ou dados históricos, esses modelos conseguem identificar padrões nestes dados de treinamento, aprendendo a analisar e classificar estes dados [Dark 2018].

2.2.1. Escolas

Dada a complexidade e a diversidade de aplicações do aprendizado de máquina, existem diferentes escolas de pensamento que abordam o tema de maneiras variadas [Berzal 2018]. Essas abordagens refletem as diversas estratégias e filosofias subjacentes à construção e ao treinamento de modelos de aprendizado de máquina.

Para melhor entender as nuances e as metodologias envolvidas, podemos classificar essas abordagens em cinco grandes famílias de pensamento [Domingos 2015, Berzal 2018]. Cada uma dessas escolas oferece perspectivas únicas e contribui de forma distinta para o avanço do campo:

- Os **Simbólicos** focam na interpretação filosófica, lógica e aprendizagem psicológica: um processo de indução que não é mais que uma dedução reversa.
- Os **Análogos** baseiam o aprendizado na extrapolação de exemplos conhecidos, fazendo julgamentos de similaridade. Seus raciocínios por analogia tem fundamen-

tos psicológicos e às vezes acabam se traduzindo em um problema de otimização matemática, como isso acontece com máquinas de vetor de suporte.
- Os **Bayesianos** usam técnicas estatísticas de inferência probabilística, baseado no teorema de Bayes, como as Redes Bayesianas.
- Os **Evolucionistas** tiram suas conclusões da teoria da evolução de Darwin e das bases da genética, das quais se inspiram para desenvolver técnicas como os algoritmos genéticos.
- Os **Conexionistas** se inspiram no cérebro humano, tentando fazer engenharia reversa de seu funcionamento com a ajuda de físicos, e neurocientistas (embora seus modelos nem sempre acabem sendo biologicamente plausível).

2.2.2. Tipos de aprendizado

No campo de aprendizado de máquina, existem basicamente as seguintes estratégias de aprendizagem [Mahesh 2020].

- O **aprendizado supervisionado** necessita de ajuda externa para o seu treinamento, na forma de dados rotulados [Mahesh 2020]. Isso significa que o conjunto de dados usado no treinamento inclui entradas com suas respectivas saídas corretas ou "rótulos". O objetivo do algoritmo é aprender a mapear essas entradas para as saídas corretas, de modo que, quando receber novos dados no futuro, ele possa prever ou classificar essas saídas com base no que aprendeu.
- No **aprendizado não supervisionado**, os algoritmos não têm acesso a respostas ou rótulos predefinidos (*i.e.* dados rotulados). Neste caso não há respostas corretas e não há "professor". Ou seja, os algoritmos são deixados por conta própria para descobrir e apresentar estruturas interessantes nos dados [Mahesh 2020]. O objetivo do algoritmo é identificar padrões ou estruturas subjacentes nos dados por conta própria (*e.g.* agrupamentos de dados).
- No **aprendizado por reforço**, o algoritmo é treinado por meio de interações com um ambiente, onde ele toma decisões e aprende a partir das consequências dessas ações [Nduati 2020]. O objetivo do agente é maximizar sua recompensa total ao longo do tempo, ajustando suas estratégias com base no *feedback* obtido. Este tipo de aprendizado é frequentemente comparado ao processo de tentativa e erro, onde o agente experimenta diferentes ações e aprende quais delas levam aos melhores resultados. Um exemplo clássico de aprendizado por reforço é o treinamento de robôs para navegação, em que o robô aprende a se mover em direção a um objetivo evitando obstáculos, ajustando seus movimentos com base nas recompensas recebidas por cada decisão tomada. Essa abordagem se destaca em problemas onde a solução não é diretamente evidente e o aprendizado precisa ser ajustado de forma dinâmica, tornando-se cada vez mais popular em áreas como automação e jogos.

Em síntese, podemos dizer que o aprendizado supervisionado trabalha com dados rotulados, o aprendizado não supervisionado busca padrões em dados não rotulados e o aprendizado por reforço trabalha sem dados. Como alternativa aos dados, o aprendizado por reforço envolve um agente que executa ações em um ambiente e recebe um *feedback* em forma de recompensas ou penalidades dinamicamente.

2.3. Redes Neurais Artificiais

Após a introdução aos fundamentos do aprendizado de máquina na seção anterior, na qual foram discutidas diferentes abordagens e métodos de treinamento, este capítulo aprofunda-se em uma das áreas mais promissoras do campo: as Redes Neurais Artificiais (RNAs). Inspiradas na arquitetura biológica do cérebro humano, as RNAs desempenham um papel crucial em tarefas complexas, como predição, classificação e geração de dados, destacando-se por sua capacidade de modelar padrões não lineares e de alto nível em grandes volumes de dados.

Iniciamos com uma introdução aos principais conceitos relacionados às Redes Neurais Artificiais (RNAs). Em seguida, discutiremos a arquitetura *feedforward*, uma das mais simples e amplamente empregadas. Posteriormente, veremos que cada neurônio aplica uma função de ativação para determinar se deve "disparar" um sinal, sendo as funções mais comuns a ReLU (Rectified Linear Unit), a Sigmoide e a Tangente Hiperbólica. O processo de aprendizado da rede é ajustado através do *backpropagation*, um algoritmo que permite a correção dos pesos nas conexões dos neurônios com base nos erros cometidos. Esses erros são calculados utilizando uma função de perda, que mede a diferença entre a saída prevista e a saída real. À medida que as redes neurais crescem em profundidade, formando várias camadas ocultas, elas se tornam parte do campo conhecido como Aprendizado Profundo (*Deep Learning*, DL), permitindo a resolução de problemas altamente complexos com grande precisão.

2.3.1. Conceitos básicos

Proposta pela escola conexionista (inspirada pelo cérebro humano), os modelos são formados por múltiplas unidades, denominados neurônios, ou elementos de processamento, que se interconectam para formar o que chamamos de redes neurais artificiais [Berzal 2018]. Diferente de outros modelos (*e.g.* árvores de decisão) de aprendizado de máquina que expressam o conhecimento de maneira explícita por meio de regras lógicas, as RNAs representam o conhecimento de forma implícita. Isso é feito através dos pesos que definem as conexões entre os neurônios [Berzal 2018]. O objetivo essencial é determinar o conjunto de pesos em cada camada que maximize a precisão das previsões do modelo. Esse processo envolve a otimização dos parâmetros da rede de modo a minimizar a função de perda, ajustando os pesos de forma iterativa até que o modelo alcance um desempenho satisfatório. O seu procedimento da construção pode ser resumida da seguinte nos seguintes passos [Berzal 2018].

- Recompilar um conjunto de dados associado ao problema.
- Desenhar uma função de custo apropriada para o problema, também conhecida como função de perda ou *loss function*. Essa função de perda reflete o desempenho da rede no contexto do problema.
- Selecionar um modelo de rede neural e estabelecer seus hiperparâmetros (tamanho, características).
- Aplicar um algoritmo de otimização para minimizar a função de custo ajustando os parâmetros da rede, visando otimizar o desempenho da rede.

A unidade computacional básica do cérebro é um neurônio. Aproximadamente 86 bilhões de neurônios podem ser encontrados no sistema nervoso humano e estão conectados com aproximadamente $10^{14} – 10^{15}$ sinapses. A Figura 2.1 mostra um desenho de um neurônio biológico (esquerda) e um modelo matemático comum (direita) [Fumo 2017].

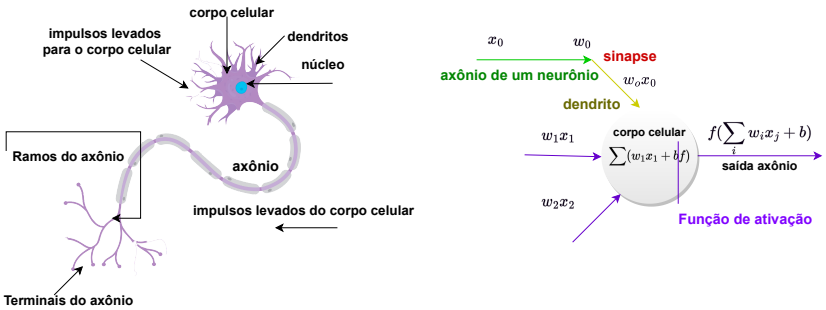

Figura 2.1. Neurônio biológico (esquerda) e artificial (direita) (fonte: [Fumo 2017]).

O neurônio também representa a unidade computacional mais básica de redes neurais, onde o limiar b_k tem o papel de aumentar ou diminuir a influência do valor da entrada líquida para a ativação do neurônio k [Fernandes 2019], conforme ilustrado na Figura 2.2. A saída do neurônio k pode ser descrita por [de Castro 2016] 1, 2:

$$y_k = f(u_k) = f(\sum_{j=1}^{m} w_{kj} + b_k) \tag{1}$$

ou

$$y_k = f(u_k) = f(\sum_{j=0}^{m} w_{kj} x_j) \tag{2}$$

Onde x_0 é um sinal de entrada de valor 1 e peso associado $w_k 0 = b_k$.

2.3.2. Arquiteturas de Redes Neurais

Embora todas as redes neurais utilizem as mesmas unidades computacionais básicas, como os neurônios são organizados pode variar entre diferentes modelos. Geralmente, essas redes seguem uma **arquitetura** bem reconhecida, também chamada de **modelo**, com variações adaptadas conforme a aplicação.

Apesar das adaptações, as arquiteturas mantêm a mesma estrutura fundamental, conforme ilustrado na Figura 2.3. A camada de entrada recebe os dados brutos e os encaminha para as camadas ocultas, que processam e transformam as informações ao identificar padrões complexos. Finalmente, a camada de saída gera o resultado, que pode ser uma previsão ou uma classificação, entre outros exemplos.

As arquiteturas podem ser classificadas em dois modelos básicos como segue.

1. ***Perceptron* de camada única**. Esta é a rede neural mais simples: consiste apenas em uma única camada de nós de saída. Denominada “camada única"porque, ao contar as camadas, não consideramos a camada de entrada. Isso ocorre porque

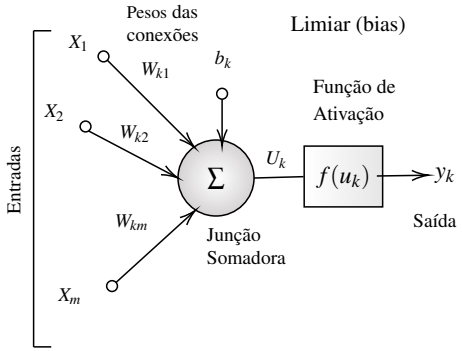

Figura 2.2. Neurônio artificial genérico (fonte: [de Castro 2016]).

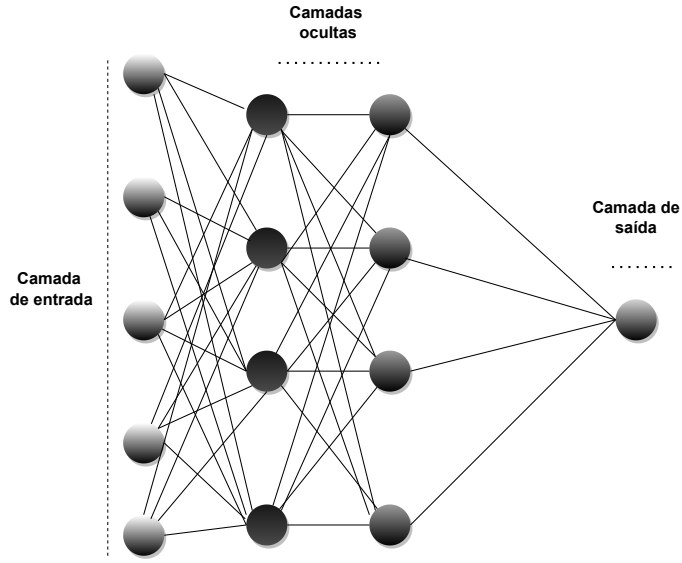

Figura 2.3. Arquitetura geral (fonte: [de Souza 2019]).

na camada de entrada não são realizados cálculos; as entradas são simplesmente transmitidas diretamente para a camada de saída por meio de uma série de pesos.

2. ***Perceptron* multicamada (MLP).** Consiste em várias camadas de unidades computacionais. Cada neurônio em uma camada tem conexões direcionadas aos neurônios da camada subsequente. MLP são mais utilizadas. e uma boa razão

é que, estas conseguem aprender representações não lineares (geralmente os dados apresentados a nós, não são linearmente separáveis), áudio ou imagens [Berzal 2018].

Uma das arquiteturas mais fundamentais (*i.e.* que serve de base para outras mais complexas) é a rede neural *feedforward* [Berzal 2018]. A *feedforward* é uma rede neural artificial na qual as conexões entre as unidades não formam ciclos. Nesta rede, a informação se move em apenas uma direção, para frente, dos nós de entrada, passando pelos nós ocultos (se houver) até chegar nos nós de saída.

2.3.3. Funções de ativação

As funções de ativação permitem que a rede aprenda e modele funções de problemas complexos e não lineares durante o seu processo de treinamento, ao definir a saída dos neurônios com base em uma entrada ou conjunto de entradas. Suponha que um circuito de chip de computador padrão seja uma rede digital de funções de ativação que pode ser "ON" (1) ou "OFF" (0), dependendo da entrada. Esse comportamento assemelha-se à operação de um *perceptron* linear em redes neurais. No entanto, é a função de ativação não linear que permite que essas redes calculem problemas não triviais usando apenas um número reduzido de nós. Em redes neurais artificiais, essa função também é chamada de função de transferência [Fumo 2017].

Cada função de ativação (ou não-linearidade) recebe um único número e realiza uma certa operação matemática fixa nele. Aqui estão algumas funções de ativações que você encontrará frequentemente, na prática, [Fumo 2017, Shah 2017]:

- **Sigmoide**. A função sigmoide transforma as entradas de qualquer magnitude em saídas dentro do intervalo $[0, 1]$, tornando-se útil para tarefas de classificação binária.
- **Tanh**. Semelhante à sigmoide, a função Tanh mapeia as entradas para saídas em um intervalo contínuo, mas variando entre $[-1, 1]$. A diferença principal é que, enquanto a sigmoide comprime os valores para $[0, 1]$, a Tanh centraliza os resultados em torno de zero, o que pode ser vantajoso em redes com muitas camadas.
- **ReLU**. A função de ativação *Rectified Linear Unit* define que todos os valores negativos são transformados em zero, mantendo apenas os valores positivos. Essa simplicidade torna a ReLU amplamente utilizada em redes neurais com muitas camadas, pois acelera o processo de treinamento e ajuda a lidar com o problema de gradientes desaparecendo (*e.g. vanishing gradients*).
- **Leaky ReLu**. A função de ativação *Rectified Linear Unit* com vazamento é uma variação da função de ativação ReLU. A principal diferença entre elas é que, enquanto a ReLU elimina completamente os valores negativos, a Leaky ReLU permite que esses valores tenham uma pequena inclinação negativa, reduzindo sua magnitude em vez de descartá-los. Essa modificação visa mitigar o problema de unidades mortas, onde neurônios deixam de aprender por receber constantemente valores negativos.

2.3.4. Processo de Treinamento

O treinamento de redes neurais envolve uma série de etapas para ajustar os pesos dos neurônios e limiares da rede para aprender padrões e estruturas a partir dos dados de

entrada. Esse processo, independentemente do tipo de aprendizado utilizado, começa geralmente com a divisão dos dados disponíveis em dois subconjuntos principais: treino (60-90% dos dados) e avaliação (10-40% dos dados) [Da Silva et al. 2017].

O conjunto de treino, como o seu nome sugere, é utilizado treinar a rede neural. Durante essa fase, a rede é alimentada com amostras desse conjunto, e os pesos são ajustados com base nos valores resultantes da **função de perda,** que é calculada a partir das saídas geradas pela rede. Os pesos da rede são ajustados utilizando algoritmos de optimização (*e.g.* ***Backpropagation***). Ou como no *Perceptron*, a sua taxa de aprendizado é modificada. Enquanto o conjunto de avaliação será utilizado para validar a capacidade de aprendizado da rede usando dados previamente não vistos. Ademais, cada iteração completa do processo de ajuste da rede com todas as amostras do conjunto de treinamento é chamada de época de treino [Da Silva et al. 2017].

2.3.4.1. Backpropagation

O algoritmo de *Backpropagation*, em português ***retropropagação***, é uma técnica fundamental para o treinamento de redes neurais. Ele realiza o ajuste fino da rede com base na taxa de erro da época (*i.e.* ciclo de treinamento) anterior. O ajuste adequado dos pesos permite reduzir as taxas de erro e tornar o modelo mais confiável, aumentando sua generalização [Johnson 2022].

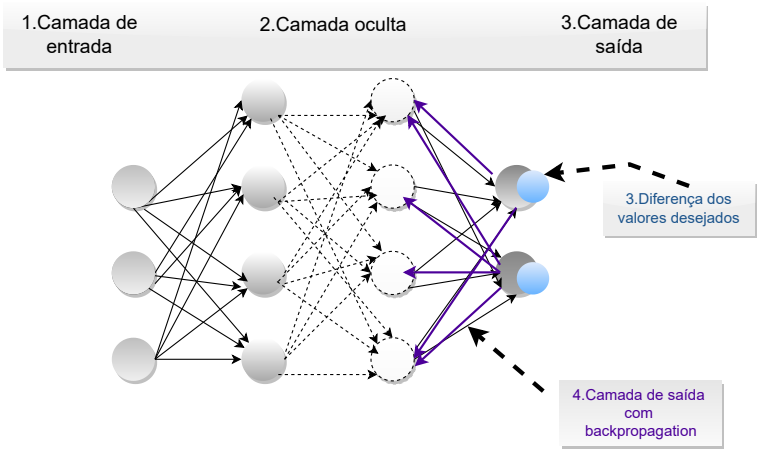

Figura 2.4. Backpropagation (fonte: [Johnson 2022]).

Em redes neurais o algoritmo de ***retropropagação*** calcula o gradiente da função de perda em relação a um único peso utilizando a regra da cadeia. Em contraste ao cálculo direto nativo, é calculada com eficiência uma camada por vez. Embora o algoritmo forneça os gradientes, ele não especifica diretamente como os aplicar para ajustar os pesos. Em vez disso, ele generaliza o cálculo do gradiente por meio da regra delta, onde os pesos são ajustados de maneira iterativa, com o objetivo de convergir para o ponto mínimo da função de erro. Esse ajuste é realizado através do gradiente descendente, um processo que visa minimizar a função de erro calculando os gradientes dos pesos e aplicando-os para atualizações na rede através de ***retropropagação***.

Considere a Figura 2.4 de exemplo de rede neural de ”retropropagação” para entender.

1. Entradas X, chegam pelo caminho pré-conectado.
2. A entrada é modelada usando pesos reais W. Os pesos são geralmente selecionados aleatoriamente.
3. Calcule a saída para cada neurônio da camada de entrada, para as camadas ocultas, para a camada de saída.
4. Calcule o erro nas saídas.
5. Erro B = Saída Real - Saída Desejada
6. Viaje de volta da camada de saída para a camada oculta para ajustar os pesos de forma que o erro diminua.

2.3.4.2. Função de perda

A função de perda retorna um valor médio para cada exemplo de treinamento e, quanto maior for esse valor, pior é o desempenho da rede naquela observação. Ou seja, a função de perda, ou de custo, pretende “levar” o resultado do treinamento em direção à convergência, atuando no processo de ajuste dos pesos da rede [Chrysostomo 2022].

A função de perda atua após o ***Backpropagation*** dos pesos. Com os valores ajustados, esses pesos são “reinseridos” na entrada da rede e continuam sendo atualizados até o final das épocas [Chrysostomo 2022].

2.3.5. Algoritmos Otimizadores

Os algoritmos otimizadores são fundamentais para o treinamento de redes neurais, pois ajustam os pesos dos neurônios de modo a minimizar a função de perda (ou erro) durante o aprendizado, permitindo que o modelo atinja maior precisão ou menor erro nas previsões. Em outras palavras, os otimizadores guiam a rede neural para encontrar a melhor configuração de parâmetros. O principal propósito desses algoritmos é justamente minimizar a função de perda, que mede o quão bem o modelo está desempenhando em relação aos dados de treinamento. Durante o treinamento, os otimizadores atualizam os pesos das conexões entre os neurônios com base nos gradientes calculados pela retropropagação, processo que permite ao modelo aprender padrões a partir dos dados de entrada.

O ciclo de otimização envolve ajustes iterativos dos pesos da rede, realizados com base nos gradientes obtidos da derivada da função de perda em relação a esses pesos. Cada passo desse processo visa mover os pesos na direção que reduz o erro, e dependendo do

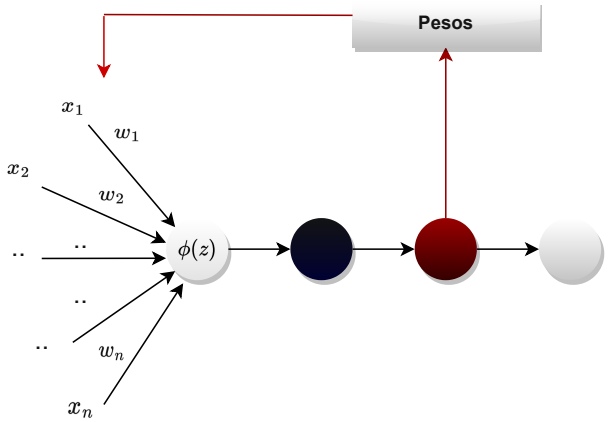

Figura 2.5. Função de perda (fonte: [Chrysostomo 2022]).

otimizador escolhido, essa atualização pode ser mais ou menos agressiva, incorporando técnicas como o uso de momentum ou adaptação da taxa de aprendizado para melhorar a eficiência do treinamento.

Existem vários algoritmos otimizadores disponíveis, cada um com características próprias para ajustar os pesos de uma rede neural. A Tabela 2.1 apresenta um resumo sobre alguns algoritmos de otimização considerados populares na literatura científica. Em seguida, discutimos cada um deles em maior nível de detalhes.

- **Stochastic Gradient Descent (SGD)**. O SGD é um dos otimizadores mais básicos e amplamente utilizados. Ele atualiza os pesos com base no gradiente da função de perda calculado para um único exemplo de treinamento (ou um pequeno lote, no caso de mini-batch SGD). Isso o torna mais rápido, mas com a desvantagem de uma alta variabilidade, já que os gradientes podem ser muito instáveis de um exemplo para outro.
- **Adaptive Moment Estimation (ADAM)**. O **Adam** é um dos otimizadores mais populares atualmente. Ele combina as vantagens do SGD com o uso de duas estimativas: a média dos gradientes (momentum) e a média dos quadrados dos gradientes. Isso permite uma adaptação dinâmica da taxa de aprendizado para cada peso da rede, o que acelera a convergência.
- **RMSprop**. O RMSprop é um otimizador que, assim como o Adam, adapta a taxa de aprendizado com base na magnitude recente dos gradientes. Ele divide o gradiente pelo quadrado da média móvel dos gradientes anteriores, o que ajuda a

Tabela 2.1. Resumo sobre algoritmos otimizadores populares

Sigla	Vantagens	Desvantagens
ADAM	Simples e eficiente em grandes bases de dados.	Oscilações ao redor do mínimo local podem tornar a convergência mais lenta.
SGD	Atualiza a taxa de aprendizado de forma adaptativa, o que o torna eficiente em problemas com gradientes ruidosos e esparsos.	Pode não se comportar bem em problemas que exigem grande precisão, pois às vezes não converge para o mínimo global de forma otimizada.
RMSprop	Excelente para problemas com gradientes que variam em magnitude ao longo do tempo.	Assim como Adam, pode ser sensível à escolha dos hiperparâmetros.
Adagrad	Funciona bem para problemas com gradientes esparsos.	Pode tornar a taxa de aprendizado muito pequena ao longo do tempo, levando a uma convergência muito lenta.
Adadelta	Resolve o problema de taxa de aprendizado decrescente.	Pode ser mais complicado de ajustar para diferentes tipos de problemas.

estabilizar a atualização dos pesos, evitando grandes oscilações.

- **Adagrad**. O Adagrad ajusta a taxa de aprendizado com base no histórico dos gradientes. Ele armazena a soma dos quadrados dos gradientes anteriores e ajusta a taxa de aprendizado para cada peso individualmente. Isso é particularmente útil para problemas onde algumas características são mais raras e requerem uma atualização mais cuidadosa.
- **Adadelta**. O Adadelta é uma extensão do Adagrad, projetada para superar o problema da taxa de aprendizado que decai muito rapidamente. Ele limita o número de gradientes acumulados em uma janela, o que mantém a taxa de aprendizado controlada.

2.3.6. *Deep Learning (DL)*

Deep Learning (DL) ou "aprendizado profundo" refere-se a uma abordagem que utiliza redes neurais compostas de muitas camadas de neurônios. Cada camada permite construir e utilizar novos recursos a partir de recursos identificados pelas camadas anteriores, sendo especialmente útil na análise dados não estruturados [Berzal 2018].

Dessa forma, as camadas não apenas identificam características que melhoram nossa capacidade de discriminação em diferentes situações, mas também elevam o nível de abstração no qual operam. As camadas são organizadas em hierarquias com vários níveis, nos quais novos recursos são descoberto automaticamente a partir de características do nível anterior [Berzal 2018].

Considere o exemplo da Figura 2.6. A camada mais à esquerda é chamada de entrada *(Input Layer)*, a camada mais à direita é a de saída *(Output Layer)*. As camadas intermediárias são camadas ocultas são valores calculados usados pela rede para realizar a transformação dos dados de entrada em padrões mais complexos que ajudam na tomada de decisões ou na previsão do resultado. Quanto mais camadas ocultas uma rede tiver entre as suas camadas de entrada e saída, mais profunda ela será. Em geral, qualquer Rede neural artificial com duas ou mais camadas ocultas é classificada como uma rede neural profunda [Wauke 2022].

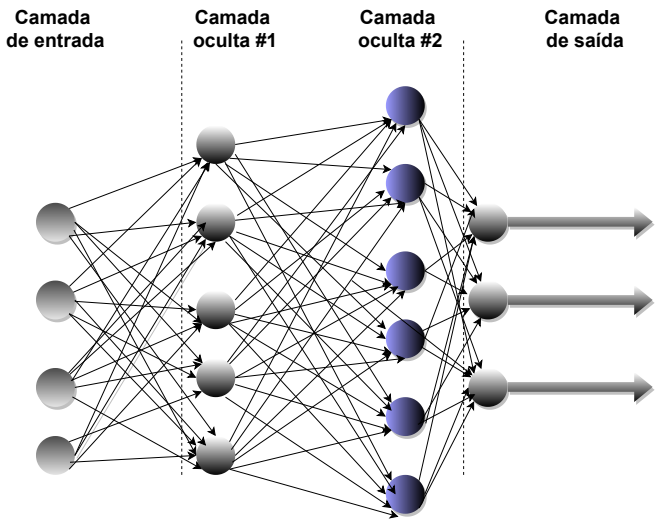

Figura 2.6. Deep Learning (fonte: [Wauke 2022]).

Em relação às técnicas de aprendizado profundo, estas baseiam-se na ideia básica de que, se formos capazes de aprender com sucesso vários níveis de representação, conseguiremos generalizar corretamente [Berzal 2018].

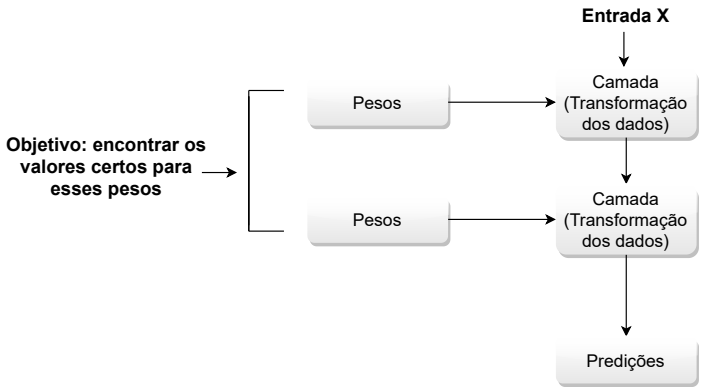

Figura 2.7. Parametrização por pesos (fonte: [Chollet 2021]).

A especificação do que uma camada deverá fazer com seus dados de entrada é

armazenada nos pesos da camada, representada por uma quantidade significativa de números. Em termos técnicos, a transformação é implementada por uma camada parametrizada por seus pesos, como mostra a Figura 2.7. Nesse contexto, a aprendizagem significa encontrar um conjunto de valores para os pesos de todas as camadas em uma rede, de modo que a rede mapeie corretamente as entradas de exemplo para seus destinos associados. O problema é que uma rede neural profunda pode conter dezenas de milhões de parâmetros. Encontrar os valores corretos para todos eles pode ser uma tarefa desafiadora, especialmente considerando que a modificação de um único parâmetro pode afetar o comportamento de todos os outros [Chollet 2021].

Na Figura 2.8 apresentamos um fluxograma contendo a atribuição de pesos e de perda em redes neurais profundas. Inicialmente, os pesos da rede recebem valores aleatórios, portanto, a rede implementa apenas uma série de transformações aleatórias. Naturalmente, sua saída está longe do ideal, e a pontuação da perda é, portanto, muito alta. No entanto, a medida que a rede processa os exemplos dos processos de rede, os pesos são ajustados gradualmente na direção correta, o que, consequentemente, leva à diminuição da pontuação da perda. Esse processo constitui o *loop* de treinamento, que, repetido um número suficiente de vezes (normalmente dezenas de iterações em milhares de exemplos), converge em valores de peso que minimizam a função de perda. Uma rede com uma perda mínima é aquela em que as saídas são tão próximas quanto possíveis dos alvos, sendo uma rede bem treinada. Embora seja um mecanismo simples, quando escalado, pode apresentar resultados impressionantes [Chollet 2021].

O aspecto fundamental na aprendizagem profunda é o uso da pontuação como um sinal de *feedback* para a realização do ajuste um mínimo no valor dos pesos, em uma direção que reduzirá a pontuação da perda para o exemplo atual (Figura 2.8). Esse ajuste é o trabalho do otimizador, que implementa o que é chamado de algoritmo *Backpropagation*: o algoritmo central em DL [Chollet 2021].

Apesar de o aprendizado profundo ser um subcampo antigo dentro do aprendizado de máquina, ele só começou a ganhar destaque a partir do início de 2010 [Chollet 2021]. Em poucos anos alcançou, nada menos que uma revolução, com resultados notáveis em problemas sensoriais, como ver e ouvir - problemas envolvendo habilidades que parecem naturais e intuitivas para os humanos, mas que, por muito tempo, foram consideradas inatingíveis para as máquinas. Obtendo avanços em áreas historicamente difíceis do aprendizado de máquina [Chollet 2021].

- Classificação de imagem ao nível próximo humano.
- Reconhecimento de fala.
- Transcrição de caligrafia.
- Tradução automática aprimorada.
- Conversão aprimorada de conversão de texto em fala.
- Assistentes digitais como o Google Now e o Amazon Alexa.
- Condução autônoma.
- Segmentação de anúncios aprimorada, usada pelo Google, Baidu e Bing.
- Resultados de pesquisa aprimorados na web.
- Capacidade de responder perguntas em linguagem natural.
- Jogo Go.

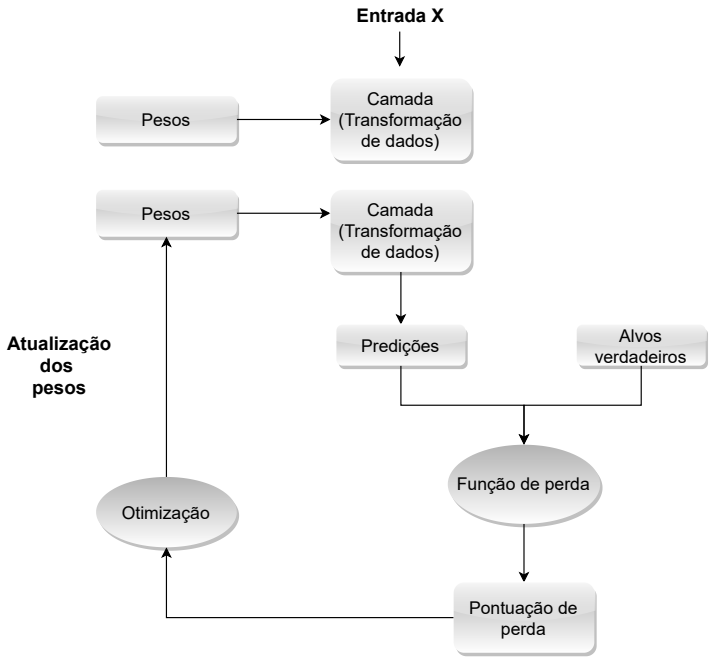

Figura 2.8. Função de Perda *Deep Learning (DL)* (fonte: [Chollet 2021]).

2.4. Generative Adversarial Networks GANs

Após a exploração das Redes Neurais Artificiais (RNAs) e seus conceitos fundamentais, como camadas, funções de ativação e o processo de treinamento, este capítulo se volta para uma abordagem mais avançada: as Redes Adversárias Generativas (GANs). As GANs representam um subtipo inovador de modelos generativos que utilizam uma dinâmica adversarial entre dois componentes principais – o gerador e o discriminador – para criar dados sintéticos de alta qualidade. Essas redes têm se destacado em diversas aplicações devido à sua capacidade de gerar amostras realistas a partir de dados brutos, oferecendo avanços significativos em áreas como geração de imagens, síntese de texto e produção de áudio.

Nesta seção abordaremos os principais aspectos das GANs, começando com uma introdução aos modelos generativos e à arquitetura adversarial que define o funcionamento das GANs. Examinaremos o processo de treinamento detalhado, distinguindo entre o treinamento do discriminador e do gerador, e discutiremos o conceito de equilíbrio de Nash, fundamental para a convergência do treinamento adversarial. Também abordaremos os desafios associados ao treinamento das GANs, as métricas de avaliação para medir seu desempenho e, por fim, as diversas aplicações em que essas redes têm sido empregadas.

2.4.1. Conceitos básicos

Os modelos generativos são uma abordagem inovadora que, como o nome sugere, conseguem "gerar" novas amostras a partir de dados existentes, praticamente indiscerníveis dos dados originais. Isso contrasta a abordagem discriminativa, na qual o modelo aprende a discriminar os grupos e calcular as probabilidades de que uma nova observação tenha um determinado rótulo, sendo, portanto, probabilísticos e não determinísticos [Foster 2019]. A Figura 2.9 apresenta uma representação de cada modelo.

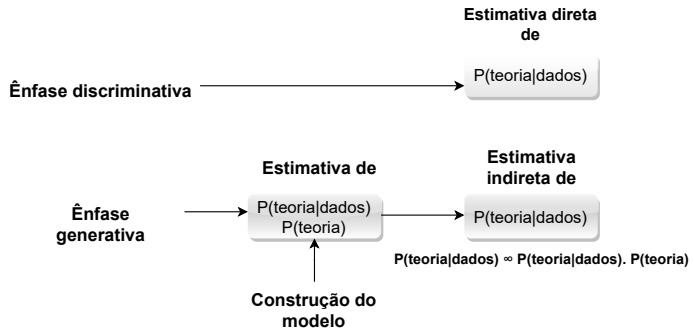

Figura 2.9. Modelos Discriminativos (acima) e Generativos (abaixo).

Em mais detalhes, na abordagem discriminativa, a probabilidade de uma determinada variável de saída y, ou classe, dado o padrão x, enquanto na generativa se aproxima a probabilidade de observar um determinado padrão x, após especificar uma classe y. Ou seja, o reconhecimento de padrões ocorre através da estimativa da probabilidade posterior $P(teoria|dados)$ [Foster 2019].

Nos modelos generativos (ou modelos que fazem uso dessa abordagem), por sua vez, o reconhecimento de padrões é realizado através da estimativa de probabilidade $P(dados|teoria)$ e a probabilidade anterior $P(teoria)$. Esses modelos, ao conseguirem estimar essas probabilidades, podem gerar novas amostras que são indiscriminadas a partir de dados previamente existentes [Foster 2019, Goodfellow et al. 2014].

2.4.2. Arquitetura Adversarial

A arquitetura das GANs tem em seu núcleo duas redes neurais menores que, como o próprio nome sugere, são opostas, e que aprendem a usar uma estrutura cooperativa de jogo de soma zero [Goodfellow et al. 2014]. Em outras palavras, esse processo pode ser explicado usando a analogia de um falsificador de arte e um especialista em arte [Creswell et al. 2018]. Nesse contexto, a rede generativa atua como o falsificador, empenhado em criar imagens que sejam o mais próximas das obras originais. Por outro lado, a rede discriminadora desempenha o papel de especialista, cujo objetivo é distinguir entre ima-

gens genuínas e cópias. Com base nos resultados fornecidos pelo especialista, a rede é ajustada; em termos mais técnicos, o discriminador determina a função de perda da rede.

A Figura 2.10 ilustra a arquitetura, destacando o cálculo da interação da perda entre os diferentes componentes. o início do processo, o gerador recebe como entrada um vetor de ruído multidimensional, composto por valores aleatórios, que é usado para sintetizar amostras artificiais e dados reais que desejamos que o gerador aprenda. O discriminador recebe duas entradas distintas: amostras reais e amostras artificiais geradas pelo gerador, determinando a probabilidade binária de que as amostras falsas sejam reais. Finalmente, para cada uma das conjunturas do discriminador, seu desempenho deve ser determinado. Usamos os resultados para ajustar tanto o gerador quanto o discriminador utilizando o método de retropropagação do erro (backpropagation) ou Descida do Gradiente Estocástico (SGD) [Torres 2020].

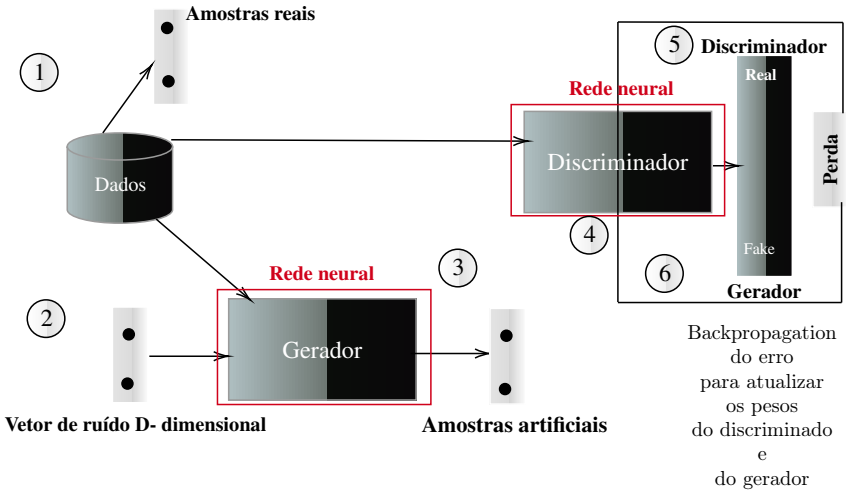

Figura 2.10. Arquitetura geral com duas redes neurais (adaptada de [Torres 2020]).

2.4.3. Processo de treinamento

GAN é definida como um jogo minimax para minimizar a perda máxima esperada. Esse teorema [Neumann 1928] estabelece que em jogos de soma zero, cada jogador sabe de antemão a estratégia de seu oponente e suas consequências, e com isso, o método de decisão se torna efetivo. Em outros termos, o gerador e o discriminador jogam um jogo de soma zero com a seguinte função objetivo [Goodfellow et al. 2014, Amin et al. 2022, Hui 2020]:

$$\min_G \max_D V(D,G) = E_{x\sim pdados(x)}[\log D(x)] + E_{z\sim pz} \log(1 - D(G(\tilde{x})))$$

- $\min_G$ Gerador empurra para baixo.
- $\max_D$ O discriminador empurra para cima.
- $\log D(x)$, capacidade do discriminador de reconhecer os dados como sendo reais.
- $\log(1 - D(G(\tilde{x})))$ Capacidade do discriminador de reconhecer a amostra do gerador como sendo falsa.

Formalmente, tanto o gerador quanto o discriminador são representados por funções diferenciáveis [Goodfellow et al. 2014]: $G(z, \theta_G)$ e $D(x, \theta_D)$. Cada um com sua função de perda, com θ_G e θ_D sendo os parâmetros dos pesos do treino.

O objetivo do treinamento é maximizar a probabilidade do discriminador prever quais são os dados reais e quais são os dados falsos. Esse processo pode ser descrito pelos seguintes passos e pelo Algoritmo 2.1 [Torres 2020].

1. Pegue um mini-lote aleatório de "dados reais" do conjunto de dados de treinamento.
2. Gere um novo mini-lote de vetores de ruído aleatório, passado como entrada para o gerador, e sintetiza um mini-lote de "dados falsos".
3. O discriminador classifica tanto "dados reais" quanto "dados falsos".
4. Os erros de classificação são calculados, e a perda total é propagada ao discriminador para atualizar os pesos e vieses deste a fim de minimizar os erros de classificação.

Algoritmo 2.1: GAN com Stochastic Gradient Descent.

for *número de iterações de treinamento* **do**
  for *s etapa* **do**
    Dada uma amostra de n ruídos $x^{(1)}, ...\xi^{(n)}$ de uma distribuição $\rho_g \xi$
    Dada uma amostra de n exemplos $x^1 ... x^n$ de uma distribuição $\rho_{data}(x)$
    Atualize D com Stochastic Gradient Descent:

$$\nabla_{\theta_d} \frac{1}{n} \sum_{i=1}^{n} [\log D(x^{(i)}) + \log(1 - D(G(\xi^i)))]$$

  end
  Dada uma amostra de n ruídos $\xi^{(1)}, ...\xi^{(n)}$ de uma distribuição $\rho_g \xi$
  Atualize G com Stochastic Gradient Descent:

$$\nabla_{\theta_g} \frac{1}{n} \sum_{i=1}^{n} [\log(1 - D(G(\xi^i)))]$$

end

2.4.3.1. Treinamento do discriminador

A Figura 2.11 representa o processo de treinamento na fase discriminadora para GANs. Nesse processo, geralmente, os parâmetros do discriminador são atualizados primeiro

e, em seguida, são utilizados para ajustar o gerador. Podemos notar que uma amostra aleatória de ruído é passada pelo bloco gerador. Como o gerador inicialmente não tem conhecimento dos valores reais ou da saída a ser produzida, ele gera resultados aleatórios, muitas vezes irrelevantes, que são chamados de recursos do gerador.

