INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Gestão preditiva de ataques de negação de serviço em redes corporativas

José Helder Gonçalves Tavares Júnior

Licenciado

Dissertação para obtenção do Grau de Mestre
em Engenharia de Eletrónica e Telecomunicações

Orientador : Doutor Nuno Miguel Abreu Luís

Júri:

Presidente: Doutor Rui António Policarpo Duarte

Vogais: Doutor Nuno Miguel Abreu Luís
Doutor Rodolfo Alexandre Duarte Oliveira

Novembro, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Gestão preditiva de ataques de negação de serviço em redes corporativas

José Helder Gonçalves Tavares Júnior

Licenciado

Dissertação para obtenção do Grau de Mestre
em Engenharia de Eletrónica e Telecomunicações

Orientador : Doutor Nuno Miguel Abreu Luís

Júri:

Presidente: Doutor Rui António Policarpo Duarte

Vogais: Doutor Nuno Miguel Abreu Luís
Doutor Rodolfo Alexandre Duarte Oliveira

Novembro, 2023

A minha família, cujo o apoio foi vital para conclusão deste projeto.

Agradecimentos

Gostaria de transmitir o meu sincero agradecimento ao Professor Miguel Luís, que desempenhou o papel de orientador exemplar ao longo de todo o processo de desenvolvimento deste trabalho. Sua orientação e disposição para enfrentar desafios foram fundamentais para aprimorar a qualidade deste projeto.

A todos os professores do ISEL a minha mais profunda apreciação por terem contribuído de maneira significativa para o sucesso deste trabalho. Suas influências foram inestimáveis e serão lembradas com gratidão nesta minha jornada acadêmica.

E por fim desejo expressar minha profunda gratidão à minha família, com um agradecimento especial à minha namorada, Jessica, que sempre esteve ao meu lado, fornecendo apoio inestimável.

Resumo

Na era atual, onde a interconexão digital se tornou ubíqua, os ciberataques são cada vez mais proeminentes. À medida que nossa sociedade se torna cada vez mais dependente da conectividade, com compras online, acesso a e-mails, consumo de média digital e a crescente adoção do trabalho remoto motivada pela pandemia de COVID-19, a segurança das redes corporativas assume um papel crítico. Muitas vezes, as empresas enfrentam desafios ao implementar medidas adequadas de segurança sobretudo devido a restrições orçamentárias, ou falta de conhecimento.

O objetivo desta dissertação visa estudar o desempenho de algoritmos de *machine learning* na deteção de acessos não legítimos a uma rede corporativa. Para alcançar esse objetivo, será desenvolvida uma topologia de rede que emule os serviços de acesso a uma rede corporativa. Em seguida, serão realizados acessos legítimos e acessos não legítimos, estes últimos baseados em ataques do tipo Negação de Serviço. Com base nos dados obtidos desses ataques, foi construído um *dataset* que foi posteriormente utilizado para treinar vários algoritmos de *machine learning*, analisados no contexto da deteção de acessos não autorizados numa rede corporativa.

Com base nos resultados dos algoritmos de *machine learning* testados foi desenvolvido um *framework* capaz de automatizar a deteção e o bloqueio de acessos não legítimos com base nas informações extraídas pelos vários modelos através da alteração da configuração da *firewall*. Esse *framework* foi validado por meio da avaliação de novos acessos, representando um passo importante na busca pela segurança das redes corporativas em um ambiente altamente interconectado.

Palavras-chave: Redes corporativas, Ataques de negação de serviço, automação, *machine learning*

Abstract

In the current era, where digital interconnectivity has become ubiquitous, cyberattacks are increasingly prominent. As our society becomes more reliant on connectivity, with online shopping, email access, consumption of digital media, and the growing adoption of remote work driven by the COVID-19 pandemic, the security of corporate networks assumes a critical role. Often, companies face challenges in implementing adequate security measures, primarily due to budget constraints or lack of knowledge.

The objective of this dissertation is to study the performance of machine learning algorithms in the detection of unauthorized access to a corporate network. To achieve this goal, a network topology that emulates the services of a corporate network access will be developed. Subsequently, legitimate accesses and unauthorized accesses, the latter based on Denial of Service attacks, will be carried out. Based on the data obtained from these attacks, a dataset was constructed, which was later used to train various machine learning algorithms, analyzed in the context of unauthorized access detection in a corporate network.

Based on the results of the tested machine learning algorithms, a framework was developed to automate the detection and blocking of unauthorized accesses based on the information extracted by the various models by altering the firewall configuration. This framework was validated through the evaluation of new accesses, representing a significant step in the pursuit of corporate network security in a highly interconnected environment.

Keywords: Corporate networks, Denial-of-Service, automation, machine learning

Índice

Lista de Figuras **xvii**

Lista de Tabelas **xix**

Lista de Abreviaturas e Siglas **xxi**

1 Introdução **1**

1.1 Motivação 1

1.2 Objetivos 3

1.3 Contribuição 4

1.4 Estrutura do documento 4

2 Estado da arte **5**

2.1 Enquadramento 5

2.2 Ataques de negação de serviço 6

2.2.1 Ataques de negação de serviço mais comuns 7

2.2.1.1 Inundação através de segmentos UDP 7

2.2.1.2 Inundação através de ICMP (Ping) 8

2.2.1.3 Inundação SYN 9

2.2.1.4 Ping da morte 10

2.2.1.5 Slowloris 10

2.2.1.6 Amplificação NTP 11
2.2.1.7 Inundação HTTP 11
2.2.2 Impacto dos DDoS no mercado 11
2.2.3 Proteção contra DDoS 13
2.2.3.1 Detecção 13
2.2.3.2 Desvio 14
2.2.3.3 Filtragem 14
2.2.3.4 Análise 15
2.2.4 Firewalls, IPS e IDS 15
2.3 *Machine Learning* 16
2.3.1 Modos de aprendizagem 17
2.3.2 Componentes do *machine leaning* 18
2.3.3 Tipos de *machine learning* 19
2.3.4 Principais algoritmos de *machine learning* 20
2.3.4.1 Regressão linear 21
2.3.4.2 Regressão logística 22
2.3.4.3 Árvores de decisão 22
2.3.4.4 Floresta aleatória 23
2.3.4.5 Redes neuronais artificiais 23
2.3.4.6 Máquina de vetores de suporte 24
2.3.5 *Features* 24
2.3.5.1 *Feature dimensionality reduction* 24
2.3.5.2 *Feature selection* 25
2.3.5.3 *Reconstruction/model-based methods* 26
2.3.6 Discussão 27
2.4 Trabalhos Relacionados 28
2.5 Conclusões 29

3 Aquisição e análise de dados **31**

3.1 Topologia de rede 31

3.2 Ferramentas utilizadas e ataques 32

3.2.1 Hping3 32

3.2.2 Slowloris 33

3.3 Experiências e coleção de dados 34

3.4 Conversão 35

3.5 Pré-processamento de dados 36

3.6 Correlação 37

3.7 Divisão do *dataset* 38

3.8 Aplicação dos algoritmos de *machine learning* 39

3.8.1 Problema de classes não balanceadas 40

3.8.2 *Floresta aleatória* 41

3.8.3 *Floresta de isolamento* 44

3.9 Cálculo da importância das *features* 45

3.10 Cálculo dos valores mínimos e máximos 47

3.11 Dataset público 49

3.12 Conclusões 50

4 *Framework* para automação e gestão preditiva de ataques de negação de serviço **51**

4.1 Desenvolvimento do *framework* 51

4.1.1 Código para o framework 53

4.1.1.1 Módulo para processamento dos dados 53

4.1.1.2 Módulo para criação de políticas de firewall 54

4.1.2 Resultados obtidos 55

4.2 Conclusões 56

5 Conclusões **59**

5.1 Resumo 59

5.2 Trabalho futuro 61

Referências **63**

A Informações extraídas da captura wirehsark e informações utlizadas no *machine learning* **i**

B Código desenvolvido para Automação **ix**

Lista de Figuras

1.1 Fluxo do pacote no sistema *firewall* Fortinet [25]. 2

2.1 Inundação UDP [12]. 8
2.2 Inundação SYN [20] 9
2.3 Inundação HTTP [11] 12
2.4 Componentes básicos do processo de *machine learning* [50]. 18
2.5 Algoritmos de *machine learning*. 21
2.6 Árvore de Decisão [2]. 22

3.1 Topologia de rede base para o estudo. 32
3.2 Exemplo de colunas que possuem apenas um valor no *dataset* recolhido. 37
3.3 Divisão do *dataset*. 39
3.4 Quantidade de dados em cada classe após *upsampling*. 41
3.5 *Features* selecionadas no modelo *floresta aleatória*. 42
3.6 Calculo da importância das *features*. 47

4.1 Sequência de eventos relativamente ao protótipo desenvolvido. 52
4.2 Resultados obtidos. 56

Lista de Tabelas

3.1 Detalhes da topologia. 32
3.2 Número de ocorrências de cada classe. 40
3.3 Tabela com os resultados obtidos com o modelo de *floresta aleatória*. . . . 43
3.4 Estatísticas para a flag igual a 1 48
3.5 Estatísticas para a flag igual a 0 49