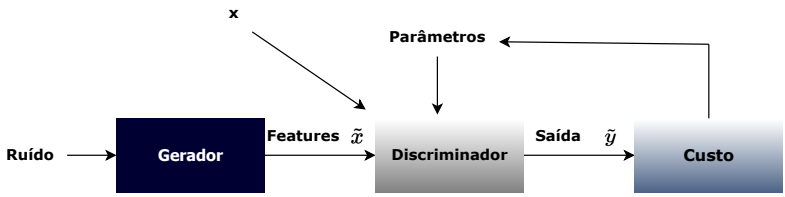

Figura 2.11. Fase Discriminadora (fonte: [Bharath 2021]).

Na próxima etapa, ambos os recursos do gerador $(\tilde{X})$ e as características reais (X) são passados pelo discriminador. Inicialmente, os recursos produzidos pelo gerador têm um desempenho terrível em comparação com os recursos reais. O discriminador também apresenta um desempenho ruim durante os estágios iniciais do procedimento de treinamento. Portanto, é essencial atualizar os parâmetros do discriminador.

A saída $(\tilde{Y})$ resultados do bloco discriminador após as características do gerador serem comparadas com as características reais. O discriminador, como discutido anteriormente, atua de forma semelhante a um classificador. Há outra etapa de comparação, onde mais uma vez recebemos a saída do discriminador e a comparamos com a saída real para calcular a função de custo geral. Finalmente, os parâmetros do discriminador são atualizados. A função que o gerador busca minimizar sobre o acerto do discriminador é a seguinte:

$$\log(1 - D(G(\tilde{x})))$$

O discriminador classifica entre $\tilde{x}$, a amostra falsa e x, os dados amostrais e a distribuição atual dos dados $\rho_{(dados(x))}$.

2.4.3.2. Treinamento do gerador

Na etapa de treinamento do gerador, o objetivo é fazer com que o discriminador classifique incorretamente um dado falso como sendo real. Em outras palavras, busca-se maximizar iterativamente a probabilidade de que o discriminador cometa esse erro. O processo de treinamento do gerador realiza as mesmas etapas que o discriminador, mas com foco em como o discriminador classifica dados falsos. Com isso, calcule a perda de classificação e propague a perda total sobre os dados para atualizar os pesos e vieses do gerador [Torres 2020].

O diagrama na Figura 2.12 é uma representação do treinamento na fase geradora. Podemos notar que nesta fase, algum ruído é fornecido ao gerador, mas uma atualização

equivalente é feita nos parâmetros dos geradores. Os parâmetros são atualizados, considerando o *feedback* recebido do discriminador. Aqui, podemos notar também que apenas os recursos gerados $(\tilde{X})$ são usados para avaliação, enquanto os recursos reais (X) não são considerados [Bharath 2021].

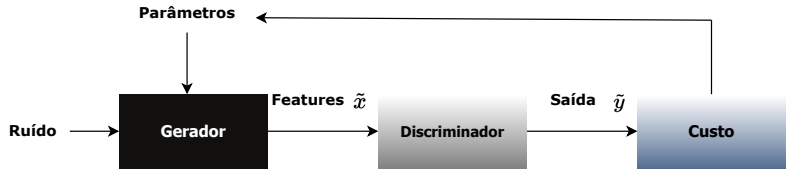

Figura 2.12. Fase Geradora (fonte: [Bharath 2021]).

O gerador desenvolve suas capacidades começando com o treinamento em ruído para gerar imagens falsas, as quais são avaliadas pelo discriminador. A função de custo é avaliada após o cálculo, e os parâmetros do gerador são atualizados. Nesta fase de treinamento, apenas os parâmetros do gerador são atualizados, enquanto os parâmetros do discriminador são desprezados.

Os parâmetros do gerador são atualizados somente quando o discriminador classifica as imagens como reais (ou seja, quando a probabilidade é igual a 1). O treinamento é realizado de forma alternada. Com o gerador realizando ajustes seus parâmetros com base no *feedback* do discriminador, aprendendo a partir das classificações corretas ou incorretas.

Ambos os modelos discriminador e gerador devem evoluir juntos durante o treinamento. É crucial manter os modelos em níveis de habilidade semelhantes desde o início. Não é desejável um discriminador superior que aprenda muito rápido, porque este se torna muito melhor em distinguir entre real e falso, enquanto o gerador nunca pode aprender rápido o suficiente para produzir imagens falsas suficientemente convincentes. Da mesma forma, se o gerador for treinado em um ritmo mais rápido, o discriminador falhará em suas tarefas, permitindo que qualquer imagem aleatória seja gerada. Portanto, é essencial garantir que tanto o gerador quanto os discriminadores sejam treinados em um ritmo consistente.

A solução Minimax é equivalente ao que é chamado de Equilíbrio de Nash [Farnia and Ozdaglar 2020], que representa o ponto em que nenhum jogador pode melhorar sua situação alterando suas próprias ações. Nas GANs, se o Equilíbrio de Nash for alcançado, é o momento oportuno para encerrar o treinamento, pois, nesse estado o gerador produz amostras tão realistas que o discriminador não consegue diferenciá-las dos dados reais.

Em outras palavras, o equilíbrio de Nash é x=y=0. Este é o único estado em que a ação do seu oponente não importa. O único estado que as ações de qualquer oponente não mudarão o resultado do jogo [Hui 2020]. O equilíbrio de Nash é alcançado quando duas condições são atendidas [Goodfellow et al. 2014]:

- O gerador produz amostras que são indistinguíveis dos dados no conjunto de treinamento $p_g = P_{dados}$.
- O discriminador pode, no máximo, adivinhar aleatoriamente quando uma amostra é real ou falsa.

2.4.3.3. Desafios e soluções do treinamento

A seguir são listadas algumas das principais dificuldades que podem ocorrer durante a fase de treinamento de uma GAN [Creswell et al. 2018], sendo, portanto, objetos de investigação na atualidade.

- **Sem equilíbrio de Nash**: a forma normal é esperado que inicialmente as funções de perda oscilem, mas à medida que o treinamento é realizado, o gerador e o discriminador se equilibram. Quando isso não acontecer, as amostras produzidas pelo gerador serão de qualidade inferior.
- **Colapso do modelo**: o gerador fica "viciado" em um pequeno conjunto de amostras que consegue enganar o discriminador, produzindo amostras semelhantes mesmo que os dados de entrada tenham muitas características. Nesses casos, o gradiente da função de perda é fixado em um valor próximo a zero.
- **Perda sem grandes informações**: não é uma tarefa trivial analisar a qualidade das amostras produzidas pelo gerador em relação ao discriminador, mesmo assumindo que quanto menor a perda do gerador, maior a qualidade das amostras produzidas. O gerador pode produzir amostras de maior qualidade mesmo quando a função de perda aumenta.

No geral, várias estratégias foram propostas para superar os desafios acima e, assim, melhorar o desempenho de GANs. Esses métodos são resumidos e ilustrados na Figura 2.13 [Razavi-Far et al. 2022].

Por exemplo, o recorte da razão de probabilidade no lado do gerador e o reponderamento da amostra no lado do discriminador foi proposto em [Wu et al. 2020b]. Além disso, [Salimans et al. 2016] discutiu sobre correspondência de recursos, discriminação de mini-lote, média histórica e normalização de lote variacional como algumas soluções para treinamento de GAN. A escolha de funções de otimização adequadas, conforme abordado por [Mullick et al. 2019], e o balanceamento da taxa de aprendizado do gerador e do discriminador pela regra de atualização de duas escalas de tempo [Heusel et al. 2017] pode melhorar a estabilidade do treinamento. Outro método amplamente utilizado é a suavização de rótulo unilateral [Szegedy et al. 2016] para evitar um discriminador muito confiável e tornar o treinamento mais robusto [Razavi-Far et al. 2022].

2.4.4. Métricas de avaliação

Assim como a escolha do método de treinamento adequado é crucial para alcançar um bom desempenho em uma determinada aplicação, é igualmente necessário escolher a métrica de avaliação correta para extrair conclusões corretas sobre o desempenho da GAN. [Theis et al. 2015].

Métricas variadas podem resultar em *trade-offs* distintos, e certos critérios de avaliação favorecem modelos específicos. Portanto, é fundamental que o treinamento e a

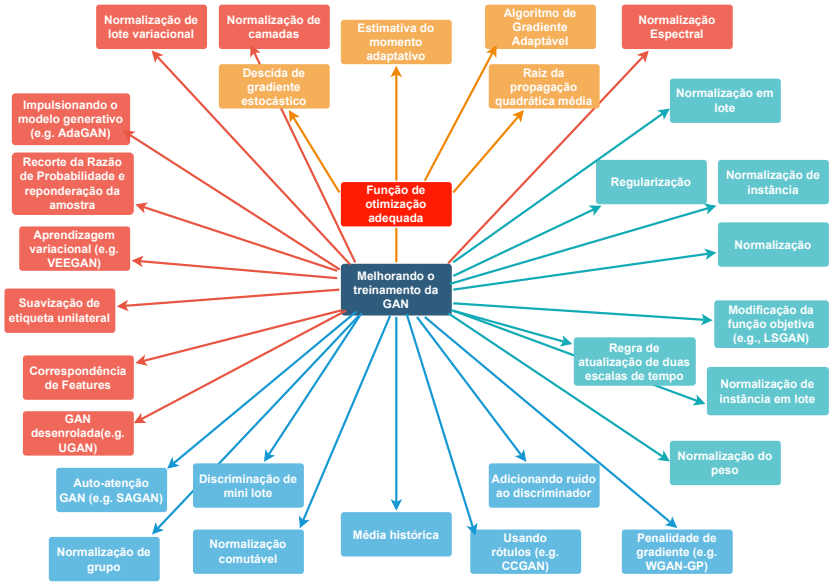

Figura 2.13. Soluções para treinamento (fonte: [Razavi-Far et al. 2022]).

avaliação correspondam à aplicação-alvo. Além disso, devemos ser cautelosos para não considerar um bom desempenho em um aplicativo como evidência de um bom desempenho em outro aplicativo [Theis et al. 2015].

A Tabela 2.2 lista diversas métricas usadas na literatura para avaliação de GANs. Essa lista não é completa e serve para demonstrar que a escolha de métricas não é trivial.

2.4.5. Aplicações gerais

Originalmente, as GANs foram desenvolvidas com o propósito de criar imagens realistas, onde o gerador visava produzir amostras que fossem visualmente indistinguíveis das imagens reais, desafiando o discriminador a diferenciá-las. Com o tempo, as GANs evoluíram de forma notável, ampliando suas aplicações para uma variedade de domínios.

- **Geração de imagem**. O trabalho de [Sage et al. 2018] explora as GANs para a criação de logos. O uso de GANs no processo que se conhece como *Data Argumentation*, para incrementar o tamanho de um *dataset* é apresentado pelo trabalho [Antoniou et al. 2017]. Criação de personagem caricatos [Jin et al. 2017], geração de rostos humanos [Hukkelås et al. 2019]. Além disso, também foram realizados trabalhos que se utilizaram de GANs para completar áreas incompletas em imagens, ao invés de criar imagens completas [Chandak et al. 2019, Li et al. 2017].
- **Síntese de imagem a partir de texto**. A síntese de imagem a partir de texto consiste na junção de GANs com o processamento de linguagem natural para

Tabela 2.2. Métricas usadas para avaliação de GAN.

Métrica	**Referência**
Average Log- likelihood	[Ian et al. 2014]
	[Theis et al. 2015]
Coverage Metric	[Tolstikhin et al. 2017]
Inception Score (IS)	[Salimans et al. 2016]
Modified Inception Score (m-IS)	[Gurumurthy et al. 2017]
Mode Score (MS)	[Che et al. 2016]
AM Score	[Zhou et al. 2017]
Fréchet Inception Distance (FID)	[Heusel et al. 2017]
Maximum Mean Discrepancy (MMD)	[Gretton et al. 2012]
The Wasserstein Critic	[Arjovsky et al. 2017a]
Birthday Paradox Test	[Arora and Zhang 2017]
Classifier Two Sample Test (C2ST)	[Steinebach 2006]
Classification Performance	[Radford et al. 2015]
	[Isola et al. 2017]
Boundary Distortion	[Santurkar et al. 2018]
NDB	[Richardson and Weiss 2018]
Image Retrieval Performance	[Wang et al. 2016]
Generative Adversarial Metric (GAM)	[Im et al. 2016]
Tournament Win Rate and Skill Rating	[Olsson et al. 2018]
NRDS	[Zhang et al. 2018b]
Adversarial Accuracy Divergence	[Yang et al. 2017]
Geometry Score	[Khrulkov and Oseledets 2018]
Reconstruction Error	[Xiang and Li 2017]
Image Quality Measures	[Wang et al. 2004]
	[Snell et al. 2017]
	[Juefei-Xu et al. 2017]
Low-level Image Statistics	[Zeng et al. 2017]
	[Karras et al. 2017]
Precision Recall and F1 score	[Lucic et al. 2018]

criar uma figura através de sua descrição. Neste campo, [Reed et al. 2016] foram pioneiros ao propor uma solução inovadora com resultados promissores para o desafio de transformar descrições textuais em imagens. A abordagem proposta

consiste em dividir o problema em dois subproblemas principais: (i) desenvolver uma forma visual e discriminativa de representar as descrições das imagens e; (ii) utilizar essa representação para gerar imagens realistas.
- **Conversão imagem-imagem**. No trabalho de [Zhu et al. 2017] essas conversões são demonstradas através dos exemplos de imagens de zebras a imagens de cavalos, fotografias a pinturas estilo Monet e fotos de inverno para fotos de verão.
- **Geração de imagens de alta resolução**. Na área da medicina diversas soluções [Chen et al. 2018, Mahapatra et al. 2019, Zheng et al. 2020], aplicam GANs para permitir maior detecção de patologias. Essas abordagens permitem melhorar a precisão no diagnóstico ao aumentar a resolução de imagens de baixa qualidade, facilitando a detecção de anomalias.

2.4.6. Arquiteturas derivadas

A expansão das aplicações das GANs foi possível graças à adaptação de suas topologias e ao desenvolvimento de variantes que se ajustam melhor às características específicas de cada área de aplicação. A Figura 2.14 apresenta um roteiro evolutivo das arquiteturas GANS. Em seguida, apresentamos um resumo sobre as arquiteturas derivadas de GANs mais proeminentes.

- **CGAN**. *Conditional Generative Adversarial Networks* (CGANs) são variantes condicionais das GANs, que utilizam informações condicionais para o gerador e discriminador, essa informação consiste em informações auxiliares dos dados (*e.g.* rótulos de classes) [Mirza and Osindero 2014]. Em contraste com as GANs tradicionais, o discriminador nas cGANs busca não apenas distinguir entre os pares das distribuições geradas e a distribuição alvo, mas também a informação condicional associada a suas amostras [Miyato and Koyama 2018].
- **DCGAN**. O *Deep Convolutional Generative Adversarial Network* (DCGAN) é uma extensão da GAN, construído usando as camadas de convolução convolucional e transposta. A principal diferença no treinamento de uma DCGAN em comparação com uma GAN tradicional está na arquitetura das redes usadas. O DCGAN foi descrito pela primeira vez no artigo [Radford et al. 2015, Tomar 2021]. A arquitetura do DCGAN consiste em: (i) um gerador, composto por camadas convolucionais transpostas, que cria imagens a partir de um vetor de ruído; e (ii) um discriminador, composto por camadas convolucionais, que analisa tanto as imagens geradas quanto as reais. Além disso, as DCGANs adotam técnicas específicas, como inicialização de pesos com distribuição normal e uso de funções de ativação como ReLU e LeakyReLU, que ajudam a estabilizar o treinamento, o que costuma ser um desafio em GANs tradicionais. Já em uma GAN tradicional, tanto o gerador quanto o discriminador geralmente são construídos com camadas densas (totalmente conectadas).
- **WGAN**. A *Wasserstein Generative Adversarial Network*, ou *Wasserstein* GAN (WGAN), é uma extensão da rede adversária generativa que melhora a estabilidade ao treinar o modelo e fornece uma função de perda que se correlaciona com a qualidade das imagens geradas [Brownlee 2021]. O desenvolvimento da WGAN tem uma motivação matemática densa, embora, na prática, exija apenas ajustes pontuais na rede adversária generativa convolucional profunda padrão estabelecida, ou DCGAN [Brownlee 2021].

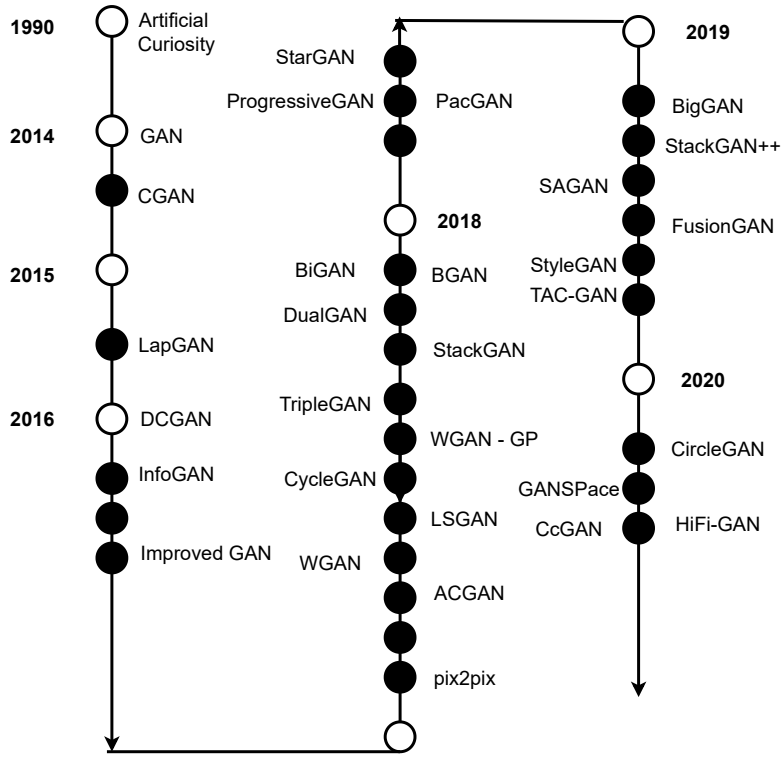

Figura 2.14. Topologias GAN (fonte: [Razavi-Far et al. 2022]).

- **InfoGAN**. InfoGAN é uma variante das GANs que utiliza função de perda de informação mútua, visando controlar mais propriedades e aprender características semânticas de dados de forma não supervisada.
- **ACGAN**. Em contraste com o modelo tradicional de GAN, o *Auxiliary Classfier Generative Adversarial Network* (ACGAN) incorpora aprendizado supervisionado, semelhante à DCGAN, permitindo que o discriminador desempenhe um papel duplo. Além de determinar se as amostras são reais ou geradas, o discriminador também classifica as amostras em suas respectivas rótulos. No entanto, o ACGAN se diferencia da DCGAN ao treinar o discriminador com uma combinação de dados reais e sintetizados.
- **StackGAN**. StackGANs utilizam uma hierarquia de CGANs para gerar imagens de alta resolução em duas etapas: primeiro cria uma imagem base de baixa resolução, depois refina para uma versão de alta resolução com maior nível de detalhe.
- **CycleGAN**. O CycleGAN é utilizado para converter imagens entre domínios sem a necessidade de pares correspondentes, preservando as características principais da imagem original. Seu objetivo é aprender o mapeamento de um domínio para

outro na ausência de imagens pareadas, permitindo a tradução de estilos e características entre diferentes conjuntos de imagens.

- **SSGAN**. *Self-Surpervised Generative Adversarial Networks* (SSGANs) utilizam uma abundância de dados não rotulados combinados com uma pequena amostra de dados rotulados para treinar o modelo de predição. Dessa forma, o discriminador atua como um classificador semi-supervisionado de múltiplas classes, aproveitando a vasta quantidade de dados não rotulados para melhorar a precisão e robustez da classificação.
- **Progressive GAN**. *Progressive* GANs são variantes de GANs com uma estrutura modificada que visam: (i) reduzir o tempo de treinamento; (ii) aumentar a variação das imagens geradas; e (iii) melhorar a estabilidade da rede para o treinamento de imagens de alta qualidade. Essas melhorias são alcançadas através da expansão progressiva da profundidade do gerador e do discriminador, com a adição gradual de camadas ao longo do processo de treinamento.
- **StyleGAN**. StyleGANs utilizam o mesmo principio de *Progressive* GANs para o treinamento e geração de imagens, mas oferecem um controle mais refinado sobre aspectos específicos das imagens geradas. Isso é alcançado por meio de: (i) um vetor latente transformado em um espaço intermediário, que permite a aplicação de diferentes estilos em várias camadas das imagens; e (ii) um mapeamento detalhado de camadas de rede e controle de ruído, que aprimora a capacidade de ajustar e refinar as características visuais das imagens geradas.
- **BIGAN**. *Bidirectional Generative Adversarial Networks* (BIGANs) utilizam um *framework* não supervisionado para o aprendizado de características, treinando tanto o gerador quanto o discriminador para mapear vetores latentes para amostras reais e vice-versa. Isso promove um aprendizado bidirecional, permitindo que o modelo não apenas gere amostras a partir de vetores latentes, mas também recupere representações latentes a partir de amostras reais, melhorando a qualidade das representações e a robustez do treinamento.
- **BGAN**. *Bayesian Generative Adversarial Networks* (BGANs) são variantes GANs que introduzem princípios bayesianos no treinamento da GAN, com o gradiente estocástico de Monte Carlo para modelar a incerteza dos pesos do gerador e discriminador, para capturar a variabilidade dos dados.

2.5. GANs na cibersegurança

As GANs têm se consolidado como uma ferramenta poderosa no campo da cibersegurança, oferecendo soluções inovadoras para desafios complexos. Sua capacidade de gerar dados sintéticos e simular cenários adversos as coloca a na vanguarda das tecnologias de defesa cibernética. Entre as aplicações mais relevantes, destaca-se o uso de GANs para a detecção e ocultação de *malware*. As GANs podem criar variantes de *malware* que enganam sistemas de detecção tradicionais, o que, por outro lado, também permite que sistemas defensivos sejam treinados com essas variantes, aumentando sua eficácia. Além disso, as GANs são empregadas na quebra de senhas, onde modelos treinados podem simular um invasor, dessa forma fortalecendo os sistemas de segurança.

Outras áreas de aplicação crucial são as de confidencialidade de informações como as de criptografia neural e esteganografia, onde as GANs são utilizadas para desenvolver sistemas de criptografia robustos e ocultação de dados, respectivamente. Além disso, elas

desempenham um papel importante na detecção de *deepfakes*, auxiliando na identificação de conteúdos falsificados que poderiam ser usados para desinformação ou fraude.

A capacidade dessas redes de simular comportamentos maliciosos e melhorar continuamente os sistemas de defesa promovem o desenvolvimento constante do campo da cibersegurança. No entanto, com esse potencial também vêm desafios, como a necessidade de equilibrar o poder de geração das GANs com a capacidade de detecção e resposta dos sistemas defensivos, devido a sua capacidade de gerar mutações de *malware*. Em suma, as GANs estão auxiliando na definição de novos paradigmas na cibersegurança, ao mesmo tempo que desafiam as convenções e impulsionam o desenvolvimento de sistemas de proteção mais inteligentes e adaptáveis.

A Tabela 2.3 relaciona diversos trabalhos que fazem uso de GANs na área de cibersegurança. Conforme pode ser observado existem diversos domínios de aplicações para GANs nesta área. No entanto, entre os principais, destacam-se:

- **Quebra de senha**. GANs podem ser usadas para gerar senhas prováveis, simulando como um invasor poderia tentar adivinhar senhas. Este processo é utilizado para testar a segurança de sistemas de autenticação e para desenvolver senhas mais seguras. Nesse sentido, diversos trabalhos buscam utilizar GANS para produzir senhas. O trabalho de [Hitaj et al. 2019], por exemplo, utiliza o WGAN aprimorado para aprender com a distribuição de dados dos bilhões de senhas vazadas e usa esse resultado para gerar adivinhações de senha de maior qualidade. Se o gerador derrotar com sucesso ambos os discriminadores, ele produzirá uma senha que nenhum discriminador poderia detectar [Soleymanzadeh and Kashef 2022].
- **Criptografia Neural**. A Criptografia Neural é um campo emergente de pesquisa que combina redes neurais e técnicas de criptografia. As GANs desempenham um papel vital neste campo [Soleymanzadeh and Kashef 2022]. Sendo utilizadas por sua capacidade como rede neural de criptografar e descriptografar dados de forma segura. Conforme exemplificado por [Wu et al. 2020a], que demonstraram em seu trabalho o conceito de criptografia biométrica, na qual eles criptografavam características faciais usando a Criptografia de Redes Adversariais Gerativas de Wasserstein. Além disso, GANs também podem ser utilizadas para simular e optimizar de criptografia ao simular atacantes, conforme demonstrado em [Abadi et al. 2016]
- **Esteganografia e Ocultação de Dados**. A esteganografia é uma técnica que utiliza algoritmos para ocultar dados em canais de comunicação, como imagens, visando garantir a segurança da informação por meio da sua obscuridade [Volkhonskiy et al. 2020]. Esse método baseia-se na ideia de esconder dados de forma que, mesmo se forem interceptados, eles não sejam facilmente detectados ou decifrados por terceiros. Ademais, nos últimos anos, devido à capacidade de GANs de gerar imagens realistas e complexas, as GANs têm sido empregadas em várias soluções que visam ocultar informações de maneira eficaz. Conforme exemplificado nas soluções propostas em [Zhang et al. 2018a, Zhang et al. 2019, Hayes and Danezis 2017], as GANs e suas variantes podem ser usadas para criar imagens esteganográficas que atingem um desempenho comparável às técnicas de esteganografia de estado da arte.
- **Detecção e ocultação de *malware***. As GANs conseguem gerar novos dados vir-

tuais por meio de dados existentes, criando um número ilimitado de variantes de código malicioso que parecem legítimos [Soleymanzadeh and Kashef 2022]. Com isso, o treinamento do classificador consegue detectar mais famílias de *malware* do que em outras abordagens discriminativas [Dutta et al. 2020, Soleymanzadeh and Kashef 2022, Madry et al. 2017]. Sua natureza adversarial concorrente é mais difícil de enganar do que outros modelos de DL, limitando problemas como *overfitting*, sobreposição de classes e diminuindo falsos positivos [Soleymanzadeh and Kashef 2022, Dutta et al. 2020].

- **Geração de Dados para Treinamento de Modelos**. As GANs têm se mostrado ferramentas poderosas na geração de dados sintéticos, que podem ser utilizados para treinar modelos de cibersegurança em diversos contextos, além do uso em análises de *malware*. Exemplos dessa aplicação incluem, mas não se limitam a, sistemas de detecção de intrusos [Bourou et al. 2021], geração de dados de ataque para simular cenários de cibersegurança [Agrawal et al. 2024], e a criação de ataques em redes de comunicação [Chalé and Bastian 2022]. Essas capacidades das GANs permitem que os modelos de cibersegurança sejam expostos a uma variedade mais ampla de cenários e padrões de ataque, melhorando sua robustez e eficácia na detecção e mitigação de ameaças reais.
- **Detecção de Deepfake**. *Deepfake* processo de substituição do rosto de uma pessoa pelo de outra, utilizando GANs [Yadav and Salmani 2019]. Embora as GANs sejam amplamente conhecidas por seu papel na criação desses *deepfakes*, elas também são utilizadas na detecção dessas falsificações. Ao treinar um discriminador, que faz parte da arquitetura das GANs, para distinguir entre conteúdo gerado artificialmente e conteúdo real, é possível aprimorar significativamente a detecção de *deepfakes* em vídeos e imagem, conforme exemplificado em [Kumar et al. 2023, Ciftci et al. 2020]

Para mitigar esses problemas, [Arjovsky et al. 2017b] propõe o uso da distância de Wasserstein (WGAN), que proporciona maior estabilidade ao treinamento. A distância de Wasserstein produz gradientes mais consistentes e evita a saturação, permitindo um aprendizado mais contínuo e reduzindo o risco de colapso de modo. Além disso, essa abordagem fornece uma curva de aprendizado mais clara, facilitando a depuração e o ajuste de hiperparâmetros, tornando o processo de otimização mais eficiente. Porém, o treinamento de GANs ainda é considerado um desafio em aberto [Zhang et al. 2024]

2.6. Resumo do Capítulo

Neste capítulo, iniciamos com uma visão geral do aprendizado de máquina, abordando suas diferentes abordagens, como o aprendizado supervisionado, não supervisionado e por reforço. Essas escolas definem as bases para o desenvolvimento de modelos de IA, moldando o modo como as redes neurais e outros algoritmos aprendem com os dados. A seguir, exploramos as Redes Neurais Artificiais (RNAs), focando em seus conceitos fundamentais, como as arquiteturas mais comuns, funções de ativação e o processo de treinamento.

Em seguida, o introduzimos as Redes Generativas Adversariais (GANs), um tipo específico de RNA que trouxe avanços significativos no campo da inteligência artificial. As GANs se destacam por sua capacidade de gerar dados sintéticos realistas, através do

Tabela 2.3. GANs na segurança cibernética (Adaptado de [Dutta et al. 2020].

Proposta	Tipo de GAN	Referência
Quebra de cifra de bancos de texto	CycleGAN	[Gomez et al. 2018]
	IWGAN	[Hitaj et al. 2019]
Quebra de senhas	IWGAN	[Nam et al. 2020]
Criptografia Neural	WGAN	[Wu et al. 2020a]
	GAN	[Abadi and Andersen 2016]
Análise de Segurança	cGAN	[Chhetri et al. 2019]
Imagem segura - Esteganografia	GAN	[Hayes and Danezis 2017]
	GAN	[Zhang et al. 2019]
	WGAN	[Shi et al. 2017]
	DCGAN	[Volkhonskiy et al. 2020]
	GAN	[Tang et al. 2017]
	AC-GAN	[Zhang et al. 2018a]
	GAN/DCGAN	[Liu et al. 2018]
Geração, ataque, detecção de malware	GAN	[Hu and Tan 2017]
	GAN	[Kawai et al. 2019]
	GAN	[Rigaki and Garcia 2018]
	WGAN	[Lin et al. 2022]
	WGAN	[Yan et al. 2019]
	AC-GAN	[Singh et al. 2019]
	LSGAN, WGAN-GP, GAN	[Corley et al. 2019]
	DAN	[Millar et al. 2020]
	cGAN	[Mazaed Alotaibi 2022]
	cGAN	[Nazari et al. 2021]
	BiGAN	[Donahue et al. 2016]
	Pix2Pix e CycleGAN	[Sern et al. 2020]
Deepfake	GAN	[Kumar et al. 2023]
	GAN	[Ciftci et al. 2020]
Geração de Dados para Treinamento de Modelos	VanillaGAN, WGAN, WGAN-GP, CopulaGAN, CTGAN e TableGAN	[Bourou et al. 2021]
	DCGAN, CGAN, LAPGAN, PGGAN, RenderGAN, StackGAN, InfoGAN e Table-GAN	[Agrawal et al. 2024]
	CTGAN	[Chalé and Bastian 2022]

confronto entre duas redes: o gerador e o discriminador. Além de apresentar os conceitos

básicos das GANs, discutimos as principais variantes que surgiram a partir do modelo original e suas aplicações em diferentes domínios, como imagens, vídeos e, mais recentemente, cibersegurança.

Por fim, foi evidenciada a importância das GANs no contexto atual de cibersegurança, onde elas podem ser usadas para a criação de novos métodos de detecção e prevenção de ameaças. A evolução das GANs e suas diversas variantes demonstram como esses modelos podem ser adaptados para atender às demandas de diferentes áreas. O conteúdo apresentado oferece uma base sólida para o entendimento e aplicação de GANs, além de abrir caminho para futuras pesquisas e avanços no uso dessas redes em cenários complexos e desafiadores.

Neste capítulo, oferecemos uma introdução acessível e clara aos conceitos fundamentais de aprendizado de máquina, redes neurais artificiais, GANs e suas aplicações na cibersegurança, para estabelecer uma base sólida para estudos mais aprofundados. Embora os tópicos abordados sejam introdutórios, eles são essenciais para o entendimento de tecnologias mais complexas que estão moldando o futuro da inteligência artificial, em geral, e a cibersegurança em particular. À medida que a área continua a evoluir rapidamente, há um enorme potencial para inovação e aplicação dessas ferramentas em diversos domínios da cibersegurança. O aprofundamento no estudo dessas técnicas certamente abrirá novas portas para avanços significativos, tanto em pesquisa quanto em soluções práticas.

2.7. Exercícios

(Q_1) **Qual é o princípio básico das Redes Generativas Adversariais (GANs)?**

a) Um modelo generativo aprende a reconstruir dados reais.
b) Um único modelo é treinado para prever a probabilidade de uma amostra ser real.
c) Um modelo generativo é usado exclusivamente para compressão de dados.
d) Dois modelos competem: um gerador avalia a autenticidade das amostras e um um discriminador gera mais amostras.
e) Dois modelos competem: um gerador cria amostras e um discriminador avalia sua autenticidade.

(Q_2) **Como o treinamento de uma GAN é formulado em termos de teoria dos jogos?**

a) Como um jogo cooperativo
b) Como um jogo de soma zero
c) Como um jogo de múltiplos jogadores
d) Como um jogo em árvore de decisão

(Q_3) **Qual é o papel do modelo gerador (G) em uma GAN?**

a) Classificar as amostras de dados
b) Capturar a distribuição dos dados e maximizar a probabilidade de enganar o discriminador
c) Avaliar a probabilidade de uma amostra ser verdadeira
d) Otimizar a função de perda do discriminador

(Q_4) **O que o modelo discriminador (D) tenta fazer em uma GAN?**

a) Estimar a probabilidade de uma amostra vir do modelo gerador
b) Estimar a probabilidade de uma amostra vir dos dados de treinamento
c) Otimizar o gerador para melhorar a precisão
d) Capturar a distribuição dos dados com base no ruído

(Q_5) **Qual das seguintes opções é uma aplicação das GANs?**

a) Otimização de redes neurais
b) Geração de imagens sintéticas para criação de avatares e animações
c) Desenvolvimento de algoritmos de clustering
d) Modelagem de redes sociais

(Q_6) **Qual é uma vantagem do uso de GANs para a geração de dados sintéticos?**

a) As GANs reduzem a complexidade do modelo discriminador
b) É possível definir a quantidade de amostras a serem geradas para cada classe
c) Elas eliminam a necessidade de pré-processamento de dados
d) Elas só geram dados textuais e tabulares

(Q_7) **Por que a geração de dados sintéticos pode ser vantajosa em relação à coleta de dados reais?**

a) Coletar dados sintéticos é mais preciso do que dados reais
b) GANs não exigem treinamento, reduzindo custos
c) O custo para gerar novas amostras é geralmente inferior ao custo de coleta de dados
d) GANs eliminam a necessidade de modelos de validação

(Q_8) **No contexto das GANs, o que representa o termo 'Z'?**

a) Uma amostra de dados de treinamento
b) Uma entrada ruidosa para o gerador
c) Um parâmetro de otimização
d) A função de perda

(Q_9) **O que representa 'X' em uma GAN?**

a) Uma entrada ruidosa para o gerador
b) A função de custo do discriminador
c) Uma amostra de treinamento
d) O vetor de pesos do modelo

(Q_{10}) **O problema de uma GAN pode ser formulado como um jogo de dois jogadores minimax, onde o objetivo é:**

a) Maximizar a função de ativação do discriminador
b) Minimizar a precisão do gerador
c) O gerador maximizar a probabilidade do discriminador cometer um erro
d) O discriminador maximizar a probabilidade do gerador aprender padrões específicos

Gabarito

(Q_1) Resposta: e
(Q_2) Resposta: b
(Q_3) Resposta: b
(Q_4) Resposta: b
(Q_5) Resposta: b
(Q_6) Resposta: b
(Q_7) Resposta: c
(Q_8) Resposta: b
(Q_9) Resposta: c
(Q_{10}) Resposta: c

Referências

Abadi, M. and Andersen, D. G. (2016). Learning to protect communications with adversarial neural cryptography. *arXiv preprint arXiv:1610.06918*.

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 308–318.

Agrawal, G., Kaur, A., and Myneni, S. (2024). A review of generative models in generating synthetic attack data for cybersecurity. *Electronics*, 13(2):322.

Amin, M., Shah, B., Sharif, A., Ali, T., Kim, K.-I., and Anwar, S. (2022). Android malware detection through generative adversarial networks. *Transactions on Emerging Telecommunications Technologies*, 33(2):e3675.

Antoniou, A., Storkey, A., and Edwards, H. (2017). Data augmentation generative adversarial networks. *arXiv preprint arXiv:1711.04340*.

Arjovsky, M., Chintala, S., and Bottou, L. (2017a). Wasserstein generative adversarial networks. In *International conference on machine learning*, pages 214–223. PMLR.

Arjovsky, M., Chintala, S., and Bottou, L. (2017b). Wasserstein generative adversarial networks. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR.

Arora, S. and Zhang, Y. (2017). Do gans actually learn the distribution? an empirical study. *arXiv preprint arXiv:1706.08224*.

Berzal, F. (2018). *Redes neuronales & deep learning: Volumen I*. Independently published.

Bharath, K. (2021). Complete guide to generative adversarial networks (gans).

Bourou, S., El Saer, A., Velivassaki, T.-H., Voulkidis, A., and Zahariadis, T. (2021). A review of tabular data synthesis using gans on an ids dataset. *Information*, 12(09):375.

Brownlee, J. (2021). How to develop a wasserstein generative adversarial network (wgan) from scratch.

Chalé, M. and Bastian, N. D. (2022). Generating realistic cyber data for training and evaluating machine learning classifiers for network intrusion detection systems. *Expert Systems with Applications*, 207:117936.

Chandak, V., Saxena, P., Pattanaik, M., and Kaushal, G. (2019). Semantic image completion and enhancement using deep learning. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, pages 1–6. IEEE.

Che, T., Li, Y., Jacob, A. P., Bengio, Y., and Li, W. (2016). Mode regularized generative adversarial networks. *arXiv preprint arXiv:1612.02136*.

Chen, Y., Shi, F., Christodoulou, A. G., Xie, Y., Zhou, Z., and Li, D. (2018). Efficient and accurate mri super-resolution using a generative adversarial network and 3d multi-level densely connected network. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 91–99. Springer.

Chhetri, S. R., Lopez, A. B., Wan, J., and Al Faruque, M. A. (2019). Gan-sec: Generative adversarial network modeling for the security analysis of cyber-physical production systems. In *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 770–775. IEEE.

Chollet, F. (2021). *Deep learning with Python*. Simon and Schuster.

Chrysostomo, N. (2022). Função de perda na rede neural: Blog gft brasil.

Ciftci, U. A., Demir, I., and Yin, L. (2020). How do the hearts of deep fakes beat? deep fake source detection via interpreting residuals with biological signals. In *2020 IEEE international joint conference on biometrics (IJCB)*, pages 1–10. IEEE.

Corley, I., Lwowski, J., and Hoffman, J. (2019). Domaingan: generating adversarial examples to attack domain generation algorithm classifiers. *arXiv preprint arXiv:1911.06285*.

Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., and Bharath, A. A. (2018). Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65.

Da Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., and dos Reis Alves, S. F. (2017). *Artificial neural networks*, volume 39. Springer.

Dark, S. (2018). *Aprendizaje Profundo Una Introducción a los Fundamentos del Aprendizaje Profundo Utilizando Python (Deep Learning Fundamentals)*. Independently Published.

de Castro, Nunes Leandro. Ferrari, G. D. (2016). *Introdução à Mineração de Dados: conceitos básicos, algoritmos e aplicações*. Saraiva, São Paulo.

de Souza, C. A. (2019). Abordagem para detecção e prevenção de intrusão em computação de nevoeiro e iot.

Domingos, P. (2015). *The master algorithm: How the quest for the ultimate learning machine will remake our world*. Basic Books.

Donahue, J., Krähenbühl, P., and Darrell, T. (2016). Adversarial feature learning. *arXiv preprint arXiv:1605.09782*.

Dutta, I. K., Ghosh, B., Carlson, A., Totaro, M., and Bayoumi, M. (2020). Generative adversarial networks in security: a survey. In *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pages 0399–0405. IEEE.

Farnia, F. and Ozdaglar, A. (2020). Do gans always have nash equilibria? In *International Conference on Machine Learning*, pages 3029–3039. PMLR.

Fernandes, K. C. (2019). Estudo da evasão de alunos de graduação utilizando educational data mining.

Foster, D. (2019). *Generative deep learning: teaching machines to paint, write, compose, and play*. O'Reilly Media.

Fumo, D. (2017). A gentle introduction to neural networks series - part 1. https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc.

Gomez, A. N., Huang, S., Zhang, I., Li, B. M., Osama, M., and Kaiser, L. (2018). Unsupervised cipher cracking using discrete gans. *arXiv preprint arXiv:1801.04883*.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *The Journal of Machine Learning Research*, 13(1):723–773.

Gurumurthy, S., Kiran Sarvadevabhatla, R., and Venkatesh Babu, R. (2017). Deligan: Generative adversarial networks for diverse and limited data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 166–174.

Hayes, J. and Danezis, G. (2017). Generating steganographic images via adversarial training. *Advances in neural information processing systems*, 30.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hitaj, B., Gasti, P., Ateniese, G., and Perez-Cruz, F. (2019). Passgan: A deep learning approach for password guessing. In *International conference on applied cryptography and network security*, pages 217–237. Springer.

Hu, W. and Tan, Y. (2017). Generating adversarial malware examples for black-box attacks based on gan. *arXiv preprint arXiv:1702.05983*.

Hui, J. (2020). Gan- ways to improve gan performance.

Hukkelås, H., Mester, R., and Lindseth, F. (2019). Deepprivacy: A generative adversarial network for face anonymization. In *International symposium on visual computing*, pages 565–578. Springer.

Ian, G., Pouget-Abadie, J., Mirza, M., Xu, B., and Warde-Farley, D. (2014). Generative adversarial nets." in advances in neural information processing systems.

Im, D. J., Kim, C. D., Jiang, H., and Memisevic, R. (2016). Generating images with recurrent adversarial networks. *arXiv preprint arXiv:1602.05110*.

Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2017). Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134.

Jin, Y., Zhang, J., Li, M., Tian, Y., Zhu, H., and Fang, Z. (2017). Towards the automatic anime characters creation with generative adversarial networks. *arXiv preprint arXiv:1708.05509*.

Johnson, D. (2022). Back propagation in neural network: Machine learning algorithm.

Juefei-Xu, F., Boddeti, V. N., and Savvides, M. (2017). Gang of gans: Generative adversarial networks with maximum margin ranking. *arXiv preprint arXiv:1704.04865*.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2017). Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Kawai, M., Ota, K., and Dong, M. (2019). Improved malgan: Avoiding malware detector by leaning cleanware features. In *2019 international conference on artificial intelligence in information and communication (ICAIIC)*, pages 040–045. IEEE.

Khrulkov, V. and Oseledets, I. (2018). Geometry score: A method for comparing generative adversarial networks. In *International conference on machine learning*, pages 2621–2629. PMLR.

Kumar, M., Sharma, H. K., et al. (2023). A gan-based model of deepfake detection in social media. *Procedia Computer Science*, 218:2153–2162.

Li, Y., Liu, S., Yang, J., and Yang, M.-H. (2017). Generative face completion. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3911–3919.

Lin, Z., Shi, Y., and Xue, Z. (2022). Idsgan: Generative adversarial networks for attack generation against intrusion detection. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 79–91. Springer.

Liu, J., Ke, Y., Lei, Y., Li, J., Wang, Y., Han, Y., Zhang, M., and Yang, X. (2018). The reincarnation of grille cipher: A generative approach. *arXiv preprint arXiv:1804.06514*.

Lucic, M., Kurach, K., Michalski, M., Gelly, S., and Bousquet, O. (2018). Are gans created equal? a large-scale study. *Advances in neural information processing systems*, 31.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.

Mahapatra, D., Bozorgtabar, B., and Garnavi, R. (2019). Image super-resolution using progressive generative adversarial networks for medical image analysis. *Computerized Medical Imaging and Graphics*, 71:30–39.

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

Mazaed Alotaibi, F. (2022). A multifaceted deep generative adversarial networks model for mobile malware detection. *Applied Sciences*, 12(19):9403.

Millar, S., McLaughlin, N., Martinez del Rincon, J., Miller, P., and Zhao, Z. (2020). Dandroid: a multi-view discriminative adversarial network for obfuscated android malware detection. In *Proceedings of the tenth ACM conference on data and application security and privacy*, pages 353–364.

Mirza, M. and Osindero, S. (2014). Conditional generative adversarial nets. *arXiv preprint arXiv 1411 1784*.

Miyato, T. and Koyama, M. (2018). cgans with projection discriminator. *arXiv preprint arXiv:1802.05637*.

Mullick, S. S., Datta, S., and Das, S. (2019). Generative adversarial minority oversampling. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1695–1704.

Nam, S., Jeon, S., Kim, H., and Moon, J. (2020). Recurrent gans password cracker for iot password security enhancement. *Sensors*, 20(11):3106.

Nazari, E., Branco, P., and Jourdan, G.-V. (2021). Using cgan to deal with class imbalance and small sample size in cybersecurity problems. In *2021 18th International Conference on Privacy, Security and Trust (PST)*, pages 1–10. IEEE.

Nduati, J. (2020). Introduction to neural networks. https://www.section.io/engineering-education/introduction-to-neural-networks/.

Neumann, J. V. (1928). Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320.

Olsson, C., Bhupatiraju, S., Brown, T., Odena, A., and Goodfellow, I. (2018). Skill rating for generative models. *arXiv preprint arXiv:1808.04888*.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Razavi-Far, R., Ruiz-Garcia, A., Palade, V., and Schmidhuber, J. (2022). *Generative Adversarial Learning: Architectures and Applications*. Springer.

Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B., and Lee, H. (2016). Generative adversarial text to image synthesis. In *International conference on machine learning*, pages 1060–1069. PMLR.

Richardson, E. and Weiss, Y. (2018). On gans and gmms. *Advances in Neural Information Processing Systems*, 31.

Rigaki, M. and Garcia, S. (2018). Bringing a gan to a knife-fight: Adapting malware communication to avoid detection. In *2018 IEEE Security and Privacy Workshops (SPW)*, pages 70–75. IEEE.

Sage, A., Agustsson, E., Timofte, R., and Van Gool, L. (2018). Logo synthesis and manipulation with clustered generative adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5879–5888.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. *Advances in neural information processing systems*, 29.

Santurkar, S., Schmidt, L., and Madry, A. (2018). A classification-based study of covariate shift in gan distributions. In *International Conference on Machine Learning*, pages 4480–4489. PMLR.

Sern, L. J., David, Y. G. P., and Hao, C. J. (2020). Phishgan: Data augmentation and identification of homoglyph attacks. In *2020 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*, pages 1–6. IEEE.

Shah, D. (2017). Activation functions. https://medium.com/@devalshah1619/activation-functions-in-neural-networks-58115cda9c96.

Shi, H., Dong, J., Wang, W., Qian, Y., and Zhang, X. (2017). Ssgan: secure steganography based on generative adversarial networks. In *Pacific Rim Conference on Multimedia*, pages 534–544. Springer.

Singh, A., Dutta, D., and Saha, A. (2019). Migan: malware image synthesis using gans. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 10033–10034.

Snell, J., Ridgeway, K., Liao, R., Roads, B. D., Mozer, M. C., and Zemel, R. S. (2017). Learning to generate images with perceptual similarity metrics. In *2017 IEEE International Conference on Image Processing (ICIP)*, pages 4277–4281. IEEE.

Soleymanzadeh, R. and Kashef, R. (2022). The future roadmap for cyber-attack detection. In *2022 6th International Conference on Cryptography, Security and Privacy (CSP)*, pages 66–70. IEEE.

Steinebach, J. (2006). El lehmann, jp romano: Testing statistical hypotheses.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Tang, W., Tan, S., Li, B., and Huang, J. (2017). Automatic steganographic distortion learning using a generative adversarial network. *IEEE Signal Processing Letters*, 24(10):1547–1551.

Theis, L., Oord, A. v. d., and Bethge, M. (2015). A note on the evaluation of generative models. *arXiv preprint arXiv:1511.01844*.

Tolstikhin, I. O., Gelly, S., Bousquet, O., Simon-Gabriel, C.-J., and Schölkopf, B. (2017). Adagan: Boosting generative models. *Advances in neural information processing systems*, 30.

Tomar, N. (2021). Dcgan: implementing deep convolutional generative adversarial network in tensorflow idiot...

Torres, J. (2020). *Python deep learning: introducción práctica con Keras y TensorFlow 2*. Marcombo.

Volkhonskiy, D., Nazarov, I., and Burnaev, E. (2020). Steganographic generative adversarial networks. In *Twelfth international conference on machine vision (ICMV 2019)*, volume 11433, pages 991–1005. SPIE.

Wang, Y., Zhang, L., and van de Weijer, J. (2016). Ensembles of generative adversarial networks, corr abs/1612.00991. *arXiv preprint arXiv:1612.00991*.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

Wauke, J. (2022). Aprendizado profundo (deep learning) vs aprendizado de máquina (machine learning) - qual é a diferença? https://jobu.com.br/2020/10/24/aprendizado-profundo-deep-learning-vs-aprendizado-de-maquina-machine-learning-qual-e-a-diferenca/.

Wu, C., Ju, B., Wu, Y., Xiong, N. N., and Zhang, S. (2020a). Wgan-e: A generative adversarial networks for facial feature security. *Electronics*, 9(3):486.

Wu, Y., Zhou, P., Wilson, A. G., Xing, E., and Hu, Z. (2020b). Improving gan training with probability ratio clipping and sample reweighting. *Advances in Neural Information Processing Systems*, 33:5729–5740.

Xiang, S. and Li, H. (2017). On the effects of batch and weight normalization in generative adversarial networks. *arXiv preprint arXiv:1704.03971*.

Yadav, D. and Salmani, S. (2019). Deepfake: A survey on facial forgery technique using generative adversarial network. In *2019 International conference on intelligent computing and control systems (ICCS)*, pages 852–857. IEEE.

Yan, Q., Wang, M., Huang, W., Luo, X., and Yu, F. R. (2019). Automatically synthesizing dos attack traces using generative adversarial networks. *International Journal of Machine Learning and Cybernetics*, 10(12):3387–3396.

Yang, J., Kannan, A., Batra, D., and Parikh, D. (2017). Lr-gan: Layered recursive generative adversarial networks for image generation. *arXiv preprint arXiv:1703.01560*.

Zeng, Y., Lu, H., and Borji, A. (2017). Statistics of deep generated images. *arXiv preprint arXiv:1708.02688*.

Zhang, K., Yang, X., Xu, L., Thé, J., Tan, Z., and Yu, H. (2024). Enhancing coal-gangue object detection using gan-based data augmentation strategy with dual attention mechanism. *Energy*, 287:129654.

Zhang, R., Dong, S., and Liu, J. (2019). Invisible steganography via generative adversarial networks. *Multimedia tools and applications*, 78(7):8559–8575.

Zhang, Z., Fu, G., Liu, J., and Fu, W. (2018a). Generative information hiding method based on adversarial networks. In *International Conference on Computer Engineering and Networks*, pages 261–270. Springer.

Zhang, Z., Song, Y., and Qi, H. (2018b). Decoupled learning for conditional adversarial networks. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 700–708. IEEE.

Zheng, T., Oda, H., Moriya, T., Sugino, T., Nakamura, S., Oda, M., Mori, M., Takabatake, H., Natori, H., and Mori, K. (2020). Multi-modality super-resolution loss for gan-based super-resolution of clinical ct images using micro ct image database. In *Medical Imaging 2020: Image Processing*, volume 11313, pages 7–13. SPIE.

Zhou, Z., Cai, H., Rong, S., Song, Y., Ren, K., Zhang, W., Yu, Y., and Wang, J. (2017). Activation maximization generative adversarial nets. *arXiv preprint arXiv:1703.02000*.

Zhu, J.-Y., Park, T., Isola, P., and Efros, A. A. (2017). Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232.

Capítulo 3

Introdução a Blockchain: Visão Geral e Conceitos Básicos

Roben Castagna Lunardi (IFRS), Regio Antonio Michelin (UNSW), Avelino Francisco Zorzo (PUCRS), Ewerton Andrade (UNIR e SIDIA), Diego Kreutz (UNIPAMPA)

Resumo. *Com o surgimento e popularização da criptomoeda Bitcoin, a utilização da tecnologia de Blockchain vem se destacando como solução para, por exemplo, garantir integridade de dados, resiliência e não-repúdio. Diversas soluções similares à plataforma Bitcoin foram criadas, como Ethereum, HyperLedger e Ripple. É importante ressaltar também que diversas pesquisas vem sendo desenvolvidas tanto para o aprimoramento da tecnologia Blockchain, como para adaptações para uso em outros domínios e aplicações. Este capítulo tem por objetivo apresentar os conceitos de uma Blockchain, através dos conceitos presentes nas Blockchains do Bitcoin, Ethereum, Hyperledger e Ripple.*

3.1. Conceitos Básicos

O conceito de Blockchain foi inicialmente introduzido para manter registro de transações de forma descentralizada e confiável em uma rede P2P. A primeira proposta foi implementada para a realização de transações da criptomoeda denominada Bitcoin [Nakamoto 2008]. Atualmente, a tecnologia Blockchain passou a ser utilizada na solução de uma diversidade significativa de problemas, como controle de transações [Min et al. 2016], votos eletrônicos [Moura and Gomes 2017], controle de direitos autorais [Kishigami et al. 2015], armazenamento e execução de trechos de código [Ethereum 2022], Internet das Coisas [Lunardi et al. 2018], Cidades Inteligentes [Michelin et al. 2018], registro de dados de saúde [Branco et al. 2020], processos de gestão de recursos humanos [Chillakuri and Attili 2021], cadeia de suprimento de alimentos agrícolas [Saurabh and Dey 2021] e serviço de DNS [Chang and Svetinovic 2016]. Muitas destas aplicações foram propostas para utilizarem características de resiliência (devido ao caráter descentralizado da rede), não-repúdio ou irretratabilidade (através do uso de assinatura nas transações) e também pela imutabilidade (forma como os blocos são interligados pelo *hash* criptográfico) [Dedeoglu et al. 2020].

Como o próprio nome indica, uma Blockchain é composta por uma cadeia de blocos[1]. Porém, é importante destacar que o conceito vai muito além da estrutura de dados. Para melhor entender o funcionamento da tecnologia Blockchain, os conceitos básicos podem ser divididos em quatro camadas, conforme ilustrado na Figura 3.1.

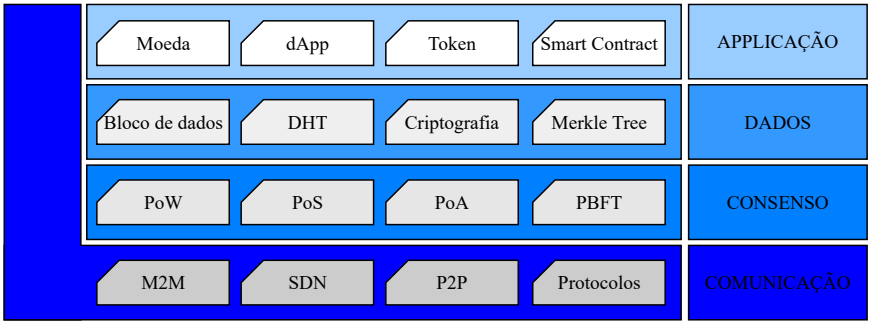

Figura 3.1. Conceitos de Blockchain em camadas [Zorzo et al. 2018].

Na **camada de comunicação** são definidas as questões arquiteturais da rede de nós que podem interagir com a Blockchain, bem como os protocolos de comunicação. Como exemplo arquitetural, pode ser citado uma rede P2P completamente descentralizada (onde todos os nós têm o mesmo papel) ou, alternativamente, a utilização de uma rede hierárquica (onde alguns nós possuem papéis específicos). As questões de arquitetura influenciam na forma de acesso e controle da Blockchain. Por exemplo, em uma Blockchain Pública todos os nós podem acessar a cadeia de blocos. Por outro lado, numa Blockchain Privada ou Consórcio somente um grupo ou conjunto de grupos de nós pode ter acesso. Adicionalmente, em uma Blockchain não permissionada (*permissionless*), todos os nós participantes têm as mesmas capacidades de produzir e acessar dados. Diferentemente, em uma Blockchain permissionada (*permissioned*), existem papéis para os nós (*e.g.*, inserir ou verificar os dados inseridos) [Dedeoglu et al. 2020]. Na prática, as definições estabelecidas na camada de comunicação podem possibilitar ou inviabilizar a utilização de determinados algoritmos de consenso.

Na **camada de consenso** é definido o algoritmo consenso a ser utilizado, que representa um conjunto de passos para garantir que os nós compartilham de uma versão válida da informação na Blockchain. O algoritmo de consenso é executado sempre que uma nova informação é proposta, garantindo que ela será inserida na Blockchains seguindo as definições e passos para inserção na rede. Existem diferentes algoritmos de consenso [Miers et al. 2019], como *Proof-of-WorK* (PoW), *Proof-of-Stake* (PoS), *Practical Byzantine Fault Tolerance* (PBFT), e *delegated Byzantine Fault Tolerance* (DBFT). Estes algoritmos podem possuir diferentes características, ocasionando grande impacto no funcionamento e no desempenho (*e.g.*, latência, vazão, uso computacional, número de mensagens trocadas) de uma Blockchain [Lunardi et al. 2019].

Na **camada de dados** devem ser definidos a estrutura de dados e como eles são relacionados, bem como os algoritmos de criptografia utilizados. Por

[1]Neste texto utilizaremos Blockchain e cadeia de blocos como sinônimos.

exemplo, os dados podem ser organizados em uma sequência única de blocos imutáveis [BitcoinFoundation 2017], na forma de um gráfico acíclico direcionado (com múltiplos ramos) [Foundation 2020], ou através de uma cadeia de blocos incrementais (blocos que podem ter novas transações) [Lunardi et al. 2020]. Além disso, dependendo da estrutura de dados, podem ser utilizadas diferentes abordagens como *Merkle Tree* ou *Hash Chains*. Complementarmente, podem ser utilizados ainda diferentes algoritmos de criptografia para assinatura (o mais comum sendo o ECDSA[2]) e funções *hash* criptográficas (*e.g.*, SHA-2[3]) [BitcoinFoundation 2017].

Na **camada de aplicação** estão presentes as interfaces para utilização da Blockchain, que definem se a cadeia de blocos irá realizar transações, as formas de comunicação e as APIs para comunicação com diferentes aplicações, por exemplo. Resumidamente, esta camada define como será o acesso de entidades externas à Blockchain. A camada de aplicação define também a possibilidade de utilização de *smart contracts* (contratos inteligentes – trechos de código que podem ser executados de forma descentralizada). Aplicações que utilizam *smart contracts*, também conhecidas como aplicações distribuídas ou simplesmente dApps, que popularizaram-se devido a características como alta disponibilidade e baixo custo computacional para quem solicita a execução (as aplicações são executadas pelos nós e não pelos usuários que solicitam a execução do código) [Nunes et al. 2020].

É importante destacar que cada Blockchain pode possuir diferentes definições em cada camada, o que podem levar a desafios particulares a cada cadeia de blocos. Por exemplo, alguns dos principais problemas encontrados em Blockchains são: (*i*) ocorrência de conflitos devido a problemas de propagação (*e.g.*, *delays* na comunicação ou blocos validados simultaneamente durante o consenso); (*ii*) quantidade de dados produzida e mantida na Blockchain (*e.g.*, a cadeia de blocos do Bitcoin ultrapassa a marca de 400GB de dados em Julho de 2022); (*iii*) tempo necessário para uma aplicação receber confirmação de inclusão de nova informação na Blockchain.

Para melhor compreender esta tecnologia, o objetivo do capítulo é fomentar a discussão sobre a utilização e o funcionamento de Blockchains, apresentando suas características e utilizando como exemplo quatro implementações (Bitcoin, Ethereum, Hyperledger e Ripple). Para alcançar o objetivo, será apresentado e discutidos neste capítulo: (*i*) os fundamentos teóricos necessários para o entendimento de uma Blockchain, incluindo conceitos fundamentais de segurança e criptografia; (*ii*) os principais conceitos utilizados para realizar o processo de inserção de um novo bloco; (*iii*) os algoritmos de consenso necessários à inclusão dos novos blocos na cadeia; e, (*iv*) algumas comparações entre as quatro Blockchains apresentadas.

O capitulo é organizado como segue. Na Seção 3.2, é apresentada a origem das Blockchains que conhecemos, isto é, uma introdução ao Bitcoin. Na sequência, é apresentada uma discussão sobre a Ethereum e a utilização de *smart contracts* (Seção 3.3. Nas Seções 3.4 e 3.5 são apresentadas as Blockchains Hyperledger, de propósito geral, e a Ripple, criada para a realização de transferências bancárias. Apresentamos ainda, na Seção 3.6, algumas comparações entre as quatro cadeias de blocos. Finalmente, na Seção

[2] https://csrc.nist.gov/glossary/term/ecdsa
[3] https://csrc.nist.gov/projects/hash-functions

3.7 e no Apêndice 3.8 são apresentados um resumo final e um conjunto de exercícios sobre conteúdo do capítulo.

Vídeos explicativos:

Como funciona a Blockchain? — Binance Academy
`https://www.youtube.com/watch?v=3rL0OIXbMio`

Blockchains: how can they be used? — Symply Explained
`https://www.youtube.com/watch?v=aQWflNQuP_o`

3.2. Bitcoin: a origem

Satoshi Nakamoto [Nakamoto 2008] publicou o artigo intitulado *Bitcoin: A peer-to-peer eletronic cash system* em meados de 2008, onde é descrita a proposta de criação de uma moeda virtual denominada Bitcoin. As principais características da proposta inicial são a descentralização, dados de transações públicos e distribuídos por todos os nós da rede, não dependência da confiança em uma entidade emissora centralizada, certificação da propriedade da moeda, e controle de gastos duplos e falsificação.

Para garantir as transações públicas descentralizadas, evitar os gastos duplos e também controlar a propriedade das moedas virtuais foi proposta uma tecnologia denominada de Blockchain. Resumidamente, a ideia principal é armazenar as transações de transferência de posse das moedas em uma Blockchain. O armazenamento das transações é realizado em blocos, onde cada bloco da cadeia possui ligação com o anterior e com o próximo de maneira sequencial, gerando um encadeamento de blocos que deu origem ao nome da tecnologia [Chervinski and Kreutz 2019].

A Blockchain utiliza em sua base algoritmos de *hash* criptográfica (*e.g.*, SHA-256) e de assinatura digital (*e.g.*, ECDSA) [Swan 2015]. Enquanto a *hash* criptográfica oferece integridade para o encadeamento dos blocos, algoritmos como o ECDSA permitem a utilização de assinaturas digitais, através de chaves pública e privada, para assegurar a autenticidade de operações na Blockchain, por exemplo. Para cada proprietário das moedas é gerado pseudo-aleatoriamente um par de chaves, cuja chave privada deve permanecer secreta. Por outro lado, a chave pública é divulgada para todos nós da rede, pois ela serve para identificar o endereço do usuário na rede do Bitcoin.

Todas as transações executadas na rede do Bitcoin são baseadas no par de chaves do usuário. Na Figura 3.2 é ilustrado como as chaves são utilizadas de maneira a efetuar a assinatura de cada transação. Para a execução de cada transação, utiliza-se como entrada a chave pública do usuário que está recebendo uma transação (*Owner 2's Public Key*, bem como a identificação da transação anterior a ela. Em seguida é calculado o *hash* criptográfico das duas informações e adicionada uma assinatura digital utilizando a chave privada do usuário que está executando a transferência dos *bitcoins* (*Owner 1's Private Key*) [Nakamoto 2008].

Para cada transação será computado um *hash* criptográfico SHA-256. No bloco será criada uma árvore, conhecida como *Merkle Tree*, que possui o somatório dos *hashes* criptográficos das transações contidas no bloco. Na árvore Merkle o *hash* criptográfico

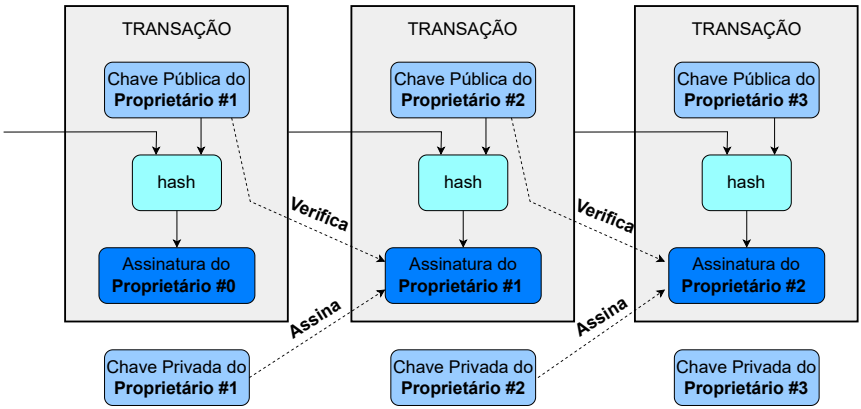

Figura 3.2. Sistema de transações do bitcoin [Nakamoto 2008]

de cada transação ficará nas folhas, enquanto que na raiz da árvore constará a soma da soma dos *hashes*. Caso haja qualquer tipo de adulteração na transação, a verificação será possível através de um dos valores do somatório dos *hashes* criptográficos, como ilustrado na Figura 3.3.

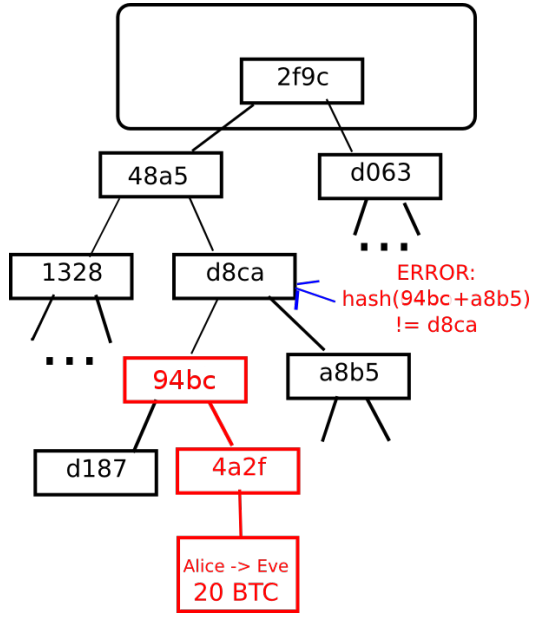

Figura 3.3. Exemplo de árvore Merkle do Bitcoin [Ethereum 2022]

O conteúdo de um bloco é ilustrado na Figura 3.4 e inclui um agrupamento de

transações. Cada bloco possui quatro campos principais, o tamanho do bloco, o cabeçalho (que contém informações de configuração e encadeamento entre os blocos), um contador de transações e o conteúdo do bloco propriamente dito, onde as transações são efetivamente armazenadas [Antonopoulos 2014].

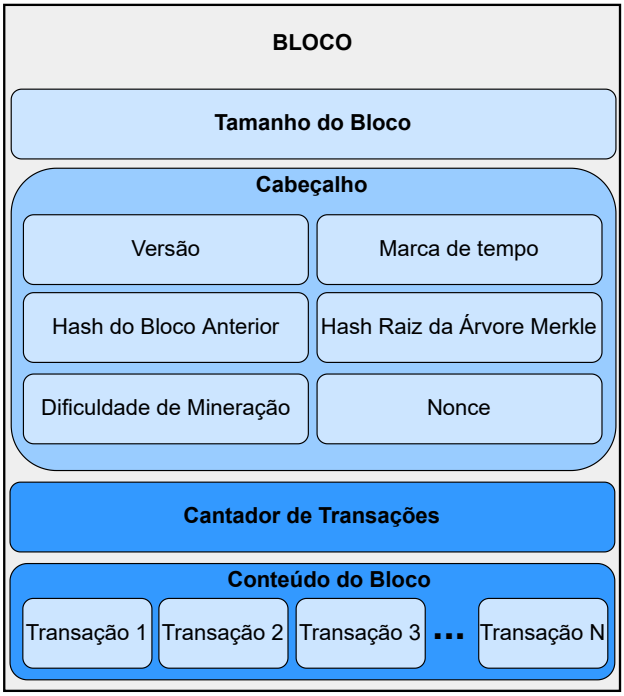

Figura 3.4. Estrutura dos blocos da Blockchain Bitcoin

A ligação entre os blocos é concretizada pelo armazenamento do resultado do *hash* criptográfico do bloco anterior no cabeçalho do novo bloco. Esse mecanismo de ligação garante a sequência entre os blocos e a integridade dos dados contidos nos blocos. Qualquer modificação nos dados de um bloco da cadeia irá comprometer tanto a sequência entre os blocos quanto a integridade dos dados dos blocos da sequência.

A Blockchain utilizada no Bitcoin funciona de modo descentralizado, utilizando uma rede P2P[4], ou seja, não possui um ponto único de falha e nem de controle. Essa Blockchain representa um sistema de registros amplo e distribuído, onde qualquer usuário pode auditar as informações contidas na cadeia de blocos [Bradbury 2016]. Resumidamente, a Blockchain do Bitcoin traz como características: (*i*) ausência de entidade única de confiança; (*ii*) inviolabilidade (em função das cadeias baseada em *hash* criptográficas); (*iii*) facilidade de auditoria (por qualquer nodo, através da verificação de assinaturas e *hash* criptográficas); e, (*iv*) auto regulável através do seu próprio sistema, ou seja, sem

[4] https://www.britannica.com/technology/P2P

auxílio humano.

3.2.1. Consenso no Bitcoin

Todo novo bloco inserido na Blockchain necessita ser validado. A validação consiste em executar um protocolo de consenso entre os nós da rede. O nó responsável pela inserção deve realizar um quebra-cabeça, que demanda tempo de processamento, cujo resultado é facilmente verificável pelos demais nós da rede. O algoritmo utilizado na rede Bitcoin é a prova de trabalho (PoW) [Nakamoto 2008].

A prova de trabalho consiste em calcular o *hash* criptográfico do bloco candidato a ser inserido na Blockchain. Entretanto, há uma dificuldade na geração do *hash* criptográfico, que é ditada por regras definidas pela rede. No caso do Bitcoin, o *hash* criptográfico deve ser iniciado por N bits zeros, onde N é definido pelo nível de dificuldade armazenado na Blockchain. Para identificar uma *hash* criptográfica que atende a regra, o minerador deve modificar o *nonce*[5] (ver Figura 3.4) até atingir o resultado esperado. Uma vez resolvido este quebra-cabeça, a informação é passada para os nós da rede. Se o valor do *hash* criptográfico estiver correto e de acordo com a regra, o bloco é inserido da Blockchain.

O trabalho de validação dos blocos é executado concorrentemente por todos os nós da rede. Quando o bloco é validado e inserido na Blockchain, o nó responsável pela solução do quebra-cabeça recebe como recompensa um valor em *bitcoins*. Esta recompensa é o incentivo da rede para os nós manterem uma cópia da Blockchain, bem como atuar ativamente validando novos blocos.

3.2.2. Discussão a partir do Bitcoin

Devido ao desperdício de processamento causado pela resolução do quebra-cabeça (*i.e.*, milhões de nós gastando energia para resolver os quebra-cabeças), outros mecanismos de consenso tem sido propostos, como a prova de posse (PoS) [Watanabe et al. 2016]. O funcionamento do PoS leva em consideração a quantidade de moedas em posse de cada nó, bem como a idade destas moedas. Através da idade das moedas de cada nó é definida a prioridade para que o mesmo participe da validação do bloco. Resumidamente, o nó que possui moedas em sua carteira por mais tempo tem mais chances de participar da validação. Uma vez validado o bloco, a idade das moedas dos nós que participaram da votação é zerada, dando chance a outros nós participarem da próxima votação. A quantidade de moedas em posse do nó é utilizada para calcular o percentual da recompensa financeira que o nó validador irá receber.

É importante ressaltar a necessidade de uma análise do custo computacional e utilização de recursos para identificar o melhor mecanismo de consenso para validação dos blocos. Além dos mecanismos de consenso baseados em PoW e PoS, existem também outras alternativas, como *Practical Byzantine Fault Tolerant* (PBFT) [Bousbiba and Echtle 2016], *Proof-of-Activy* (PoA) [Bentov et al. 2014] e *Proof-of-Publication* (PoP) [Wang et al. 2015].

[5] https://academy.binance.com/en/glossary/nonce

Vídeos explicativos:

But how does bitcoin actually work? — 3Blue1Brown
https://www.youtube.com/watch?v=bBC-nXj3Ng4

Dificuldade de mineração - simplesmente explicada — Symply Explained
https://www.youtube.com/watch?v=o1gOyhU6XEw

3.3. Ethereum

A Blockchain Ethereum [Ethereum 2022] tem como a finalidade, além do armazenamento das transações de criptomoedas entre partes [Nakamoto 2008], o armazenamento de trechos de códigos (executáveis), denominados *smart contracts*. Um contrato inteligente pode ser visto como um trecho de código que é inserido na Blockchain, cuja principal característica é ser auto-executável e possuir uma identificação única. Na cadeia de blocos, um *smart contract* é ativado quando houver uma transação endereçada a ele. A operação programada no contrato inteligente é executada em todos os nós da rede, utilizando como entrada as informações passadas na transação que disparou o contrato.

Os *smart contracts* atuam como agentes autônomos dentro da Blockchain e possuem um comportamento pré-definido, o que os torna confiáveis para prover lógica a ser executada dentro da Blockchain. Os contratos inteligentes tornaram possível as organizações autônomas descentralizadas (*decentralized autonomous organization* - DAO) [Christidis and Devetsikiotis 2016], entidades existentes na Blockchain cujo comportamento pode ser modificado através de *smart contracts*, onde um contrato pode incorporar outro contrato através de seu endereço. Resumidamente, os contratos inteligentes e as DAO representam as principais inovações da Blockchain Ethereum quando comparada à Bitcoin.

A criptomoeda da Ethereum, o *ether*, pode ser utilizada tanto como moeda digital, quanto para pagamento de taxas de execução dos *smart contracts*. Como o preço do *ether* pode oscilar e atingir valores elevados, foram propostas pequenas frações da moeda para as taxas. Na Blockchain da Ethereum é armazenado o número inteiro da menor unidade disponível, denominada *wei*. Para entender os diferentes tipos de unidades da moeda que podem ser transmitidos na Ethereum, é apresentada uma lista com os valores correspondentes de cada opção:

- 1 *wei* = 1 *wei* (menor fração transmitida na Ethereum) ou 10^{-18} *ether*
- 1 *kwei* =10^{3} *wei* ou 10^{-15} *ether*
- 1 *mwei* =10^{6} *wei* ou 10^{-12} *ether*
- 1 *gwei* =10^{9} *wei* ou 10^{-9} *ether*
- 1 *szabo* =10^{12} *wei* ou 10^{-6} *ether*
- 1 *finney* =10^{15} *wei* ou 10^{-3} *ether*
- 1 *ether* = 10^{18} *wei* ou 1 *ether*

Além das informações sobre a criptomoeda e os *smart contracts*, os blocos da Blockchain da Ethereum possuem outros campos e dados, que são também diferentes dos apresentados anteriormente para o bloco do Bitcoin, os quais podem ser observados na Figura3.4. Resumidamente, segue uma descrição dos itens armazenados nos blocos da Ethereum:

- *parentHash* - Keccak[6] 256 bits do *header* (cabeçalho) do seu bloco antecessor;
- *ommersHash* - keccak 256 bits da lista de "*tios/ommers*" que compõe este bloco;
- *beneficiary* - Endereço de 160 bits para onde as taxas coletadas com a mineração do bloco são transferidas;
- *stateRoot* - Keccak 256 bits representando o *hash* da raiz, após todas transações serem executadas e finalizadas;
- *transactionsRoot* - Keccak 256 bits *hash* da raiz da árvore populada com cada transação que compõe o bloco;
- *receiptsRoot* - Keccak 256 bits *hash* da raiz da árvore de recebedores das transações contidas no bloco;
- *logsBloom* - filtro composto por informações indexáveis, contida em cada transação de *log*;
- *difficulty* - uma valor correspondendo a dificuldade do presente bloco. Calculado com base na dificuldade do bloco anterior e tempo;
- *number* - número de blocos entre o *genesis* (*number* 0) e o bloco atual;
- *gasLimit* - valor representando o limite de "gas" que pode ser gasto por bloco;
- *gasUsed* - valor representando o total de "gas" utilizado nas transações deste bloco;
- *timestamp* - tempo em segundos representando a data em que o bloco foi inserido;
- *extraData* - *array* de bytes contendo dados relevantes para o bloco. Tamanho máximo de 32 bytes;
- *mixHash* - 256 bits de *hash* usado para provar em conjunto com o campo *nonce*, de que o bloco atual possui "computação suficiente";
- *nonce* - 64 bits *hash* que em conjunto com o *mixHahs* prova que o bloco possui computação suficiente.

Diferentemente do Bitcoin, a Ethereum, além da verificação das transações, traz o conceito de estado das contas para dentro dos blocos. Como pode ser observado na Figura 3.5, além das informações das transações, também são armazenados os estados finais das contas. O estado armazenado sempre será o resultante do estado inicial e das transações existentes no bloco, tanto de criptomoedas quanto de códigos.

Para confirmar uma transação, o minerador segue os seguintes passos de validação:

1. verifica se a transação está correta, isto é, verifica se o remetente possui saldo suficiente, se as assinaturas estão corretas e se o campo de *nonce* está correto;
2. calcula as taxas a serem pagas pela transação (baseadas no **startgas** e **gasprice**) e subtrai do remetente o valor das taxas;
3. retira uma quantidade de "**gas**" por byte para pagar a transação;
4. transfere o valor da transação para o destinatário;
5. se a transação possui falta de saldo ou "**gas**" (utilizado para pagar taxas) insuficiente, reverte-se os estados das contas, exceto as taxas pagas, isto é, o minerador continua com o que já recebeu;
6. se não ocorreu problema, retorna o "**gas**" excedente para o usuário, salva os estados finais das contas e salva as taxas do minerador.

[6] A versão do Keccak utilizada pela Ethereum é anterior à versão alterada pelo NIST que é adotada no SHA-3

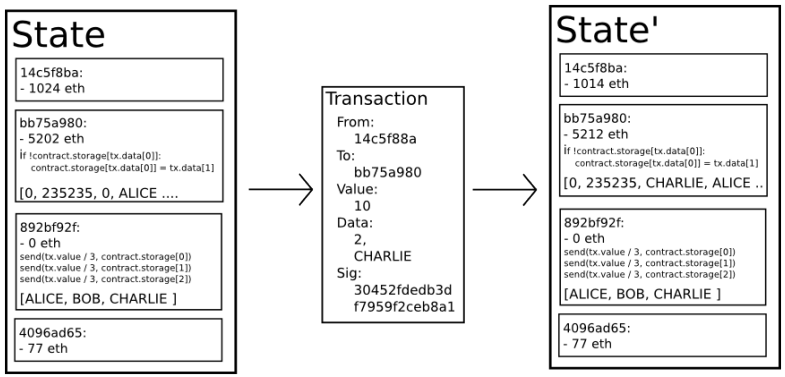

Figura 3.5. Transação na Ethereum [Ethereum 2022]

3.3.1. Consenso

Até setembro de 2022, o consenso da Ethereum era realizado de forma similar ao Bitcoin, adotando um algoritmo de PoW, ou prova de trabalho, para criar e inserir um bloco na Blockchain. O algoritmo de mineração até então utilizado pela Ethereum era o EtHash, uma versão de gerador de *hash* criptográfica que exige mais da memória, ou seja, não apenas processamento, como ocorre no Bitcoin, tornando-o menos vantajoso para processadores ASIC[7], atualmente os mais utilizados na rede do Bitcoin. Assim como no Bitcoin, a dificuldade do algoritmo PoW da Ethereum era ajustável.

A partir de Setembro de 2022, a Ethereum passa a adotar o Proof-of-Stake (PoS), ou prova de participação em tradução livre. Neste algoritmo, para participar da mineração, os nós devem fazer depósitos de uma ”cota”de criptomoedas via smart contract, para que provem que possuem participação na Ethereum. Em caso de um comportamento indesejado do nó participante (geração de blocos/transações inválidas ou extremamente demorados), essa cota será retida pela blockchain, causando prejuízo ao nó malicioso/defeituoso. Na Ethereum esta cota é de 32 Ethers.

No PoS, esses nós que participam do processo de consenso são chamados de validadores. A escolha de qual validador iniciará o consenso é aleatória e o tempo de geração de blocos passa a ser fixo (a cada 12 segundos). Para cada novo bloco é escolhido aleatoriamente um novo validador para gerar o bloco. Para validar o bloco gerado, é eleito um comitê de validadores também escolhidos aleatoriamente.

Vale ressaltar que somente serão adicionadas nos blocos as transações que forem validadas, ou seja, aquelas que chegarem ao estado final. Independente de a transação ser de criptomoedas ou execução de *smart contracts*, ela é inserida na Blockchain somente quando o estado das contas chegar no valor final, como pode ser visto na Figura 3.6.

[7] https://www.pcmag.com/encyclopedia/term/asic

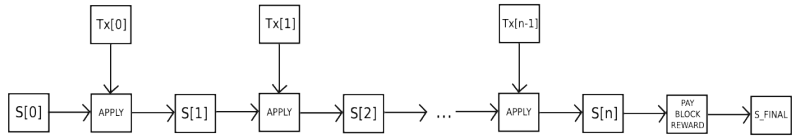

Figura 3.6. Estados da Conta [Ethereum 2022]

Apesar do processo de validação ser exaustivo por incorporar diversas atividades, os proponentes garantem que o esforço recompensa pelo fato de evitar problemas relacionados ao armazenamento de informações indevidas das contas e por manter o estado válido atualizado de cada conta na Blockchain. Intuitivamente, a expectativa é que será mais fácil verificar o estado de cada conta em momentos futuros, uma vez que a Blockchain armazena o último estado válido de cada conta.

Vídeos explicativos:

Vitalik Buterin explains Ethereum — Ethereum
https://www.youtube.com/watch?v=TDGq4aeevgY

Smart contracts — Symply Explained
https://www.youtube.com/watch?v=ZE2HxTmxfrI

3.4. Hyperledger

A Hyperledger Foundation[8] é uma organização sem fins lucrativos que reúne todos os recursos e infraestrutura necessários para desenvolver ecossistemas de código aberto para projetos de Blockchain. O principal objetivo definido pela Hyperledger é a busca pela definição de um padrão de interligação de diferentes soluções da indústria que utilizam a tecnologia de Blockchain.

Os principais *frameworks*, considerados como *Graduated Hyperledger Projects* em Julho de 2022[8], de Blockchain disponibilizados pela Hyperledger são:

1. **Fabric**[9] é uma implementação da tecnologia de Blockchain que utiliza algoritmos de consenso PBFT [Castro et al. 1999] e oferece suporte a *smart contracts*. A rede do Fabric é composta por dois tipos de nós: *validating peers* (nós de validação) responsáveis por manter a cadeia, executando o algoritmo de consenso e as transações; e *non-validating*, ou nós não validadores, responsáveis por interconectar os diferentes clientes e validar as transações entre os *peers*, mas sem executar as transações.
2. **Iroha**[10] consiste em uma implementação de módulos que podem ser utilizados para a composição de uma Blockchain ou para a integração com outros projetos.

[8] https://www.hyperledger.org/
[9] https://www.hyperledger.org/use/fabric
[10] https://www.hyperledger.org/use/iroha

Dentre os módulos providos pelo Iroha, podem ser destacadas as bibliotecas de suporte para iOS e Android, uma variante do algoritmo de consenso dos generais bizantinos, aqui denominada de Sumeragi, biblioteca para *broadcast* em rede P2P e uma biblioteca para serialização de transações.