Lista de Abreviaturas e Siglas

ACK Acknowledge. 9
API Application Programming Interface. 14

BGP Border Gateway Protocol. 14

CDN Content Delivery Network. 9
CPSs Ciber-Físicos. 11, 12
CPU Central Processing Unit. 6
CSV Comma-Separated Values. 35

DDoS Distributed Denial of Service. 1, 5, 6, 7, 9, 11, 12, 13, 14, 15, 16, 28, 29, 61
DNS Domain Name System. 14, 28
DoS Denial of Service. 2, 3, 4, 5, 6, 31, 35, 40, 56, 59, 60, 61

FTP File Transfer Protocol. 31

HTTP Hypertext Transfer Protocol. 10, 11, 31, 33, 34

ICMP Internet Control Message Protocol. 7, 8, 32
IDS Intrusion Detection System. 12, 15
IoT Internet of Things. 13, 28
IP Internet Protocol. 9, 10, 11, 14, 15, 34, 52, 53, 55, 56, 61
IPS Intrusion Prevention System. 15, 16, 53, 54, 55, 56, 57, 61
ISP Internet Service Provider. 11, 14, 28

LDAP Lightweight Directory Access Protocol. 28

NGFW Next Generation Firewall. 15, 16
NTP Network Time Protocol. 11, 28

PCAP Packet Capture. 35, 52, 60, i
Ping Packet Inter-Network Groper. 10, 32

RAM Random-Access Memory. 6

SMOTE Técnica de sobreamostragem sintética da classe minoritária. 40, 41
SSH Secure Shell. 54
SYN Synchronize. 9, 34, 49, 50, 62
SYN-ACK SYNchronize-ACKnowledgement. 9

TCP Transmission Control Protocol. 9, 32, 34, 46

UDP User Datagram Protocol. 7, 8, 11, 32
UTM Unified Threat Management. 16

VPN Virtual Private Network. 31

WAF Web Application Firewall. 14, 15
WAN Wide Area Network. 35

1

Introdução

Este capítulo descreve os objetivos e motivações para o desenvolvimento desta dissertação, indicando onde o mesmo pode contribuir para o desenvolvimento de soluções de cibersegurança através de técnicas de *machine learning* e de mecanismos de automação. Por fim é apresentada a estrutura do documento.

1.1 Motivação

Os ataques cibernéticos, normalmente designados de ciberataques, estão a tornar-se cada vez mais comuns, tendo como alvo mais cobiçado as empresas, desde as mais pequenas até às de maior dimensão. Tal ocorre porque, cada vez mais, as atividades quotidianas estão relacionadas com a utilização da Internet, tal como compras online e trabalho remoto. Estes ataques podem assumir diversas formas, mas os ataques de negação de serviço merecem destaque devido à sua simplicidade e eficácia [43].

Os ataques de Negação de Serviço Distribuído (Distributed Denial of Service (DDoS)) podem causar impactos significativos na disponibilidade, segurança e reputação de uma empresa. Portanto, é de suma importância que as organizações adotem medidas proativas para se defenderem contra esses ataques. A salvaguarda contra essa categoria de ataque pode ser abordada de diversas maneiras, incluindo a aquisição de equipamentos especializados, embora isso possa envolver custos consideráveis. Em contrapartida, empresas de menor dimensão dependem frequentemente exclusivamente de um *firewall* para proteção suas redes.

Como base para esta pesquisa, será utilizado um *firewall* da Fortinet, que viabiliza a implementação de políticas de mitigação de Ataques de Negação de Serviço (Denial of Service (DoS)). Esses dispositivos oferecem a possibilidade de estabelecer políticas que, quando configuradas adequadamente, não apenas podem otimizar os recursos do próprio *firewall*, mas também proteger o alvo do ataque em questão.

A Figura 1.1 esquematiza esquematiza o fluxo de um pacote por meio de um *firewall*, apresentando também todos os perfis de segurança que são aplicados ao pacote durante esse processo. Esses perfis de segurança podem incluir inspeção de pacotes, filtragem de conteúdo, verificação de integridade, autenticação de origem, e criptografia.

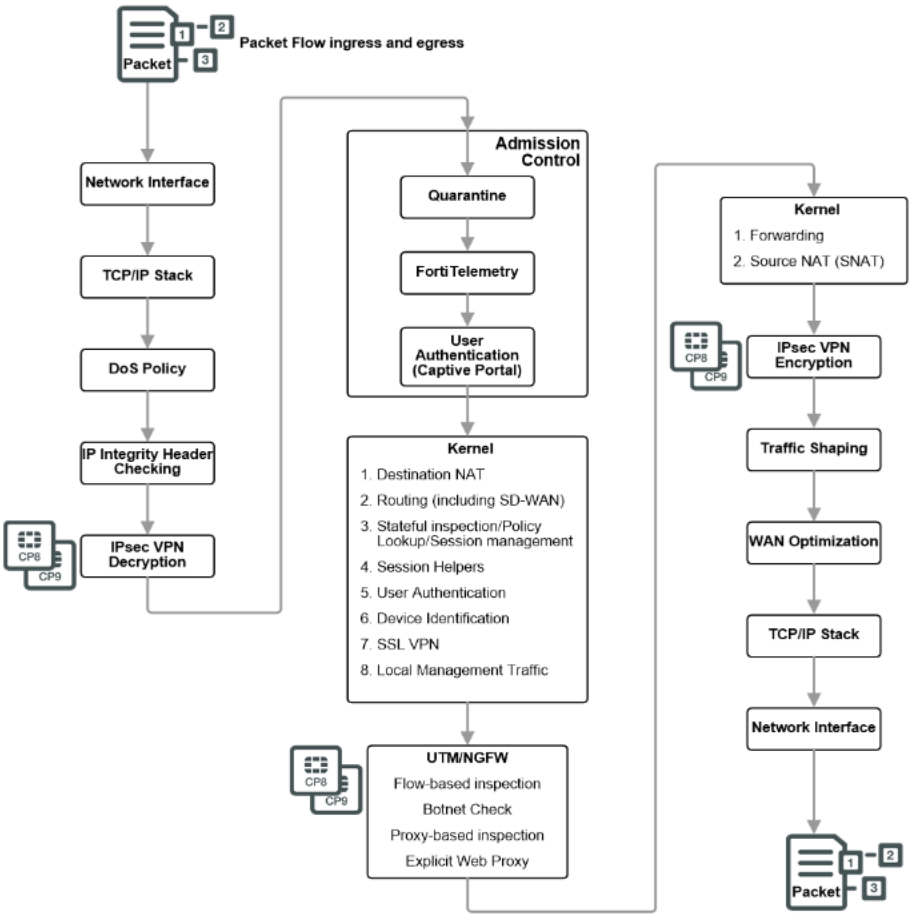


Figura 1.1: Fluxo do pacote no sistema *firewall* Fortinet [25].

Outro ponto importante para o desenvolvimento deste trabalho é o *machine learning*. Atualmente, o *machine learning* é aplicado em diversas áreas, como reconhecimento de fala, processamento de imagem e vídeo, previsão de procura, deteção de fraudes financeiras, entre outras. Aplicado de forma correta em cibersegurança pode oferecer a possibilidade de deteção automatizada de ameaças em tempo real [45] . Desta forma

será possível detetar o fluxo do atacante em tempo real, e aplicar de maneira automatizada políticas contra DoS na *firewall* em questão.

1.2 Objetivos

Esta dissertação visa desempenhar um papel fundamental na detecção e mitigação automatizadas de ataques de negação de serviço, concentrando-se especialmente em empresas de pequeno e médio porte. Essa abordagem se justifica pela realidade de que tais empresas muitas vezes enfrentam limitações significativas de recursos para investir em soluções de proteção. A ênfase não se restringe apenas à questão econômica; considera-se também a menor escala de tráfego e complexidade inerentes a empresas de menor porte. Essas características, por sua vez, contribuem para a redução das dimensões dos desafios enfrentados, tornando possível desenvolver estratégias eficientes e economicamente acessíveis para proteger essas organizações contra ameaças cibernéticas. Para atingir esse objetivo, foram delineadas as seguintes tarefas:

- Revisão da literatura com o objetivo de entender os vários tipos de ataques de negação, técnicas de *machine learning* e a sua aplicabilidade no problema a resolver, assim como trabalhos que já façam uso destas técnicas para a deteção e prevenção de ataques de negação;

- O desenvolvimento de um ambiente de testes, para emular o acesso de rede de uma empresa à Internet. Sobre este ambiente de testes serão efetuados acessos legítimos e não legítimos por forma a construir uma base de dados para posterior análise;

- Aplicação de diferentes técnicas de *machine learning* "floresta aleatória e floresta de isolamento"com o objetivo de identificar os acesso não legítimos, ou seja, os que representam um ataque à rede interna, e consequente implementação de regras para o seu bloqueio;

- A partir dos dados recolhidos e analisados, desenvolver *scripts* visando automatizar a aplicação de políticas de *firewall*, criando dessa forma uma proteção para o sistema sobre ataques.