3. **Sawtooth**[11] é uma implementação de uma Blockchain distribuída voltada para aplicações de empresas, visando manter *smart contracts* com garantia das particularidades de cada empresa. O projeto Sawtooth utiliza um algoritmo de consenso denominado *Proof of Elapsed Time* (PoET), que consiste em um sorteio, entre os *peers* confiáveis, para identificar quem será o líder da validação do bloco a ser inserido.
4. **Aries**[12] prove um *framework* de interoperabilidade para iniciativas e soluções voltadas para a criação, transmissão e armazenamento de credenciais digitais verificáveis. O Aries pode ser visto como uma infraestrutura para interações P2P voltadas para Blockchain. O projeto utiliza recursos criptográficos providos pelo Hyperledger Ursa[13], que oferece funções de gerenciamento distribuído e seguro de chaves secretas.
5. **Besu**[14] é um cliente Ethereum voltado para ambientes de consórcio e projetado para ser amigável para casos de uso baseados em redes permissionadas públicas e privadas. O Besu é capaz de executar testes em redes como Rinkeby[15], Ropsten[16] e Görli[17], incorporando diversos algoritmos de consenso, como PoW, PoA[18] (IBFT, IBFT 2.0), Etherhash e Clique[19].
6. **Indy**[20] prove ferramentas, bibliotecas e componentes reutilizáveis para o provimento de identidades digitais voltadas para Blockchains ou outros *ledgers* distribuídos. O principal foco da Indy é fornecer interoperabilidade entre domínios, aplicações e outros "silos" de sistemas Blockchain. O *framework* pode ser utilizado em modo *standalone*[21] ou em modo de interoperabilidade com outras Blockchains.

3.4.1. Consenso

Na Figura 3.7 é apresentado o sistema geral de consenso utilizado por projetos Hyperledger. É importante ressaltar que é uma definição básica do mecanismo de consenso, isto é, cada *framework* Hyperledger pode implementar o consenso de formas ligeiramente diferentes.

Na proposta do Hyperledger, o consenso é atingido através da execução de duas atividades:

[11] https://www.hyperledger.org/use/sawtooth
[12] https://www.hyperledger.org/use/aries
[13] hyperledger.org/use/ursa
[14] https://www.hyperledger.org/use/besu
[15] https://www.rinkeby.io
[16] https://ropsten.etherscan.io/
[17] https://goerli.net/
[18] https://www.kaleido.io/blockchain-blog/consensus-algorithms-poa-ibft-or-raft
[19] https://consensys.net/blog/quorum/hyperledger-besu-understanding-proof-of-authority-via-clique-and-ibft-2-0-private-networks-part-1/
[20] https://www.hyperledger.org/use/hyperledger-indy
[21] https://www.connectpos.com/what-is-a-standalone-system/

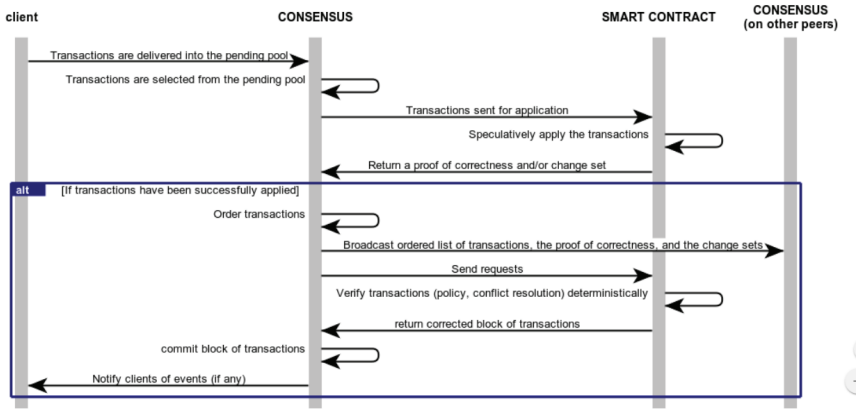

Figura 3.7. Fluxo geral de consenso de projetos Hyperledger

- **Ordenação de transações** é um serviço que deve ser implementado para garantir a ordenação das transações, agrupadas em blocos, enviadas pelos clientes;
- **Validação de transações** é um mecanismo para execução da validação das transações, onde devem ser considerados *smart contracts* que definem a lógica de validação de uma transação.

Além da execução das atividades, duas propriedades devem ser garantidas pelos *frameworks* Hyperledger:

- ***Safety*** - cada nó deve garantir que dada a mesma sequencia de entradas, resulte sempre na mesma saída e o algoritmo deve se comportar da mesma maneira em todos os nós da rede; e
- ***Liveness*** - cada nó da rede eventualmente irá receber cada transação submetida.

Tendo como base a definição dada na Figura 3.7, bem como as propriedades almejadas pela rede, cada *framework* Hyperledger apresenta a sua proposta específica de mecanismo de consenso, que pode objetivar resolver problemas específicos de diferentes cenários. Entretanto, um ponto comum em todos os projetos Hyperledger é o fato de os algoritmos de consenso buscarem atingir o consenso num tempo na casa de segundos. Esse objetivo pode ser um fator decisivo na escolha da tecnologia de Blockchain. Por exemplo, na rede do Bitcoin o consenso é alcançado normalmente com tempo superior a 10 minutos, o que é muito tempo para a maioria das aplicações atuais, onde os usuários esperam confirmar as suas transações na ordem de milisegundos. Na prática, uma confirmação de 10 minutos pode ser inviável para a maioria das transações bancárias e comerciais atuais.

Em projetos como o Hyperledger Fabric, podem ser utilizados algoritmos baseados em votação, como o PBFT. Isto leva a um custo computacional menor quando comparado com soluções baseadas em PoW, como é o caso do Bitcoin, por exemplo. Porém, é importante destacar que um protocolo PBFT gera um maior número de mensagens na rede (*i.e.*, maior tráfego de dados na rede), uma vez que todos os nós que participam do consenso devem votar para decidir se um bloco pode ou não ser inserido na Blockchain.

O principal aspecto negativo de soluções baseadas em PBFT é a escalabilidade. Uma rede com um grande número de participantes pode gerar um tráfego enorme de mensagens e, além disso, aumentar o tempo do consenso devido a dispersão dos nós e latência da rede, por exemplo. Para contornar este problema, o Hyperledger lançou diferentes opções de consenso e *frameworks*, que devem ser selecionados de acordo com o cenário de aplicação.

Vídeos explicativos:

Hyperledger Fabric Explainer Video — Hyperledger
`https://www.youtube.com/watch?v=1ORrdusUzeg`

What is Hyperledger Fabric? — BlockGeek
`https://www.youtube.com/watch?v=k4KKrQOV6SE`

3.5. Ripple

O Ripple [Armknecht et al. 2015] é um sistema de pagamento e moeda digital diferente da maioria das Blockchains e, atualmente (Julho de 2022), está entre as sete principais criptomoedas segundo o CoinMarkeCap[22]. Além de oferecer a criptomoeda XRP, o Ripple tem como propósito realizar transações financeiras, entre diferentes entidades, através da sua rede. Apesar de utilizar um protocolo de consenso distribuído [Schwartz et al. 2014, Chase and MacBrough 2018], o desenvolvimento e manutenção é gerido pela empresa Ripple.

Para realizar as transações de forma descentralizada, o Ripple conta com três tipos principais de nós na rede:

- **Usuários** responsáveis por solicitar ou enviar pagamentos, como usuários comuns e bancos com requisições de transferência;
- **Criadores de Mercado** (*market makers*) que atuam como facilitadores das transações;
- **Servidores de Validação** responsáveis por executar o algoritmo de consenso e validar transações do sistema.

Os usuários Ripple utilizam chaves públicas e privadas. A chave publica serve para identificar o usuário nas transações e para validar a transação através da verificação da assinatura digital de quem realizou a transação. Por outro lado, a chave privada é utiliza para assinar a transferência realizada. Uma transferência pode utilizar o XRP (criptomoeda do Ripple) ou ser realizada através de transações IOU ("*I Owe You*", expressão em inglês para "eu te devo uma"), quando o usuário que recebe a transação aceitar este tipo de transação.

Supondo que um usuário A deseja realizar um transferência não XRP para B. Esta transferência somente será possível se B confia em A, isto é, B aceita transações IOU de A. O problema deste tipo de transação reside no fato de que B deve criar um canal seguro com A. Para isso, B deve ter um canal de confiança (*trust line*) para receber o valor que A

[22] `https://coinmarketcap.com`

deseja enviar para B. Usualmente, o nó A irá realizar a transferência através de *Criadores de Mercado*, desde que os mesmos possuam fundos para realizar as transações. Podem ser utilizados quantos *Criadores de Mercados* forem necessários para que a transação chegue até B. O protocolo do Ripple tenta definir o menor caminho para que a transação seja realizada entre A e B [Armknecht et al. 2015]. Como exemplo, na Figura 3.8 o usuário A deseja enviar $100 para B. Para enviar o valor, são utilizados *Criadores de Mercado*, representados por U1, U2, U3, e U4. Como U3 possui a "confiança" de apenas $90 com U1, a transferência de A para B é realizada por U1, U2 e U4.

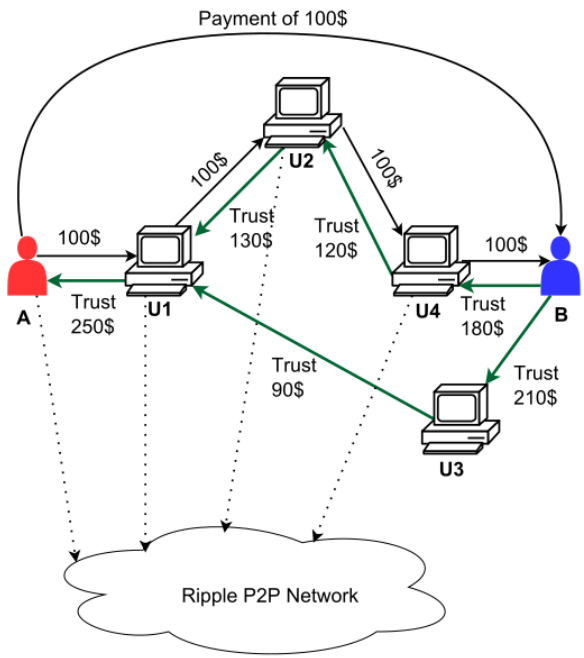

Figura 3.8. Transação IOU de A para B no valor de $100 [Armknecht et al. 2015]

Uma forma mais simples de utilizar a rede Ripple é aditar a criptomoeda XRP, que surgiu como uma forma de evitar "spam" de transações (gasta-se XRP em cada transação) e como moeda de troca universal entre usuários ou entidades bancárias. Com o XRP, diferentes bancos podem realizar transações em moedas que não possuam conversão disponível no momento da transação, por exemplo, servindo como "moeda intermediária". Para efetuar as transações, o Ripple utiliza o RTX (*Ripple Transaction Protocol*) [Todd 2015]. Uma visão geral das transações pode ser vista na Figura 3.9 .

As transações no Ripple podem ser de seis tipos:

- **Payment** - transação tradicional de envio de valores de uma conta para outra;
- **AccountSet** - transação para alterar opções de uma conta, como cancelar transações que não forem validadas;

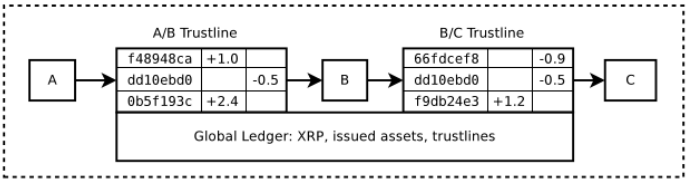

Figura 3.9. Transações no Ripple [Todd 2015]

- **SetRegulatory** - transação para definir ou mudar a chave de assinatura usada por uma entidade;
- **OfferCreate** - transação que expressa a intenção da troca de moedas;
- **OfferCancel** - transação que remove uma oferta de troca de moedas;
- **TrustSet** - transação que cria ou modifica um canal confiável entre duas contas.

Um problema comum encontrado em transações descentralizadas é a possibilidade de existência de nós maliciosos. Além da possibilidade de um nó tentar realizar uma transação fraudulenta (*e.g.*, em nome de outro nó ou sem fundos), um usuário pode tentar realizar o que é conhecido como gasto duplo (*double spending*) [Karame et al. 2012], ou seja, realizar uma transação, em que há fundos necessários, repetidas vezes antes que a rede atualize os valores de saldo do usuário. Para resolver este problema, o Ripple, assim como as demais Blockchains, utiliza um algoritmo de validação e consenso das transações realizadas pelos usuários. Desta forma, somente transações válidas são persistidas na Blockchain do Ripple, também conhecida como Ledger [Schwartz et al. 2014] ou Global Ledger [Todd 2015]. Esta Ledger global é comum para todos os nós, como pode ser observado na Figura 3.10.

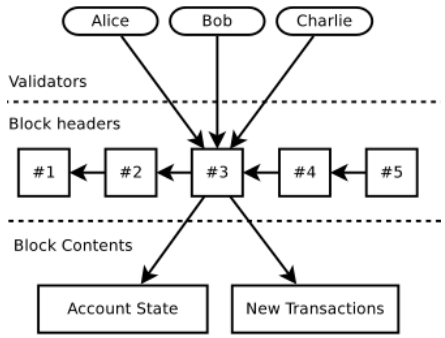

Figura 3.10. Ripple Global Ledger [Todd 2015]

Na cadeia Ripple Ledger, cada bloco possui: (*i*) cabeçalho com o *hash* criptográfico SHA-256 do bloco anterior, número de sequência, bit indicando a validade (ou não) do bloco, *timestamp*, apontamento para o conteúdo (transações) do próprio bloco e para o estado (saldo) de cada conta através de uma árvore Merkle; (*ii*) as transações propriamente ditas, representadas pelos quadrados verdes na Figura 3.11; e (*iii*) informações

sobre as contas, como identificador, as relações de confiança e saldo, representadas por triângulos azuis na Figura 3.11).

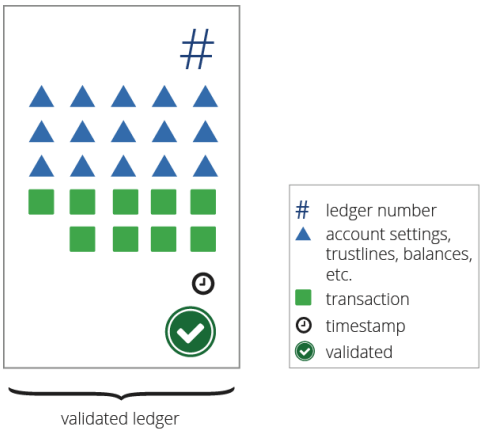

Figura 3.11. Composição de um bloco do Ripple [Ripple 2017]

Diferentemente do Bitcoin, as informações sobre saldo das contas e incrementos e decrementos realizados são armazenados na Blockchain. Os *Servidores de Validação* são os responsáveis por criar e gerir os blocos através do protocolo de consenso. Portanto, o protocolo de consenso não fica na Blockchain, mas sim nos *Servidores de Validação*, que mantém uma lista de outros servidores confiáveis chamados de *Unique Node List* (UNL).

3.5.1. Consenso no Ripple

Atualmente, o Ripple, a XRP Ledger Foundation e a Coil são conhecidas por publicar listas padrão recomendadas de validadores de alta qualidade, observando aspectos de desempenho, identidades comprovadas e política de TI [XRP Ledger 2022]. Conforme especificação do Ripple, recomenda-se que existam pelo menos 100 *Servidores de Validação* na UNL de forma a assegurar uma baixa probabilidade de ataques. Um ataque pode ocorrer quando pelo menos 20% dos *Servidores de Validação* não valida propositalmente uma transação, para impedir que ela seja concretizada. Estudos indicam que a probabilidade desse tipo de ataque ocorrer diminui drasticamente com mais de 100 nós confiáveis na UNL de cada *Servidores de Validação*.

Sempre que ocorre uma nova proposta de bloco, o mesmo é incluído na Ledger em aberto (*Open Ledger*) do *Servidor de Validação* que o criou. As transações que são validadas irão permanecer no bloco, enquanto aquelas que não receberem aprovação serão descartadas. Na Figura 3.12, as transações representadas por hexágono, triângulo e circunferência (em verde) foram validadas, enquanto que as representadas por estrela e quadrado em vermelho foram descartadas.

Após a definição das transações a serem incluídas no bloco, faz-se necessário que pelos menos 50% dos *Servidores de Validação* concordem com a alteração para, então, o bloco ser efetivamente propagado para outros *Servidores de Validação*. É importante

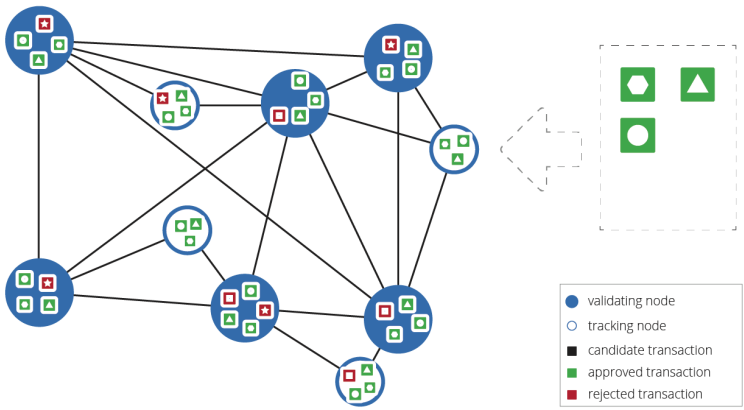

Figura 3.12. Transações validadas e descartadas na Ripple [Ripple 2017]

destacar que apenas os votos dos *Servidores de Validação* presentes na UNL serão levados em consideração. Quando o bloco proposto atinge 80% de validações positivas, a Ledger é considerada como último estado fechado (*Last Closed Ledger*). Finalmente, o novo bloco será propagado para ser atualizado na Blockchain dos demais *Servidores de Validação* (Figura 3.13).

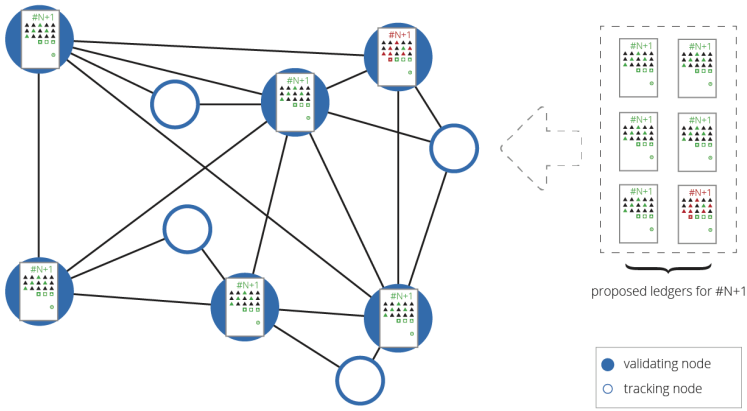

Figura 3.13. Propagação de um novo bloco na Ripple [Ripple 2017]

A quantidade de XRP necessária e o tempo de espera para realizar transações são definidos no algoritmo de consenso. A atualização destes parâmetros é realizado de forma automática pela rede, entre os próprios *Servidores de Validação* do Ripple.

O método de consenso adotado pelo Ripple possibilita a realização de transações de forma mais dinâmica. Porém, apesar do Ripple indicar que transações são realizadas

em poucos segundos, algumas transações podem levar mais tempo (*e.g.*, até 40 segundos [Armknecht et al. 2015]). Um aspecto a ser considerado no Ripple é a definição dos *Servidores de Validação* confiáveis a serem inseridos na UNL. Uma UNL maior pode garantir maior resiliência, menor risco à ataques, mas também aumentar o tempo de validação das transações.

Vídeos explicativos:

O que é Ripple? Explicando Ripple para Iniciantes — Binance Academy
`https://www.youtube.com/watch?v=Uy5e26c6P2Y`

Ripple and XRP - Part 7: Consensus vs. Proof-of-Work — Official Ripple Channel
`https://www.youtube.com/watch?v=RZqsUaDBgTY`

3.6. Comparação entre as Blockchains

Enquanto que as Blockchains Bitcoin, Ethereum e Ripple estão entre as criptomoedas com maior mercado, a HyperLedger foi criada e é mantida pela Linux Foundation, contando com diversas empresas prestando suporte ao seu desenvolvimento. Como pode ser observado na Tabela 3.1, exceto a Hyperledger, todas as outras possuem uma criptomoeda. Apesar disso, todas elas, com exceção do Bitcoin, possuem outras opções de utilização, para além de criptomoedas. Por exemplo, a Ethereum e Hyperledger suportam a utilização de *smart contracts* de propósito geral e a Ripple permite realizar transferências bancárias.

Tabela 3.1. Tabela Comparativa entre Blockchains

	Moeda	**Usos**	**Consenso**	**Acesso**	**Organização**
Bitcoin	Bitcoin	Moeda Digital	PoW	Público	Bitcoin Foundation
Ethereum	Ether	Moeda Digital e *Smart Contracts*	PoW; PoS	Público/ Privado	Ethereum Foundation
Hyperledger	-	*Smart Contracts*, Diversos (Indústria 4.0)	PBFT, Sumeragi e PoET	Privado / Consórcio e *Permissioned*	Linux Foundation
Ripple	XRP	Moeda Digital, Moeda Intermediária, e Transações Bancárias	Ripple	Público/ Privado*	Ripple Labs

Enquanto que a Bitcoin e a Ethereum 1.0 utilizam o algoritmo de consenso baseado em *Proof-of-Work* (PoW), a Ethereum 2.0 utiliza para *Proof-of-Steak* (PoS). Já a Hyperledger utiliza diferentes algoritmos de consenso em seus diversos *frameworks*. Por exemplo, a Hyperledger Fabric utiliza o *Practical Byzantine Fault Tolerance* (PBFT). Por

fim, o Ripple utiliza um algoritmo de consenso próprio, denominado Ripple consensus, que é similar ao PBFT.

Exceto a Bitcoin, todas as demais podem ser utilizadas para criação de Blockchains privadas. Apesar disso, é importante destacar que a Ripple não provê suporte e nem recomenda a utilização em modo privado. Ainda, vale ressaltar que a Hyperledger não provê uma Blockchain para utilização no modo público.

Todas as Blockchains estudadas possuem pelo menos uma empresa mantendo a respectiva cadeia de blocos. Com exceção do Bitcoin, as outras três Blockchains possuem também grandes empresas como membros ou apoiadores. Apesar de todas as Blockchains possuírem forte apoio de bancos e empresas financeiras, a Ripple [Ripple 2017] é a que possui apoio quase que exclusivo deste ramo. Já a Ethereum e Hyperledger possuem aporte diversificado, especialmente de empresas do ramo de tecnologia.

3.7. Resumo do Capítulo

Neste capítulo foram apresentadas quatro Blockchains: Bitcoin, Ethereum, Hyperledger e Ripple. Com o objetivo de introduzir conceitos básicos de funcionamento das Blockchains, o capítulo iniciou com a apresentação do Bitcoin, considerada a primeira e mais popular Blockchain. Na sequência, foram introduzidas as Blockchains Ethereum, que incorpora o conceito de *smart contracts* e, a Hyperledger, que oferece versatilidade e diferentes opções de utilização. Finalmente, a Ripple foi apresentada como um exemplo de uma Blockchain com alto desempenho para a realização de transações e com propósito de auxiliar o mercado financeiro na transferência de valores monetários.

Um dos principais fatores que diferenciam as Blockchains estudadas é o algoritmo de consenso adotado. Apesar da Blockchain do Bitcoin e da Ethereum 1.0 utilizarem o PoW, existem pequenas mudanças na Ethereum, como na validação dos estados, que permitem a adoção de *smart contracts*. Já na Hyperledger o principal algoritmo utilizado baseia-se na maioria absoluta dos usuários (maior que 2/3), similar ao que é proposto pela Ripple, que necessita de 80% de validação. Outro fator importante é sobre quem mantém os dados e como ocorre o acesso a Blockchain. Blockchains como Bitcoin, Ethereum e Ripple permitem o uso sem a necessidade de manter um nó ativo, ou seja, não é necessário manter infraestrutura computacional para utilizá-las. Todavia, a Hyperledger permite a adoção de Blockchain privado e com maior controle de acesso a mesma. Ainda, as Blockchains públicas podem permitir a criação de instâncias privadas, sendo uma alternativa para aplicações que necessitem de maior controle ao acesso a Blockchain.

3.8. Exercícios

(Q_1) **Leia as frases a seguir sobre a tecnologia de Blockchain:**

I - A estruturada de dados mais comum em Blockchains é composto por uma cadeia de blocos encadeados pelo resultado do *hash* criptográfico do cabeçalho bloco anterior.

II - Não é comum utilizar assinatura digital em Blockchains.

III - Bitcoin não é considerado uma Blockchain por possuir estrutura conhecida como *Direct Acyclic Graph* (DAG).

Sobre as afirmações, pode-se afirmar que estão corretas:

a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_2) **Leia as frases a seguir sobre o Bitcoin:**

I - O Bitcoin utiliza SHA-256 como função *hash* criptográfica e ECDSA para as assinaturas digitais.

II - Os mineradores são recompensados por produzirem blocos válidos.

III - O algoritmo de consenso utilizado no Bitcoin é o *Practical Byzantine Fault Tolerance* (PBFT).

Sobre as afirmações, pode-se afirmar que estão corretas:

a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_3) **Leia as frases a seguir sobre a Ethereum:**

I - A Ethereum 1.0 utiliza exatamente o mesmo algoritmo de consenso que o Bitcoin, baseado na produção de *hash* criptográfico utilizando o algoritmo SHA-256.

II - A Ethereum permite a utilização de *smart contracts* com salvamento do estado das variáveis.

III - Os *smart contracts* na Ethereum podem ser implementados utilizando diferentes linguagens de programação, como JAVA, C++, e Ruby.

Sobre as afirmações, pode-se afirmar que estão corretas:

a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_4) **Leia as frases a seguir sobre a Hyperledger:**

I - A criptomoeda do Hyperledger Fabric é o XRP.

II - O Hyperledger Sawtooth utiliza o algoritmo de consenso *Proof-of-Work* (PoW).

III - A principal característica dos projetos Hyperledger é poder ser utilizado em ambientes permissionados (*permissioned*), com vazão maior (número de transações por segundo) maior que o Bitcoin.

Sobre as afirmações, pode-se afirmar que estão corretas:

a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_5) **Leia as frases a seguir sobre a Ripple:**

I - A Ripple tem por objetivo realizar transações de moedas entre entidades utilizando a moeda XRP.

II - Todos os nós na Ripple possuem os mesmos papéis e capacidades de inserção na Blockchain.

III - O algoritmo de consenso da Ripple é baseado em *Servidores de Validação* e requer que pelo menos 80% desses aprovem a transação para que ela possa ser considerada válida.

Sobre as afirmações, pode-se afirmar que estão corretas:

a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_6) **Leia as frases a seguir sobre a estrutura de dados da Blockchain:**

I - As funções de *hash* criptográfico são utilizadas em Blockchains apenas para assinar digitalmente o dado, com a função de garantir a integridade do próximo bloco.

II - O custo computacional para verificar um *hash* criptográfico é muito elevado, por este motivo o algoritmo de consenso garante a validade do dado sem que o *hash* criptográfico do bloco precise ser validado.

III - Qualquer alteração em um bloco é facilmente verificável, pois a alteração de apenas um bit irá produzir valor de *hash* criptográfico diferente.

Sobre as afirmações, pode-se afirmar que estão corretas:

a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_7) **Leia as frases a seguir sobre a estrutura de dados da Blockchain:**

I - Na Blockchain do Bitcoin a assinatura digital tem a função de garantir que o nó que assinou a transação é quem originou a mesma (não-repúdio).

II - Para assinar digitalmente o usuário deve utilizar a chave privada, enquanto a chave pública é utilizada para verificar a assinatura.

III - Independente do dado a ser assinado, a assinatura resultante sempre terá o mesmo valor.

Sobre as afirmações, pode-se afirmar que estão corretas:
a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_8) **Leia as frases a seguir sobre smart contracts na Ethereum:**

I - A execução de *smart contracts* na Ethereum permite que códigos sejam executados de forma distribuída na Blockchain.
II - Para consultar valores de um contrato, sempre deve se pagar o "*gas*" devido ao processamento realizado pelo nó da Blockchain.
III - Para executar uma função de um contrato que possua alteração de estado da variável, existe uma cobrança de "*gas*" devido ao processamento realizado pelo nó da Blockchain.

Sobre as afirmações, pode-se afirmar que estão corretas:
a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_9) **Leia as frases a seguir sobre *smart contracts* na Ethereum:**

I - Em um *smart contract* pode ser definido que apenas o criador do contrato execute uma função.
II - Os resultados da execução de um *smart contract* podem ser facilmente adulterados na Blockchain da Ethereum, pois não é realizado consenso para esse tipo de transação.
III - A Ethereum não possui suporte oficial à *smart contracts*, o mesmo devendo ocorrer somente após 2022.

Sobre as afirmações, pode-se afirmar que estão corretas:
a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

(Q_{10}) **Leia as frases a seguir sobre aplicação de Blockchains:**

I - Blockchains só podem ser utilizadas para transações de criptomoedas.
II - Diferentes pesquisas e empresas vêm propondo a adoção de Blockchains para resolver problemas em diferentes áreas, como saúde, Internet das Coisas (IoT) e sistemas de registros de nomes (DNS).
III - Não existem Blockchains para ambientes privados, como por exemplo, para utilização por uma empresa ou por um conjunto de organizações.

Sobre as afirmações, pode-se afirmar que estão corretas:
a) Apenas I
b) Apenas II
c) Apenas III
d) Apenas I e II
e) Apenas I e III

Gabarito

(Q_1) Resposta: a
(Q_2) Resposta: d
(Q_3) Resposta: b
(Q_4) Resposta: c
(Q_5) Resposta: e
(Q_6) Resposta: c
(Q_7) Resposta: d
(Q_8) Resposta: e
(Q_9) Resposta: a
(Q_{10}) Resposta: b

Referências

Antonopoulos, A. M. (2014). *Mastering Bitcoin: Unlocking Digital Crypto-Currencies*. O'Reilly Media, Inc., 1st edition.

Armknecht, F., Karame, G. O., Mandal, A., Youssef, F., and Zenner, E. (2015). Ripple: Overview and outlook. In Conti, M., Schunter, M., and Askoxylakis, I., editors, *Trust and Trustworthy Computing: 8th International Conference, TRUST 2015, Heraklion, Greece, August 24-26, 2015, Proceedings*, pages 163–180, Cham. Springer International Publishing.

Bentov, I., Lee, C., Mizrahi, A., and Rosenfeld, M. (2014). Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract]y. *SIGMETRICS Perform. Eval. Rev.*, 42(3):34–37.

BitcoinFoundation (2017). Bitcoin foundation – supporting education, adoption and development in bitcoin. https://bitcoinfoundation.org/.

Bousbiba, O. and Echtle, K. (2016). A fast byzantine fault-tolerant diagnostic agreement protocol for synchronous distributed systems. In *ARCS 2016; 29th International Conference on Architecture of Computing Systems*, pages 1–11.

Bradbury, D. (2016). Blockchain's. *Engineering Technology*, 11(10):44–47.

Branco, V., Lippert, B., Lunardi, R., Nunes, H., Neu, C., Zorzo, A., Pirolla, D., Bernucio, R., and Spacov, S. (2020). Modelo de negócio para saúde colaborativa usando smart contracts: caso tokenhealth. *Revista Brasileira de Computação Aplicada*, 12(1):134–144.

Castro, M., Liskov, B., et al. (1999). Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186.

Chang, T. H. and Svetinovic, D. (2016). Data analysis of digital currency networks: Namecoin case study. In *2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 122–125.

Chase, B. and MacBrough, E. (2018). Analysis of the XRP ledger consensus protocol. *CoRR*, abs/1802.07242.

Chervinski, J. O. and Kreutz, D. (2019). Introdução às tecnologias dos blockchains e das criptomoedas. *Revista Brasileira de Computação Aplicada*, 11(3):12–27.

Chillakuri, B. and Attili, V. P. (2021). Role of blockchain in hr's response to new-normal. *International Journal of Organizational Analysis*.

Christidis, K. and Devetsikiotis, M. (2016). Blockchains and Smart Contracts for the Internet of Things. *IEEE Access*, 4:2292–2303.

Dedeoglu, V., Dorri, A., Jurdak, R., Michelin, R. A., Lunardi, R. C., Kanhere, S. S., and Zorzo, A. F. (2020). A journey in applying blockchain for cyberphysical systems. In *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*, pages 383–390.

Dedeoglu, V., Jurdak, R., Dorri, A., Lunardi, R. C., Michelin, R. A., Zorzo, A. F., and Kanhere, S. S. (2020). *Blockchain Technologies for IoT*, pages 55–89. Springer Singapore, Singapore.

Ethereum (2022). A Next-Generation Smart Contract and Decentralized Application Platform. https://ethereum.org/en/whitepaper/.

Foundation, I. (2020). IOTA - Next Generation Blockchain. https://iota.org/.

Karame, G. O., Androulaki, E., and Capkun, S. (2012). Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917.

Kishigami, J., Fujimura, S., Watanabe, H., Nakadaira, A., and Akutsu, A. (2015). The blockchain-based digital content distribution system. In *2015 IEEE Fifth International Conference on Big Data and Cloud Computing*, pages 187–190.

Lunardi, R. C., Alharby, M., Nunes, H. C., Dong, C., Zorzo, A. F., and van Moorsel, A. (2020). Context-based consensus for appendable-block blockchains. In *2020 IEEE International Conference on Blockchain (Blockchain)*, pages 401–408.

Lunardi, R. C., Michelin, R. A., Neu, C. V., Nunes, H. C., Zorzo, A. F., and Kanhere, S. S. (2019). Impact of consensus on appendable-block blockchain for iot. In *2019 16th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, page 228–237. Association for Computing Machinery.

Lunardi, R. C., Michelin, R. A., Neu, C. V., and Zorzo, A. F. (2018). Distributed access control on IoT ledger-based architecture. In *2018 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, pages 1–7.

Michelin, R. A., Dorri, A., Steger, M., Lunardi, R. C., Kanhere, S. S., Jurdak, R., and Zorzo, A. F. (2018). Speedychain: A framework for decoupling data from blockchain for smart cities. In *2018 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services (MobiQuitous)*, pages 145–154, New York, NY, USA. ACM.

Miers, C. C., Koslovski, G. P., Pillon, M. A., Jr., M. A. S., Carvalho, T. C. M. B., Rodrigues, B. B., and ao H. F. Battisti, J. (2019). Análise de mecanismos para consenso distribuído aplicados a blockchain. In Henriques, M. A. A., Terada, R., and Batista, D. M., editors, *Minicursos do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 91–139. SBC.

Min, X., Li, Q., Liu, L., and Cui, L. (2016). A permissioned blockchain framework for supporting instant transaction and dynamic block size. In *2016 IEEE Trustcom/BigDataSE/ISPA*, pages 90–96.

Moura, T. and Gomes, A. (2017). Blockchain voting and its effects on election transparency and voter confidence. In *Proceedings of the 18th Annual International Conference on Digital Government Research*, dg.o '17, pages 574–575, New York, NY, USA. ACM.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. https://bitcoin.org/bitcoin.pdf.

Nunes, H. C., Lunardi, R. C., Zorzo, A. F., Michelin, R. A., and Kanhere, S. S. (2020). Context-based smart contracts for appendable-block blockchains. In *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9.

Ripple (2017). The XRP Ledger Consensus Process. https://ripple.com/build/xrp-ledger-consensus-process/.

Saurabh, S. and Dey, K. (2021). Blockchain technology adoption, architecture, and sustainable agri-food supply chains. *Journal of Cleaner Production*, 284:124731.

Schwartz, D., Youngs, N., and Britto, A. (2014). The Ripple Protocol Consensus Algorithm. https://ripple.com/files/ripple_consensus_whitepaper.pdf.

Swan, M. (2015). *BlockChain*. O'Reilly Media, 1 edition.

Todd, P. (2015). Ripple Protocol Consensus Algorithm Review. https://raw.githubusercontent.com/petertodd/ripple-consensus-analysis-paper/master/paper.pdf.

Wang, Y., Wu, Q., Qin, B., Chen, X., Huang, X., and Zhou, Y. (2015). Group-oriented proofs of storage. In *Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '15, pages 73–84, New York, NY, USA. ACM.

Watanabe, H., Fujimura, S., Nakadaira, A., Miyazaki, Y., Akutsu, A., and Kishigami, J. (2016). Blockchain contract: Securing a blockchain applied to smart contracts. In *2016 IEEE International Conference on Consumer Electronics (ICCE)*, volume 3, pages 467–468.

XRP Ledger (2022). Which UNL should i select? https://xrpl.org/faq.html.

Zorzo, A. F., Nunes, H. C., Lunardi, R. C., Michelin, R. A., and Kanhere, S. S. (2018). Dependable IoT using blockchain-based technology. In *2018 Eighth Latin-American Symposium on Dependable Computing (LADC)*, pages 1–9.

Capítulo

4

Introdução à Vulnerabilidades e Ataques em Blockchains e Criptomoedas

Diego Kreutz (UNIPAMPA), Rodrigo Mansilha (UNIPAMPA), Igor Ferrazza Capeletti (UNIPAMPA), Laura Caroline Tschiedel (UNIPAMPA), Vinicius Nunez (UNIPAMPA), Luciano Vargas (UNIPAMPA), Roben Lunardi (IFRS), Claudio Schepke (UNIPAMPA)

Resumo. *Neste capítulo, apresentamos uma análise abrangente das principais vulnerabilidades e ataques associados à tecnologia Blockchain e às criptomoedas, contextualizando sua relevância em sistemas distribuídos modernos. Exploramos um total de 17 ataques, detalhando desde os mecanismos por trás de cada ameaça até as contramedidas atualmente conhecidas. Exemplos incluem o ataque de 51%, que pode comprometer a segurança de toda a rede ao permitir o controle majoritário por um único grupo, e o gasto duplo, que explora falhas no tempo de confirmação de transações. Além disso, abordamos vulnerabilidades como aquelas relacionadas a contratos inteligentes, que, quando mal desenvolvidos, podem expor sistemas a graves riscos de reentrada e inconsistências no estado. Ademais, também discutimos o impacto prático desses ataques, que já resultaram em perdas financeiras substanciais, como no caso da exploração de carteiras no ambiente Ethereum. Paralelamente, apresentamos medidas preventivas e corretivas, como a aplicação de algoritmos de consenso robustos, o uso de verificações adicionais em transações e a implementação de soluções avançadas contra ameaças emergentes, como a computação quântica. Apesar dos avanços no campo, enfatizamos que diversos desafios ainda permanecem, especialmente no que tange à mitigação de ataques complexos e à evolução de novos modelos de segurança para Blockchains. Concluímos destacando a necessidade de contínua pesquisa e inovação para fortalecer a resiliência das redes distribuídas contra ameaças futuras.*

4.1. Introdução

O Bitcoin [Nakamoto 2008], criado em 2008, tem revolucionado o mercado financeiro. Até então, esse mercado dependia de uma *cadeia de confiança* entre organizações públicas (*e.g.*, Banco Central do Brasil, Fundo Monetário Internacional) e privadas (*e.g.*, Itaú

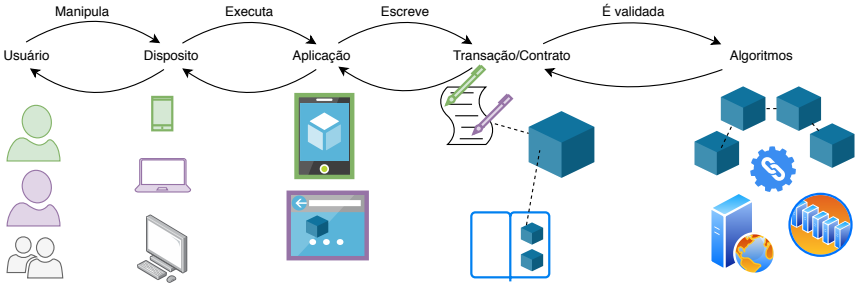

Figura 4.1. Visão Geral sobre Blockchains

e HSBC) para oferecer garantias quanto a certos quesitos de segurança do registro de transações financeiras (*i.e.*, livro-razão). Esses requisitos podem ser sintetizados da seguinte forma:

- *Integridade* - garante que um registro não será modificado de maneira indesejada;
- *Autenticidade* - garante que um registro poderá ser modificado apenas por quem lhe é permitido (também conhecido como integridade de fonte); e,
- *Disponibilidade* - garante que um registro poderá ser acessado quando for solicitado por alguém permitido.

Em contraste com o sistema financeiro tradicional, que configura um livro-razão organizado em uma hierarquia enraizada em órgão centralizador (*e.g.*, Banco Central), o Bitcoin atende os quesitos de segurança entre participantes que não necessariamente têm uma relação de confiança entre si. Muitas vezes, na rede do Bitcoin os participantes estão dispersos em larga escala numa rede de sobreposição par-a-par (P2P), formada por nós equivalentes entre-si. O Bitcoin é considerado uma tecnologia disruptiva, pois cria digitalmente uma entidade de confiança distribuída descentralizada, eliminando a necessidade de uma terceira parte de confiança [Greve et al. 2018]. A principal tecnologia viabilizadora do Bitcoin é conhecida como Blockchain.

O funcionamento de uma Blockchain é sintetizada na Figura 4.1. Pessoas manipulam dispositivos computacionais executando uma aplicação que emprega algum serviço de livro-razão. Esse livro é mantido de forma compartilhada por uma rede P2P. Registros das transações são armazenados como blocos e são encadeados criptograficamente em uma lista. A adição de novas entradas e novos blocos passa obrigatoriamente por determinados mecanismos que garantem os requisitos de segurança do sistema.

A *autenticidade* das fontes das informações em uma Blockchain é garantida pela utilização de assinaturas digitais. Os códigos de verificação da *integridade* das transações (*i.e.*, resumos criptográficos, também conhecidos como *hash* criptográfica) são armazenados em blocos de dados interligados entre si, onde cada bloco é ligado apenas ao bloco anterior a ele, resultando em uma cadeia sequencial. Antes de ser adicionado à cadeia, cada bloco é validado por um processo computacional, usualmente um cálculo matemático denominado *mineração*, como é o caso dos algoritmos de prova de trabalho, mais conhecidos como PoW (*Proof-of-Work*) [Wright 2008]. Esse processo pode ser executado por

um classe especial de nó, denominada minerador [Garay et al. 2015], que podem implementar diferentes algoritmos de consenso de acordo com as necessidades e decisões de projeto de cada Blockchain [Miers et al. 2019]. Em algumas Blockchains os mineradores podem se associar em grupos denominados *mining pools* [Silva and Rodrigues 2016]. Em uma Blockchain pública e não permissionada, como a do Bitcoin, qualquer nó da rede pode inserir novas entradas, mas novos blocos só podem ser adicionados com a concordância dos demais participantes. A *disponibilidade* das informações é suportada pela guarda compartilhada na rede P2P.

A tecnologia Blockchain tem despertado um interesse crescente da academia [Zheng et al. 2018, Monrat et al. 2019, Li et al. 2020a, Berdik et al. 2021, Deepa et al. 2022] e da indústria (*e.g* Ripple[1], HyperLedger[2], Binance[3], GoLedger[4]) [Al-Jaroodi and Mohamed 2019, Li and Zhou 2021, Chen et al. 2022]. Blockchains podem ser utilizadas em uma variedade de domínios e aplicações, como comunicações em cadeia de fornecimento, contratos inteligentes e gerenciamento de identidade digital [Pilkington 2016]. Por exemplo, a tecnologia pode ser utilizada para agregar valor (*e.g.*, rastreabilidade) à cadeia de suprimentos [Toyoda et al. 2017, Pal and Kant 2019, Galvin 2017]. Infelizmente, esse crescente valor de mercado também tem atraído a atenção de criminosos, especialmente para explorar vulnerabilidades das interfaces de Blockchains [Zhao et al. 2017], contratos inteligentes [Sayeed et al. 2020], entre outros componentes de sistemas baseados em Blockchain [Peck 2016, Li et al. 2020a, Guo and Yu 2022, Kausar et al. 2022].

Neste capítulo apresentamos uma introdução às vulnerabilidades, ataques e contra-medidas conhecidas no âmbito da tecnologia Blockchain. O restante do capítulo está organizado como segue. Na Seção 4.2 são apresentados exemplos de vulnerabilidades e soluções técnicas para Blockchains. Na sequência (Seção 4.3), discutimos as formas conhecidas de ataques e contra-medidas conceituais para Blockchain. Finalmente, apresentamos as considerações finais na Seção 4.4. Adicionalmente, apresentamos uma lista de exercícios de múltipla escolha no Apêndice 4.5.

4.2. Vulnerabilidades

Nesta seção apresentamos uma descrição sucinta das principais vulnerabilidades relacionadas às Blockchains. Os tipos de vulnerabilidades mais comuns estão sumarizados na Tabela 4.1. Essas vulnerabilidades podem ser organizadas considerando o componente alvo do sistema geral, conforme apresentado anteriormente na Figura 4.1. Na sequência, detalhamos as características e os principais impactos das vulnerabilidades em Blockchains.

4.2.1. Dispositivo

O dispositivo é frequentemente utilizado como identificador único do usuário, através de um par de chaves privada e pública. Enquanto a chave pública representa a identificação em uma Blockchain, a chave privada (secreta) assegura ao usuário a segurança das suas

[1] https://ripple.com/
[2] https://www.hyperledger.org/
[3] https://www.binance.com
[4] https://goledger.com.br

operações. É importante ressaltar que em caso de perda da chave privada, as chances de recuperá-la são mínimas, visto que as Blockchains são tipicamente descentralizadas e não possuem controle de nenhum governo ou empresa. Portanto, no caso de a chave cair em mãos erradas, será muito difícil identificar o comportamento do criminoso e conseguir recuperar as informações modificadas na Blockchain.

Problema. Uma vulnerabilidade de utilização de chaves assimétricas foi descoberta no esquema de assinatura digital por Curvas Elípticas ECDSA (*Elliptic Curve Digital Signature Algorithm*) [Mayer 2016] . Foi identificado que o algoritmo `secp256k1` não gera aleatoriedade suficiente durante o processo de assinatura. Isso torna possível ao atacante recuperar a chave através de ataques de curva inválida, que exploram vulnerabilidades de implementação do algoritmo. A implementação do `secp256k1` é também vulnerável a erros causados pela multiplicação escalar desprovida de fórmulas uniformes. **Solução**. Para resolver o problema, os desenvolvedores são aconselhados a utilizar outros algoritmos de curvas, como o `Curve25519`, considerados eficientes e seguros.

Tabela 4.1. Exemplos de tipos de vulnerabilidades

Tipo	**Especificação**	**Solução**	**Blockchains**
Chave Primária	Falta de aleatoriedade no algoritmo `secp256k1`	Utilizar outros algoritmos, como `curve25519`	Blockchains que usam `secp256k1`
Aplicativos	Falhas de implementação nos aplicativos	Substituir ou corrigir os aplicativos	Ethereum
Capacidade de transações	Falha de *design* pode alterar o ID da transação	Autenticação de dois fatores (2FA)	Bitcoin
Contratos inteligentes	Vulnerabilidades de reentrada, *timestamp*, ordem das transações e exceções mal tratadas	Monitorar a rede, verificar as exceções e definir gás para emitir uma transação.	Ethereum
Operações criptográficas	O atacante explora vulnerabilidades de *hashes* baseadas em curvas elípticas.	Utilizar curvas criptográficas sem *backdoors*	Nenhuma até o momento
Operações de *hash*	SHA-256 sofre de ataque de comprimento e extensão	Utilizar duplo *hash*	Bitcoin
Computação Quântica	Computação quântica irá quebrar as curvas elípticas atuais	Curvas elípticas super singulares e algoritmos pós- quânticos	Maioria é suscetível

4.2.2. Aplicativos

Existem diversos aplicativos que interagem com as Blockchains. Uma das formas de interação é através de códigos executados na Blockchain, mais conhecidos como *Smart*

Contracts, ou contratos inteligentes. Esses aplicativos são frequentemente vistos como elos fracos, pois são criados por desenvolvedores diversos, muitos deles pouco experientes, o que potencializa a proliferação de vulnerabilidades de segurança que afetam o ecossistema das Blockchains. Por exemplo, um ataque a vulnerabilidades de um DAO (*Decentralized Autonomous Organizations*)[5] da Blockchain Ethereum levou a um prejuízo de 60 milhões de dólares em criptomoedas que foram saqueadas [del Castillo 2016]. **Problema**. Esse ataque explorou uma vulnerabilidade de um DAO responsável pelas regras de porcentagem de votos que são necessários para ter acesso a um fundo. O prejuízo ocorreu por meio de um *bug* encontrado em um dos recursos do sistema, que possibilita a retirada de fundos ou transferência de uma porcentagem do mesmo para um DAO filho. **Solução**. Esse problema foi resolvido pela divisão da Blockchain em duas cadeias, utilizando um *fork* na cadeia de blocos da Ethereum. Isso permitiu que o dano fosse parcialmente revertido, isto é, foi realizada uma recuperação parcial do valor desviado.

Um segundo exemplo envolve aplicativos de carteira também da Blockchain Ethereum. Nesse caso, um usuário do aplicativo Parity conseguiu destruir um valor aproximado de 355 milhões de dólares, que na época correspondia a aproximadamente 513.743 Ethers (*i.e.*, unidades da criptomoeda da Ethereum) [Russell 2017]. **Problema**. O aplicativo de carteira Parity colocava seus contratos *multisig* em uma biblioteca com vulnerabilidades. No Parity, um usuário qualquer era capaz de criar sua própria biblioteca como se fosse uma carteira. A partir da sua biblioteca, o usuário tinha a liberdade de invocar uma função de inicialização e, em seguida, uma de eliminação, destruindo muitos Ethers. **Solução**. O problema de implementação foi corrigido posteriormente. Entretanto, não houve forma de interromper o incidente quando a falha foi explorada, pois as demais carteiras eram dependentes do contrato com a biblioteca do usuário, tornando o prejuízo irreversível.

4.2.3. Transação e Contratos Inteligentes

Uma implementação da realização da transação inadequada pode levar a falhas e incidentes de segurança. **Problema**. O identificador da transação pode ser alterado e levar a uma falha na capacidade de transmissão por haver maleabilidade nas transações após sua criação. Essa vulnerabilidade foi explorada no gerenciador de carteiras da Mt. Gox [Chen et al. 2019]. Como não era realizada uma verificação por parte do gerenciador Mt. Gox, os atacantes conseguiram sacar fundos de forma desonesta. O Mt. Gox gerenciava as transações por identificador. Com a manipulação do identificador, os indivíduos maliciosos solicitavam a retirada de um valor e, em seguida, alegavam que a transação havia falhado. **Solução**. Confirmação com validação da transação, evitando que uma troca de identificador leve a problemas como o ocorrido com o Mt. Gox.

Os contratos inteligentes são códigos executados de forma distribuída e podem ser classificados como um tipo especial transação. Defeitos no desenvolvimento do código do contrato podem gerar diferentes tipos de vulnerabilidades associadas ao *timestamp* dos contratos, às dependências na ordem das transações, às exceções mal tratadas, aos problemas de reentrada, e à falta de mecanismo de bloqueio do contrato. Nas Blockchains, cada bloco possui um *timestamp* que é configurado pelo minerador de acordo com seu horário local do sistema. Alguns gatilhos de contratos inteligentes necessitam do *ti-*

[5] https://ethereum.org/en/dao/

mestamp. **Problema**. Se um atacante conseguir alterar o *timestamp*, as operações dos contratos inteligentes poderão ficar vulneráveis. **Solução**. Implementação de um serviço de rótulos de tempo (*timestamp*) descentralizado.

Problema. Quando contratos são chamados por outros contratos (*e.g.*, contrato P é chama o contrato F, que contém inconsistências), o contrato chamado pode falhar e retornar como falso [Li et al. 2020b]. Se o primeiro contrato não verificar corretamente as exceções do contrato chamado F, o contrato chamador P corre o risco de estar vulnerável. **Solução**. Os contratos devem verificar as exceções e o resultado das chamadas de outros contratos invocados.

Problema. A vulnerabilidade de reentrada ocorre quando um contrato inteligente muda de estado ao final da execução, isto é, há um estado intermediário vulnerável. No estado intermediário, o atacante consegue realizar diversas chamadas repetidas para o mesmo contrato a fim de obter Ethers, por exemplo. **Soluções**. Como uma primeira solução, podemos indicar a visibilidade das transações, isto é, deixar as transações visíveis por um curto período de tempo antes de serem executadas. Assim, os observadores da rede conseguirão monitorar e tomar medidas antes do atacante conseguir incluir as transações em um bloco. Uma segunda forma de evitar a vulnerabilidade de reentrada é através do estabelecimento de um limite mínimo de unidades de "gás", isto é, taxa a ser paga. A ideia é deixar execuções recursivas complexas sem "gás", pois passaria a existir um limite por transação ou por execução de contrato inteligente.

Outras técnicas também podem ser utilizadas para verificar vulnerabilidades em contratos inteligentes da Ethereum. Por exemplo, um verificador pode ser implementado para analisar o *bytecode* dos contratos inteligentes Ethereum [Luu et al. 2016]. Um verificador pode analisar o *bytecode* dos contratos inteligentes, que são armazenados na Blockchain da Ethereum, seguindo o modelo de execução da Ethereum *Virtual Machine* (EVM)[6].

4.2.4. Vulnerabilidades Transversais

Algumas das vulnerabilidades são transversais aos componentes do sistema geral apresentados na Figura 4.1. Essas vulnerabilidades, apresentadas nesta seção, podem afetar diferentes componentes do sistema geral de Blockchains.

Algoritmos Criptográficos

Numa Blockchain, a segurança e a integridade dos dados são tipicamente garantidas através de resumos criptográficos e algoritmos de assinatura digital nas transações, blocos e cadeias de blocos. **Problema**. Vulnerabilidades das curvas elípticas utilizadas nos algoritmos criptográficos podem possibilitar uma manipulação maliciosa de dados. Um atacante, com o auxílio de um gerador pseudo-aleatório, pode chegar a um padrão numérico capaz de quebrar a criptografia. Esse tipo de vulnerabilidade é também conhecido como *backdoor* (porta dos fundos) de algoritmos criptográficos. **Solução**. Evitar curvas elípticas de baixa qualidade, como a `secp256k1`.

[6] https://ethereum.org/en/developers/docs/evm/

Operações de *hash*

Algoritmos de *hash* criptográfica, como o SHA-256, são considerados seguros por instituições como o NIST[7]. **Problema**. O SHA-256 é suscetível a ataques de extensão e comprimento. Se o ataque for realizado de maneira correta, uma mensagem assinada recebe dados fornecidos pelo atacante, que altera a mensagem original. **Solução**. Utilizar um *hash* duplo, o que inviabiliza o ataque de extensão e comprimento [Taskinsoy 2019, Pham et al. 2020], ou algoritmos de resumo criptográfico de terceira geração, isto é, SHA-3[8].

Computação Quântica

Problema. O avanço da computação quântica ameaça a resistência dos algoritmos baseados em curvas elípticas, que são baseados em logaritmos discretos. **Solução**. Utilização de algoritmos pós-quânticos baseados em curvas elípticas super singulares e desenvolvimento de Blockchains resistentes a ataques de computadores quânticos, também conhecidas como *Quantum Resistant Ledgers*.

É importante ressaltar que, nos últimos anos, o NIST manteve um programa de criação de padrões de algoritmos criptográficos resistentes à computação quântica (*quantum-resistant cryptography* ou *post-quantum cryptography*[9]). Em julho de 2022, o NIST finalmente anunciou os quatro primeiros algoritmos resistentes a ataques de computadores quânticos[10], vencedores de uma competição que durou seis anos.

4.3. Ataques e Contra-medidas

Frequentemente, os ataques são direcionados para as criptomoedas com valor financeiro mais elevado, como Bitcoin e Ethereum, visando obter lucros significativos. Uma grande parcela dos ataques concentra-se nos nodos da rede e seus relacionamentos na Blockchain. Por exemplo, ataques como o Timejacking, Eclipse, 51% e DDoS tentam isolar os nodos, enviar transações falsas, controlar a maior parte da rede e sobrecarregar o sistema para validar transações forjadas na rede.

Com o objetivo de identificar o panorama geral atual, nesta seção apresentamos os principais ataques conhecidos a Blockchains, sintetizados na Tabela 4.2. Adicionalmente, discutimos também contra-medidas conhecidas, ou soluções propostas, para cada ataque e identificamos ainda algumas das Blockchains afetadas.

4.3.1. Gasto Duplo

O ataque de gasto duplo (do inglês *double spending*) [Karame et al. 2012] ocorre quando um usuário utiliza uma mesma unidade de criptomoeda várias vezes para realizar

[7] https://csrc.nist.gov/projects/hash-functions

[8] https://www.nist.gov/news-events/news/2015/08/nist-releases-sha-3-cryptographic-hash-standard

[9] https://csrc.nist.gov/Projects/post-quantum-cryptography

[10] https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms

Tabela 4.2. Ataques, contra-medidas e Blockchains afetadas

Ataque	**Especificação**	**Contra-medida(s)**	**Blockchains**
Gasto Duplo	Utilizar a mesma criptomoeda N vezes em transações	Algoritmos de validação	Bitcoin
Finney	Emitir uma transação duplicada para a vítima	Aguardar mais confirmações	Bitcoin
Sybil	Estabelecer diversos nodos maliciosos	Verificar o *ID* e os recursos esperados	Bitcoin
Eclipse	Controlar as conexões de um nodo	Implementar limites de comunicação na rede	Bitcoin
51%	Possuir mais de 50% do poder de *hash* ou moeda total	Aguardar confirmações e observar padrões	Ethereum, Bitcoin
Vector 76	Combinação dos ataques Gasto Duplo e Finney	Mais confirmações por transação	Bitcoin
Time-jacking	Um nó é isolado com um *timestamp* diferente	Aumentar conexões entre os nós	Bitcoin
Crypto-jacking	Utilizar recursos do computador da vítima para minerar	Métodos de detecção de mineração	Monero
DDoS	Enviar inúmeras solicitações para sobrecarregar o sistema	Filtros para bloquear e limitar o tráfego de rede	Bitcoin
Spam	Enviar *spams* para atrasar a criação de blocos	Provar tarefas e gerar *tokens*	Nano, Ethereum
Sequestro BGP	Redirecionar o tráfego e atrasar mensagens de rede	Desconectar o atacante ou alterar a configuração	Bitcoin
Liveness	Atrasar a confirmação de uma transação	Bloquear nodos que não têm vivacidade	Bitcoin, Ethereum
Equilíbrio	Realizar gasto duplo com baixo poder computacional	Algoritmos de validação	Bitcoin
Mineração egoísta	Mineradores acatando blocos honestos	Diminuir a lucratividade da mineração egoísta	Bitcoin
Retenção	Obter lucros sem precisar gastar muitos recursos	Algoritmos para evitar bloqueio de retenção	Bitcoin
Suborno	Forçar o comportamento da rede para maior recompensa	Mais nodos para inviabilizar concentração de recursos	Baseadas em PoW
Verifier's Dilemma	Comportamento desleal de mineradores	Produção de transações incorretas por outros nodos	Baseadas em PoW

múltiplas transações. Esse ataque pode ser implementado através da exploração do protocolo de envio de transações em Blockchains que operam com o algoritmo de consenso PoW, como o Bitcoin. Nesse caso, o atacante pode explorar o tempo de intermediação entre o início e a confirmação das transações para realizar o ataque, como ilustrado na Figura 4.2.

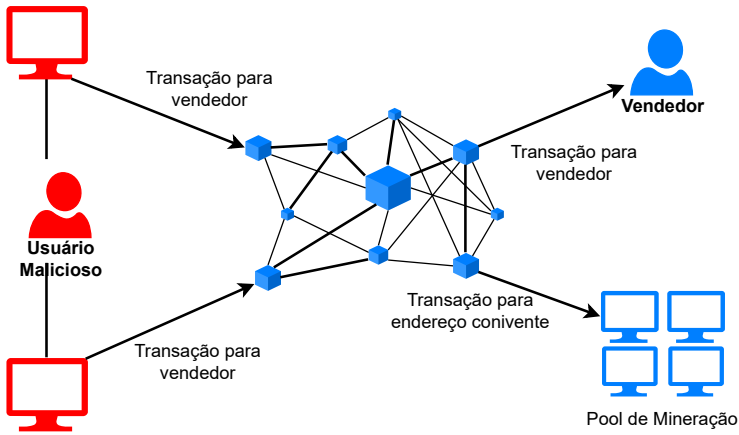

Figura 4.2. Ataque de Gasto Duplo

Num primeiro momento, o atacante precisa identificar o endereço das duas vítimas (vendedores). Em seguida, ele realiza uma transação com cada uma das vítimas utilizando as mesmas unidades de criptomoeda, como Bitcoins. O ataque é concretizado somente se as vítimas aceitarem a transação sem nenhuma confirmação. Caso as vítimas aceitem a transação sem nenhuma confirmação, o atacante poderá utilizar a mesma criptomoeda em duas ou mais transações.

O ataque de gasto duplo explora uma falha de confirmação da transação [Karame et al. 2012]. Isso ocorre com alguma frequência devido ao tempo de resposta necessário para validar as transações na rede. Para minimizar esse tempo, muitos sistemas concretizam a transação antes dela ter sido validada na rede.

Contra-medida. Criptomoedas como a Bitcoin utilizam um protocolo que recomenda aguardar a validação de pelo menos 6 (seis) nodos de forma a identificar e evitar tentativas de gasto duplo. Cálculos estatísticos indicam que 6 confirmações são suficientes para um bom equilíbrio entre tempo de espera e confiança na validação.

Vídeo explicativo:

What is Double Spending — 99 bitcoins

`https://www.youtube.com/watch?v=cOc7V64HUDQ`

4.3.2. Finney

No ataque Finney um minerador malicioso, em uma variação do gasto duplo de moeda, engana a vítima com uma transação que será invalidada, como ilustrado na Figura 4.3. O ataque inicia com a emissão de uma transação para uma conta controlada pelo atacante. Em seguida, o atacante realiza a mesma transação com a vítima. No final, a transação com a conta controlada pelo atacante será validada por meio de um bloco que ainda não havia sido divulgado e que contém auto-transações geradas pelo próprio atacante, enquanto que a transação com a vítima será cancelada.

O ataque Finney explora a vulnerabilidade de mecanismos de confirmação de pagamento e transferências. Em muitos casos, esses mecanismos não verificam a procedência do bloco que realiza a transferência. Apesar de esses mecanismos agilizarem a confirmação do pagamento para o usuário, reduzindo o tempo de espera, eles não validam suficientemente a transação, permitindo que o ataque ocorra.

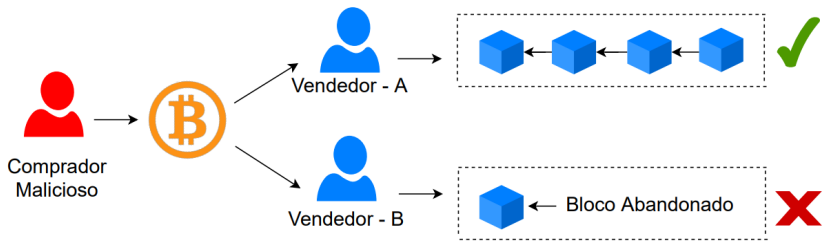

Figura 4.3. Ataque Finney

Contra-medida. O ataque Finney pode ser evitado, no momento da transação, se aguardarmos confirmações extras. No caso da Bitcoin é recomendado aguardar 6 confirmações distintas. É importante ressaltar que o número de configurações pode estar relacionado também ao valor da transação. Se o valor não for significativo, uma confirmação é o suficiente, pois o custo do ataque é relativamente alto para o atacante devido a necessidade de mineração do bloco contendo as suas auto-transações.

Vídeo explicativo:

What is double spending? — BitByBit Veve Drops

`https://www.youtube.com/watch?v=87bHzNsZs7M`

4.3.3. Sybil

O ataque Sybil [Douceur 2002] é antigo e bem conhecido na comunidade de sistemas P2P. Uma especialização do ataque Sybil, aplicada a uma Blockchain, consiste em o atacante criar várias identidades para controlar as informações transmitidas, filtrando transações que serão encaminhadas para o nodo alvo [Dasgupta et al. 2019]. Com essa estratégia, o atacante consegue controlar a visão da Blockchain do nodo vítima e utilizar-se disso para realizar outros ataques ao nodo, como o de gasto duplo. Um exemplo de ataque Sybil sendo realizado em uma Blockchain é ilustrado na Figura 4.4.

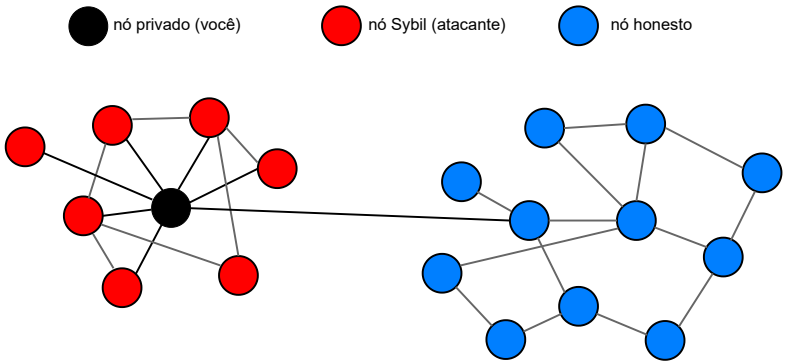

Figura 4.4. Ataque Sybil

No ataque Sybil, o atacante subverte o sistema de reputação de uma rede P2P, criando um grande número de identidades, que são utilizadas para obter uma influência na rede. Ataques Sybil a uma Blockchain podem comprometer os protocolos de anonimato descentralizados, aumentando a possibilidade de identificar as identidades reais dos usuários da rede [Biryukov and Pustogarov 2015].

Contra-medidas. Uma medida de proteção é verificar se a identidade possui os mesmos recursos que uma identidade normal e independente. Outra medida é garantir que os nodos da rede tenham conexões com nodos diversos, dificultando a efetivação do ataque Sybil ilustrado na Figura 4.4.

Vídeo explicativo:

Ataques Sybil - Segurança na Blockchain — Binance Academy

`https://www.youtube.com/watch?v=-EKhIBUQjcA`

4.3.4. Eclipse

O princípio do ataque Eclipse, em uma Blockchain, é controlar as conexões de entrada e saída do usuário, isolando a vítima de outros pares, limitando a visão da rede e fazendo com que a vítima gaste poder computacional em visões insignificantes da Blockchain [Chicarino 2019]. No exemplo da Figura 4.5, o atacante possui 117 conexões de entrada e 8 conexões de saída, criando uma rede para realizar o ataque contra nodos que aceitam as suas conexões de entrada [Heilman et al. 2015].

Existem os ataques Eclipse baseados em rede de *bots* e de infraestrutura [Shalini and Santhi 2019]. O ataque baseado em redes de *bots* consiste em controlar um número potencialmente grande de vítimas, que tornam-se os *bots* da rede. Já o ataque baseado em infraestrutura consiste em controlar um provedor de serviço de acesso a Internet ou uma empresa que detêm endereços IP contíguos.

Contra-medida. Para evitar o ataque Eclipse é necessário implementar controle das entradas e saídas de um nodo na Blockchain. Esse controle pode ser implementado

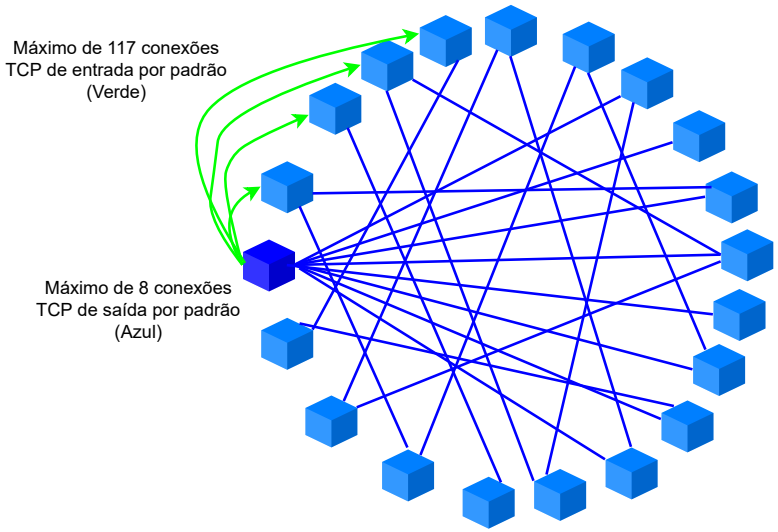

Figura 4.5. Ataque Eclipse

através de limites de comunicação, dificultando o trabalho do atacante, que poderia ser obrigado a controlar uma quantidade muito grande de nodos para ter sucesso no ataque.

Vídeo explicativo:

Blockchain Lecture 4.5 - Network Security - Eclipse Attacks — Arthur Gervais

`https://www.youtube.com/watch?v=ih_EKMs8nQE`

4.3.5. 51%

O mecanismo de consenso de algumas Blockchains é vulnerável ao ataque de 51%, que, como o próprio nome sugere, consiste em controlar a maioria do poder de mineração de blocos. Com 51% do poder de mineração, o atacante pode explorar e modificar a cadeia de blocos, eliminando ou inserindo transações, conforme ilustrado na Figura 4.6.

Apesar de ter um custo muito elevado, o ataque de 51% é viável em Blockchains baseadas nos algoritmos de consenso PoW e PoS. Em Blockchains com consenso PoW, quando um único minerador ou grupo cooperado de mineradores forem responsáveis por mais de 50% do poder de *hash* da rede, o ataque poderá ser lançado. Já em Blockchains de consenso PoS, o ataque poderá ser lançado quando um minerador possuir mais de 50% da quantidade total de moedas em circulação.

A partir do ataque de 51%, os atacantes podem explorar diversas vulnerabilidades para benefício próprio, como:

- Modificar as operações de mineração dos mineradores;
- Alterar e excluir a ordem das transações;
- Inverter transações e modificar as operações de confirmação de transações; e

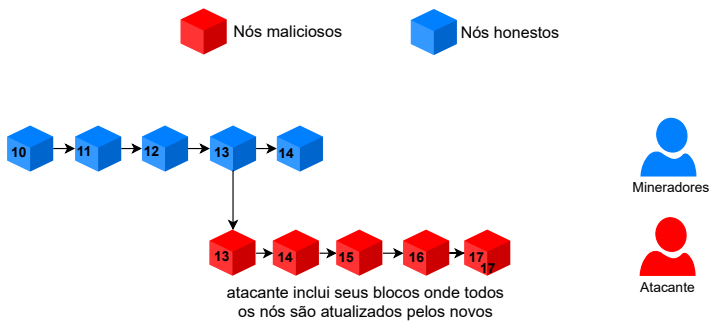

Figura 4.6. Ataque 51%

- Iniciar um ataque de gasto duplo, ou seja, gastando a mesma moeda diversas vezes.

Contra-medidas. Existem diferentes contra-medidas que podem ser aplicadas para mitigar o ataque de 51% [Sayeed and Marco-Gisbert 2019, Shrestha and Nam 2019, Ye et al. 2018, Yang et al. 2019]. Por exemplo, o ataque pode ser prevenido se evitarmos participar de *pools* de mineração formadas por grupo de mineradores que possuem uma grande capacidade de mineração. Um segundo exemplo de contra-medida é desenvolver algoritmos de consenso que dificultem que poucas máquinas possam ter controle de mais de 50% da capacidade de geração de blocos. Por exemplo, podem ser utilizados algoritmos baseados em votação (*e.g.*, PBFT, dBFT, FBA, Raft [Sankar et al. 2017, Xiao et al. 2020]) ou provas que utilizem mais memória, armazenamento ou outro recurso (*e.g.*, PoW com alto requisito de memória, PoS, PoC [Xiao et al. 2020]) que desfavoreçam máquinas unitárias com alta taxa de produção de blocos.

Vídeo explicativo:

O que é um Ataque de 51%? Explicado para Iniciantes — Binance Academy

https://www.youtube.com/watch?v=BuTj9raHQOU

4.3.6. Vector 76

O ataque de Vector 76, ilustrado na Figura 4.7, segue o padrão dos modelos de gasto duplo, explorando vulnerabilidades potencializadas pela descentralização, como os protocolos de consenso distribuídos. Consequentemente, o foco do Vector 76 tem sido principalmente as criptomoedas descentralizadas, como é o caso da Bitcoin.

O Vector 76 é utilizado frequentemente para explorar casas de câmbio (compra e venda) de criptomoedas. O ataque pode ser caracterizado como uma combinação dos ataques de Gasto Duplo e Finney. O atacante pré-minera um bloco com uma transação que transfere algum valor para uma casa de câmbio. O bloco é então enviado para os demais nodos da casa de câmbio. Assim que a transação é confirmada, o atacante envia uma nova transação para retirar o valor (*i.e.*, cancelamento da transação). Ao mesmo tempo, o atacante envia também uma transação que gasta as mesmas moedas que foram depositadas anteriormente. Com base no algoritmo de resolução de *forks*, se a transação em conflito

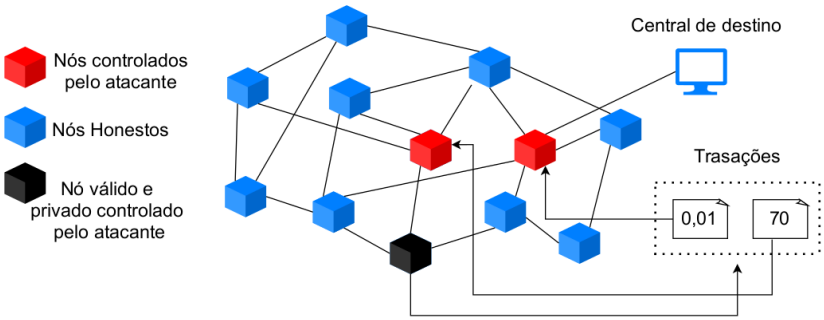

Figura 4.7. Ataque Vector 76

pertencer à cadeia mais longa, o atacante terá sucesso. Para ter mais efetividade no Vector 76, o atacante age como um minerador malicioso assumindo o controle de dois nodos ativos e bem posicionados da Blockchain. Esse controle é importante, pois quanto mais transações passam por esses nodos, maior é a probabilidade do ataque ser bem sucedido.

Contra-medida. A utilização de Blockchains com maior controle na inserção das transações é uma contra-medida importante. Para além disso, é necessário também ampliar o número de conexões de entrada e saída de cada nodo e aumentar o número de confirmações por transação.

4.3.7. Timejacking

O comportamento do ataque Timejacking é ilustrado na Figura 4.8. No Timejacking os nodos maliciosos trabalham para isolar um nodo de destino da Blockchain. Se o *timestamp* de um nodo não estiver correto, ele será rejeitado e isolado dos demais, criando margem para o ataque. Para alterar o tempo de rede, o atacante tenta desviar o carimbo de data e hora (*timestamp*) com a conexão para um alvo com nodos que emitem um tempo incorreto [Narayanan et al. 2016]. Assim que o nodo estiver isolado na rede, o atacante passa a enviar transações fraudulentas até que uma delas seja confirmada.

Contra-medidas. Uma das soluções para evitar o ataque Timejacking é estabelecer um limite superior de tempo na criação dos carimbos de data e hora do bloco e na criação do bloco. Outras contra-medidas incluem: diminuir os intervalos de tempo aceitáveis entre os nodos, utilizar apenas pares confiáveis, e monitorar a rede e desconectar os nodos quando houver atividade suspeita. Adicionalmente, antes de aceitar uma transação, é necessário verificar se há confirmações o suficiente e também utilizar o tempo mediano da Blockchain na validação dos blocos.

4.3.8. Cryptojacking

No ataque Cryptojacking, ilustrado no diagrama da Figura 4.9, o atacante objetiva utilizar o poder computacional de sua vítima, sem o consentimento dela, para minerar blocos e, subsequentemente, receber recompensas por produzir novos blocos. Um ataque Cryptojacking pode diminuir significativamente a capacidade de processamento da vítima, além de ampliar o consumo de energia, ou mesmo causar um superaquecimento na máquina da vítima [Braga et al. 2017].

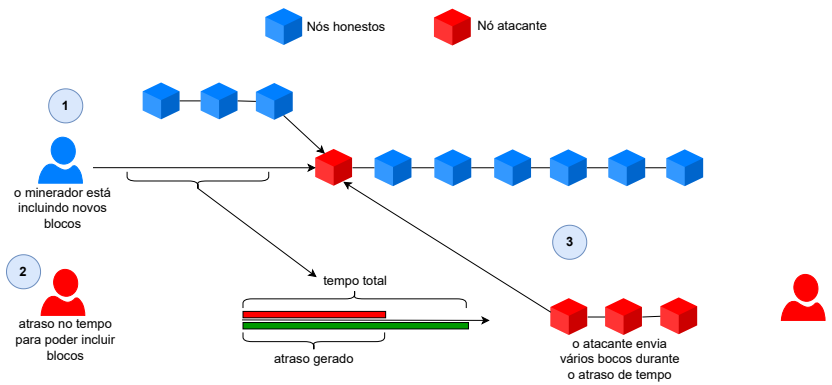

Figura 4.8. Ataque Timejacking

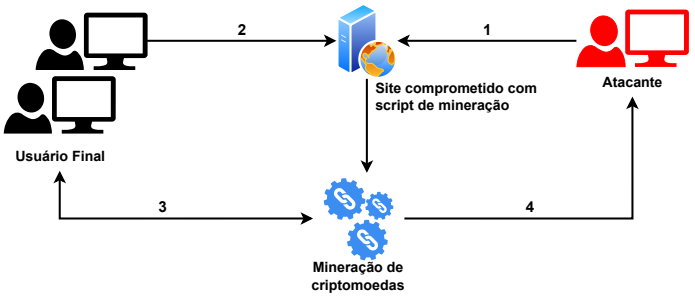

Figura 4.9. Ataque Cryptojacking

Conforme detalhado na Figura 4.9, o primeiro passo (1) do ataque Cryptojacking [Melo 2018] é o atacante comprometer um site Web onde irá injetar um *script* de mineração. O usuário final, ao acessar o site comprometido (2), irá executar o *script*, iniciando a mineração (3). Finalmente, o resultado da mineração será enviado ao atacante (4).

Contra-medidas. As principais contra-medidas ficam a cargo do usuário (ou vítima) e consistem em detectar a utilização da máquina para Cryptojacking. A detecção pode ser realizada através do monitoramento do desempenho de máquina, observando a utilização de recursos como a CPU (*Central Processing Unit*)[11]. Outras medidas para evitar esse tipo de ataque incluem a utilização de extensões de anti-criptomineração e a adoção de bloqueadores de anúncios.

Vídeo explicativo:

What is cryptojacking? Why criminals love this con — IDG TECHtalk

[11] https://www.britannica.com/technology/central-processing-unit

https://www.youtube.com/watch?v=Gvkl4tJnbik

4.3.9. DDoS

Um ataque DoS (*Denial of Service*) tem por objetivo inundar o alvo com requisições, sobrecarregado o sistema alvo de modo que requisições legítimas sejam prejudicadas (*e.g.*, não atendidas). Resumidamente, o ataque compromete a disponibilidade do sistema alvo.

Um ataque DDoS (*Distributed Denial of Service*) pode ser interpretado como uma amplificação do DoS, sendo realizado de forma distribuída através da utilização de recursos de diversos, como milhares de *hosts*. O fato de ser distribuído amplifica e potencializa significativamente o ataque, tornando-se mais efetivo na tarefa de interromper os serviços do sistema alvo.

No caso de Blockchains, um ataque DDoS de larga escala pode deixar os principais nodos da rede fora do ar por algum período de tempo, facilitando o trabalho do atacante em ações maliciosas subsequentes, como adulteração nos resultados de validação da Blockchain. Um ataque DDoS é capaz de desconectar completamente um nodo da rede da Blockchain [Biryukov and Pustogarov 2015]. A Figura 4.10 ilustra esquematicamente um ataque de DDoS a máquinas que representam nodos de uma rede de uma Blockchain.

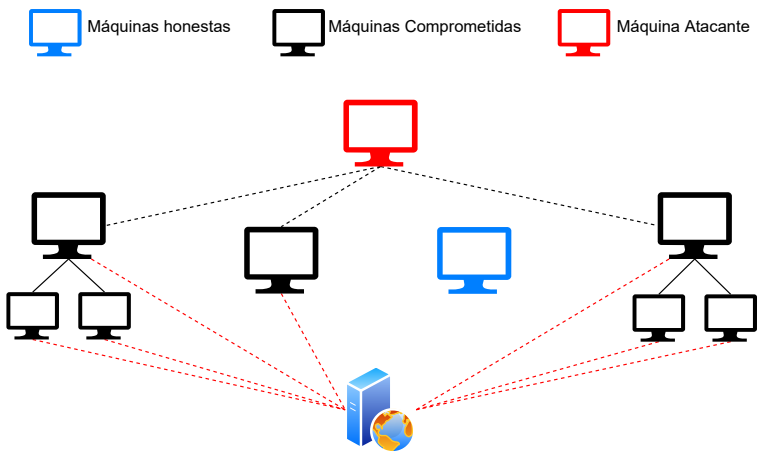

Figura 4.10. Ataque de DDoS

Contra-medida. Medidas de prevenção de ataques de DDoS podem incluir limitação de tráfego de rede através de filas de prioridade e filtros de pacotes para bloquear tráfegos maliciosos.

Vídeo explicativo:

DDoS attack on blockchain — how does it attack the blockchain — Rajneesh Gupta

https://www.youtube.com/watch?v=YiwLs4XlzdA

4.3.10. Spam

Conhecemos o ataque de Spam como UCE (*Unsolicited Commercial Email*), isto é, um e-mail que é encaminhado de forma indevida. No caso de Blockchains, o ataque de Spam, ilustrado na Figura 4.11, gera atrasos na comunicação e sincronização da rede e na criação dos blocos, afetando transações confirmadas [Begum et al. 2020, Paavolainen et al. 2018]. Segundo análises técnicas, um ataque de Spam estressou a rede Bitcoin por 18 meses [Parker 2017]. Criptomoedas como a Nano também sofreram com ataques de Spam, levando nodos a ficar fora de sincronia e causando lentidão significativa na rede da Blockchain [Nogueira 2021, Harper 2021, Dalton 2019]. Ao enviar vários *spams* para a rede da Blockchain, um atacante é capaz de causar atrasos a criação de blocos, diminuir o número de nodos que podem ser alcançados, tornar lenta a sincronização, e até mesmo gerar uma interrupção completa da rede.

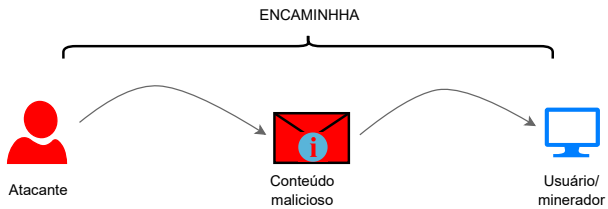

Figura 4.11. Ataque de Spam

Algumas vezes, transações de *spam* fazem parte de um ataque de negação de serviço, que pode objetivar evitar que outros usuários gastem recursos na Blockchain, como criptomoedas. Em outros casos, o objetivo pode ser mais modesto, como espalhar conteúdo indesejado ou "sujeira" na rede com transações rastreáveis.

Contra-medidas. Há diferentes técnicas que podem ser utilizadas para evitar ataques de Spam, como prioridade baseada na taxa das transações, prioridade baseada na vazão, prioridade baseada na reputação e filtragem após o fato [Dalton 2019]. A maioria das Blockchains utiliza taxas de transações para desencorajar *spam*. Por exemplo, a Bitcoin prioriza transações com altas taxas, o que desestimula usuários a criar transações sem necessidade. A Bitcoin possui também uma taxa de latência mínima, isto é, um usuário não pode enviar transações com taxa zero.

Vídeo explicativo:

Let's talk about Nano spam! — Patrick Luberus

`https://www.youtube.com/watch?v=kkIevVVvACU`

4.3.11. Sequestro BGP

O BGP (*Border Gateway Protocol*) é um protocolo de rede que gerencia como os pacotes IP são encaminhados aos destinos [Rekhter et al. 2006, Kent et al. 2000]. No ataque de sequestro BGP, como ilustrado na Figura 4.12, os atacantes objetivam interceptar o tráfego da rede da Blockchain, mais especificamente as conexões dos mineradores, redirecionando o tráfego para um *pool* de mineração controlado pelo atacante.