1.3 Contribuição

Tendo em conta os atuais desafios de segurança que redes corporativas enfrentam, o trabalho desenvolvido pode ser resumido nos seguintes pontos:

- Discutir e apresentar um apanhando das principais literaturas já desenvolvidas com temáticas semelhante;
- Estudo das diferentes tipos e técnicas de *Machine Learning* , aplicadas em ataques de DoS em redes corporativas, com o objetivo de compreender a mais adequada para ambientes corporativos;
- Automatizar a proteção de redes corporativas de pequeno e médio porte, fazendo uso da linguagem Python na criação dos scripts, e políticas de *firewall*

1.4 Estrutura do documento

Para além deste capítulo, o documento apresenta a seguinte estrutura:

- **Capítulo 2 - Estado da arte:** introduz o leitor aos diversos conceitos endereçados neste trabalho, desde os diversos tipos de ataques em redes de comunicações até às diferentes técnicas de *machine learning*, terminando com uma discussão dos trabalhos relacionados com esta dissertação;
- **Capítulo 3 - Aquisição e análise de dados:** Este capítulo aborda o desenvolvimento do ambiente de teste que será utilizado como base neste trabalho, o método de coleção de dados, e uma análise preliminar aos mesmos;
- **Capítulo 4 - Framework para automação:** Nesta secção será elaborada uma estrutura para a automatização de bloqueio de acessos não legítimos, desde a conversão do arquivo de captura gerado via Wireshark, até à implementação do modelo de deteção no conjunto de dados. Posteriormente, utilizaremos o modelo de automação com base nas informações adquiridas por meio de *machine learning*.
- **Capítulo 5 - Conclusões:** Fornece um resumo abrangente de todo o progresso alcançado nos capítulos anteriores, acompanhado das considerações finais. Subsequentemente, serão abordados os elementos que identificamos como fundamentais para futuras investigações, a fim de dar seguimento ao desenvolvimento empreendido neste estudo.

2

Estado da arte

No capítulo 2 são apresentados os conceitos fundamentais relativamente às temáticas abordadas nesta dissertação. Na secção 2.1 é feito um enquadramento ao tema dos ciberataques. A secção 2.2 discute os vários tipos de ataques de negação de serviço, o impacto deste tipo de ataques no mercado e mecanismos de proteção. Neste capítulo são explorados também os conceitos de *machine learning*, apresentados na secção 2.3, e como esta tecnologia pode ser aplicada para combater os ataques DoS ou DDoS. Por fim, na secção 2.4 são apresentados alguns trabalhos desenvolvidos na mesma temática desta dissertação. A compreensão destes conceitos é crucial para a proteção de sistemas e redes contra ataques de negação de serviço, que podem causar danos significativos à disponibilidade de serviços na Internet.

2.1 Enquadramento

A evolução da era digital tem sido evidenciada ao longo dos anos, sendo que durante a pandemia do COVID-19 houve um aumento significativo das vendas e negócios *online*. Para acompanhar as tendências de mercado, muitas empresas viram-se obrigadas a migrar as suas operações para plataformas *online*, transferindo inclusive a totalidade dos seus negócios para o meio digital. Este movimento resultou num aumento do investimento em modelos de negócios *online* por parte das empresas [28].

Naturalmente, surgiu também uma tentativa maior em realizar todo o tipo de ataques informáticos, os ciberataques),a essas mesmas empresas, quer durante a fase anterior

à pandemia, quer hoje em dia [28]. Ciberataques estão em níveis nunca antes observados: de acordo com um estudo realizado pela empresa de segurança Check Point, os ataques a níveis globais aumentaram 28% no terceiro trimestre de 2022 em comparação com o mesmo período de 2021 [10]. O tipo de ataque a realizar pode variar na forma e no conteúdo, desde ataques *ransomwars, phishing, trojans* até os amplamente reconhecidos ataques de negação de serviço.

2.2 Ataques de negação de serviço

Num ataque de DoS, também conhecido por ataque de negação de serviço, um utilizador não legítimo utiliza uma única ligação com a Internet para explorar uma vulnerabilidade de software ou inundar um alvo com solicitações falsas, geralmente na tentativa de esgotar os atributos do servidor, por exemplo Random-Access Memory (RAM) e Central Processing Unit (CPU). [35].

Por outro lado, os ataques de DDoS são lançados de vários dispositivos ligados, distribuídos por toda a Internet. Esta característica de várias pessoas e vários dispositivos geralmente são mais difíceis de evitar, principalmente devido ao grande volume de dispositivos envolvidos. Ao contrário dos ataques DoS de fonte única, os ataques DDoS tendem a atingir a infraestrutura de rede na tentativa de saturar com grandes volumes de tráfego [35].

Os ataques de DDoS envolvem vários dispositivos *online*, conhecidos como *botnet*, que são usados para sobrecarregar uma máquina de destino com tráfego falso, e quando feitos com sucesso podem afetar o funcionamento dessa mesma máquina, tornando-a inoperacional, razão pela qual este tipo de ataques é muito procurado por *hackers*, vândalos cibernéticos, extorsionistas e qualquer outro utilizador com intenções não legítimas, com o objetivo de criar danos [34].

Contrariamente a outros tipos de ciberataques, os DDoS não tentam violar o perímetro de segurança. Ao invés, tornam a(s) máquina(s) indisponíveis para os utilizadores legítimos, desabilitando os atributos *online* das mesmas, sejam elas empresas ou particulares [28]. Os DDoS também podem ser utilizados como *esconderijos* para outras atividades maliciosas e inclusive para derrubar dispositivos de segurança [34]. Os ataques DDoS podem ocorrer de forma curta ou repetidamente. Independentemente disso, a indisponibilidade da máquina afetada, ou da rede por detrás dessa máquina, pode permanecer por vários dias ou semanas, consoante o nível de recuperação da entidade e das defesas implementadas. Posto isto, o peso na receita final pode ser astronómico: para além da perda do consumidor que passa a existir, pode mesmo criar

danos cuja irreversibilidade se demonstra extremamente dispendiosa, quer a nível financeiro, quer a nível de reputação [34].

Os ataques DDoS são ameaças virtuais muito utilizadas principalmente pela sua facilidade de execução [52]. Perante este cenário, têm vindo a ser desenvolvidos novos estudos para encontrar novas estratégias e aprimorar técnicas e soluções já existentes de combate ao DDoS [42].

Não se sabe ao certo quando é que os DDoS surgiram, uma vez que a informação disponível para consulta relata várias documentações e situações, que não foram formalmente oficializadas, que aconteceram ainda na década de 70, 80 e 90. Apenas se consegue garantir que já estão em funcionamento há mais de duas décadas e cada vez mais potenciadas à medida que a era digital também evolui.

2.2.1 Ataques de negação de serviço mais comuns

Os ataques de negação de serviço podem se manifestar em diversas formas, direcionando sua atenção para as variadas camadas do modelo TCP/IP. Este tipo de ataque procura explorar vulnerabilidades em pontos específicos da infraestrutura de rede, buscando sobrecarregar recursos críticos e incapacitar serviços essenciais. A capacidade de adaptação dos ataques de negação de serviço é notável, pois podem variar desde ataques volumétricos que inundam a largura de banda, até ataques de exaustão de recursos que exploram vulnerabilidades em sistemas alvo.

Abaixo estão elencados os principais tipos de ataques de negação de serviço e uma explicação sobre como esses ataques são executados pelos indivíduos mal-intencionados. Cada tipo de ataque possui características específicas que visam explorar diferentes fraquezas nos sistemas alvo. Compreender a natureza desses ataques é crucial para implementar medidas eficazes de mitigação e proteção da infraestrutura de rede contra tais ameaças persistentes.

2.2.1.1 Inundação através de segmentos UDP

Uma inundação de User Datagram Protocol (UDP), por definição, é qualquer ataque DDoS que inunda um alvo com pacotes UDP. O objetivo do ataque é inundar portas aleatórias numa máquina remota. Isto faz com que a máquina verifique repetidamente o aplicativo que está nessa porta e (quando nenhum aplicativo for encontrado) responda com um pacote Internet Control Message Protocol (ICMP) do tipo *Destino*

inalcançável. Esse processo esgota os recursos da máquina, o que pode levar à inacessibilidade [34]. A Figura 2.1 ilustra esse processo, onde máquinas controladas pelo atacante enviam pacotes UDP visando esgotar os recursos do servidor.

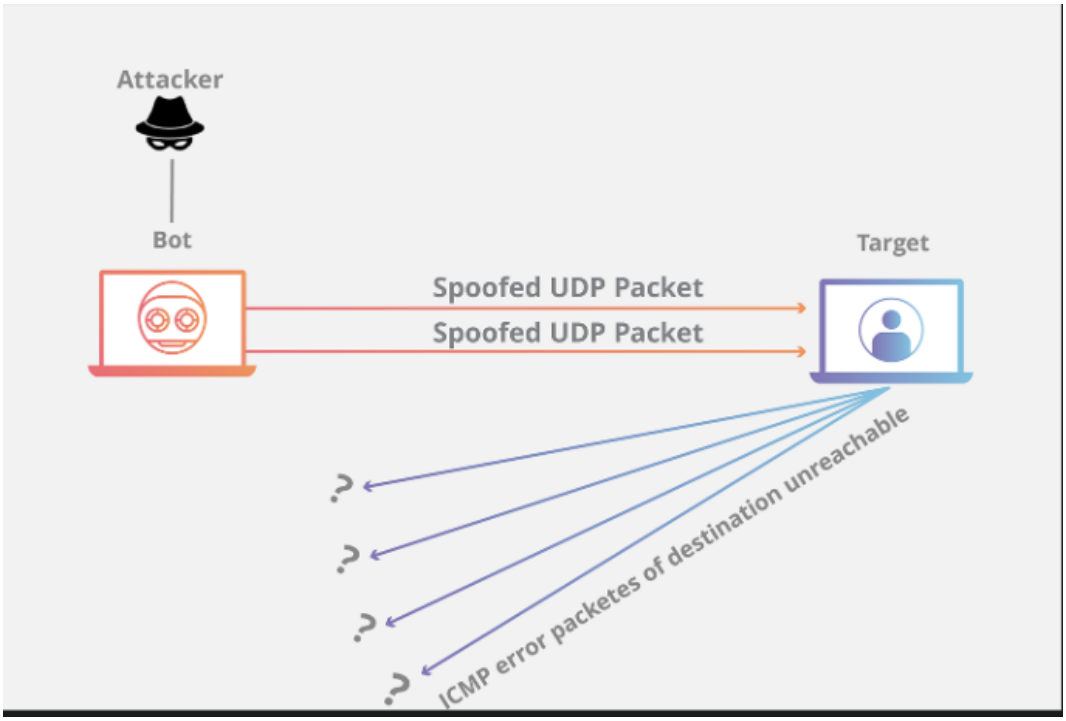


Figura 2.1: Inundação UDP [12].