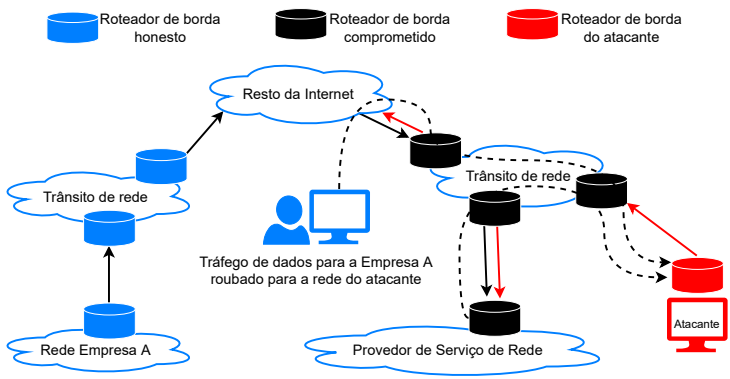

Figura 4.12. Ataque de Sequestro BGP

Nas Blockchains, os ataques de sequestro BGP ocorrem em nível de nodo ou rede. No caso da Bitcoin [Apostolaki et al. 2017], o sequestro de alguns grandes *pools* de mineração é o suficiente para produzir um efeito significativo na rede, como atrasar a velocidade de propagação dos blocos ou até mesmo dividir a rede da Blockchain. É importante ressaltar que o principal desafio do sequestro BGP é a distribuição do poder de mineração da rede, o que afeta diretamente o número de prefixos IP que precisam ser sequestrados para o ataque ser bem sucedido.

Contra-medidas. A primeira e mais eficaz contra-medida é utilizar implementações do BGP que incorporam extensões de segurança capazes de evitar o ataque ao protocolo. Um segundo exemplo de contra-medida é utilizar sistemas como BGPMon [Yan et al. 2009] para realizar o monitoramento de anúncios desonestos. Uma vez detectada a ocorrência de anúncios desonestos, é necessário alterar a configuração do BGP ou ainda forçar a desconexão do atacante.

Vídeo explicativo:

GTS 32: Ataques a Infraestrutura BGP e medidas de contenção — NICbrvideos

`https://www.youtube.com/watch?v=NS2L9Xxvqb8`

4.3.12. Liveness

O objetivo de um ataque Liveness é atrasar o máximo possível o tempo de confirmação de transações de um usuário qualquer através de uma rede privada [Kiayias and Panagiotakos 2017], como ilustrado na Figura 4.13. Esse ataque foi demonstrado em Blockchains como a Bitcoin e a Ethereum e pode ser dividido em três etapas: (a) preparar o ataque, (b) realizar a negação de transação, e (c) retardar a Blockchain.

Na primeira etapa o atacante constrói uma rede privada contendo uma cadeia de blocos mais longa do que a rede pública. Na segunda etapa o atacante nega a transação, isto é, evita que o *hash* da transação seja gravado na rede pública, mantendo o bloco desta

hash na rede privada. Finalmente, na terceira etapa o atacante retarda ao máximo a adição da *hash* da transação na rede pública. Quando sua rede privada estiver suficientemente grande, ou seja, maior que a lista de blocos atual da rede pública, o atacante irá publicar os blocos, assegurando que sua lista privada de blocos se efetive na rede pública. O ataque é finalizado quando a *hash* da transação for verificada como validada na rede pública.

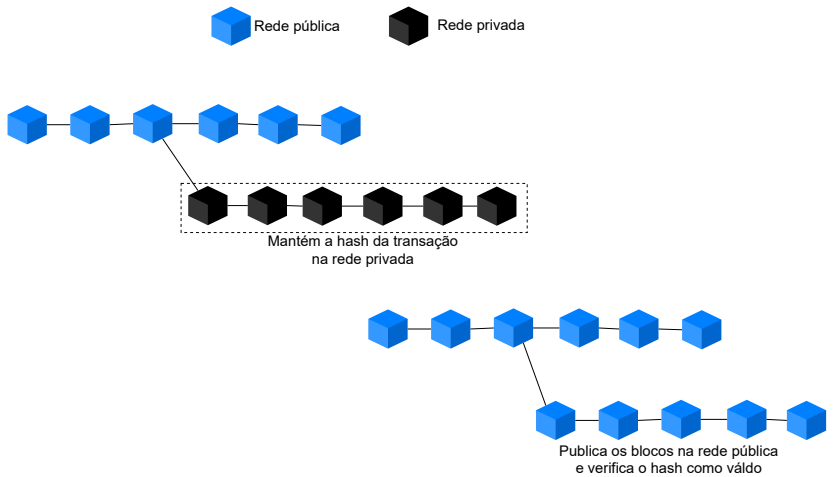

Figura 4.13. Ataque de Liveness

Contra-medida. Na rede da Blockchain, podemos classificar como divergentes os nodos que não apresentarem vivacidade (*liveness*), isto é, aqueles que não apresentarem trocas de mensagens com outros nodos presentes na rede pública. Nesses casos, quando a comunicação com outros nodos da rede pública não existir, os nodos da rede privada do atacante serão bloqueados.

4.3.13. Equilíbrio

No ataque de Equilíbrio o atacante, com poder de computação inferior, consegue interromper por um período curto de tempo as conexões entre subgrupos com poder de mineração iguais [Natoli and Gramoli 2016]. O atacante estrutura a Blockchain como uma árvore DAG (*Directed Acyclic Graph*) e divide em dois subgrupos, conforme ilustrado na Figura 4.14. O atacante envia transações em um subgrupo e emite blocos de mineração no segundo subgrupo, criando atrasos nas cadeias de blocos entre os subgrupos. O objetivo é permitir que o subgrupo de blocos consiga superar o de transações, aumentando a probabilidade de o atacante conseguir reescrever os blocos. Resumidamente, explorando o atraso entre os dois subgrupos, o atacante consegue utilizar as mesmas moedas em transações distintas, ou seja, o ataque de equilíbrio viola a persistência do prefixo principal de blocos e permite ao atacante gastar duas vezes as mesmas moedas [Li et al. 2020a].

Contra-medida. Por ser um modelo de ataque com o mesmo princípio do ataque Sybil (ver Seção 4.3.3), que facilita ataques de gasto duplo, as contra-medidas podem ser

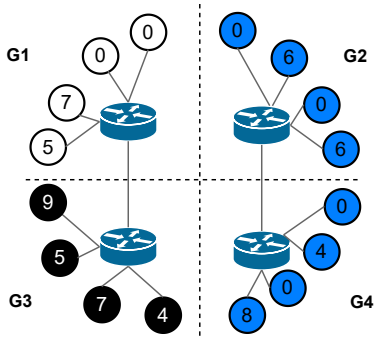

Figura 4.14. Exemplo de particionamento do DAG do Ataque de Equilíbrio

as mesmas. Por exemplo, utilizar algoritmos de validação como prova de trabalho para saber se os nodos possuem poder computacional esperado pela rede.

4.3.14. Mineração Egoísta

O ataque de mineração egoísta (*selfish mining*), ilustrada na Figura 4.15, é caracterizado por um grupo cooperado de mineradores (*pool* de mineração) que realizam ataques contra blocos honestos ou contra outros grupos cooperados de mineração da rede da Blockchain [Azimy and Ghorbani 2019]. Tipicamente, o que um ataque egoísta faz é forçar nodos honestos da rede a realizar cálculos inúteis, em uma cadeia de blocos pública obsoleta, para blocos que não serão incluídos na Blockchain.

O ataque de mineração egoísta é iniciado a partir de um *pool* de mineração controlado pelo atacante. Quando novos blocos forem adicionado à Blockchain do atacante, eles não serão propagados na rede pública. Consequentemente, existe uma probabilidade de outros mineiros estarem minerando com a cadeia de blocos defasada. Assim que conseguir obter uma cadeia de blocos mais longa do que a da rede pública, o atacante realiza a publicação seletiva de blocos a fim de desacreditar o trabalho realizado pelo resto da rede pública. O objetivo é invalidar os blocos que os demais nodos da rede mineraram.

Contra-medidas. Uma contra-medida é impor limites. Por exemplo, quando um minerador aprender sobre ramos concorrentes de mesmo comprimento da cadeia de blocos, ele deve propagar em todos os ramos e escolher aleatoriamente em qual irá minerar [Eyal and Sirer 2014]. Essa estratégia diminuiria a capacidade da rede privada aumentar substancialmente por meio do controle da propagação de dados. Outra contra-medida é diminuir a lucratividade da mineração egoísta utilizando carimbos de data e hora (*timestamps*) imprevisíveis (*freshness preferred*) para penalizar os mineradores que retêm blocos [Heilman 2014]. Um terceiro exemplo de contra-medida é a estratégia de *publish or perish* [Zhang and Preneel 2017], na qual a política de resolução de *forks* irá negligenciar blocos que não foram publicados em tempo e preferir aqueles que incorporam links para blocos predecessores concorrentes. Consequentemente, um bloco que é mantido em segredo não irá contribuir com os *forks*, desestimulando esse tipo de ataque.

Vídeo explicativo:

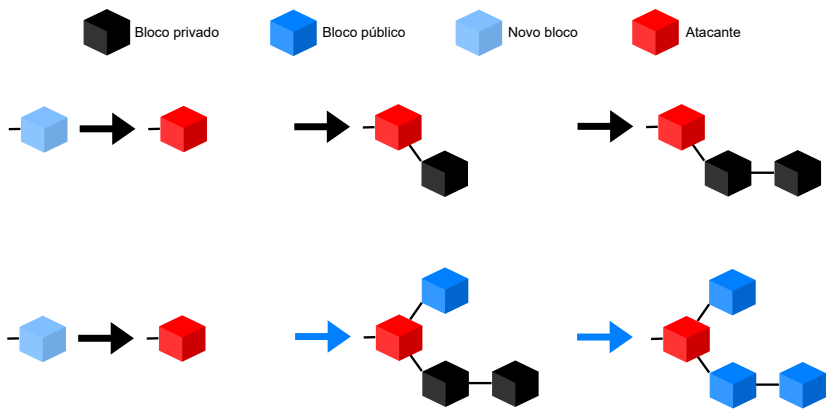

Figura 4.15. Ataque de Mineração Egoísta

Cryptoeconomics - 3.5 - Bitcoin: Selfish Mining — Cryptoeconomics Study
`https://www.youtube.com/watch?v=SWKjSEi-9pg`

4.3.15. Retenção

O ataque de retenção (*withholding attack*) [Eyal 2015, Luu et al. 2015a] é ilustrado na Figura 4.16. Ele é executado em grupos de mineradores que utilizam poder computacional combinado na mineração para obter lucros dividindo custos. Na prática, o ataque consiste em um gerente de um *pool* de mineração enviar alguns de seus mineiros para outros *pools*, onde eles irão apenas fingir que estão trabalhando na produção de resumos criptográficos.

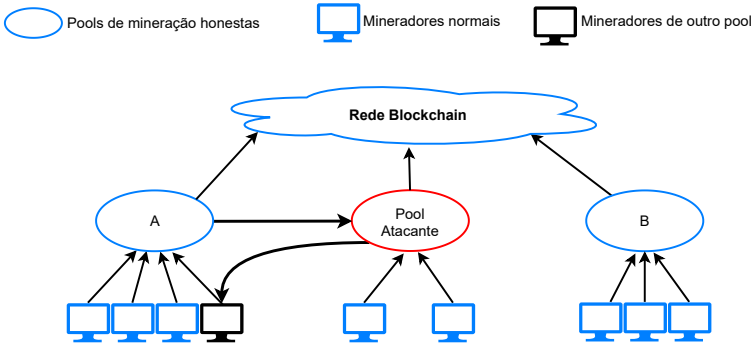

Figura 4.16. Ataque de Retenção

Em Blockchains como a Bitcoin, os gestores dos *pools* de mineração dificilmente conseguem identificar e caracterizar esses mineiros maliciosos. Consequentemente, apesar de não trabalharem, esses mineiros conseguem obter a recompensa, proporcional ao

poder de computação, da *pool* de mineração. É importante destacar que há diferentes estratégias de retenção que podem ser utilizadas, resultando em funções de recompensa (*i.e.*, ganhos) ligeiramente distintas [Luu et al. 2015a, Tosh et al. 2017, Kim and Hahn 2019, Qin et al. 2020].

Contra-medidas. Assim como no ataque de mineração egoísta (ver Seção 4.3.14), podemos diminuir a lucratividade do ataque de retenção utilizando carimbos de data e hora imprevisíveis para penalizar os mineradores retentores de blocos [Heilman 2014]. Outras estratégias podem incluir algoritmos que definem um intervalo de tempo aceitável para receber ou minerar um novo bloco de um minerador honesto [Solat and Potop-Butucaru 2017].

4.3.16. Suborno

O ataque de suborno (*bribery*) [Bonneau et al. 2016, Wang et al. 2022] é ilustrado na Figura 4.17. Ele leva em consideração o consenso da Blockchain para forçar o comportamento da rede para aumentar sua recompensa ou manipular as transações que serão inseridas primeiro. Nesse ataque, o minerador malicioso precisa ter um alto poder aquisitivo disponível para conseguir gerar lucro, considerando que é necessário uma transação de valor elevado na rede.

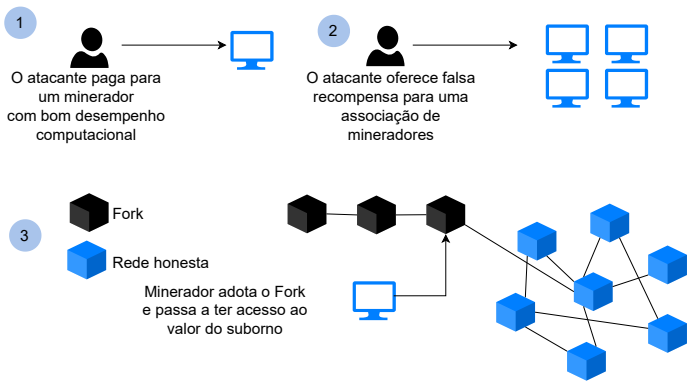

Figura 4.17. Ataque de Suborno

O principal risco do ataque de suborno é o minerador malicioso perder o seu investimento. Por outro lado, se o ataque for bem sucedido, o atacante poderá realizar múltiplas transações para obter gasto duplo [Bonneau et al. 2016]. Como o atacante terá controle sobre a rede, será possível definir transações a serem inseridas em blocos e posteriormente desfazer os mesmos blocos, gerando uma cadeia maior de blocos sem a transação com o gasto duplicado da criptomoeda.

Existem três métodos para infiltrar o suborno na rede. O primeiro caso é quando o minerador malicioso paga a outro minerador, proprietário de um bom desempenho computacional, para minerar os blocos que lhe são atribuídos. No segundo caso, o atacante cria um *pool* de mineração, oferecendo um retorno maior pelo trabalho. Finalmente, em

Blockchains como a Bitcoin, o minerador pode ainda criar um *fork* que irá conter o dinheiro do suborno, permitindo que qualquer minerador que adote esse *fork* tenha acesso ao suborno.

Contra-medida. Não conhecemos contra-medida para o ataque de suborno até o momento. Entretanto, é importante ressaltar que alugar uma grande capacidade de mineração via suborno pode ser impraticável em Blockchains como a Bitcoin. De qualquer forma, é importante ressaltar que, apesar de pouco provável, grandes empresas ou governos poderiam adquirir ou alugar um enorme poder computacional para interferir na operação de Blockchains específicas, por exemplo.

Vídeo explicativo:

How to Model the Bribery Attack: A Practical Quantification Method in Blockchain — ESORICS

`https://www.youtube.com/watch?v=r9YoSz7sGa0`

4.3.17. Verifier's Dilemma

O ataque Verifier's Dilemma diz respeito ao dilema de verificar (ou não) todas as transações que um minerador recebe [Alharby et al. 2020]. Esse ataque está diretamente relacionado ao comportamento desleal de mineradores na produção de novos blocos. Um minerador malicioso pode não validar uma transação custosa, como o processamento de um contrato inteligente, e utilizar esse tempo para minerar novos blocos.

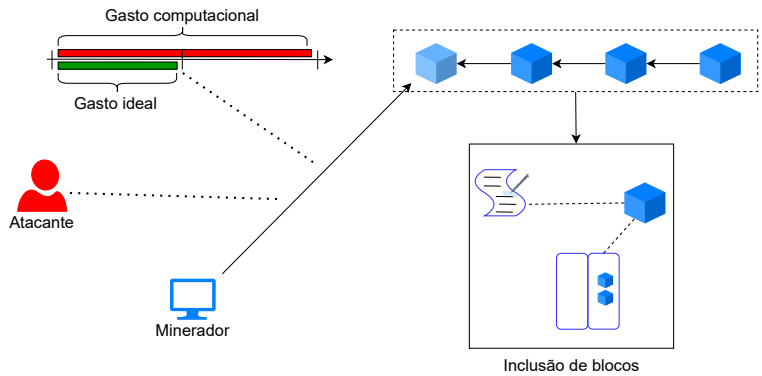

Figura 4.18. Ataque de Verifier's Dilemma

No ataque Verifier's Dilemma o minerador malicioso foca na mineração de blocos para obter as recompensas, mesmo que não tenha contribuído para o processo de validação das transações. Esse comportamento malicioso, se utilizado por um conjunto grande de nodos da rede, pode permitir que transações sem verificação sejam incluídas na Blockchain.

Contra-medida. Diminuir a vantagem do minerador malicioso. Uma forma de detectar a ação de minerados maliciosos é incluir a produção proposital de transações in-

corretas [Luu et al. 2015b]. Como os mineradores maliciosos não verificam as transações, irão produzir blocos contendo essas transações incorretas. Logo, quando esses blocos forem propagados para os demais nodos da rede, os blocos podem ser identificados e descartados pelos mineradores que realizam a verificação. Consequentemente, os mineradores maliciosos deixam de receber as recompensas pelos blocos minerados que contém transações incorretas.

Vídeo explicativo:

CESC2017 - Jason Teutsch - A Scalable Verification Solution for Blockchains — Blockchain at Berkeley

`https://www.youtube.com/watch?v=QPHowJxEkJQ`

4.4. Resumo do Capítulo

Neste capítulo apresentamos uma relação das principais vulnerabilidades e dos principais ataques a Blockchains. Por exemplo, na Seção 4.3 descrevemos dezessete ataques e respectivas contra-medidas. Como podemos observar, a maioria dos ataques vem sendo demostrada ou aplicada a criptomoedas de alto valor de mercado, como a Bitcoin e a Ethereum. Em alguns casos, vulnerabilidades e ataques a Blockchains geraram prejuízos significativos (*e.g.*, 355 milhões de dólares [Russell 2017]). Embora existam contra-medidas para a maioria dos ataques, ainda há casos desafiadores e de alto impacto, como o ataque de 51% (ver Seção 4.3.5) e suborno (ver Seção 4.3.16), sem solução. É importante destacarmos que a tecnologia de Blockchains ainda é relativamente recente, ou seja, ainda temos um longo caminho pela frente no sentido de mitigarmos os ataques conhecidos e novos, que ainda deverão surgir.

4.5. Exercícios

(Q_1) **Alguns ataques em Blockchains conseguem realizar múltiplas transações com uma mesma moeda e obter vantagens sobre outros usuários da rede. Assinale os pares de ataques a seguir e identifique aqueles que possibilitam ao atacante utilizar múltiplas vezes uma mesma moeda.**

(I) Finney e Liveness
(II) Gasto Duplo e Equilibrio
(III) Cryptojacking e Ataque 51%
(IV) Finney e Sybil
(V) Gasto Duplo e Timejacking

Assinale a alternativa CORRETA:

(a) Apenas I e II
(b) Somente II
(c) Apenas II e III
(d) Apenas II e IV
(e) Apenas I, II e V

(Q_2) **Alguns ataques se beneficiam das Blockchains explorando formas de manipular a rede. Leia e analise as afirmações a seguir.**

(I) O ataque DDoS sobrecarrega o sistema com requisições e dados, levando um ou mais nodos principais da rede a ficar *off-line*, para forjar resultados de validação, por exemplo.
(II) Através de ataques Sybil, usuários maliciosos sempre conseguem encontrar as identidades reais dos usuários.
(III) Em Blockchains que utilizam consenso PoW, o ataque 51% pode ser realizado quando o usuário tiver mais de 50% do total de moedas.
(IV) O ataque Cryptojacking permite ao atacante ajudar um usuário, acessando a sua máquina para minerar criptomoedas.
(V) Um ataque de Gasto Duplo ocorre quando o usuário malicioso identifica o endereço de duas vítimas e utiliza a mesma criptomoeda para realizar a transação com ambas. O ataque é concluído quando as duas vítimas aceitam a transação.

Marque a alternativa CORRETA:

(a) Somente I
(b) I e III
(c) II e IV
(d) I, III e V
(e) I e V

(Q_3) **No ataque de 51%, quando um usuário tiver mais de 50% do poder de *hash* criptográfico em Blockchains de consenso PoW ou tiver mais de 50% do total de moedas em cadeias de blocos de consenso PoS, o atacante consegue controlar a Blockchain. Analise as afirmações a seguir com relação aos possíveis e prováveis objetivos de ataques de 51%.**

(I) Contribuir com as operações de mineração dos mineradores.
(II) Alterar e excluir a ordem das transações.

(III) Controlar 100% dos mineradores da rede.
(IV) Inverter transações e modificar as operações de confirmação de transações.
(V) Invalidar blocos minerados pelos demais mineradores.
(VI) Iniciar um ataque de gasto duplo, gastando a mesma moeda diversas vezes.

Assinale a alternativa INCORRETA:

(a) Apenas III e V
(b) Somente II
(c) Apenas I e III
(d) Apenas V e VI
(e) Apenas II e IV

(Q_4) **Analise as afirmações a seguir com relação a ataques em Blockchains.**

(I) Para mitigar ataques Eclipse, podemos realizar o controle das entradas e saídas de um nodo através de limites de comunicação entre os nodos.
(II) Ataques de Sequestro BGP são rápidos e fáceis de solucionar, mesmo em sistemas que não incorporam nenhuma extensão de segurança BGP na rede.
(III) Os ataques de Equilíbrio e Sybil podem ser prevenidos utilizando algoritmos de validação como prova de trabalho, ou seja, os nodos precisam resolver enigmas matemáticos com alto custo computacional, por exemplo.
(IV) Apesar de existirem contra-medidas para os ataques de Gasto Duplo, o ataque pode se concretizar se um usuário aceitar transações sem nenhuma confirmação de outros nodos.
(V) Não existe solução para detectar ou impedir um ataque de Cryptojacking.
(VI) Existem diversas contra-medidas que podem ser adotadas para prevenir ataques de Timejacking, aumentar os intervalos de tempo aceitáveis entre os nodos da rede, bloquear o tráfego da rede, banir 50% dos mineradores da rede ou ainda incluir um número pseudo-aleatório em cada transação.

Assinale a alternativa em que todas as afirmações são VERDADEIRAS:

(a) Apenas II e V
(b) Apenas III e VI
(c) Apenas I, III e IV
(d) Apenas I, VI e V
(e) Apenas I, III e VI

(Q_5) **As *backdoors* são exemplos de vulnerabilidades presentes em curvas elípticas, que permitem um usuário malicioso realizar ataques. Assinale a alternativa CORRETA que apresenta (a) um algoritmo de curvas elípticas que conhecidamente gera chaves criptográficas vulneráveis a ataque de curva inválida e (b) um exemplo de curva elíptica que resolve o problema.**

(a) SK-128 e secp256k1.
(b) Curve25519 e SAFER SK-64.
(c) secp256k1 e Curve25519.
(d) SK-40 e SHA-256.

(Q_6) **Alguns aplicativos associados a Blockchains possuem vulnerabilidades e, consequentemente, viabilizam ataques que podem levar a prejuízos significativos. Com relação às vulnerabilidades conhecidas em aplicativos, analise as afirmações a seguir.**

(I) Através de uma DAO responsável pelas regras de votos para acessar fundos, um atacante pode realizar transferências para uma DAO filha de forma repetida e desviar valores do fundo.

(II) O gerenciador Mt. Gox, utilizado pelas criptomoedas Bitcoin, Ethereum e Nanocoin, não verifica o ID da transação. Consequentemente, um usuário malicioso pode modificar o ID da transação para sacar fundos de forma desonesta.

(III) Uma vulnerabilidade na chave privada foi descoberta no algoritmo ECDSA (*Elliptic Curve Digital Signature Algorithm*), causada pelo algoritmo `secp256k1`, que gera aleatoriedade robusta durante o processo de assinatura.

(IV) O aplicativo de carteira Parity, utilizado na Blockchain Ethereum, continha falhas em sua biblioteca. O aplicativo permitia ao usuário criar sua própria biblioteca em forma de carteira e invocar uma função de inicialização e outra de eliminação, que destrói as criptomoedas desta carteira.

Assinale a alternativa que contém apenas exemplos VERDADEIROS e CORRETOS de vulnerabilidades conhecidas de aplicativos:

(a) Apenas III e VI
(b) Apenas I, II e IV
(c) Apenas I, II e III
(d) Apenas II e III
(e) Apenas I e IV

(Q_7) **Analise as afirmações e identifique aquelas que podem ser apontadas como formas de prevenir ataques de gasto duplo.**

(I) Verificação da identidade e dos recursos do nodo.

(II) Utilizar prova de trabalho, onde cada nodo participante da mineração deva participar de uma competição para resolver um enigma criptográfico custoso.

(III) Resolver o problema dos Generais Bizantinos para confirmar uma transação, isto é, são necessárias mais de uma confirmação para uma bom equilíbrio de tempo gasto e confiança da validação.

Assinale a alternativa CORRETA:

(a) Apenas I está correta.
(b) Apenas I e II estão corretas.
(c) Apenas III está correta.
(d) Todas as alternativas estão corretas.
(e) Nenhuma das alternativas está correta.

(Q_8) **Em qual dos ataques apresentados o atacante visa utiliza o poder computacional de sua vítima, sem o seu consentimento, para minerar criptomoedas?**

(a) Timejacking

(b) Gasto Duplo
(c) Cryptojacking
(d) Finney
(e) Nenhuma das alternativas.

(Q_9) **Analise as afirmações a seguir.**

(i) O ataque de Spam é tipicamente utilizado para gerar atrasos na comunicação e sincronização da rede.
(ii) Com o avanço da tecnologia quântica, Blockchains e criptomoedas atuais, como Bitcoin, podem implementar novos algoritmos que resistem a ataques de computadores quânticos.
(iii) Não há como detectar e nem prevenir os ataques Sybil.
(iv) O ataque Timejacking utiliza do recurso computacional de sua vitima, sem o seu consentimento, para minerar ativamente criptomoedas.
(v) A computação quântica ameaça algumas implementação de algoritmos criptográficos utilizados em Blockchains, porém, novas implementação pós-quânticas vem oferecendo alternativas resistentes a ataques pós-quânticos.

Assinale a alternativa que contém somente afirmações VERDADEIRAS:

(a) Apenas I e IV estão corretas.
(b) Apenas I e II e V estão corretas.
(c) Apenas III e V estão corretas.
(d) Todas as alternativas estão corretas.
(e) Nenhuma das alternativas está correta.

(Q_{10}) **As contra-medidas de (a) verificação de identidade e (b) utilização de recursos de prova de trabalho são recomendadas em conjunto para mitigar qual dos ataques a seguir?**

(a) Sybil
(b) Sequestro BGP
(c) Equilíbrio
(d) Eclipse
(e) DDoS

Gabarito

(Q_1) Resposta: (d)
(Q_2) Resposta: (e)
(Q_3) Resposta: (c)
(Q_4) Resposta: (c)
(Q_5) Resposta: (c)
(Q_6) Resposta: (e)
(Q_7) Resposta: (d)
(Q_8) Resposta: (c)
(Q_9) Resposta: (b)
(Q_{10}) Resposta: (a)

Referências

Al-Jaroodi, J. and Mohamed, N. (2019). Blockchain in industries: A survey. *IEEE Access*, 7:36500–36515.

Alharby, M., Lunardi, R. C., Aldweesh, A., and van Moorsel, A. (2020). Data-driven model-based analysis of the ethereum verifier's dilemma. In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 209–220. IEEE.

Apostolaki, M., Zohar, A., and Vanbever, L. (2017). Hijacking bitcoin: Routing attacks on cryptocurrencies. In *2017 IEEE symposium on security and privacy (SP)*, pages 375–392. IEEE.

Azimy, H. and Ghorbani, A. (2019). Competitive selfish mining. In *2019 17th International Conference on Privacy, Security and Trust (PST)*, pages 1–8. IEEE.

Begum, A., Tareq, A., Sultana, M., Sohel, M., Rahman, T., and Sarwar, A. (2020). Blockchain attacks analysis and a model to solve double spending attack. *International Journal of Machine Learning and Computing*, 10(2):352–357.

Berdik, D., Otoum, S., Schmidt, N., Porter, D., and Jararweh, Y. (2021). A survey on blockchain for information systems management and security. *Information Processing & Management*, 58(1):102397.

Biryukov, A. and Pustogarov, I. (2015). Bitcoin over tor isn't a good idea.

Bonneau, J., Felten, E. W., Goldfeder, S., Kroll, J. A., and Narayanan, A. (2016). Why buy when you can rent? bribery attacks on bitcoin consensus. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.698.738&rep=rep1&type=pdf.

Braga, A. M., Marino, F. C. H., and dos Santos, R. R. (2017). Segurança de aplicações blockchain além das criptomoedas. In Nunes, R. C., Canedo, E. D., and de Sousa Júnior, R. T., editors, *Minicursos do XVII Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, volume 1 of *SBSeg 2017*, chapter 3, pages 1–50. Sociedade Brasileira de Computação. https://sol.sbc.org.br/livros/index.php/sbc/catalog/book/84.

Chen, W., Wu, J., Zheng, Z., Chen, C., and Zhou, Y. (2019). Market manipulation of bitcoin: Evidence from mining the mt. gox transaction network. In *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pages 964–972.

Chen, Y., Lu, Y., Bulysheva, L., and Kataev, M. Y. (2022). Applications of blockchain in industry 4.0: A review. *Information Systems Frontiers*, pages 1–15.

Chicarino, V. R. L. (2019). *Uma heurística para a detecção de ataques ao mecanismo de consenso por Prova de Trabalho em Blockchain*. PhD thesis, Mestrado em Computação - Programa de Pós-Graduação em Computação- Universidade Federal Fluminense.

Dalton, M. (2019). Spam attacks: Crypto strategies for surplus transactions. https://cryptobriefing.com/spam-attacks-crypto-strategies-for-surplus-transactions/.

Dasgupta, D., Shrein, J. M., and Gupta, K. D. (2019). A survey of blockchain from security perspective. *Journal of Banking and Financial Technology*, 3(1):1–17.

Deepa, N., Pham, Q.-V., Nguyen, D. C., Bhattacharya, S., Prabadevi, B., Gadekallu, T. R., Maddikunta, P. K. R., Fang, F., and Pathirana, P. N. (2022). A survey on blockchain for big data: approaches, opportunities, and future directions. *Future Generation Computer Systems*.

del Castillo, M. (2016). The DAO attacked: Code issue leads to $60 million ether theft. https://www.coindesk.com/markets/2016/06/17/the-dao-attacked-code-issue-leads-to-60-million-ether-theft/.

Douceur, J. R. (2002). The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer.

Eyal, I. (2015). The miner's dilemma. In *2015 IEEE Symposium on Security and Privacy*, pages 89–103. IEEE.

Eyal, I. and Sirer, E. G. (2014). Majority is not enough: Bitcoin mining is vulnerable. In *International conference on financial cryptography and data security*, pages 436–454. Springer.

Galvin, D. (2017). Ibm and walmart: Blockchain for food safety. *PowerPoint presentation*.

Garay, J., Kiayias, A., and Leonardos, N. (2015). The bitcoin backbone protocol: Analysis and applications. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 281–310. Springer.

Greve, F. G., Sampaio, L. S., Abijaude, J. A., Coutinho, A. C., Valcy, Í. V., and Queiroz, S. Q. (2018). Blockchain e a revolução do consenso sob demanda. *Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC)-Minicursos*.

Guo, H. and Yu, X. (2022). A survey on blockchain technology and its security. *Blockchain: Research and Applications*, 3(2):100067.

Harper, C. (2021). Nano's network flooded with spam, nodes out of sync. https://www.coindesk.com/tech/2021/03/11/nanos-network-flooded-with-spam-nodes-out-of-sync/.

Heilman, E. (2014). One weird trick to stop selfish miners: Fresh bitcoins, a solution for the honest miner. In *International Conference on Financial Cryptography and Data Security*, pages 161–162. Springer.

Heilman, E., Kendler, A., Zohar, A., and Goldberg, S. (2015). Eclipse attacks on {Bitcoin's}{peer-to-peer} network. In *24th USENIX Security Symposium (USENIX Security 15)*, pages 129–144.

Karame, G. O., Androulaki, E., and Capkun, S. (2012). Double-spending fast payments in bitcoin. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 906–917.

Kausar, F., Senan, F. M., Asif, H. M., and Raahemifar, K. (2022). 6g technology and taxonomy of attacks on blockchain technology. *Alexandria Engineering Journal*, 61(6):4295–4306.

Kent, S., Lynn, C., and Seo, K. (2000). Secure border gateway protocol (s-bgp). *IEEE Journal on Selected areas in Communications*, 18(4):582–592.

Kiayias, A. and Panagiotakos, G. (2017). On trees, chains and fast transactions in the blockchain. In *International Conference on Cryptology and Information Security in Latin America*, pages 327–351. Springer.

Kim, S. and Hahn, S.-G. (2019). Mining pool manipulation in blockchain network over evolutionary block withholding attack. *IEEE Access*, 7:144230–144244.

Li, L. and Zhou, H. (2021). A survey of blockchain with applications in maritime and shipping industry. *Information Systems and e-Business Management*, 19(3):789–807.

Li, X., Jiang, P., Chen, T., Luo, X., and Wen, Q. (2020a). A survey on the security of blockchain systems. *Future Generation Computer Systems*, 107:841–853.

Li, X., Jiang, P., Chen, T., Luo, X., and Wen, Q. (2020b). *A survey on the security of blockchain systems*, volume 107. Elsevier.

Luu, L., Chu, D.-H., Olickel, H., Saxena, P., and Hobor, A. (2016). Making smart contracts smarter. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, pages 254–269.

Luu, L., Saha, R., Parameshwaran, I., Saxena, P., and Hobor, A. (2015a). On power splitting games in distributed computation: The case of bitcoin pooled mining. In *2015 IEEE 28th Computer Security Foundations Symposium*, pages 397–411. IEEE.

Luu, L., Teutsch, J., Kulkarni, R., and Saxena, P. (2015b). Demystifying incentives in the consensus computer. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 706–719.

Mayer, H. (2016). Ecdsa security in bitcoin and ethereum: a research survey. *CoinFaabrik, June*, 28:126.

Melo, K. (2018). Cryptojacking: O que é? como se proteger? https://minutodaseguranca.blog.br/cryptojacking-o-que-e-como-se-proteger/.

Miers, C. C., Koslovski, G. P., Pillon, M. A., Jr., M. A. S., Carvalho, T. C. M. B., Rodrigues, B. B., and ao H. F. Battisti, J. (2019). Análise de mecanismos para consenso distribuído aplicados a blockchain. In Henriques, M. A. A., Terada, R., and Batista, D. M., editors, *Minicursos do XIX Simpósio Brasileiro de Segurança da Informação e de Sistemas Computacionais*, pages 91–139. SBC.

Monrat, A. A., Schelén, O., and Andersson, K. (2019). A survey of blockchain from the perspectives of applications, challenges, and opportunities. *IEEE Access*, 7:117134–117151.

Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. *Decentralized Business Review*, page 21260.

Narayanan, A., Bonneau, J., Felten, E., Miller, A., and Goldfeder, S. (2016). *Bitcoin and cryptocurrency technologies: a comprehensive introduction*. Princeton University Press.

Natoli, C. and Gramoli, V. (2016). The balance attack against proof-of-work blockchains: The R3 testbed as an example. *CoRR*, abs/1612.09426.

Nogueira, N. (2021). Criptomoeda Nano é alvo de ataques de spam e sofre com lentidão na rede. https://portaldobitcoin.uol.com.br/criptomoeda-nano-e-alvo-de-ataques-de-spam-e-sofre-com-lentidao-na-rede/.

Paavolainen, S., Elo, T., and Nikander, P. (2018). Risks from spam attacks on blockchains for intemet-of-things devices. In *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 314–320.

Pal, A. and Kant, K. (2019). Using blockchain for provenance and traceability in internet of things-integrated food logistics. *Computer*, 52(12):94–98.

Parker, L. (2017). Bitcoin spam attack stressed network for at least 18 months, claims software developer. https://bravenewcoin.com/insights/bitcoin-spam-attack-stressed-network-for-at-least-18-months-claims-software-developer.

Peck, M. E. (2016). Ethereum's $150-million blockchain-powered fund opens just as researchers call for a halt. *IEEE Spectrum. Institute of Electrical and Electronics Engineers,(28 May 2016)*.

Pham, H. L., Tran, T. H., Phan, T. D., Le, V. T. D., Lam, D. K., and Nakashima, Y. (2020). Double SHA-256 hardware architecture with compact message expander for bitcoin mining. *IEEE Access*, 8:139634–139646.

Pilkington, M. (2016). Blockchain technology: principles and applications. In *Research handbook on digital transformations*. Edward Elgar Publishing.

Qin, R., Yuan, Y., and Wang, F.-Y. (2020). Optimal block withholding strategies for blockchain mining pools. *IEEE Transactions on Computational Social Systems*, 7(3):709–717.

Rekhter, Y., Li, T., and Hares, S. (2006). A border gateway protocol 4 (bgp-4). Technical report, IETF.

Russell, J. (2017). A major vulnerability has frozen hundreds of millions of dollars of ethereum. https://techcrunch.com/2017/11/07/a-major-vulnerability-has-frozen-hundreds-of-millions-of-dollars-of-ethereum/.

Sankar, L. S., Sindhu, M., and Sethumadhavan, M. (2017). Survey of consensus protocols on blockchain applications. In *2017 4th international conference on advanced computing and communication systems (ICACCS)*, pages 1–5. IEEE.

Sayeed, S. and Marco-Gisbert, H. (2019). Assessing blockchain consensus and security mechanisms against the 51% attack. *Applied sciences*, 9(9):1788.

Sayeed, S., Marco-Gisbert, H., and Caira, T. (2020). Smart contract: Attacks and protections. *IEEE Access*, 8:24416–24427.

Shalini, S. and Santhi, H. (2019). A survey on various attacks in bitcoin and cryptocurrency. In *2019 International Conference on Communication and Signal Processing (ICCSP)*, pages 0220–0224. IEEE.

Shrestha, R. and Nam, S. Y. (2019). Regional blockchain for vehicular networks to prevent 51% attacks. *IEEE Access*, 7:95033–95045.

Silva, G. and Rodrigues, C. K. d. S. (2016). Mineração individual de bitcoins e litecoins no mundo. *Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2016), Niterói, Rio de Janeiro, Brasil*.

Solat, S. and Potop-Butucaru, M. (2017). Brief announcement: Zeroblock: Timestamp-free prevention of block-withholding attack in bitcoin. In *International Symposium on Stabilization, Safety, and Security of Distributed Systems*, pages 356–360. Springer.

Taskinsoy, J. (2019). Bitcoin and turkey: A good match or a perfect storm? *Available at SSRN 3477849*.

Tosh, D. K., Shetty, S., Liang, X., Kamhoua, C. A., Kwiat, K. A., and Njilla, L. (2017). Security implications of blockchain cloud with analysis of block withholding attack. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 458–467. IEEE.

Toyoda, K., Mathiopoulos, P. T., Sasase, I., and Ohtsuki, T. (2017). A novel blockchain-based product ownership management system (poms) for anti-counterfeits in the post supply chain. *IEEE Access*, 5:17465–17477.

Wang, Y., Wang, Z., Zhao, M., Han, X., Zhou, H., Wang, X., and Koe, A. S. V. (2022). Bsm-ether: Bribery selfish mining in blockchain-based healthcare systems. *Information Sciences*, 601:1–17.

Wright, C. S. (2008). Bitcoin: a peer-to-peer electronic cash system. *Available at SSRN 3440802*.

Xiao, Y., Zhang, N., Lou, W., and Hou, Y. T. (2020). A survey of distributed consensus protocols for blockchain networks. *IEEE Communications Surveys & Tutorials*, 22(2):1432–1465.

Yan, H., Oliveira, R., Burnett, K., Matthews, D., Zhang, L., and Massey, D. (2009). BGPmon: A real-time, scalable, extensible monitoring system. In *2009 Cybersecurity Applications & Technology Conference for Homeland Security*, pages 212–223. IEEE.

Yang, X., Chen, Y., and Chen, X. (2019). Effective scheme against 51% attack on proof-of-work blockchain with history weighted information. In *2019 IEEE International Conference on Blockchain (Blockchain)*, pages 261–265. IEEE.

Ye, C., Li, G., Cai, H., Gu, Y., and Fukuda, A. (2018). Analysis of security in blockchain: Case study in 51%-attack detecting. In *2018 5th International conference on dependable systems and their applications (DSA)*, pages 15–24. IEEE.

Zhang, R. and Preneel, B. (2017). Publish or perish: A backward-compatible defense against selfish mining in bitcoin. In *Cryptographers' Track at the RSA Conference*, pages 277–292. Springer.

Zhao, X., Chen, Z., Chen, X., Wang, Y., and Tang, C. (2017). The dao attack paradoxes in propositional logic. In *2017 4th International Conference on Systems and Informatics (ICSAI)*, pages 1743–1746.

Zheng, Z., Xie, S., Dai, H.-N., Chen, X., and Wang, H. (2018). Blockchain challenges and opportunities: A survey. *International journal of web and grid services*, 14(4):352–375.

Capítulo

5

Detecção de Anomalias em Redes de Computadores

Raul Ceretta Nunes (UFSM), Márcia Henke (UFSM)

Resumo. *Detecção de anomalias é um tópico que tem sido estudado em diversas áreas de pesquisa e domínios de aplicação. Na área de segurança computacional sistemas de detecção de intrusão costumam apoiar atividades de centros de operações de segurança cibernética na identificação de ameaças. A automatização do processo de detecção pode ser orientada a detecção de assinaturas de ataques e/ou a detecção de anomalias. Diferentes abordagens e técnicas tem sido propostas para observar novos ataques em redes computacionais. Esse capítulo apresenta as principais abordagens e técnicas na área de detecção de anomalias no nível de redes com objetivo de prover uma compreensão básica sobre o assunto.*

5.1. Introdução

Os três principais princípios da segurança da informação são: confidencialidade, integridade e disponibilidade. O princípio da confidencialidade visa limitar o acesso não autorizado à informação, enquanto o princípio da integridade tende a garantir a fidedignidade e correção da informação. Já o princípio da disponibilidade aspira garantir que a informação esteja sempre disponível e acessível. Como no meio digital o acesso à informação se dá pelas redes de computadores, um dos meios de garantir a segurança da informação é atuar no monitoramento de sistemas e na segurança em redes. Este capítulo aborda como isto tem sido feito na área de detecção de anomalias em redes de computadores.

A proteção de sistemas e redes de computadores é um campo amplo e complexo e compreende uma das principais atividades de Centros de Operações de Segurança Cibernética (*Cyber Security Operations Center* - CSOC) e de Grupos de Resposta a Incidentes de Segurança (*Computer Security Incident Response Team* – CSIRT). A abordagem convencional para proteger os sistemas é projetar e configurar mecanismos de segurança, como firewalls, mecanismos de autenticação, redes privadas pessoais (*Virtual Private Network* – VPN), os quais procuram criar um "escudo" protetor, o que também é conhecido por perímetro de segurança [Moraes 2010]. No entanto, tais mecanismos de segurança quase sempre têm vulnerabilidades e geralmente não são suficientes para

garantir a segurança completa da infraestrutura e para evitar ataques que estão sendo continuamente adaptados para explorar as fraquezas do sistema, estas muitas vezes causadas por projetos descuidados e falhas de implementação. Esse cenário de insegurança gera muitos incidentes e incrementa a necessidade por tecnologias de segurança que possam monitorar sistemas e identificar ataques cibernéticos. O desafio é a diversidade de incidentes oriundos de uma diversidade de ataques cibernéticos. A detecção e resposta a esses ataques é crucial para que se possa manter a funcionalidade dos sistemas protegidos.

Sistemas de Detecção de Intrusão (*Intrusion Detection System* – IDS) têm por finalidade identificar ações oriundas de atacantes que possam ameaçar redes de computadores ou sistemas computacionais. Em seu cerne, algoritmos de detecção aplicam métodos que possibilitam a análise de dados, a fim de identificar características que indiquem perfis típicos de ataques. No contexto da detecção de intrusões, um dos grandes desafios dos projetistas consiste na escolha do algoritmo mais promissor, ou dos algoritmos mais promissores, para cada um dos múltiplos tipos de ataques.

A diversidade de tipos de ataque conduz a uma diversidade de tipos de dados e a variações nos processos de automatização da detecção. Os tipos de dados que podem ser analisados compreendem dados dos *hosts* e/ou das redes que conectam *hosts* e a análise pode ser orientada a detecção de assinaturas de ataques e/ou a detecção de anomalias. Logo, diferentes abordagens e técnicas tem sido adotadas para detectar ataques e intrusões em redes computacionais. A literatura relacionada à detecção tanto de assinaturas quanto de anomalias no tráfego de redes geralmente é classificada na categoria "detecção de intrusão" e as técnicas e métodos são implantados em IDS.

Esse capítulo apresenta as principais abordagens e técnicas de detecção de anomalias com objetivo de prover uma compreensão básica sobre a área quando se está interessado na análise em nível de redes, ou seja, a análise de elementos indicativos contidos no tráfego de redes.

O capítulo está organizado como segue. A Seção 5.2 apresenta a classificação de IDS e destaca onde a detecção de anomalia está no contexto de um IDS. A Seção 5.3 descreve as principais abordagens utilizadas para detecção de anomalias. A Seção 5.4 destaca que, na prática, a escolha de características (parâmetros do tráfego de rede) é um fator chave na análise de anomalias. A Seção 5.5 demonstra que o estudo e desenvolvimento de técnicas de detecção de anomalias pode ser apoiado por diferentes tecnologias, tal como ferramentas e bases de dados, e que estão disponíveis aos interessados pela área. A Seção 5.6 indica as métricas mais utilizadas para avaliar o desempenho das soluções de detecção de anomalias. Finalmente, a Seção 5.7 encerra o capítulo reforçando os principais elementos abordados.

5.2. Sistemas de Detecção de Intrusão e a Detecção de Anomalias

Um IDS é uma ferramenta de software que fornece apoio aos profissionais da área de segurança computacional, dos quais são esperadas ações apropriadas para mitigar problemas de segurança tal como indisponibilidade de serviços ou comprometimento de informações. Do ponto de vista arquitetural, um IDS consiste de cinco componentes chave: coletores de dados, base de conhecimento, detector, componente de configuração e componente de resposta. A Figura 5.1 ilustra a correlação desses componentes.

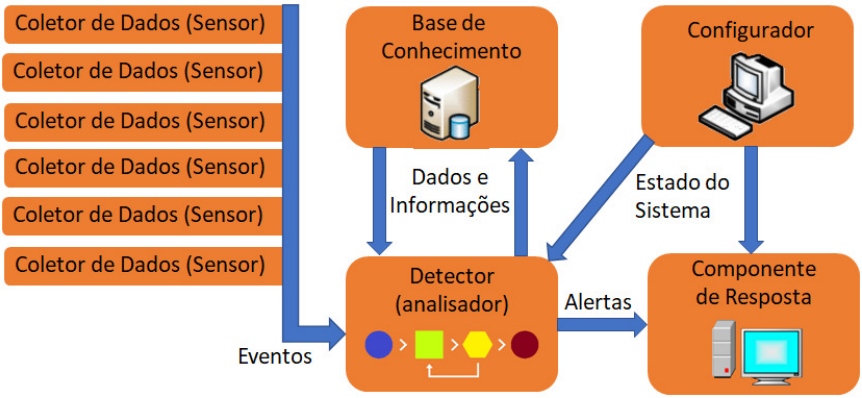

Figura 5.1. Componentes de um sistema de detecção de intrusão.

Num IDS, a coleta de dados é realizada usando sensores. Em redes de computadores, estes sensores correspondem a softwares escaneadores (*sniffers*) do tráfego da rede. A base de conhecimento contém regras e informações relevantes para a identificação de intrusões. O detector é o componente que congrega os algoritmos responsáveis por identificar intrusões/ataques usando os dados armazenados na base de conhecimento e os obtidos dos sensores. O componente de configuração provê o estado corrente do IDS e meios de configurá-lo e o componente de resposta é responsável por iniciar ações apropriadas baseado nas intrusões detectadas.

A classificação de um IDS depende de três fatores: da fonte de dados/informações (baseado no *host*, baseada na rede ou híbrida), da arquitetura de sua rede (centralizada ou distribuída) e da técnica de detecção utilizada (baseada em assinaturas, baseada em anomalias ou híbrida).

A seguir é apresentada a classificação dos IDS quando considerado a abordagem utilizada pela fonte de informações (Seção 5.2.1) e quando considerado a abordagem de análise utilizada no componente Detector (Seção 5.2.2).

5.2.1. IDS baseado em rede e IDS baseado em *host*

De acordo com a abordagem de coleta de dados adotada (fonte de informações), os IDS são classificados como baseados em rede ou baseados em *host*, ou híbridos. Os IDS baseados rede coletam e analisam informações sobre pacotes de dados que trafegam pela rede, tal como volume de tráfego, número de conexões, fluxos, pacotes perdidos, dentre outros. Por outro lado, os IDS baseados em *host* analisam dados coletados nos computadores que hospedam um serviço, tal como uso de memória, processamento, disco, de processos, variáveis de ambiente, dentre outros. A abordagem híbrida se refere a soluções que procuram agregar verificações de tráfego à monitoração de atividades no sistema.

Pela característica das fontes de informações, os IDS baseados em rede são voltados à observação de ações externas sobre a rede de computadores monitorada, tal como um ataque de negação de serviço, e os IDS baseados em *host* são vocacionados à observa-

ção de ações internas, tal como software malicioso atuando de dentro da rede monitorada.

Em qualquer dos casos, os dados que são capturados para análise tendem a ser sensíveis, ou seja, podem apresentar restrições quanto a segurança e privacidade. Por isso, a adoção de dados anonimizados sempre deve ser a preferida para análise, por exemplo, em detecção de *spam*, onde envolve muitas informações dos usuários de e-mails.

5.2.2. IDS baseado em assinaturas e IDS baseado em anomalias

De acordo com a abordagem de análise de dados os IDS são classificados como baseados em assinaturas e baseado em anomalias.

Um IDS baseado em assinatura opera sobre uma base de conhecimento que contém um dicionário predefinido de padrões de ataque. Nesse IDS o componente detector verifica se algum padrão detectado no tráfego de rede corresponde a um ou mais dos padrões de ataque predefinidos. Caso haja a identificação de um padrão da base, a ocorrência de uma intrusão/ataque é sinalizada. Os IDS baseados em assinaturas fornecem bons resultados de detecção para ataques previamente conhecidos (na base de conhecimento), ou seja, resultam em baixos índices de falsos positivos (indicações de ataque equivocadas). No entanto, não são capazes de detectar ataques ainda desconhecidos [Garcia-Teodoro et al. 2009], conhecidos como ataques do dia zero (ataques novos ou nunca observados).

Um IDS baseado em anomalia opera sobre uma base de conhecimento que contém informações capazes de permitir ao componente detector verificar se os dados coletados correspondem a um padrão de comportamento normal dos usuários, *hosts* e redes. Caso haja a identificação de um comportamento não usual, desconhecido, a ocorrência de uma intrusão/ataque é sinalizada. Note que diferente de um detector baseado em assinatura, esse tipo de IDS pode ser menos preciso, dado que pode resultar em muitos falsos positivos. Porém, uma grande vantagem desse tipo de IDS é poder detectar ataques de dia zero.

Atualmente, o avanço das técnicas de aprendizagem de máquina tem possibilitado melhorias significativas para o funcionamento dos IDS baseados em anomalias, proporcionando resultados com qualidade cada vez maior [Liang et al. 2017] [Smitha et al. 2019] [Althubiti et al. 2018]. Pode-se, novamente, exemplificar com a detecção de mensagens spam, onde um algoritmo de aprendizagem de máquina é treinado a partir de características (palavras) que compõem uma mensagem, seja do assunto do e-mail, corpo do e-mail ou informações do cabeçalho da mensagem, minimizando os falsos positivos.

5.3. Abordagens de Detecção de Anomalias

Uma característica chave para a compreensão das abordagens de detecção de anomalias no contexto de redes de computadores é compreender que a Internet é uma das poucas plataformas operacionais existentes que não possui centros de controle. Tal característica serve tanto como fator de sucesso (ampliação permanente e liberdade para desenvolvimentos) quanto de insucesso (suscetibilidade a novas vulnerabilidades e ponto de falhas). Este contexto leva a uma inevitável situação de ocorrência de intrusão (invasão ou tentativa de), o que amplia a necessidade de distinção entre comportamentos considerados normais e comportamentos considerados maliciosos, ou anormais. Logo, na tentativa de lidar com esse lado negativo da Internet, várias soluções de detecção de anomalias têm sido desenvolvidas ao longo dos anos. Esta seção apresenta algumas dessas soluções.

5.3.1. Aprendizagem de Máquina

O aprendizado de máquina, uma subárea da inteligência artificial, corresponde a algoritmos e métodos com capacidade de aprender de maneira automatizada o comportamento de dados observados. As abordagens mais conhecidas de aprendizado de máquina são: aprendizado supervisionado, não supervisionado e por reforço.

No aprendizado supervisionado o aprendizado de um modelo comportamental é realizado com base em dados previamente conhecidos, chamados dados rotulados. Através dos dados rotulados (entradas conhecidas) os algoritmos (modelos comportamentais) podem ser testados e ajustados para obterem os melhores resultados (saídas esperadas). Uma vez ajustado o modelo (treinado para a relação entre entrada e saída), dado uma nova entrada pode ser inferida a saída. Exemplos de métodos de aprendizagem supervisionada: baseados em separabilidade (entropia), como árvores de decisão (*decision tree*) e florestas randômicas (*random forest*), e baseados em particionamento, como SVM (*support vector machines*) e KNN (*K nearest neighbor*) .

No aprendizado não supervisionado o aprendizado de um modelo comportamental é realizado com base em dados não conhecidos, chamados dados não rotulados. Os algoritmos devem descobrir por si só, por meio da exploração dos dados (análises de similaridades ou dissimilaridades), os possíveis relacionamentos entre eles, o que permite identificar grupos de interesse. Note que dependendo dos dados de entrada (também chamados de características ou *features*), o método pode apresentar diferentes resultados e precisões. Exemplos de métodos de aprendizagem não supervisionada: baseados em clusterização, tal como K-Means, e baseados em associação, tal como Apriori.

No aprendizado por reforço o modelo comportamental é obtido por um método interativo de "tentativa e erro" e avaliação dos resultados. A cada interação, o método recebe como entrada uma indicação do estado atual do ambiente, escolhe uma ação a ser tomada e produz uma saída. Quando o modelo acerta, recebe uma recompensa, do contrário uma penalidade. A análise do resultado permite que o modelo ajuste sua ação na próxima interação, reforçando qualquer sucesso ou penalizando qualquer resultado negativo. O objetivo é reconhecer o conjunto de ações que potencializam os acertos. Logo, trata-se de uma aprendizagem a partir de um mapeamento de entrada-saída alcançada através de interação continuada com o ambiente para minimizar um dado objetivo de desempenho [Haykin, 2008].

A seguir, estão descritas as principais técnicas de aprendizagem de máquina utilizadas para detecção de anomalias no nível de redes.

5.3.1.1. Técnicas de Aprendizado Supervisionado

Há diferentes técnicas de aprendizagem de máquina supervisionada sendo aplicadas no contexto de detecção de anomalias em redes de computadores, dentre as quais Árvores de Decisão (*Decision Tree* - DT), Máquinas de Vetores de Suporte (*Support Vector Machine* - SVM) e Redes Neurais (*Neural Networks*) [Eltanbouly et al. 2020].

Os métodos de detecção de anomalias supervisionados costumam empregar conjuntos de dados conhecidos para classes de interesse, tal como normais, ataques, des-

conhecidos. Por exemplo, com base em um conjunto de dados considerados normais (exemplos conhecidos) busca-se identificar o modelo comportamental que dado uma entrada normal indique que a entrada é da classe normal. A dificuldade consiste em obter um modelo que consiga inferir corretamente a classe dada uma entrada não conhecida.

Árvore de Decisão: uma árvore de decisão é composta por nós e folhas, onde os nós representam um teste sobre uma das entradas e as folhas representam o rótulo da classe. A ideia chave do algoritmo de árvore de decisão é particionar recursivamente um conjunto de treinamento (raiz da árvore) até que cada subconjunto (nó) contenha entradas de uma única classe (nó de decisão). O algoritmo consiste em construir uma árvore que permita mapear as entradas (conjunto de treinamento) para as classes corretas, ou seja, que represente o modelo comportamental. No algoritmo, inicialmente todas as instâncias de treinamento são atribuídas à raiz do nó, ou seja, a um único espaço de decisão. Na prática, o algoritmo funciona como segue.

Para cada amostra no conjunto de dados, divida o espaço de decisão em regiões de decisão para cada valor de amostra e calcule uma função de custo (por exemplo, ganho de informação) para cada divisão realizada. Então identifique o recurso e o valor do recurso correspondente que leva à melhor divisão (por exemplo, para classificação: ganho máximo de informação, para regressão: soma residual mínima dos quadrados). Essa combinação de valor de recurso constitui a condição de divisão. Particione todas as instâncias em regiões de decisão com base na condição de divisão. Para cada região de decisão, continue este processo até que uma condição de parada seja alcançada. Esta condição de parada sinalizará a classificação da amostra.

Florestas Randômicas (*Random Forest*): muito empregada em problemas de classificação, regressão e outras tarefas, esta técnica utiliza um conjunto (*ensemble*) de classificadores de árvores de decisão e é considerada como pertencente a abordagem de aprendizagem em conjunto (*ensemble learning*) [Primartha and Tama 2017]. Cada árvore cresce de acordo com um vetor aleatório, independente e identicamente distribuído nas árvores. Cada árvore do conjunto gera um voto para a classe mais popular do vetor de entrada. A diversidade da floresta pode ser obtida tomando amostra de um conjunto de atributos, do conjunto de dados, ou apenas alterando arbitrariamente alguns parâmetros da árvore de decisão. Existem dois parâmetros de floresta aleatória que podem ser ajustados: o número de variáveis a serem escolhidas em cada nó, e que é comumente fixado em todos os nós, e o número de árvores que constroem a floresta. Esta técnica pode criar padrões com muita eficiência sobre *datasets* balanceados e ter uma ótima performance no que diz respeito a classificação de padrões [Prashanth et al. 2008].

Máquinas de Vetores de Suporte (SVM): É uma técnica de classificação amplamente aplicada em problemas de segurança de redes tais como detecção de phishing [Miyamoto *et al*. 2009] e detecção de intrusos [Xiao, H. *et al*. 2007]. Basicamente, o funcionamento de SVM pode ser descrito da seguinte forma: dadas duas classes e um conjunto de instâncias de treinamento cujas amostras pertencem a essas classes, a SVM constrói um hiperplano que divide o espaço de características em duas regiões, maximizando a margem de separação entre as mesmas. Esse hiperplano é conhecido como hiperplano de separação ótimas. As amostras desconhecidas (exemplos de teste) são então mapeadas para esse mesmo espaço, e atribuídas a uma das classes [Alpaydim, E.

2010]. A principal vantagem de SVM é a baixa probabilidade de erros de generalização. Quanto à utilização de SVM para detecção de intrusão, por exemplo, há duas vantagens principais. A primeira está relacionada à rapidez do algoritmo em fase de uso, uma vez que o desempenho em tempo real é de primordial importância para esse tipo de aplicação. A segunda razão é a escalabilidade, pois SVM é relativamente insensível ao número de pontos de dados. Dessa forma, a taxa de precisão na classificação não depende diretamente da dimensão do espaço de características [Mukkamala, S. *et al*. 2002]. Entretanto, a SVM tem como desvantagem a demanda por elevada complexidade computacional na fase de treinamento, fato que pode inviabilizar o uso de SVM em problemas que necessitem de treinamento durante a operação do sistema (*necessidade de reajuste periódico dos parâmetros do modelo de classificação*).

K Vizinhos Mais Próximos (KNN - K *Nearest Neighbor*): o princípio básico desta técnica de detecção é classificar uma amostra de dados desconhecida (entrada) em uma das classes já conhecidas, por exemplo, classes Normal e Anormal. A técnica consiste num classificador que independe de parâmetros para sua operação, ou seja, a classificação é realizada apenas pela medida de distância entre os vizinhos, o que pode ser obtido através da métrica de distância Euclidiana. Nesta técnica, o k corresponde ao número de vizinhos próximos usados para realizar a classificação do dado desconhecido, logo ele influencia na precisão da classificação. Por exemplo, se considerado um $k = 1$, pode haver o dado desconhecido pode estar próximo a um dado Normal e ser classificado como Normal. Porém, se considerado um $k = 2$, pode ser observado que o dado desconhecido é mais próximo de dois Anormais do que de dois Normais, logo é classificado como Anormal. Assim, o número ideal para k deve ser obtido através de testes com dados conhecidos. A precisão da técnica é dependente da definição do k.

Redes Neurais: correspondem a técnicas computacionais que procuram representar um modelo matemático inspirando-se na estrutura neural de seres inteligentes e é uma abordagem muito popular para trabalhar com classificação, assim como o que ocorre na detecção de anomalias. Por ser uma técnica de aprendizagem de máquina baseada em camadas de aprendizagem, esta técnica pertence a classe de técnicas de aprendizado profundo (*deep learning*). Diferente das técnicas tradicionais de aprendizado supervisionado, com árvores de decisão, as redes neurais podem lidar com conjuntos de dados não lineares e não balanceados [Kwon et al. 2018]. O funcionamento básico de uma rede neural visa adquirir conhecimento através da experiência, ou seja, gerar predições usando uma coleção de nodos interconectados (neurônios) que são organizados em camadas (de entrada, de saída e ocultas). Os neurônios da rede criam uma soma ponderada a partir da entrada que recebem e então transformam essa soma ponderada usando algum tipo de função não linear, tal como *logit*, tangente hiperbólica ou função linear retificada [Eck 2018]. Uma vez que a informação chega à camada de saída, ela é coletada e convertida em previsões. Dependendo da complexidade dos dados para os quais as previsões são desejadas, as redes neurais podem ter várias camadas ocultas e as funções usadas dentro dos neurônios podem variar. Algumas redes neurais de aprendizado supervisionado são da classe *Convolutional Neural Network*, que processa dados bi-dimensionais com informação espacial numa única direção da entrada para a saída. Outra classe de redes neurais são as recorrentes (*Recurrent Neural Network*), que usa laços de realimentação das saídas para as entradas das camadas (*feedback loops*) para aprender no processamento de dados

sequenciais com informação de tempo.

5.3.1.2. Técnicas de Aprendizado Não-Supervisionado

No aprendizado não-supervisionado não há presença de um professor, ou seja, não existe instâncias ou exemplos rotulados. O algoritmo de aprendizagem de máquina aprende a representar, geralmente agrupando, as entradas submetidas de acordo com uma medida de qualidade. Essas técnicas são utilizadas principalmente quando o objetivo for encontrar padrões ou tendências que auxiliem no entendimento dos dados, desta forma detectam valores discrepantes apenas com base nas propriedades intrínsecas das instâncias de dados. O principal objetivo na aplicação de técnicas com abordagem de aprendizado não supervisionado para detecção de anomalias é usar uma rotulagem automática de amostras de dados, ainda não rotuladas. Este objetivo é motivado pelo fato de os dados rotulados serem muito difíceis de se obter. As técnicas mais empregadas para aprendizado Não-Supervisionado são clusterização e regras de associação, descritas na sequencia. [Patterson and Gibson 2017].

Clusterização, ou agrupamento de dados, é um método capaz de descrever objetos em grupos (ou *clusters*), de maneira que objetos de um mesmo grupo sejam mais semelhantes entre si do que objetos de grupos distintos. Seu funcionamento está baseado na estimativa de densidade, ou seja, na estimativa da função de densidade de probabilidade que melhor descreve a variável aleatória de interesse. A hipótese básica do método quando aplicado na detecção de anomalias é que os dados anômalos ou de ataque formam uma pequena porcentagem do total dos dados e, com base nesta hipótese, anomalias e ataques podem ser detectados com base no tamanho dos *clusters*. *Clusters* grandes correspondem a dados normais e o restante dos pontos de dados, fora de *clusters*, são classificados como *outliers*, ou seja, anomalias ou ataques [Syarif et al. 2012]. Para exemplificar, considere a seguinte algoritmo de clusterização para detecção de anomalia no tráfego de rede. Inicie com um conjunto vazio de *clusters* e gere *clusters* com uma única passagem pelo conjunto de dados. Calcule a distância entre cada nova instância (novo dado) e seu centroide (ponto central do agrupamento). O *cluster* com a menor distância é selecionado, e se essa distância for menor ou igual a largura do *cluster*, a instância é atribuída a esse *cluster*, caso contrário, cria-se um novo *cluster* com esta instância como centroide. Note que ao final, as instâncias que não pertencerem a algum *cluster* poderão ser consideradas anomalias. Um algoritmo de clusterização clássico é o *K-Means*.

Regras de Associação é uma implicação da forma X -> Y, onde, lê-se X implica em Y, desde que X e Y sejam subconjuntos da base de dados analisada e os conjuntos de itens não tenham interceptação. No aprendizado não supervisionado por regras de associação, existem duas medidas que são calculadas: a confiança e o suporte. A confiança é a probabilidade condicional, P(Y|X), que é o que normalmente é calculado. Para poder dizer que a regra vale com suficiente confiança, esse valor deve ser próximo de 1 e significativamente maior que P(Y), a probabilidade geral de as pessoas comprarem Y. Há também o interesse em maximizar o suporte da regra, pois mesmo que haja uma dependência com um valor de confiança forte, se o número desses clientes for pequeno, a regra não vale nada. O suporte mostra a significância estatística da regra, enquanto a confiança mostra a força da regra [Alpaydin 2014].

5.3.2. Outras Abordagens

Além das técnicas de aprendizagem de máquina, que dominam as proposições de detecção identificando *outliers* ou anomalias, são também exploradas outras técnicas para identificar e barrar o que é considerado anômalo. Esta seção apresenta as ideias por traz da técnica de filtragem, anti-* e combinação de classificadores.

Filtragem

Dentre as abordagens para mitigar as atividades maliciosas, tem-se os filtros ou *firewalls*. Em redes de computadores, estes filtros geralmente são instalados entre a rede interna de uma organização e a Internet e estabelecem uma ligação controlada, ou seja, constroem o que é chamado de perímetro de segurança ou muro de segurança externa. Desta forma, os filtros protegem a rede organizacional contra atividades maliciosas provenientes da Internet, estabelecendo um único ponto de entrada e saída como maneira de impor segurança. Basicamente, *firewalls* utilizam regras (filtros) que definem o que deve ser feito. Desta forma, todo o tráfego que entra ou sai da rede, ou máquina, é comparado com as regras e o resultado é uma ação, geralmente permitir ou negar o tráfego. Como exemplos dessas regras, pode-se considerar restrições à velocidade de navegação, *download* ou *upload* de arquivos.

Os tipos de *firewalls* mais empregados são os com filtragem de pacotes, filtragem de estados e filtragem de aplicação (*gateway* de aplicação). A filtragem baseada em pacotes emprega um conjunto de regras que pode negar ou deixar um pacote passar, baseando-se nas informações contidas em cabeçalhos da camada de rede ou de transporte (endereço IP de origem e destino, endereço de porta de origem e destino e o tipo de protocolo), auxiliado por uma tabela de filtragem para escolher quais pacotes devem ser descartados. Já a filtragem de estados, baseia-se nos estados das conexões, verificando somente o primeiro pacote de cada conexão, sendo este pacote aceito, os demais pacotes são filtrados de acordo com as informações da conexão na tabela de estados. Já a filtragem de aplicação, trata a camada de aplicação, abstrai portas e protocolos e foca no conteúdo dos pacotes a procura de indícios de anomalias como, por exemplo, sequências de caracteres específicos (palavras ou frases) que indiquem a presença de ataques, código maliciosos de aplicações como *BitTorrent, Dropbox, WhatsApp*, DNS.

Contudo, a filtragem apresenta deficiências para limitar o tráfego malicioso. Um ponto a considerar é a dependência de verificações e configurações manuais que em muitas situações fazem com que dados legítimos acabem sendo filtrados. Outro ponto é que a abordagem de filtro se apresenta mais eficiente em aplicações específicas, como filtragem de *proxies* e *gateways* de aplicação, que avaliam um fluxo particular de determinada aplicação, ou aplicações. Note que para cada nova aplicação que se aplica a técnica de filtragem, uma configuração específica pode ser necessária.

Software anti-*

Os softwares anti-* são outra solução para tratamento de anomalias. Esses softwares, além de utilizar de banco de dados com atividades maliciosas ou anomalias já conhecidas e a partir desse conhecimento comparam ações no ambiente de monitoramento, também, utilizam-se de algoritmos, geralmente estatísticos, para identificar esse tipo de atividade maliciosa. Conhecidos como sistemas de detecção de intrusão baseados em as-

sinaturas. Alguns exemplos desses softwares são: antivírus, antispyware, antiphishing e antispam. Porém, para bom desempenho desses softwares são necessárias atualizações de forma automatizada, pois novas assinaturas de atividades maliciosas surgem diariamente.

Como indicado no primeiro livro da série Unihacker [Amaral et al. 2021], os softwares anti-* são bem conhecidos. Os antivírus são softwares que tem como objetivo identificar e remover vírus de um computador. Os antispywares abordam o combate a programas e códigos espiões como *spyware*, *adware*, *keyloggers*. Softwares antiphishing combatem atividades fraudulentas que são emitidas através de sites *Web* ou através de e-mail que servem de vetor para esse tipo de atividade maliciosa.

Geralmente esse tipo de abordagem é integrada nos navegadores *Web* ou embutido em correios eletrônicos. Os softwares antispam se baseiam na filtragem de mensagens, que não foram solicitadas, usando os campos do cabeçalho ou do conteúdo da mensagem. A partir desses campos é possível verificar o endereço de origem, nome do remetente e o assunto de uma mensagem e, desta forma, validar como uma mensagem solicitada ou uma mensagem não solicitada. A desvantagem com esse tipo de solução é a dependência em relação aos erros de configuração, pois é preciso definir regras do que se deseja ou não receber e isso implica definir de quais são os remetentes, endereços e assuntos que consideram válidos. Isso implica nas famosas listas negras (*blacklists*), listas brancas (*whitelists*) e listas cinza (*greylists*) que são empregadas na filtragem de cabeçalho. Também, pode-se filtrar a partir do conteúdo da mensagem, que busca por palavras chaves. Sendo configurado corretamente, esse filtro torna-se bastante eficiente. O filtro baseado no conteúdo não considera a origem das mensagens. Atualmente, muitos dos filtros de mensagens se utilizam do mecanismo de aprendizagem de máquina, como por exemplo, e-mails *web*. Observa-se que a abordagem baseada em filtragem é interessante, mas possui limitadores e estratégias baseadas na exploração de técnicas de aprendizagem de máquina.

Combinação de Classificadores

A combinação de classificadores tem como objetivo minimizar a desafiadora tarefa de um único classificador resolver problemas específicos, pois através da combinação da decisão de classificadores menos específicos, busca-se obter uma visão mais abrangente para o problema a ser solucionado. Por exemplo, para o problema de detecção de spam foi possível reduzir a taxa de falsos positivos com a abordagem de conjunto de classificadores para uma classificação binária [Lutz et al. 2018]. Existem diversas razões para a combinação de múltiplos classificadores, algumas dessas razões são encontradas em [Jain et al., 2000] e listadas a seguir.

1. Um desenvolvedor tem acesso a diferentes classificadores, treinados sem um contexto diferente e para uma representação ou descrição inteiramente diferente do mesmo problema;
2. Muitas vezes, mais de um conjunto de treino é disponível, coletados em um tempo diferente ou em um ambiente diferente. Esses conjuntos de treinamento podem ainda usar características diferentes;
3. Diferentes classificadores treinados sobre o mesmo dado não diferem somente em seu desempenho global. Possivelmente, os classificadores apresentam fortes diferenças locais. Cada classificador tem sua própria região no espaço de caracte-

rísticas na qual atua com um desempenho melhor;
4. Alguns classificadores, tais como redes neurais, mostram diferentes resultados com diferentes inicializações dos parâmetros devido à aleatoriedade inerente ao procedimento de treino. Nesse tipo de situação, normalmente é feita uma escolha da melhor rede e descarte das demais. Porém, é possível combinar várias redes, na tentativa de fortalecer a aprendizagem dos dados.

Nota-se que combinar conjuntos de características diferentes, sessões de treinamento diferentes, técnicas de classificação, pode resultar em um *pool* de classificadores cuja saída é combinada para assim melhorar a assertividade da classificação global. Nesta direção, pode-se considerar classificadores híbridos e conjuntos de classificadores como dois tipos de combinação de classificadores. Um classificador híbrido [Tsai *et al*. 2009] parte da ideia de combinar diversas técnicas de aprendizagem de máquina com objetivo de melhorar a atuação de forma geral do sistema. Sendo uma abordagem hibrida composta por duas partes funcionais, a primeira emprega os dados de entrada e retorna um resultado intermediário, e a segunda opera nos resultados intermediários para apresentar o resultado final.

Classificadores podem inclusive ser selecionados de maneira dinâmica através do conceito de redes de conselheiros [Quincozes et al. 2021]. Com este tipo de solução os detectores rotulados como conselheiros fornecem uma etapa adicional para a seleção dos melhores classificadores, permitindo resolver eventuais conflitos de classificação. Os detectores que analisam resultados díspares podem trocar conselhos para melhorar a precisão geral, selecionando apenas classificadores confiáveis para compor a decisão final de classificação, melhorando a confiabilidade do resultado. Além disso, os detectores podem realizar auto-aprendizagem atualizando seu conjunto de dados de treinamento após receber conselhos de detectores especializados sobre diferentes tipos de ataque.

5.4. Detecção de Anomalias na Prática

Detecção de anomalias aborda uma ampla gama de problemas em diversas vertentes na área de redes computadores, logo explora comportamentos em tráfego de protocolos HTTP, pacotes UDP/TCP, IoT (por exemplo: Zigbee [Moreno and Ruíz 2007], 6LoWPAN [Miguel et al. 2018], MQTT [Quincozes et al. 2019], CoAP [Ansari et al. 2018]). Esse cenário leva a uma série de parâmetros de análise que dizem respeito como os dados se apresentam para se identificar o que seria ou não uma anomalia em determinado fluxo de dados. Esta seção apresenta algumas reflexões que abordam essa diversidade da distribuição dos dados, métodos e/ou técnicas, anomalia de redes, tipos de dados, conjunto de dados (*dataset*), salientando, brevemente, alguns aspectos e obstáculos.

Existem vários tipos de anomalias de tráfego de rede e cada autor pesquisa este tópico endereçando-os de forma diferente. Na pesquisa de [Fernandes, G. *et al*. 2019], por exemplo, os autores apresentam as anomalias de rede categorizando-as em duas propriedades relevantes: de acordo com sua *natureza*, agrupados por como elas são caracterizadas, independentemente de serem maliciosas ou não; e de acordo com seu *aspecto causal*, distinguido de acordo com sua causa, tanto no aspecto malicioso quanto no não malicioso. A categorização de anomalias com base em sua natureza depende do contexto em que uma anormalidade é encontrada, ou de como ela ocorreu, sendo que ela pode ser ou não uma anomalia. Este aspecto pode direcionar como o sistema irá lidar e entender as anomalias

mineradas e detectadas. Com base em sua natureza, são apresentadas três categorias de anomalias: anomalias pontuais, anomalias coletivas e anomalias contextuais. Uma anomalia pontual é o desvio de uma instância de dados individual do padrão/comportamento usual. Uma anomalia coletiva ocorre quando apenas uma coleção de instâncias de dados semelhantes se comporta de forma anômala em relação ao conjunto de dados completo. Anomalias contextuais ou condicionais são eventos considerados anômalos dependendo do contexto em que se encontram.

De acordo com [Barford et al. 2002], as anomalias também podem ser agrupadas em quatro categorias: eventos operacionais de configuração incorreta ou falha; *flash crowd* de uso legítimo, mas anormal; anomalias de medição; e anomalias de abuso de rede ou ataques maliciosos (ou simplesmente, ataques de rede). Os eventos operacionais correspondem a configuração incorreta ou falhas, considerados não maliciosos, que podem ocorrer em um sistema de rede principalmente por falhas de hardware, *bugs* de software ou erros humanos. Os *flash crowds* são grandes eventos que geram uma inundação de requisições legítimas no tráfego, similares a um ataque de negação de serviço, o que pode ocorrer quando há um rápido crescimento de usuários tentando acessar um determinado recurso de rede, causando um aumento dramático na carga de um servidor. Por exemplo, o lançamento de uma promoção em uma *Black Friday* pode gerar um esgotamento de recurso do servidor. Embora não seja malicioso, se não houver tempo suficiente para reagir e fornecer os recursos necessários para lidar com a demanda de sobrecarga, esses eventos *flash* podem levar a uma indisponibilidade do serviço [Pan et al. 2014]. As anomalias de medição são problemas relacionados a infraestrutura de coleta de dados, ou seja, relacionados às medições necessárias para avaliar os comportamentos. Exemplos são a perda de dados de fluxo causada pela sobrecarga do roteador. Uma anomalia por abuso de rede corresponde a um conjunto de ações maliciosas com o objetivo de interromper, negar, degradar ou destruir informações e serviços de sistemas de redes de computadores. Este tipo de evento pode comprometer, além da integridade da rede, a confidencialidade e disponibilidade, dado que invasores com acessos indevidos ao sistema podem comprometer serviços e dados. Acessos indevidos podem ocorrer por meio de algumas abordagens como [Ghorbani et al. 2010]: engenharia social, *bugs* de software como manuseio de arquivos temporários, uso de uma identidade falsa, acesso não autorizado, inundando-o com nomes de usuários válidos e senhas arbitrárias para bloquear usuários autênticos.

As tentativas de categorizações e seus exemplos, demonstram que, na prática, uma etapa essencial para construir um sistema de detecção de anomalias é escolher a *fonte de dados da rede*, pois a origem do conjunto de dados selecionado pode direcionar quais tipos de anomalias o sistema pode detectar.

A partir desse contexto a caracterização precisa dos dados resulta no melhor desempenho do sistema de detecção de anomalias [Fernandes, G. *et al*. 2019]. Algumas das fontes de dados utilizadas são: Tcpdump [The Tcpdump Group 2022], SNMP (*Simple Network Management Protocol*) [Thottan and Ji 2003] e IP *flow* [Aitken et al. 2013]. Tcpdump é uma ferramenta de análise de pacotes utilizada para monitorar pacotes em uma rede de computadores. Permite analisar os cabeçalhos dos pacotes TCP/IP que passam pela interface de rede e é utilizada para realizar monitoramento e manutenção em redes de computadores. Também é explorada por estudantes para entender o funcionamento da pilha de protocolos TCP/IP. Porém, na prática, este tipo de dado pode apresen-

tar informações limitadas. Outra fonte de dados é o protocolo SNMP que apresenta uma estrutura cliente-servidor (gerentes SNMP e agentes SNMP) [Thottan and Ji 2003]. Os dados SNMP têm sido usados em sistemas de detecção de intrusão, pois são úteis quando se trata de coletar dados precisos de atividade de rede em um único nível de *host*. Todos os dados coletados são armazenados, como objetos SNMP, em um banco de dados hierárquico denominado MIB (*Management Information Base*). Objetos SNMP são dados de tráfego resumidos, construídos pela agregação de dados brutos coletados principalmente por ferramentas de *dump* TCP [Marnerides et al. 2014]. Uma significativa vantagem é que o SNMP é um protocolo implantado com dados refinados disponíveis, sendo usado em ferramentas tradicionais de gerenciamento de rede para medir parâmetros de desempenho e volume de tráfego, onde é vital entender quais endereços IP são a origem e o destino do tráfego e quais portas TCP/UDP estão gerando tráfego [Fernandes, G. *et al.* 2019].

Outra fonte de dados é o IP *flow*, tecnologia de gerenciamento que atende o desenvolvimento de novos serviços e a crescente complexidade das redes. Essa complexidade leva à necessidade de informações mais detalhadas sobre os dados transmitidos, o que é essencial para entender o comportamento das aplicações, usuários, departamentos de negócios e outras estruturas que dependem da rede para sua operação. Neste sentido, o uso de ferramentas e protocolos de gerenciamento de fluxo permite a construção de um banco de dados detalhado composto por informações essenciais de tráfego, possibilitando o melhor entendimento de aspectos mais subjetivos da operação da rede. Dessa forma, esse tipo de coleta vai além dos limitados contadores de bits e pacotes fornecidos pelo SNMP para caracterizar traços mais específicos no tráfego, mostrando tendências e comportamento da rede [Fontugne and Fukuda 2011].

Na mesma linha, e para mitigar restrições, a Cisco apresentou o protocolo NetFlow em 1996, sendo o pioneiro na introdução da estrutura de fluxo [Claise, B. 2004]. Todos os pacotes que constituem um fluxo têm um conjunto de propriedades comuns, incluindo endereços IP de origem/destino e portas TCP/UDP, VLAN e TOS (Tipo de Serviço), assim a NetFlow introduziu uma nova prática para auxiliar o gerenciamento de rede, incorporada aos dispositivos de rede (*switches*), que captura todos os pacotes que passam pelo *switch* e os agrega em fluxos IP de acordo com suas propriedades comuns. NfSen [NfSen: NetFlow sensor 2011] e nTop [nTop 2016] são as aplicações gráficas mais comuns que permitem a análise de dados. Além do NetFlow, existem outros protocolos que surgiram com o mesmo propósito. O sFlow foi introduzido pela InMon Corp. em 2001 [Panchen et al. 2001] e [Duffield, N. 2004]. No ano de 2008, a *Internet Engineering Task Force* (IETF) padronizou a exportação de informações de fluxo IP de roteadores, sondas e *switches*, introduzindo o protocolo IPFIX (*IP Flow Information Export*) [Aitken et al. 2013]. O IPFIX foi baseado no NetFlow versão 9 e recentemente, dois aprimoramentos do NetFlow apareceram, o Flexível NetFlow e NetFlow-lite [Deri, L. *et al.* 2011].