Como forma de mitigar este tipo de ataque, a maioria dos sistemas operacionais implementam um limite de respostas destes pacotes ICMP como forma de interromper os ataques que exigem uma resposta ICMP. Uma desvantagem desta abordagem é que pacotes podem ser filtrados neste processo, pacotes úteis que podem ser utilizados por ferramentas de monitorização da rede [12].

2.2.1.2 Inundação através de ICMP (Ping)

Em semelhança à inundação UDP, uma inundação ICMP sobrecarrega o recurso de destino com pacotes ICMP *Echo Request* (utilizados na ferramenta *ping*), geralmente enviando pacotes com uma elevada frequência, sem esperar pelas devidas respostas. Esse tipo de ataque pode consumir largura de banda de saída e entrada, pois os servidores da vítima geralmente tentam responder com pacotesICMP *Echo Reply*, resultando numa significativa desaceleração geral de desempenho do sistema [34].

Para proteção contra esse tipo de ataque, desativar servidores e dispositivos de rede para não responder a solicitações ICMP pode ser uma alternativa, porém estas podem ser necessárias para a monitorização da rede. Outra alternativa é contratar soluções

de Content Delivery Network (CDN). Essa estratégia tira o custo de recursos tanto da largura de banda quanto da capacidade de processamento do servidor visado e o coloca na rede *anycast*[1] do CDN [16].

2.2.1.3 Inundação SYN

Um ataque DDoS de inundação Synchronize (SYN) explora uma fraqueza conhecida na sequência de conexão Transmission Control Protocol (TCP) (o "aperto de mão de três etapas"), em que uma solicitação SYN para iniciar uma conexão TCP com um *host* deve ser respondida com uma resposta SYNchronize-ACKnowledgement (SYN-ACK) desse *host* e em seguida, confirmado por uma resposta Acknowledge (ACK) do solicitante. Num cenário de inundação SYN, o solicitante envia várias solicitações SYN, mas não responde à resposta SYN-ACK do *host* ou envia as solicitações SYN de um endereço Internet Protocol (IP) falsificado. De qualquer forma, o sistema *host* continua a aguardar a confirmação de cada uma das solicitações, vinculando recursos até que nenhuma nova ligação possa ser feita e, resultando em negação de serviço [34]. A Figura 2.2 ilustra esse processo, onde máquinas controladas pelo atacante enviam pacotes SYN-ACK.

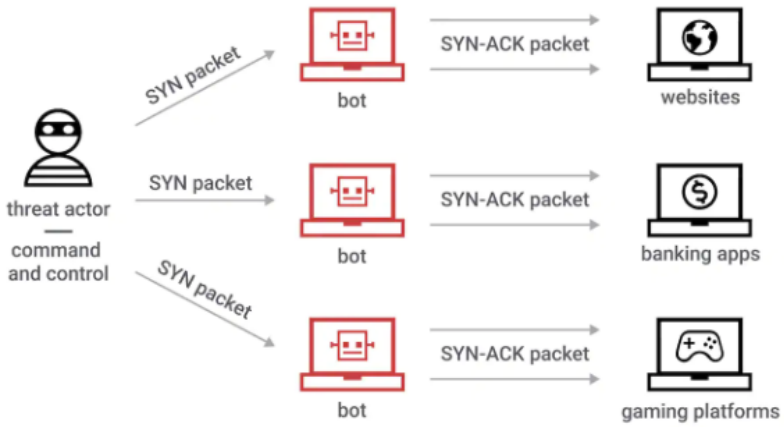


Figura 2.2: Inundação SYN [20]

.

[1] *Anycast* é uma técnica de rede em que o mesmo endereço IP é atribuído a múltiplos dispositivos, mas apenas um dispositivo responde a um determinado pacote de dados, geralmente aquele que está fisicamente mais próximo do remetente.

2.2.1.4 Ping da morte

O ataque denominado Packet Inter-Network Groper (Ping) da morte (Ping of Death (PoD)) consiste em um ataque realizado por um usuário não legítimo, no qual múltiplos Pings mal formados ou maliciosos são enviados a um computador. O comprimento máximo de um pacote IP, incluindo o cabeçalho, é de 65.535 bytes. Contudo, a camada de ligação geralmente impõe limites ao tamanho máximo da trama, por exemplo, em redes Ethernet o limite é de 1500 bytes. Dessa forma, um datagrama IP de grandes dimensões é fragmentado em vários datagramas IP, conhecidos como fragmentos, e o *host* destinatário remonta os fragmentos IP resultando no datagrama completo. Num ataque de Ping da morte, após a manipulação maliciosa do conteúdo do fragmento, o destinatário recebe um pacote IP com mais de 65.535 bytes quando remontado, podendo sobrecarregar os *buffers* de memória alocados para o pacote e causando uma negação de serviço [34].

Existem diferentes formas de mitigar esses ataques, conforme mencionado por [13], uma maneira seria acrescentar verificações ao processo de remontagem para garantir que a restrição de tamanho máximo do pacote não será excedida após remontagem do mesmo. Outra solução é criar um *buffer* de memória com espaço suficiente para lidar com pacotes que excedam o máximo estabelecido

2.2.1.5 Slowloris

Slowloris é um ataque altamente direcionado, permitindo que um servidor da Web derrube outro servidor, sem afetar outros serviços ou portas na rede de destino. O *Slowloris* faz isso mantendo o maior número possível de ligações com o servidor Web de destino, abertas o máximo tempo possível. São criadas através de ligações com o servidor de destino, mas enviando apenas uma solicitação parcial. O *Slowloris* envia constantemente mais cabeçalhos Hypertext Transfer Protocol (HTTP), mas nunca conclui uma solicitação. O servidor de destino mantém cada uma dessas ligações falsas abertas. Este processo, eventualmente, atinge a capacidade máxima de ligações simultâneas, levando à negação de ligações adicionais de clientes legítimos [34].

Para servidores vulneráveis a este tipo de ataque existem diferentes maneiras de abrandar o mesmo. A utilização de proteção baseada em nuvem, como um serviço de *proxy* reverso, e limitando o numero máximo de conexões de um mesmo endereço IP, são algumas delas [15].

2.2.1.6 Amplificação NTP

Nos ataques de amplificação de Network Time Protocol (NTP), o agressor explora servidores NTP acessíveis publicamente para sobrecarregar um servidor de destino com tráfego UDP. O ataque é definido como um ataque de amplificação porque a proporção de consulta para resposta em tais cenários está entre 1:20, 1:200 ou até mais. Isso significa que qualquer invasor que obtenha uma lista de servidores NTP abertos (por exemplo, usando uma ferramenta como *Metasploit* ou dados do Open NTP Project) pode facilmente gerar um ataque DDoS devastador de alta largura de banda e volume [34].

Uma solução de mitigação deste tipo de ataques passa por se fazer uma verificação do IP de origem impedindo que pacotes falsificados saiam da rede [14]. Isso se deve ao fato de que estas solicitações UDP são enviadas por *botnet* do invasor e precisam de ter um endereço de IP de origem falsificado para o endereço de IP da vítima. Um componente crucial para reduzir a eficácia dos ataques de amplificação baseados em UDP é que os Internet Service Provider (ISP)s rejeitem qualquer tráfego interno com endereços de IP falsificados, cabendo ao ISP criar *blackhole*[2] para este tipo de tráfego.

2.2.1.7 Inundação HTTP

Num ataque DDoS de inundação HTTP, o invasor explora solicitações HTTP GET ou POST aparentemente legítimas para atacar um servidor ou aplicativo da Web. As inundações de HTTP não usam pacotes mal formados, técnicas de *spoofing* ou reflexão e exigem menos largura de banda do que outros ataques para derrubar o site ou servidor de destino. O ataque é mais eficaz quando força o servidor ou aplicativo a alocar o máximo de recursos possível em resposta a cada solicitação [34]. A Figura 2.3 demonstra esse processo, onde máquinas controladas pelo atacante enviam pacotes HTTP *request*.

2.2.2 Impacto dos DDoS no mercado

Os Sistemas Ciber-Físicos (CPSs) consistem num conjunto de componentes em rede, incluindo sensores, unidades de processamento de controle e dispositivos de comunicação aplicadas à monitorização e gestão de infra-estruturas físicas. Os CPSs são

[2] *blackhole* (ou buraco negro) em redes de computadores é uma técnica usada para descartar pacotes de rede indesejados, enviando-os para um *buraco negro*. Quando um pacote é enviado para o buraco negro, ele é descartado imediatamente, sem processamento adicional, economizando recursos do servidor ou da rede e evitando a sobrecarga.

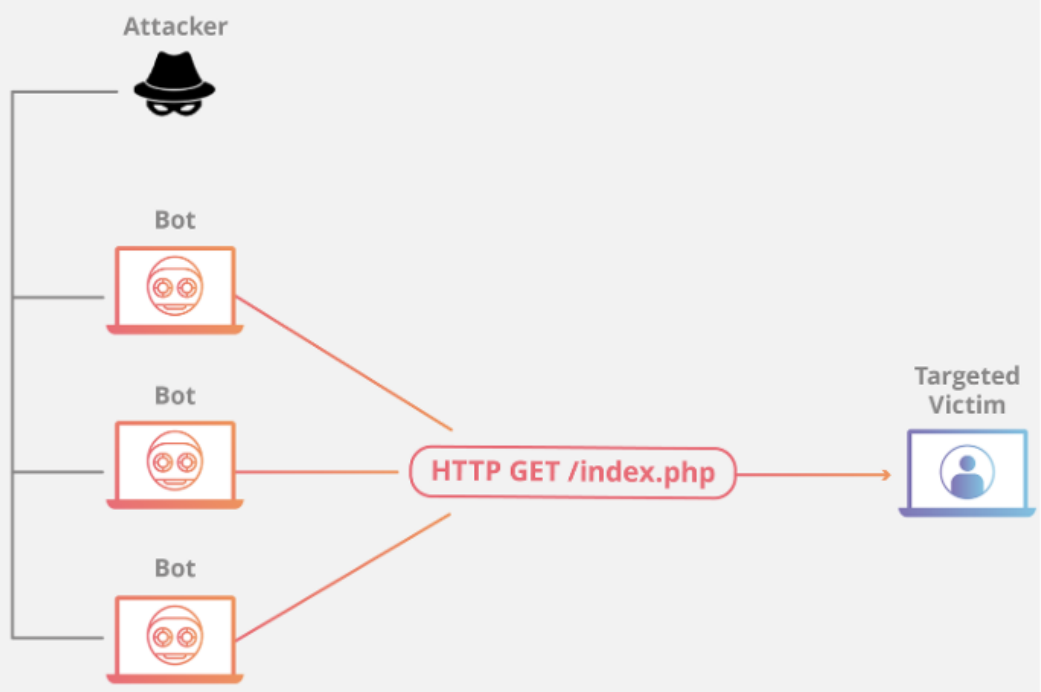


Figura 2.3: Inundação HTTP [11] .

normalmente utilizados para aplicações críticas de segurança, como aviação, instrumentação, sistemas de defesa e controlo de infra-estruturas críticas, nomeadamente energia elétrica, recursos hídricos e sistemas de comunicação. Como consequência, potenciais ataques cibernéticos e físicos podem levar ao desaparecimento de informações, danos económicos extensos e destruição de infra-estruturas críticas. Nos últimos anos têm vindo a surgir sistematicamente novas ameaças avançadas contra os CPSs, incluindo os DDoS. Além disso, os sistemas tradicionais de Intrusion Detection System (IDS) têm vindo a ter dificuldade em detetar com eficiência esses novos ataques, o que prova que é necessário desenvolver sistemas de deteção de intrusão que sejam mais sofisticados e altamente eficazes a proteger os CPSs. Alguns desses sistemas de deteção de instrução já envolvem técnicas bem avançadas de processamento de sinal, como é o caso das soluções baseadas em *machine learning* [42].

Os DDoS são também utilizados para atacar outras fontes, como as nuvens virtuais de informação, chamadas computação em nuvem (*clouds* ou *cloud computing*) onde as pessoas preservam a sua informação e mais que um utilizador pode aceder a esse ambiente compartilhado, usando o mesmo recurso. A computação em nuvem tem vindo a ganhar muita atenção devido ás suas características, como o custo-benefício e a prestação de serviços sob demanda. Mas, apesar das suas vantagens, este conceito de ambiente compartilhado numa nuvem pode ter a sua segurança ameaçada e comprometida, bem como a disponibilidade do serviço. Além disso, muitas empresas optam por utilizar a

computação em nuvem, o que mais uma vez incentiva aos ataques cibernéticos. Mais uma vez, os DDoS têm um papel muito grande e pesado nos ataques cibernéticos a estas nuvens virtuais e compete a um provedor de serviços em nuvem (CSP) ter a capacidade de garantir que a segurança e a disponibilidade dos serviços e recursos se mantém assegurada, para garantir o compromisso com o cliente [5].

Internet of Things (IoT) caracteriza uma rede de dispositivos coletivamente conectados e a tecnologia que auxilia a comunicação entre os dispositivos e a nuvem, e também entre os próprios dispositivos. Os dispositivos de IoT são amplamente usados em muitos setores devido às vantagens nos ecossistemas sociais, industriais e empresariais, incluindo cidades inteligentes, agricultura inteligente, medicina inteligente, logística inteligente, etc.. No entanto, os DDoS representam uma ameaça muito séria à segurança dos sistemas de IoT. Os invasores podem explorar facilmente as vulnerabilidades dos dispositivos IoT e controlá-los como parte de *botnets* para lançar ataques DDoS. Isto é possível de acontecer uma vez que os dispositivos IoT são limitados por recursos com memória escassa e recursos de computação [33].

As grandes empresas comerciais e entidades de renome têm proteção mais acentuada contra os DDoS uma vez que têm capacidade de arcar com medidas mais defensivas e rigorosas contra ciberataques. No entanto, os novos empreendedores e negócios mais familiares e/ou pequenos estão numa situação diferente pois não conseguem sustentar esses recursos de segurança da mesma forma [28].

2.2.3 Proteção contra DDoS

De forma a se proteger um alvo de ataques de DDoS é necessário recorrer aos vários estágios de mitigação: deteção, desvio, filtragem e análise.

2.2.3.1 Detecção

A identificação de desvios de fluxo de tráfego podem sinalizar o surgimento de um ataque DDoS. A eficácia é medida pela sua capacidade de reconhecer um ataque o mais cedo possível, sendo o principal objetivo o de detetar instantaneamente [36].

Para isto existem diversas análises, incluindo o *machine learning*, que será abordado neste trabalho, e a inteligência artificial tem uma capacidade de se adaptar a diversos problemas, de naturezas diferentes, apresentando técnicas que buscam a deteção de ataques tem tempo real, a partir de informações observadas na rede.

Outra forma de deteção é através do uso de ferramentas de análise de dados de tráfego. Estas ajudam a detetar alguns desses sinais reveladores de um ataque de DDoS, tais como [11]:

- Uma quantidade suspeita de tráfego com origem num único endereço de IP ou de uma faixa de endereços IP;
- Uma enxurrada de tráfego de utilizadores com o mesmo perfil de comportamento, como tipo de dispositivo, geolocalização ou versão de navegador web;
- Um aumento inexplicável de solicitações de uma mesma página ou ponto de terminação;
- Padrões de tráfego incomuns, como picos em horários inusitados ou padrões que parecem artificiais (por exemplo um pico a cada 10 minutos).

2.2.3.2 Desvio

O tráfego é redirecionado para longe de seu destino por meio de encaminhamento Domain Name System (DNS) ou Border Gateway Protocol (BGP), e é tomada uma decisão entre filtrá-lo ou descartá-lo completamente. O encaminhamento DNS está sempre ativo, portanto, pode responder a ataques rapidamente e é eficaz contra-ataques no aplicativo e na rede. O encaminhamento BGP é sempre ativo ou sob procura [36].

Uma outra forma de desvio, disponível para praticamente todos os administradores de rede é criar uma rota tipo *blackhole* Se um ativo da Internet estiver enfrentando um ataque de DDoS, o ISP do ativo poderá enviar todo o tráfego do site para um *blackhole* como uma forma de defesa [11].

2.2.3.3 Filtragem

Nesta etapa o tráfegoDDoS é eliminado, geralmente identificando padrões que distinguem instantaneamente entre tráfego legítimo (ou seja, humanos, chamadas de Application Programming Interface (API) e *bots* de mecanismos de pesquisa) e visitantes mal-intencionados. A capacidade de resposta tem de conseguir bloquear um ataque sem interferir na experiência dos utilizadores legítimos. O objetivo é que a sua solução seja completamente transparente para os visitantes do *website* [36].

O Web Application Firewall (WAF) é uma ferramenta capaz de ajudar a mitigar o ataque de DDoS na camada 7. Ao ser posicionado entre a Internet e o servidor de origem,

um WAF poderá atuar como um *proxy* reverso e proteger o servidor alvo de determinados tipos de tráfego maliciosos, realizando a filtragem dos mesmo. Ao filtrar as solicitações de acordo com uma série de regras usadas para identificar ferramentas de DDoS, os ataques na camada 7 podem ser impedidos [11].