Com foco nos dados, em [Kalinichenko et al. 2014] encontra-se uma classificação de diferentes métodos de acordo com a variabilidade de dados e sua aplicabilidade em diferentes casos. Considera-se que os dados para detecção de anomalias se apresentam em três formas: i) *Dados Métricos* quando cada objeto em um conjunto de dados possui um determinado conjunto de atributos que permite operar com noções de "distância" e "proximidade"; ii) *Dados Evolutivos*, apresentados em Sequências Discretas, Séries Temporais e Multidimensionais; e iii) *Dados Multiestruturados*, dados que são apresen-

tados de forma não estruturada, semiestruturada ou estruturada. Diante deste contexto, os métodos abordados por [Kalinichenko et al. 2014], estão relacionados com essas três formas de dados. Por exemplo, para atributos que operam com noções de distância e proximidade costuma-se adotar métodos como clusterização, KNN, regressão linear, PCA (*Principal Component Analisys*) [Abdi and Williams 2010] para redução de características, métodos de máxima probabilidade e métodos estatísticos como Markov, Chebyshev, Chernoff [Cansado and Soto 2008] [Zhu, X. 2007]. Já para dados de séries temporais e multidimensionais os métodos aplicados são: *Hidden Markon Model* [Aggarwal 2013], técnicas baseadas em Kernel, baseadas em janelas, método SVM, filtro Kalman e modelos de alta regressão [Greff et al. 2015]. Para dados não estruturados, encontra-se os seguintes métodos ou técnicas: técnica de tf-idf LSA (*Latent, Semantic Analysis*) [Susan T. Dumais 2005], métodos de teoria de grafos [Aggarwal 2013], *Minimal Description Length principle* [Noble and Cook 2003] [Eberle and Holder 2007] [Chakrabarti 2004], *Bayesian Models* [Heard et al. 2010], *Markov Random Field*, *EM algorithm* [Silva and Willett 2008], assim como LOF (*Local Outlier Factor*) [Bhuyan et al. 2013]

As categorizações motivam também avaliações de adequação de técnicas de aprendizagem profunda (*deep learning*). Em [Chalapathy R.and Chawla S. 2019] é abordado um agrupamento sobre técnicas de detecção de anomalias profunda (*Deep Anomaly Detection*) em diferentes categorias baseada em suposições implícitas e na abordagem adotada. São considerados diferentes aspectos para detecção de anomalia profunda, como, por exemplo, a natureza dos dados que podem ser considerados como dados sequenciais ou não sequenciais e, que segundo os autores, tem uma correlação direta com o tipo de técnica que se aplica. Para entrada de dados sequencias, as técnicas para detecção de anomalia profunda tendem a se encaixar melhor em CNN (*Convolution Neural Networks*), RNN (*Recurrent Neural Network*) e LSTM (*Long Short Term Memory Networks*), enquanto que para dados não sequenciais as técnicas que tendem a se encaixar melhor são as de CNN, AE (*Autoencoders*) e suas variantes. Salienta-se que uma técnica de aprendizagem profunda é essencialmente uma rede neural com três ou mais camadas.

Quando considerado o domínio de aplicações, pode-se avaliar também aspectos como a natureza dos dados, as técnicas de detecção de anomalias profundas, além de outros aspectos que podem ser relevantes. Por exemplo, em domínios de sistemas de detecção de intrusão baseados em *host* (*Host-Based Intrusion Detection Systems - HIDS*) as técnicas de detecção de anomalias profundas (*Deep Anomaly Detection - DAD*) precisam lidar com o comprimento variável e a natureza sequencial dos dados. As técnicas DAD têm que modelar os dados da sequência ou calcular a semelhança entre as sequências. Algumas técnicas já aplicadas em HIDS são: CNN (*Convolution Neural Networks*), LSTM (*Long Short Term Memory Networks*), GRU (*Gated Recurrent Unit*), DNN (*Deep Neural Networks*) e SPN (*Sum Product Networks*). Diferentemente, no domínio de sistemas de detecção de intrusão baseado em redes (*Network Intrusion Detection Systems- NIDS*), que lida com o monitoramento de toda a rede em busca de tráfego suspeito, é interessante examinar cada pacote na rede. Logo, devido ao comportamento de *streaming* em tempo real, a natureza dos dados possui comportamento de *big data*, com alto volume, velocidade e variedade. Salienta-se que neste caso, os dados da rede também têm um aspecto temporal associado a eles. Algumas das técnicas de DAD de sucesso para NIDS são: CNN, LSTM, RNN, AE, RBM (*Restricted Boltzmann Machines*), DCA (*Dilated Convolution Autoen-*

coders), DBN (*Deep Belief Network*), SAE (*Stacked Autoencoders*), GAN (*Generative Adversarial Networks*) e CVAE (*Convolutional Variational Autoencoder*). Um desafio enfrentado pelas técnicas DAD na detecção de intrusão é que a natureza das anomalias continua mudando à medida que os intrusos adaptam seus ataques de rede para evitar as soluções de detecção de intrusão existentes.

Dentro do domínio de análise baseada em redes, outro grande foco de trabalho tem sido o da Internet das Coisas (*IoT - Internet of Things*). A IoT é identificada como uma rede de dispositivos que estão interconectados com softwares, servidores, sensores, dentre outros dispositivos. Logo, no campo da IoT, dados gerados por estações meteorológicas, *tags* de identificação por radiofrequência (RFID), componentes de infraestrutura e alguns outros sensores são enquadrados principalmente como dados sequenciais de séries temporais. A detecção de anomalias nessas redes IoT busca identificar o comportamento fraudulento e defeituoso em dados gerados por escalas massivas de dispositivos interconectados. Os principais desafios associados à detecção de *outliers* em dados da IoT são relacionados a heterogeneidade de dispositivos interconectados, o que torna a compreensão do tráfego mais complexa, uma vez que dispositivos distintos podem ter comportamentos distintos. O que se observa é que as técnicas que melhor atendem a demanda da IoT são: AE (*Autoencoders*), DBN (*Deep Belief Network*) e LSTM (*Long Short Term Memory*) [Chalapathy R.and Chawla S. 2019].

Para [Mohammadi et al. 2018] os dados de IoT são classificados como de *big data*, mas são diferentes, sendo necessário explorar as propriedades dos dados de IoT e tais diferenças em relação a dados tradicionais. Algumas características importantes sobre os dados de IoT: são dados de *streaming* em grande escala; possuem heterogeneidade, ou seja, vários dispositivos de aquisição de dados IoT reúnem informações diferentes; apresentam correlação de tempo e espaço, onde, os dispositivos sensores são conectados a um local específico e, portanto, têm um local e um carimbo de data/hora para cada um dos itens de dados; são dados com alto grau de ruído, devido a pequenos pedaços de dados em aplicativos de IoT, estando sujeito a erros e ruídos durante a aquisição e transmissão.

Finalmente, a partir do contexto exibido nesta seção, observa-se que a literatura demonstra que a detecção de anomalia tem aspectos bem distintos considerando a natureza dos dados e seu domínio de aplicação. Desta forma, as técnicas empregadas para detecção de anomalias também sofrem grande influência dos dados e características disponíveis (*features*), o que pode levar a variações de desempenho das técnicas na classificação do que é ou não uma anomalia. Considerar as diferenças de características nos dados para análise de tráfego em conjunto com técnicas que apesentam melhor desempenho frente a tais dados é primordial para a construção de um sistema de detecção de intrusão que apresente uma assertividade promissora e de confiança ao sistema.

5.5. Principais Tecnologias de Apoio

Para realizar trabalhos na área de detecção de anomalias em redes de computadores, diferentes tecnologias podem ser utilizadas e exploradas, dentre as quais bases de dados com tráfego de rede com e sem anomalias/ataques (Seção 5.5.1), ferramentas para escaneamento de tráfego (Seção 5.5.2), bem como bibliotecas com algoritmos diversos (Seção 5.5.3), dentre os quais os de aprendizado de máquina, para implantação mais facilitada da análise de dados por ferramentas de análise.

5.5.1. Conjuntos de Dados para Testes e Validação

O maior desafio para avaliação de sistemas de detecção de anomalias é a falta de conjuntos de dados (*datasets*) públicos adequados para testes e validação. A principal causa dessa falta é a sensibilidade dos dados. Tipicamente, a inspeção do tráfego de rede pode revelar informações sensíveis como dados pessoais, comunicações confidenciais e segredos de negócios. Diante desse risco, os detentores de tráfego de redes se sentem ameaçados e criam barreiras organizacionais e legais que impedem sua utilização. Para lidar com a falta de dados reais, pesquisadores costumam adotar *datasets* normalmente públicos.

Atualmente, para permitir que técnicas de detecção de anomalias sejam avaliadas, muitos conjuntos de dados foram criados e estão disponíveis à desenvolvedores. Em [Braei and Wagner 2020] são apresentados vários *datasets* e muitos deles são considerados *benchmarks* para detecção de anomalias e incluem dados reais e sintéticos.

Os primeiros *datasets*, tal como o DARPA 98/99, o KDDCup99 e o NSL-KDD, foram muito utilizados, mas acabaram ao longo do tempo apresentando problemas do tipo [Moustafa and Slay 2016]: (1) indisponibilidade de padrões de comportamento de ataques modernos; (2) falta de similaridade do que era considerado um padrão normal de tráfego décadas atrás e o que é considerado um padrão normal atualmente; e (3) diferença na distribuição de tipos de ataque nos conjuntos de treinamento e de testes. Para sobrepor estes problemas novos *datasets* foram propostos e estão disponíveis. A seguir alguns *datasets* são apresentados, incluindo a versão atualizado do NSL-KDD.

NSL-KDD – O *dataset* NSL-KDD (*Network Security Laboratory - Knowledge Discovery in Databases*) [Shahriar M. and Amin N. 2017] é um *dataset* balanceado e corresponde a uma versão atualizada do *dataset* KDDCup 1999, na qual o número de registros duplicados foi reduzido [Dhanabal and Shantharajah 2015]. Há quatro arquivos neste *dataset* (Train+, Train-, Test+ e Test-) e cada um é composto por 42 características (*features*) e rótulos de informações. Os rótulos são categorizados como *Normal* ou pertencente a um dos quatro grupos de ataque (DOS, U2R, R2L ou PROBE). Se houver interesse, os grupos de ataques podem ser agrupados dentro de uma categoria Anomalia. Há duas possibilidades de experimentos de classificação com este *dataset*: classificação binária ou classificação múltipla.

Kyoto 2006+ – Este *dataset* público contém, aproximadamente, 3 anos de dados de tráfego de rede real, de novembro de 2006 a agosto de 2009, coletados a partir de diversos tipos *honeypots* e organizados para avaliação de técnicas de detecção de anomalias [Song et al. 2011]. Há 24 características neste *dataset* (14 convencionais e 10 adicionais) e rótulos indicando sessões normais (1), ataques conhecidos (-1) e ataques desconhecidos (-2).

MAWILab – Este *dataset* é público e é composto de tráfego de rede capturado todos os dias a cada 15 minutos num *link* de dados entre o Japão e os Estados Unidos desde 2001 [Callegari et al. 2016] [Fontugne et al. 2010]. Todo fluxo de dados do *dataset* MAWILab é classificado por rótulos baseados em probabilidade e obtidos da saída de um detector que combina 4 técnicas de detecção de anomalias para indicar a presença de anomalias. Estes rótulos são usados como referência para análises de qualidade dos modelos testados. Há 4 categorias de tráfego rotuladas: *anomalous*, *suspicious*, *notice* e *benign*, e as anomalias de cada traço de tráfego são identificadas por característica como

endereço IP, informação de porta, etc.

UNSW-NB15 – É um *dataset* para detecção de intrusão de rede e foi gerado com a ferramenta IXIA PerfectStorm em laboratório no ano de 2015. O *dataset* contém um tráfego híbrido com comportamentos de atividades reais de um tráfego moderno e também com comportamentos de um tráfego de ataques contemporâneos. Oferece 100 GB de tráfego gerado com a ferramenta tcpdump, contém 49 características, 2 rótulos (*binary* e *multi-class*) e 9 tipos diferentes de ataques (*Fuzzers*, *Analysis*, *Backdoors*, *DoS*, *Exploits*, *Generic*, *Reconnaissance*, *Shellcode* e *Worms*) [Nour Moustafa and Jill Slay 2022]. Os registros no conjunto de treinamento e no conjunto de testes incluem dados normais e ataques.

Naturalmente, outros *datasets* são disponíveis, tal como os seguintes conjuntos de dados sugeridos para experimentações diversas: CIC-IDS2017 [Sharafaldin et al. 2018], CTU-13 [García et al. 2014] e ISCX'12 [Canadian Institute of Cybersecurity 2022]. Diversos *datasets* são apresentados em [Ring et al. 2019a] e *links* para estes *datasets* estão disponíveis em [Ring et al. 2019b].

5.5.2. Escaneadores de Tráfego de Redes

A proteção de sistemas computacionais no nível de rede costuma iniciar pela proteção do perímetro da rede via filtragem de pacotes através de *firewall* e servidores *proxy*. Estes mecanismos representam uma proteção significante contra ameaças que modificam o IP de origem dos pacotes e exploram vulnerabilidades nos protocolos e aplicações criando ou modificando pacotes IP. A filtragem pode ocorrer tanto para o tráfego de entrada quanto para o tráfego de saída. Ameaças externas são limitadas pela filtragem do tráfego de entrada e a participação de dispositivos internos em atos maliciosos na rede externa é limitada pela filtragem do tráfego de saída. Em ambos os casos há a ação de um administrador na elaboração, configuração e implementação de um conjunto de regras de filtragem.

As regras de filtragem implantadas nos *firewalls* costumam ser estáticas e podem não ser suficientes para conter ataques sobre equipamentos da rede interna ou partindo desta. Nestes casos, os mecanismos de defesa que empregam algoritmos de detecção de intrusão, especialmente os IDS baseados em redes e detecção de anomalias, permitem identificar ameaças ou ataques no tráfego que está passando pelos limites da rede, ou seja, passando pelos *firewalls* de borda da rede.

Neste cenário de controle de perímetro, um IDS atua como um monitor do tráfego de rede para comportamentos anormais e suspeitos e podem enviar alertas para o administrador de rede. Embora os IDS possam ser empregados para atuarem de maneira automática na contenção de uma intrusão, por exemplo ajustando regras de filtragem dinamicamente sem intervenção humana, esta prática tem seu custo e pode não ser adequada em muitas organizações. De qualquer maneira, na prática, a atuação de algoritmos de detecção de anomalias no nível de rede depende de dados de rede e estes devem ser adquiridos por monitores de tráfego de rede (*network sniffers*).

Uma ferramenta de monitoramento de rede muito utilizada no escaneamento de tráfego como apoio a ações de detecção de intrusão é o Wireshark [Richard Sharpe 2022], um software *open source* que executa em diversos sistemas operacionais e plataformas

(Windows, OS X, Linux, Unix). Em algumas distribuições Linux ele já vem inclusive pré-instalado, tal como o que ocorre no Kali Linux. Ele suporta um grande conjunto de protocolos e pode ser operado por linha de comando, o que facilita a integração em sistemas de IDS, ou por interface gráfica com o usuário. Através da interface gráfica o Wireshark pode ser usado diretamente como analisador de rede [Ndatinya et al. 2015], mas através de sua API de programação a biblioteca *libwireshark* pode ser utilizada para criar aplicações próprias (extensão, *plugin* ou mesmo um IDS completo).

5.5.3. Bibliotecas de Algoritmos

Uma biblioteca de algoritmos é uma coleção de métodos utilitários, classes e módulos que viabilizam o agrupamento e a reutilização de códigos. Reusar código que já foi produzido ou escrito por outras pessoas/desenvolvedores é um processo natural para ganhar tempo de desenvolvimento e também incorporar nas próprias implementações códigos de boa qualidade já desenvolvidos e testados. Deste modo, as bibliotecas desempenham um papel fundamental na vida de um desenvolvedor, pois otimizam o desenvolvimento de um programa. Neste sentido, as bibliotecas de *Machine Learning* vêm tendo uma grande abrangência em sua utilização para detecção de anomalias em redes de computadores. Algumas das mais populares bibliotecas na área de aprendizagem de máquina são a WEKA [Bhatia P. 2019], a mlpy [mlpy 2022], a scikit-learn [Pedregosa et al. 2011] [sklearn 2022] e TensorFlow [TensorFlow 2022].

A biblioteca WEKA é uma coleção de ferramentas de visualização e de algoritmos de aprendizado de máquina que podem ser empregados na análise de dados e modelagem preditiva, ações necessárias para detecção de anomalias. A biblioteca pode ser utilizada na preparação de dados, classificação, regressão, agrupamento, mineração de regras de associação e visualização, assim como, pode ser utilizada academicamente para avaliar algoritmos para detecção de anomalias e se ter mais domínio sobre a implementação das técnicas e métodos desejados. O conjunto de algoritmos implementados são escritos em linguagem java.

O Python, como linguagem de programação, têm sido preferido em desenvolvimentos de de computação científica, ciência de dados e aprendizado de máquina, principalmente por permitir aumentar o desempenho e a produtividade ao permitir o uso de bibliotecas de baixo nível e APIs limpas de alto nível. Sua biblioteca padrão [Python 2022] é muito extensa, contendo módulos integrados que fornecem bom nível de acesso à funcionalidades do sistema. Adicionalmente, boas bibliotecas de algoritmos de aprendizagem de máquina foram construídas para desenvolvedores Python, dentre elas a mlpy e a scikit-learn. A mlpy fornece uma ampla gama de métodos de aprendizado de máquina de última geração para problemas supervisionados e não supervisionados e visa encontrar um compromisso razoável entre modularidade, manutenibilidade, reprodutibilidade, usabilidade e eficiência. A mlpy é multiplataforma, funciona com Python 2 e 3 e é distribuído sob licença GPL3 [Raschka *et al*. 2020]. A scikit-learn, assim como a mlpy, explora as bibliotecas numérica (NumPy) e científica (SciPy) do Python e oferece vários algoritmos de aprendizagem de máquina para classificação, regressão e agrupamento de dados.

TensorFlow é uma biblioteca *open source* criada em 2015 pela Google com algoritmos de aprendizagem de máquina que permitem definir, treinar e implantar modelos de aprendizagem profunda. O processamento de aprendizagem profunda exige o uso efici-

ente de unidades de processamento gráfico (GPU) e de infraestruturas de processamento paralelo e distribuído para treinar os modelos sobre grandes volumes de dados e para descobrir novos métodos tal como os de funções de ativação em redes neurais. A biblioteca TensorFlow oferece bons suportes para exploração de algoritmos de aprendizagem profunda (principalmente redes neurais) em ambientes heterogêneos e distribuídos.

5.6. Métricas

As equipes de projeto de detectores de anomalias, na prática, são desafiadas a validar comparativamente o desempenho/qualidade dos seus algoritmos. Por exemplo, para alguém que utiliza um detector de anomalia, é um problema o detector dizer que ocorreu um ataque/anomalia, quando na realidade não ocorreu. Esta situação classificada como falso positivo que pode consumir o tempo dos analistas de segurança. Outro problema é o detector dizer que não ocorreu ataque/anomalia quando, de fato, ocorreu, o que pode permitir ao atacante invadir com sucesso o sistema sem ser percebido. Nesse sentido, há métricas que permitem avaliar os modelos e métodos projetados.

Uma das principais métricas utilizadas em detecção de anomalias é a Matriz de Confusão. A Matriz de Confusão é composta pelas seguintes taxas de classificação: falso positivo (*False Positive* - FP), classificação equivocada de tráfego legítimo como malicioso; falso negativo (*False Negative* - FN), tráfego com conteúdo malicioso erroneamente classificado como não malicioso; verdadeiro positivo (*True Positive* – TP), tráfego malicioso classificado corretamente como malicioso; e verdadeiro negativo (*True Negative* – TN), tráfego legítimo classificado corretamente como legítimo.

A partir das taxas dispostas na Matriz de Confusão, são derivadas outras métricas importantes e muito utilizadas, tal como: (i) acurácia (*Accuracy*); (ii) precisão (*Precision*); sensibilidade (*Recall*) e Medida-F1 (*F1-Score*) [Ibrahim, Siraj and Din 2017].

A seguir, a Seção 5.6.1 detalha a Matriz de Confusão e a Seção 5.6.2 detalha outras métricas que derivam da Matriz de Confusão.

5.6.1. Matriz de Confusão

A Matriz de Confusão é utilizada para aplicar a medida de desempenho escolhida. A referida matriz permite a visualização do desempenho de um algoritmo e é tipicamente utilizada em experimentos com aprendizagem de máquina. Cada coluna da Matriz de Confusão representa a classificação da amostra atribuída pelo algoritmo de aprendizagem, enquanto cada linha representa a classe real da amostra. A Tabela 1.1 apresenta a Matriz de Confusão, composta pelas seguintes taxas de classificação: TP (Verdadeiro Positivo); FP (Falso Positivo); FN (Falso Negativo) e TN (Verdadeiro Negativo).

Tabela 5.1. Matriz de Confusão

Classe Atual	Classe Esperada	
	Anomalia	Não Anomalia
Anomalia	TP	FN
Não Anomalia	FP	TN

Para avaliar o período em que um classificador mantém seu conhecimento válido, a taxa de erro ($Error_{rate}$) é utilizada como métrica. Ela permite avaliar o desempenho de

um modelo treinado em um ambiente supervisionado, ou seja, com amostras devidamente rotuladas em anomalia e não-anomalia, e posteriormente testado com amostras que não participaram do treinamento. A Equação 1 define a taxa de erro baseada nos erros de falso positivo e falso negativo.

$$Error_{rate} = \frac{FP + FN}{TP + FN + TN + FP} \tag{1}$$

5.6.2. Outras Métricas

A métrica de acurácia (***Accuracy***) é definida para expressar a proporção de itens classificados corretamente, ou seja, expressar ambos, a taxa de classificações corretas de fluxo de ataque (TP) e a de fluxo normal (TN). Esta métrica é expressa pela Equação 2.

$$Accuracy = \frac{TP + TN}{TP + FN + TN + FP} \cdot 100\% \tag{2}$$

A métrica de precisão (***Precision***) também é definida para expressar a proporção de itens classificados corretamente, porém, visa expressar apenas a taxa de classificações corretas de fluxo de ataque. Esta métrica é expressa pela Equação 3:

$$Precision = \frac{TP}{TP + FP} \cdot 100\% \tag{3}$$

A métrica de sensibilidade (***Recall***) expressa a habilidade do sistema corretamente detectar o ataque quando, de fato, o ataque ocorre. Na literatura, essa métrica também é referida como taxa de verdadeiros positivos (*True Positive Rate*). Esta métrica é expressa pela Equação 4:

$$Recall = \frac{TP}{TP + FN} \cdot 100\% \tag{4}$$

A medida-F1 (***F1-Score***) é uma métrica que indica o balanço entre precisão e sensibilidade e, por isso, é utilizada em casos onde falsos positivos e falsos negativos possuem impactos diferentes para o detector de anomalias. Um valor alto dessa métrica significa que a acurácia é relevante, ou seja, que os valores de TP, TN, FP, FN não apresentam grandes distorções. É adotada também como uma medida de confiabilidade da acurácia. Definida pela média harmônica entre a precisão e a sensibilidade, esta métrica é expressa pela Equação 5:

$$F1 - Score = 2 \cdot \frac{Recall \cdot Precision}{Recall + Precision} \tag{5}$$

Em conjunto com a Matriz de Confusão, essas métricas são uma importante ferramenta para determinar o desempenho de um detector de anomalias. Elas proporcionam um ferramental experimental que, a partir dos valores obtidos para a Matriz de Confusão (fáceis de medir), habilitam a análise comparativa de diferentes modelos/métodos de detecção.

5.7. Resumo do Capítulo

Este capítulo apresenta uma visão geral sobre as principais abordagens e técnicas para detecção de anomalias em redes, com foco na análise de elementos indicativos no tráfego de rede. A detecção de anomalias visa identificar comportamentos que se desviam do padrão normal, apontando possíveis ameaças à segurança. Sistemas de detecção de intrusão (IDS) são ferramentas essenciais nesse contexto, buscando mitigar ataques cibernéticos por meio da identificação de comportamentos anômalos.

Os IDS possuem como componente central o detector, que aplica algoritmos para identificar intrusões. Essas detecções podem ser realizadas de forma supervisionada, utilizando dados rotulados, ou não supervisionada, baseada em padrões nos dados. Recentemente, observa-se uma tendência crescente no uso de técnicas de aprendizado profundo, que têm demonstrado eficácia na construção de detectores de anomalias. Contudo, o sucesso desses sistemas depende de uma seleção criteriosa dos dados de entrada, cujas características, como protocolos e número de pacotes, são cruciais para o processo de identificação.

Pesquisas na área frequentemente utilizam datasets públicos para testar e validar algoritmos, permitindo o desenvolvimento de novas técnicas e ajustes, como a seleção das melhores características e parâmetros. Ferramentas para monitoramento de tráfego e bibliotecas de algoritmos são amplamente exploradas. Além disso, métricas de avaliação desempenham papel vital na validação da eficácia dos detectores.

Este capítulo aborda o funcionamento dos IDS (Seção 5.1), as abordagens para detecção de anomalias (Seção 5.2), o uso de aprendizado de máquina com destaque para aprendizado profundo (Seção 5.3), a seleção de dados de entrada (Seção 5.4), e a importância dos datasets públicos (Seção 5.5). Por fim, discute as métricas de avaliação (Seção 5.6), fundamentais para medir a performance dos detectores. A detecção de anomalias permanece um campo em constante evolução, sendo crucial para o fortalecimento da segurança em redes contra ameaças cibernéticas.

5.8. Exercícios

(Q_1) **O que é FALSO afirmar com relação a Sistemas de Detecção de Intrusão (*Intrusion Detection Systems* – IDS):**

(a) IDS tem por finalidade identificar ações oriundas de atacantes que possam ameaçar redes de computadores ou sistemas computacionais.
(b) Do ponto de vista arquitetural, um IDS consiste de cinco componentes chave: coletores de dados, base de conhecimento, detector, componente de configuração e componente de resposta.
(c) Os IDS baseados em *host* analisam dados coletados nos computadores que hospedam um serviço, tal como uso de memória, de processamento, de disco, de processos, variáveis de ambiente, dentre outros.
(d) A classificação de um IDS depende de dois fatores: da fonte de dados/informações e da técnica de detecção utilizada
(e) Em um IDS a base de conhecimento contém regras e informações relevantes para a identificação de intrusões.

(Q_2) **Um sistema de detecção de intrusão (IDS) pode ser baseado em: assinaturas e anomalias. Os baseados em assinaturas operam sobre uma base de conhecimento que contém informações capazes de permitir ao componente detector verificar se os dados coletados correspondem a um padrão de comportamento normal dos usuários, *hosts* e redes.**

(a) Falso
(b) Verdadeiro

(Q_3) **O aprendizado de máquina, uma subárea da inteligência artificial, corresponde a algoritmos e métodos com capacidade de aprender de maneira automática o comportamento de dados observados. As abordagens mais conhecidas de aprendizado de máquina são: aprendizado supervisionado, não supervisionado e por reforço.**

(a) Falso
(b) Verdadeiro

(Q_4) **O que é FALSO com relação a aprendizagem de máquina?**

(a) O aprendizado de máquina, uma subárea da inteligência artificial, corresponde a algoritmos e métodos com capacidade de aprender de maneira automática o comportamento de dados observados.
(b) As abordagens mais conhecidas de aprendizado de máquina são: aprendizado supervisionado, não supervisionado e por reforço.
(c) Exemplos de métodos de aprendizagem supervisionada: clusterização, regras de associação.
(d) Os métodos de detecção de anomalias supervisionados costumam empregar conjunto de dados conhecidos para classes de interesse, tal como normais, ataques, desconhecidos.
(e) O principal objetivo na aplicação de técnicas com abordagem de aprendizado não supervisionadas para detecção de anomalias é usar uma rotulagem automática de amostras de dados, ainda não rotuladas.

(Q_5) **Entre as soluções utilizadas para segurança da Internet, tem-se a filtragem ou *firewall*. Basicamente, *firewalls* utilizam regras (filtros) que definem o que deve ser feito. Desta forma, todo o tráfego que entra ou sai da rede ou máquina é comparado com as regras e o resultado é uma ação, geralmente permitir ou negar o tráfego.**

(a) Falso
(b) Verdadeiro

(Q_6) **Outra abordagem para solução de anomalias são os softwares anti-*, programas desenvolvidos para detectar e eliminar potenciais ameaças aos computadores e redes como, por exemplo, antivírus, anti-spyware, anti-phishing e anti-spam. Antivírus são usados no combate a programas e códigos espiões como *adware e keyloggers*.**

(a) Falso
(b) Verdadeiro

(Q_7) **O que é FALSO em relação a conjunto de classificadores:**

(a) Diferentes classificadores treinados sobre o mesmo dado não diferem somente em seu desempenho global, possivelmente, os classificadores apresentam fortes diferenças locais. Cada classificador tem sua própria região no espaço de características na qual atua com um desempenho melhor.
(b) Uma razão para combinar diversos classificadores é o desenvolvedor ter acesso a diferentes classificadores e poder treina-los em contextos diferentes;
(c) Diferentes conjuntos de características, diferentes conjuntos de treinamento, diferentes métodos de classificação, ou diferentes sessões de treino, podem resultar em um conjunto de classificadores cuja saída apresenta classificações iguais.
(d) Classificadores híbridos e conjuntos de classificadores podem ser considerados dois tipos de combinação de classificadores;
(e) A combinação de classificadores tenta superar a difícil tarefa de construir um único classificador para resolver problemas específicos, através da combinação da decisão de classificadores menos específicos, ou menos definidos para o problema a ser solucionado.

(Q_8) **Preencha adequadamente, com (S) para técnicas de Aprendizado Supervisionado, com (NS) para aprendizado Não-Supervisionado, ou com (OA) para Outras Abordagens:**

(a) ( ) Árvore de decisão
(b) ( ) Filtragem
(c) ( ) SVM (*Support Vector Machine*)
(d) ( ) Clusterização
(e) ( ) Combinação de Classificadores

(Q_9) **Analise as afirmações a seguir e marque (V) para Verdadeiro e (F) para Falso.**

(a) ( ) Um dos principais desafios para construir bons conjuntos de dados (*datasets*) para testes e avaliações é a sensibilidade dos dados.

(b) ( ) Há poucos (menos de 10) conjuntos de dados disponíveis para testes e avaliações.
(c) ( ) Para a geração de um conjunto de dados deve ser utilizado dados reais, e não dados sintéticos (gerados por ferramentas de geração de tráfego).
(d) ( ) Uma ferramenta muito utilizada para monitoramento de rede e detecção de intrusão é o Wireshark, que oferece uma API para o desenvolvedor.
(e) ( ) Os desenvolvedores de algoritmos para detecção de anomalias costumam se apoiar em bibliotecas de algoritmos de aprendizagem de máquina.

(Q_{10}) **Analise as afirmações a seguir.**

I. A Matriz de Confusão é uma métrica de desempenho utilizada na avaliação de algoritmos de detecção de anomalia que relaciona o resultado obtido (classe esperada da amostra) ao resultado real (classe atual da amostra).
II. A taxa de erros de um dado algoritmo relaciona o quantitativo global de erros (FP e FN) com o total de resultados (corretos e incorretos).
III. A métrica de acurácia (*Accuracy*) corresponde a taxa de verdadeiros positivos.
IV. A métrica *FI-Score* é uma métrica que indica também a confiabilidade da acurácia, dado que um valor alto de *FI-Score* significa que a acurácia é relevante.
V. As medidas de uma Matriz de Confusão costumam ser difíceis de medir, mas permitem realizar análises comparativas.

Marque a resposta correta.

(a) Estão corretas apenas I, II e IV.
(b) Estão corretas apenas I e III.
(c) Estão corretas apenas III e V.
(d) Estão corretas apenas II, IV e V.
(e) Todas as afirmações estão corretas.

5.8.1. Gabarito

(Q_1) d
(Q_2) a
(Q_3) b
(Q_4) c
(Q_5) a
(Q_6) b
(Q_7) c
(Q_8) a = S, b = AO, c = S, d = NS, e = AO
(Q_9) V-F-F-V-V
(Q_{10}) a

Referências

Abdi, H. and Williams, L. J. (2010). Principal component analysis. *WIREs Computational Statistics*, 2(4):433–459.

Aggarwal, C. C. (2013). *An Introduction to Outlier Analysis*, pages 1–40. Springer New York, New York, NY.

Aitken, P., Claise, B., and Trammell, B. (2013). Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011.

Alpaydim, E. (2010). Introduction to machine learning. The MIT Press, Cambridge, Massachusetts, EUA, 537 páginas.

Alpaydin, E. (2014). *Introduction to Machine Learning*. The MIT Press.

Althubiti, S., Nick, W., Mason, J., Yuan, X., and Esterline, A. (2018). Applying long short-term memory recurrent neural network for intrusion detection. In *SoutheastCon 2018*, pages 1–5.

Amaral, E. M. H., Kreutz, D., Andrade, E. R., and Lunardi, R. C. (2021). *Fundamentos de Segurança I*. EDIURCAMP, 1ª edition.

Ansari, D. B., Rehman, A. U., and Ali, R. (2018). Internet of things (iot) protocols: a brief exploration of mqtt and coap.

Barford, P., Kline, J., Plonka, D., and Ron, A. (2002). A signal analysis of network traffic anomalies. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet Measurment*, IMW '02, page 71–82, New York, NY, USA. Association for Computing Machinery.

Bhatia P. (2019). Data mining and data warehousing: Principles and practical techniques. Cambridge: Cambridge University Press. doi:10.1017/9781108635592.

Bhuyan, M. H., Bhattacharyya, D. K., and Kalita, J. K. (2013). Network anomaly detection: methods, systems and tools.

Braei, M. and Wagner, S. (2020). Anomaly detection in univariate time-series: A survey on the state-of-the-art. *CoRR*, abs/2004.00433.

Callegari, C., Giordano, S., and Pagano, M. (2016). Statistical network anomaly detection: An experimental study. In Doss, R., Piramuthu, S., and Zhou, W., editors, *Future Network Systems and Security*, pages 12–25, Cham. Springer International Publishing.

Canadian Institute of Cybersecurity (2022). Iscx dataset. http://www.iscx.ca/datasets/ (Acessado em 22 de agosto de 2022).

Cansado, A. and Soto, A. (2008). Unsupervised anomaly detection in large databases using bayesian networks. *Applied Artificial Intelligence*, 22(4):309–330.

Chakrabarti, D. (2004). Autopart: Parameter-free graph partitioning and outlier detection. In *European conference on principles of data mining and knowledge discovery*, pages 112–124. Springer.

Chalapathy R.and Chawla S. (2019). Deep learning for anomaly detection: A survey. arXiv preprint arXiv:1901.03407.

Claise, B. (2004). Rfc 3954: Cisco systems netflow services export version 9 (pp. 1–33). https://tools.ietf.org/html/rfc3954.

Deri, L. *et al*. (2011). Increasing data center network visibility with cisco netflow-lite. In *International Conference on Network and Service Management*, page 1–6.

Dhanabal, L. and Shantharajah, S. (2015). A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. In *International Journal of Advanced Research in Computer and Communication Engineering*, volume 4, page 446–452.

Duffield, N. (2004). Sampling for passive internet measurement: A review. statistical science, 19, 472–498.

Eberle, W. and Holder, L. (2007). Discovering structural anomalies in graph-based data. In *Seventh IEEE International Conference on Data Mining Workshops (ICDMW 2007)*, pages 393–398.

Eck, A. (2018). Neural networks for survey researchers. *Survey Practice*, 11(1).

Eltanbouly, S., Bashendy, M., AlNaimi, N., Chkirbene, Z., and Erbad, A. (2020). Machine learning techniques for network anomaly detection: A survey. In *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, pages 156–162.

Fernandes, G. *et al*. (2019). A comprehensive survey on network anomaly detection. *Telecommunication Systems*, 70:447–489.

Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010). MAWILab: Combining Diverse Anomaly Detectors for Automated Anomaly Labeling and Performance Benchmarking. In *ACM CoNEXT '10*, Philadelphia, PA.

Fontugne, R. and Fukuda, K. (2011). A hough-transform-based anomaly detector with an adaptive time interval. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, SAC '11, page 471–477, New York, NY, USA. Association for Computing Machinery.

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

García, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Ghorbani, A. A., Lu, W., and Tavallaee, M. (2010). *Network Attacks*, pages 1–25. Springer New York, NY.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., and Schmidhuber, J. (2015). LSTM: A search space odyssey. *CoRR*, abs/1503.04069.

Heard, N. A., Weston, D. J., Platanioti, K., and Hand, D. J. (2010). Bayesian anomaly detection methods for social networks. *The Annals of Applied Statistics*, 4(2):645–662.

Kalinichenko, L., Shanin, I., and Taraban, I. (2014). Methods for anomaly detection: A survey. In *CEUR Workshop Proceedings*, volume 1297, page 2025.

Kwon, D., Natarajan, K., Suh, S. C., Kim, H., and Kim, J. (2018). An empirical study on network anomaly detection using convolutional neural networks. In *ICDCS*, pages 1595–1598.

Liang, J., Zhao, W., and Ye, W. (2017). Anomaly-based web attack detection: A deep learning approach. ICNCC 2017, page 80–85, New York, NY, USA. Association for Computing Machinery.

Lutz, G. P., Ost, L., and Henke, M. (2018). Construção de conjunto de classificadores baseado na diversidade do espaço de características e algoritmos de aprendizagem para detecção de spam.

Marnerides, A., Schaeffer-Filho, A., and Mauthe, A. (2014). Traffic anomaly diagnosis in internet backbone networks: A survey. *Computer Networks*, 73:224–243.

Miguel, M. L., Jamhour, E., Pellenz, M. E., and Penna, M. C. (2018). Sdn architecture for 6lowpan wireless sensor networks.

Miyamoto *et al*. (2009). An evaluation of machine learning-based methods for detection of phishing sites. Em Proceedings of the 15th International Conference on Advances in Neuro-Information Processing, pp. 539-546.

mlpy (2022). mlpy - machine learning python. https://mlpy.sourceforge.net/.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., and Guizani, M. (2018). Deep learning for iot big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials*, 20(4):2923–2960.

Moraes, A. F. d. (2010). *Segurança em Redes: Fundamentos*. Editora Érica, 1 edition.

Moreno, J. and Ruíz, D. (2007). Informe técnico: Protocolo zigbee (ieee 802.15. 4).

Moustafa, N. and Slay, J. (2016). The evaluation of network anomaly detection systems: Statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set. *Information Security Journal: A Global Perspective*, 25(1-3):18–31.

Mukkamala, S. *et al*. (2002). Intrusion detection using neural networks and support vector machines. Proceedings of the International Joint Conference on Neural Networks. pp. 1702-1707.

Ndatinya, V., Xiao, Z., Manepalli, V. R., Meng, K., and Xiao, Y. (2015). Network forensics analysis using wireshark. *International Journal of Security and Networks*, 10(2):91–106.

NfSen: NetFlow sensor (2011). Nfsen: Netflow sensor, 2011. http://nfsen.sourceforge.net/. Accessed August 7, 2022.

Noble, C. C. and Cook, D. J. (2003). Graph-based anomaly detection. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '03, page 631–636, New York, NY, USA. Association for Computing Machinery.

Nour Moustafa and Jill Slay (2022). The unsw-nb15 dataset. https://research.unsw.edu.au/projects/unsw-nb15-dataset acessado: 12 agosto 2022.

nTop (2016). ntop. http://www.ntop.org/. Accessed August 7, 2022.

Pan, J., Hu, H., and Liu, Y. (2014). Human behavior during Flash Crowd in web surfing. *Physica A: Statistical Mechanics and its Applications*, 413(C):212–219.

Panchen, S., McKee, N., and Phaal, P. (2001). InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks. RFC 3176.

Patterson, J. and Gibson, A. (2017). *Deep Learning: A Practitioner's Approach*. O'Reilly Media, Inc., 1st edition.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Prashanth, G., Prashanth, V., Jayashree, P., and Srinivasan, N. (2008). Using random forests for network-based anomaly detection at active routers. In *2008 International Conference on Signal Processing, Communications and Networking*, pages 93–96.

Primartha, R. and Tama, B. A. (2017). Anomaly detection using random forest: A performance revisited. In *2017 International Conference on Data and Software Engineering (ICoDSE)*, pages 1–6.

Python (2022). Library python. https://docs.python.org/3/library/index.html acessado: 12 de agosto 2022.

Quincozes, S., Emilio, T., and Kazienko, J. (2019). Mqtt protocol: fundamentals, tools and future directions.

Quincozes, S. E., Raniery, C., Ceretta Nunes, R., Albuquerque, C., Passos, D., and Mossé, D. (2021). Counselors network for intrusion detection. *International Journal of Network Management*, 31(3):e2111. e2111 nem.2111.

Raschka *et al*. (2020). Machine learning in python: Main developments and technology trends in data science, machine learning, and artificial intelligence. https://www.mdpi.com/2078-2489/11/4/193 Journal Information, vol 11, number 4, article number 193, ISSN 2078-2489 - DOI 10.3390/info11040193.

Richard Sharpe, Ed Warnicke, U. L. (2022). *Wireshark User's Guide - Version 3.7.3*. Wireshark.org, 1 edition.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019a). A survey of network-based intrusion detection data sets. *Computers & Security*, 86:147–167.

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., and Hotho, A. (2019b). A survey of network-based intrusion detection data sets.

Shahriar M. and Amin N. (2017). A new deep learning approach for anomaly base ids using memetic classifier. International Journal of Computers, Communications & Control, 12(5).

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *International Conference on Information Systems Security and Privacy (ICISSP)*.

Silva, J. and Willett, R. (2008). Detection of anomalous meetings in a social network. In *42nd Annual Conference on Information Sciences and Systems*, pages 636–641. IEEE.

sklearn (2022). scikit-learn - machine learning in python. https://scikit-learn.org.

Smitha, R., Hareesha, K., and Kundapur, P. P. (2019). Machine learning approach for web intrusion detection: Mamls perspective. *Soft Computing and Signal Processing*, pages 119–133.

Song, J., Takakura, H., Okabe, Y., Eto, M., Inoue, D., and Nakao, K. (2011). Statistical analysis of honeypot data and building of kyoto 2006+ dataset for nids evaluation. BADGERS '11, page 29–36, New York, NY, USA. Association for Computing Machinery.

Susan T. Dumais (2005). Latent semantic analysis. Annual Review of Information Science and Technology 38: 188. doi:10.1002/aris.1440380105.

Syarif, I., Prugel-Bennett, A., and Wills, G. (2012). Unsupervised clustering approach for network anomaly detection. volume 293.

TensorFlow (2022). Library tensorflow. https://www.tensorflow.org/overview?hl=pt-bracessado em 15 agosto de 2022.

The Tcpdump Group (2022). tcpdump: command-line packet analyzer. https://www.tcpdump.org/.

Thottan, M. and Ji, C. (2003). Anomaly detection in ip networks. *IEEE Transactions on Signal Processing*, 51(8):2191–2204.

Tsai *et al*. (2009). Intrusion detection by machine learning: A review. Expert Syst. Appl., vol. 36, no. 10, pp. 11994–12000.

Xiao, H. *et al*. (2007). Intrusion detection using ensemble of svm classifier. Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FKSD 2007), pp. 45-49.

Zhu, X. (2007). Cs838-1 advanced nlp : The em algorithm k-means clustering, (6), 1–6.