Os Next Generation Firewall (NGFW), são um pouco mais abrangentes e podem realizar essa filtragem dos diferentes tipos de ataques de DDoS. O dispositivo a ser utilizado neste trabalho para cumprir com esta etapa de mitigação, o *firewall* da Fortigate, permite a criação de políticas especificas para proteção dos diferentes tipos de DDoS, além de permitir a implementação de WAF para proteção de servidores internos.

2.2.3.4 Análise

A análise do sistema pode ajudar a compilar informações sobre o ataque, tanto para identificar o(s) infrator(es) quanto para melhorar a resiliência futura, podendo ser necessário uma análise manual detalhada. Técnicas avançadas de análise de segurança podem oferecer visibilidade granular do tráfego de ataque e compreensão instantânea dos detalhes do ataque [36].

2.2.4 Firewalls, IPS e IDS

Assim como os ataques se foram adaptando à nova realidade, as defesas para DDoS também se foram adaptando e evoluindo, como parte das tecnologias de proteção existentes, como *firewalls* e IPs. Os *firewalls* foram os primeiros dispositivos de ponto de estrangulamento usados para separar redes confiáveis de não confiáveis. Posteriormente surgiram os IDSs e os Intrusion Prevention Systems (IPSs) [26].

Os *firewalls* convencionais (filtros de pacotes, *proxies* ou *firewalls* de inspeção de estado) examinam os cabeçalhos dos pacotes para identificar se há uma regra que permite o tráfego de uma determinada origem para um determinado destino, por exemplo, e descartam as tentativas de ligação de uma fonte não permitida ou para um destino proibido. Os *firewalls* são capazes de observar se uma sessão foi estabelecida ou não pelos *peers* (cliente e servidor) tentando estabelecer uma conversa (uma ligação). Uma vez estabelecida uma sessão, um dispositivo de *firewall* mantém o estado de todas as ligações permitidas pela política de segurança, desde o momento em que a mesma começa até que termine. Essas informações são mantidas numa tabela de sessão, mas um *firewall* está sujeito a ataques uma vez que há um limite de ligações na tabela de sessão e ultrapassando o limite não é possível adicionar mais ligações a essa tabela [26].

Quase todos os *firewalls* e IPS oferecem algum nível de defesa contra DDoS. Alguns dispositivos Unified Threat Management (UTM) ou NGFW oferecem serviços anti-DDoS e podem mitigar muitos ataques DDoS. Ter um dispositivo para *firewalls*, IPS e DDoS é mais fácil de gerir e menos complexo de implementar, mas um único dispositivo para fazer toda a proteção pode ser facilmente sobrecarregado com ataques DDoS volumétricos [26].

As proteções contra DDoS nos *firewalls* ou IPS podem afetar o desempenho geral de um único dispositivo, resultando em taxas de transferência reduzidas e maior latência para utilizadores finais. Devido a isso, a habilitação de mecanismos anti-DDoS em dispositivos *firewall* ou IPS deve ser feita com cuidado e a implantação de proteções anti-DDoS dedicadas, além do *firewall* ou IPS, é recomendada em ambientes altamente críticos [26].

2.3 *Machine Learning*

Atualmente, a magnitude da coleta diária de dados em escala global é extraordinária, impulsionada pela proliferação de dispositivos conectados à internet, plataformas digitais e interações cotidianas. Essa imensa quantidade de dados abrange uma diversidade impressionante de formatos, incluindo texto, áudio, imagens, além de registros detalhados, como transações comerciais, formando um panorama complexo e rico em informações. [41]. Deste modo, e com uma quantidade tão grande de dados, torna-se importante saber como extrair, da forma mais adequada, informação a partir destes. Quando se está perante a uma pequena quantidade de dados, pode-se fazer essa extração de forma analógica ou manual, no entanto, e tendo em conta que a quantidade de dados disponíveis para análise aumenta significativamente de dia para dia, torna-se uma tarefa impossível para que os seres humanos consigam processar todos estes dados, fazendo, ao mesmo tempo, uma análise. Quando tal acontece recorre-se à tecnologia, nomeadamente, aos computadores ou outro tipo de máquinas, de forma a processar e extrair informações, de forma mais rápida e eficaz [41]

O que acontece é que uma aprendizagem automática é tão relevante para tantas indústrias e sectores nos dias de hoje, onde, através da procura de padrões nos dados fazendo uso da aplicação de algoritmos computacionais e descobrindo regularidades, pode-se classificar esses dados em categorias, facilitando, deste modo a extração de informação de um conjunto grande de dados. É neste sentido que surge o *machine learning*.

O *machine learning* pode ser definido como sendo uma aplicação de inteligência artificial que permite que os sistemas aprendam a melhorar com a sua experiência sem que sejam programados, de forma explícita. Tal veio suprir uma das grandes diferenças entre os seres humanos e os computadores. Enquanto um ser humano tende a melhorar, de forma automática, a forma como resolve e lida com os problemas, através da aprendizagem baseada na análise dos erros cometidos anteriormente e na sua resolução e correção, procurando novas abordagens para abordar esse mesmo problema, um programa de computador tradicional não olha para os resultados das suas tarefas, sendo, por isso, incapazes de melhorar o seu comportamento [9]. O *machine learning* refere-se à capacidade de desenvolver aplicações que aprendem e aprimoram seu desempenho ao longo do tempo. Isso significa que, ao serem expostas a dados e experiências, essas aplicações podem, por exemplo, aperfeiçoar a precisão de previsões, otimizar recomendações ou identificar padrões complexos de maneira autônoma..

2.3.1 Modos de aprendizagem

O *machine learning* pode ser classificado tendo em conta os seus tipos de estratégias de aprendizagem, que são geralmente identificados tendo em conta a quantidade de inferência que a aplicação ou programa de computador é capaz de realizar. Deste modo, identificam-se as seguintes estratégias de aprendizagem subjacentes ao *machine learning* [8].

Aprendizagem de rotina (*Route Learning*) diz respeito a estratégias que todos os programas de computador tradicionais utilizam. Neste tipo de estratégias todo o conhecimento do programa é implementado diretamente pelo programador, não se verificando deste modo a realização de qualquer tipo de inferências. Deste modo, o programa não apresenta a capacidade de desenhar qualquer tipo de conclusões ou transformar-se a partir das informações que transmite.

A aprendizagem a partir de instruções (*Learning from Instructions*) abarca todos os programas e aplicações que são capazes de transformar informação numa determinada linguagem de entrada para uma linguagem interna. Apesar do conhecimento de como esta transformação pode ser realizada de forma eficiente ela pode ainda ser dada pelo programador. Esta requer pequenas formas de inferência por parte do programa ou aplicação. Deste modo é definido um nível separado ao nível de sistema de aprendizagem comparativamente com a aprendizagem de rotina.

A aprendizagem por exemplos (*Learning from Examples*) é das estratégias mais comuns utilizadas e permite uma maior flexibilidade, permitindo aos programas e aplicações

o desenvolvimento de competências completamente desconhecidos, ou possibilita que estes encontrem estruturas e padrões desconhecidos nos dados.

2.3.2 Componentes do *machine leaning*

No que concerne aos componentes do processo de *machine learning*, podem ser considerados quatros componentes básicos: i) definição do problema; ii) dados; iii) representação; e iv) algoritmos e treino, tal como ilustrado na Figura 2.4.

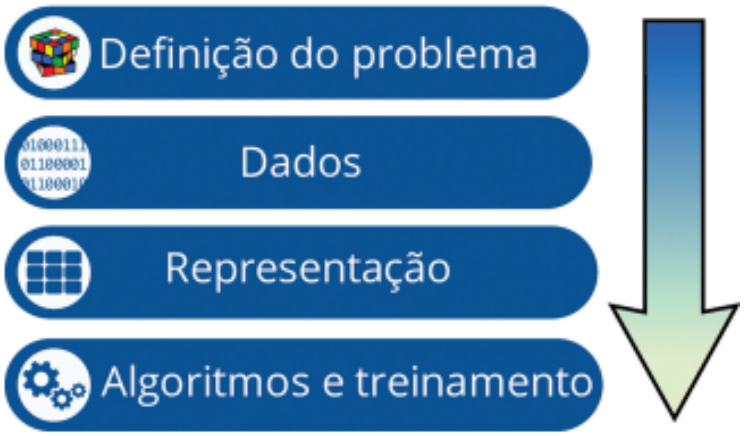


Figura 2.4: Componentes básicos do processo de *machine learning* [50].

O primeiro componente essencial do *machine learning* é, então, a definição do problema. Neste componente o problema de aprendizagem define-se por um determinado conjunto de dados conhecido, X, denominado de *input features* (ou características ou atributos), prever ou aproximar uma função de interesse desconhecido $y = f(X)$, em função dos dados conhecidos. Já um elemento de X é denominado de vector de *features* ou, simplesmente, de entrada [50].

No contexto do segundo componente, os dados, comumente referidos como entradas (*inputs*), desempenham um papel fundamental no desenvolvimento de qualquer processo de *machine learning*. A qualidade e quantidade desses dados são determinantes para os resultados alcançados. A qualidade dos dados é crucial, exigindo que sejam verdadeiramente representativos do problema em questão, consistentes e enriquecidos com informações relevantes [50].

O terceiro componente denomina-se por representação, sendo também conhecido por impressão digital *fingerprint* ou descritor, e este diz a respeito sobre a determinação da capacidade de desempenho do processo de *machine learning*. O algoritmo apenas

irá aprender a inferência desejada se estiverem representadas as variáveis necessárias [29].

O último componente diz respeito aos algoritmos e treino. A tarefa de construção e treino dos algoritmos varia de caso para caso, não existindo nenhum algoritmo que seja mais adequado do que outro. Deste modo, a escolha e conhecimento do algoritmo de aprendizagem a utilizar é uma etapa essencial no que se refere à construção de um processo de *machine learning* [50].

2.3.3 Tipos de *machine learning*

Michel afirma em [9] que existem vários tipos de *machine learning*, tendo em conta a estrutura do problema. Os mais utilizados são o *machine learning* supervisionado e o *machine learning* não supervisionado, identificando-se, também, a aprendizagem semi-supervisionada e de reforço.

As técnicas de *machine learning* sem supervisão requerem apenas a entrada de *feature values* nos dados de treino, e o algoritmo de aprendizagem descobre as estruturas escondidas nos dados de treino baseados neles de forma implícita. Exemplos de técnicas de *machine learning* não supervisionadas são as técnicas de agrupamento que tentam dividir e agrupar os dados em grupos coerentes. Um exemplo destas técnicas diz respeito à segmentação de mercado, onde as pessoas são agrupadas segundo os seus comportamentos sociais e a categorização dos artigos é realizada de acordo com os seus temas mais populares, envolvendo, deste modo, tarefas de agrupamento e aprendizagem não supervisionada. A extração de padrões constitui, assim, uma conhecida forma de *machine learning* não supervisionado [4].

O outro tipo de *machine learning* bastante comum, é o *machine learning* supervisionado. A diferença deste para o não supervisionado é que o supervisionado necessita do valor da variável de saída para cada amostra de treino. Deste modo, cada amostra de treino é representada sob a forma de um par onde se encontram os valores de entrada e saída. O algoritmo vai, então, treinar um modelo que irá prever o valor das variáveis de entrada e de saída recorrendo às características que foram previamente definidas no processo. Se a variável de saída é valorizada de forma contínua, então, ao modelo preditivo dá-se o nome de função de regressão. Um exemplo de uma função de regressão é a previsão da temperatura do ar numa determinada altura do ano. Se a variável de saída for um conjunto discreto de valores, então estamos perante um modelo classificador. O diagnóstico médico automático, onde os doentes são classificados como tendo ou não uma determinada doença através da análise dos sintomas, é um bom exemplo de um modelo classificador [4].

Existe também o *machine learning* semi-supervisionado, encontrando-se entre os dois tipos anteriores, podendo este ser mais vantajoso visto que os dados não rotulados são mais acessíveis que os dados de alta qualidade rotulados. Este tipo de *machine learning* opera como um pequeno conjunto de dados de treino rotulados (ou seja, supervisionados) e um conjunto de dados maior não rotulados (ou seja, não supervisionados). Quando se treina um modelo de *machine learning* de previsão semi-supervisionado, os algoritmos têm a capacidade de analisar tantos os valores de saída supervisionados assim como a distribuição dos dados não supervisionados [4].

O último tipo de *machine learning* diz respeito ao *machine learning* de reforço, onde se vai observar um reforço da aprendizagem. Neste tipo a aprendizagem dá-se de um modo sensivelmente diferente quando comparado com os outros tipos, uma vez que estes aprendem tendo em conta exemplos, teorias lógicas, funções e probabilidades. Já o *machine learning* de reforço diz respeito ao modo como os agentes conseguem aprender quando não existem exemplos rotulados. É portanto um método de aprendizagem que vai interagir com o ambiente, produzindo, deste modo, ações, e descobrindo no mesmo tanto recompensas como erros. Assim, este tipo permite que programas de computador e aplicações determinem de forma automática qual o comportamento ideal dentro de um determinado contexto, de modo a que o desempenho do agente seja maximizado. É portanto necessária a existência de um sistema simples de recompensa e *feedback* de modo que o agente aprenda qual a melhor ação [44].

2.3.4 Principais algoritmos de *machine learning*

Existe uma grande variedade de algoritmos, conforme Figura 2.5, que podem ser utilizados em *machine learning*, sendo que cada um apresenta uma finalidade específica. Para atingir esse objetivo, será realizada uma descrição detalhada de cada algoritmo, destacando suas distinções individuais, vantagens e limitações. Além disso, será explorado como esses algoritmos podem ser aplicados em diversos ambientes, evidenciando sua adaptabilidade e eficácia em cenários distintos. Este exame minucioso proporcionará uma compreensão mais profunda dessas ferramentas, permitindo uma escolha informada sobre a seleção e implementação dos algoritmos mais apropriados para diferentes contextos em aplicações de machine learning.

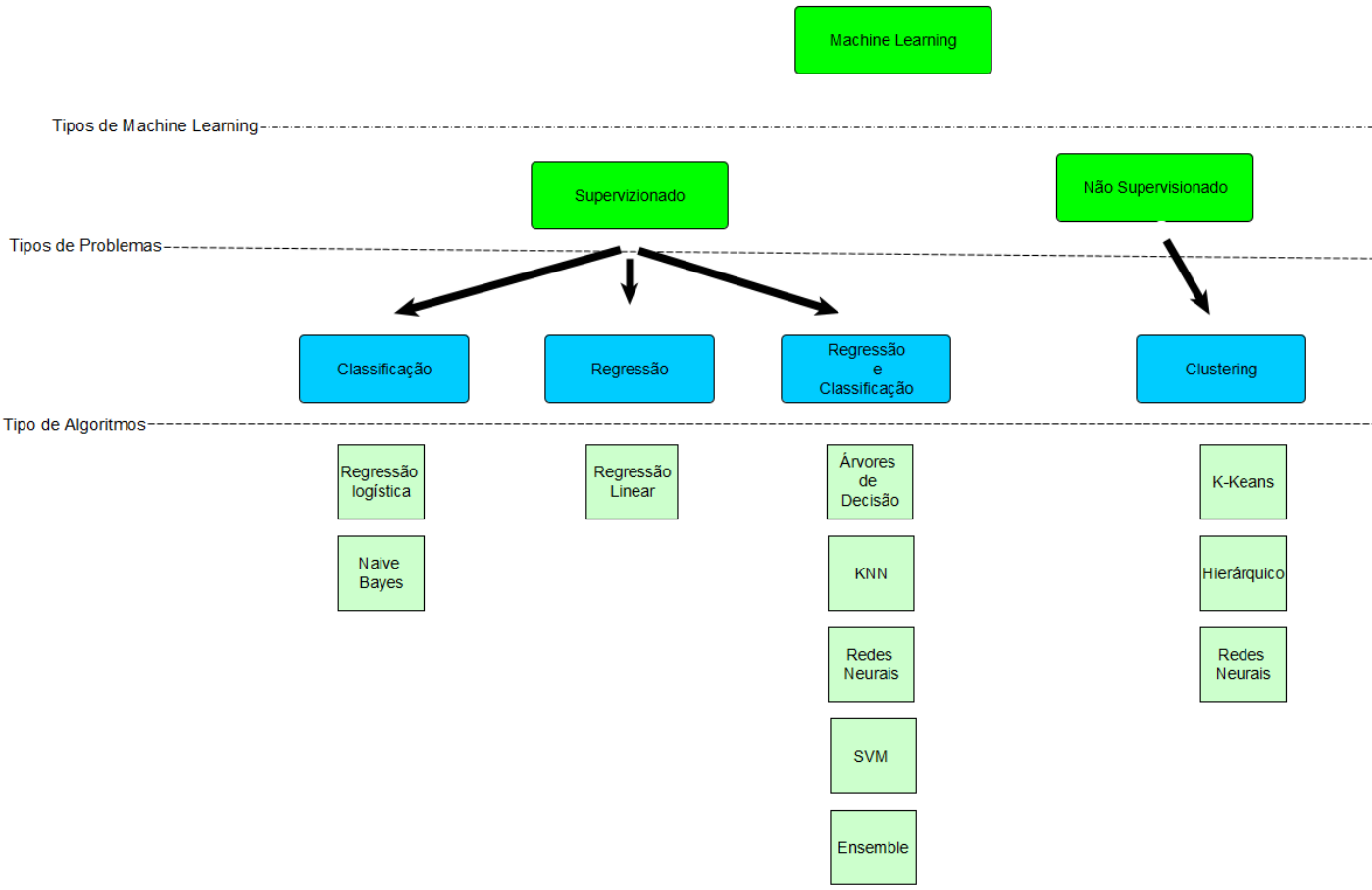


Figura 2.5: Algoritmos de *machine learning*.

2.3.4.1 Regressão linear

A regressão linear diz respeito a um dos algoritmos mais utilizados dentro do *machine learning* e consiste numa reta tração a partir de uma determinada relação, representada num diagrama de dispersão. Esta reta sistematiza a relação entre os dados de duas variáveis, sendo, deste modo, muito útil para a realização de previsões [55]. O resultado da regressão linear é sempre um valor, sendo utilizada de forma adequada quando o *dataset* apresenta alguma tendência de crescimento ou diminuição constante.

A regressão linear é um modelo atraente porque a representação é muito simples. A representação é uma equação linear que combina um conjunto específico de valores de entrada (x) e a solução para a qual é a saída prevista para esse conjunto de valores de entrada (y). Como tal, os valores de entrada (x) e o valor de saída são numéricos [7]. A equação abaixo ilustra um problema simples de regressão linear:

$$y = B_0 + B_1 \cdot x$$

2.3.4.2 Regressão logística

O modelo de regressão logística é bastante semelhante ao modelo de regressão linear, estabelecendo uma relação entre as variáveis explicativas e a probabilidade de ocorrer ou não o fenómeno em estudo, permitindo a criação de uma variável binária para estimar a probabilidade de se classificar como (1) sucesso, ou (0) fracasso.

Este algoritmo é especialmente aplicado a modelos lineares para resolver problemas de classificação. Assim, busca-se prever a probabilidade de cada observação pertencer a uma das classes da resposta em interesse. Para a estimativa dessa probabilidade, a variável resposta do conjunto de treino deve ser modelada de acordo com uma distribuição binomial [49].

Este método é aplicado a modelos de classificação, sendo o *output* uma variável categórica, isto é, calcula-se uma probabilidade cujo valor classifica o *output* como pertencente a uma categoria ou classe, sendo este condicionado por X, ou seja, $P(Y|X)$ [49].

2.3.4.3 Árvores de decisão

Uma árvore de decisão refere-se a um grafo que ajuda a determinação de um conjunto específico de ações, onde se podem observar vários ramos em que cada ramo corresponde a um resultado diferente ou uma possível reação a um problema. A Figura 2.6 representa de maneira simplificada uma árvore de decisão com dois nós, onde o resultado é ir à praia ou não ir.

Figura 2.6: Árvore de Decisão [2].

Em *machine learning* este é um tipo de aprendizagem supervisionado, onde os dados são divididos em várias entidades possíveis relativamente a um parâmetro específico. O método de aprendizagem supervisionada utiliza a árvore de decisão como modelo

preditivo para analisar a observação de um item nos ramos para concluir o valor-alvo do item nas folhas. Os nós de decisão representam a divisão dos dados, e as folhas representam os resultados [24].

As árvores de decisão é um dos métodos poderosos comummente utilizados em vários campos, tais como a aprendizagem mecânica, o processamento de imagens e a identificação de padrões [56]. As árvores de decisão são um modelo sucessivo que une uma série de testes básicos de forma eficiente e coesa, onde uma característica numérica é comparada com um valor limiar em cada teste [56].

2.3.4.4 Floresta aleatória

Uma Floresta aleatória é um algoritmo de aprendizagem supervisionada por máquina que é construído a partir de algoritmos baseados em árvores de decisão, até que se forme uma "floresta". Este algoritmo estabelece o resultado com base nas previsões das árvores de decisão. Prevê tendo em conta média dos resultados de várias árvores. Ao aumentar o número de árvores aumenta a precisão do resultado [39].

Os mesmos autores referem as seguintes vantagens na utilização deste algoritmo: demora menos tempo a treinar quando comparados com outros algoritmos; consegue prever a produção com alta precisão, mesmo para um grande conjunto de dados, correndo de forma eficiente, e por fim, é capaz de manter a precisão quando falta uma grande proporção de dados.

As florestas aleatórias utilizam diferentes técnicas para melhorar o seu desempenho. O "Feature Bagging"é uma técnica utilizada no algoritmo florestas aleatórias do scikit-learn para melhorar o desempenho das árvores de decisão em problemas de aprendizado de máquina. Enquanto o florestas aleatórias já utiliza o método de Bagging, que consiste em criar várias árvores de decisão treinadas em subconjuntos aleatórios do conjunto de dados de treinamento, o "Feature Bagging"acrescenta uma camada adicional de aleatoriedade ao processo. A principal ideia por trás do "Feature Bagging"é a seleção aleatória de características (ou variáveis) durante a construção de cada árvore de decisão .

2.3.4.5 Redes neuronais artificiais

Di Franco define em [22] esta abordagem como sendo caracterizada pela utilização de algoritmos para extrair informação de um grande conjunto e heterogéneo de dados. A característica mais importante deste tipo de algoritmos é que a sua estrutura replica a rede de neurónios do cérebro humano. Deste modo as redes neuronais artificiais são

sistemas compostos por vários nós que se ligam e se encontram em várias ramificações. As redes neuronais artificiais aprendem, então, por via de atualização e ampliações dessas ligações e ramificações.

Matematicamente, uma rede neuronal artificial é uma aplicação não linear em relação aos seus parâmetros θ, que associa uma entrada x a uma saída $y = f(x, \theta)$. Para simplificar, assume-se que y é uni-dimensional, mas também pode ser multidimensional. A aplicação f tem uma forma particular que precisaremos. As redes neuronais podem ser utilizadas para regressão ou classificação.

Como é habitual em aprendizagem estatística, os parâmetros θ, são estimados a partir de uma amostra de aprendizagem. A função para minimizar não é convexa, levando a minimizadores locais. O sucesso do método veio de um teorema de aproximação universal devido aos estudos de Cybenko e Hornik . Além disso, Le Cun propôs uma eficiente forma de calcular o gradiente de uma rede neuronal, chamada realimentação, que permite facilmente obter um minimizador local do critério quadrática [38].

2.3.4.6 Máquina de vetores de suporte

O conceito de máquina de vetores de suporte (support vector machine) é visto como sendo um modelo supervisionado de *machine learning* que recorre a algoritmos para resolver os problemas de classificação em dois grupos. Este algoritmo tem duas importantes vantagens: maior velocidade e melhor desempenho com um número limitado de amostras. Isto torna este algoritmo adequado para a resolução de problemas de classificação de texto [3].

Neste algoritmo cada item referente aos dados que vão ser traçados como um ponto no espaço n-dimensional (onde n é um número de características que apresenta), sendo o valor de cada característica o valor de uma coordenada específica. Depois, efetua-se a classificação encontrando o hiperplano que diferencia muito bem as duas classes [3]. Em suma, os vetores de apoio são, simplesmente, as coordenadas de observação individual.

2.3.5 *Features*

2.3.5.1 *Feature dimensionality reduction*

Conjuntos de dados que possuem uma grande quantidade de características são conhecidos como dados de alta dimensão. Frequentemente, esses conjuntos contêm informações redundantes, incluindo fatores relacionados ou duplicados. Surge, então, a

necessidade de redução de dimensão (*feature dimensionality reduction*) para eliminar essas interferências, utilizando recursos já existentes para formar um espaço de recursos de baixa dimensão [37]. A redução de dimensão é uma técnica essencial na análise de dados de alta dimensão, pois não apenas simplifica a representação dos dados, mas também preserva as informações mais relevantes. Isso resulta em conjuntos de dados mais gerenciáveis e eficientes para análise e modelagem, contribuindo para aprimorar a precisão e a eficácia de algoritmos de aprendizado de máquina e análise estatística.

Contudo, o processo de mapeamento de dados de alta dimensão para baixa dimensão através de projeções leva à perda de informação, inevitavelmente. O problema que precisa ser resolvido no momento, é obter dados de redução úteis do conjunto de dados de alta dimensão para atender à precisão de reconhecimento e aos requisitos de armazenamento sob a premissa de manter as características essenciais dos dados originais de maneira ideal. No entanto, em muitas situações práticas, a identificação e aquisição destas características são demasiado desafiantes, o que por sua vez vai tornar a redução da dimensão uma das tarefas essenciais e mais desafiantes em *machine learning* [37].

Segundo [53] a redução da dimensão também reduz drasticamente o custo do processo de *machine learning* e permite resolver problemas complexos com modelos simples. É também, uma técnica especialmente útil em modelos preditivos, pois são conjuntos de dados que contém um grande número de características de entrada, e torna a sua função mais complicada.

2.3.5.2 *Feature selection*

Há mais que uma forma de assegurar a redução da dimensionalidade em *machine learning*, sendo uma delas a seleção de atributos *(feature selection)*. Este, por sua vez, seleciona um subconjunto de atributos originais que contém as informações que são relevantes manter [48].

Segundo [48], estudos mais recentes têm feito um esforço enorme em aplicar algoritmos de seleção de atributos ao desenvolver modelos de previsão de rendimento de culturas (colheita em agricultura), sendo que estes podem ser divididos em três grupos: (1) aplicar seleção de atributos sem avaliar a sua contribuição; (2) investigar qual o método de seleção de atributos mais adequado; (3) aplicar seleção de atributos e indicar a sua eficácia em comparação com a abordagem de não seleção de atributos. Este tipo de subdivisão pode ser aplicada, também, a outros meios e setores.

Já [57] divide os métodos de seleção de atributos em dois: seleção de atributos não

supervisionada e supervisionada. Na seleção de atributos não supervisionada, o relacionamento da variável de destino não é considerado. Aqui, determina-se a correlação entre os atributos, sendo que se houvesse dois atributos correlacionados não haveria necessidade de ambos. No caso da seleção de atributos supervisionado, o mesmo vai ter em conta o relacionamento de destino - portanto, o relacionamento entre cada um dos atributos e o destino (ou) o rótulo será usado na seleção de atributos. Os métodos que se enquadram na seleção de atributos supervisionados incluem métodos de filtro, métodos de *wrapper* e métodos incorporados.

2.3.5.3 *Reconstruction/model-based methods*

Como já referido, *machine learning* é utilizado para realizar diversas análises. O desempenho dessas análises num modelo de *machine learning* vai depender muito da configuração de hiperparâmetros correspondentes. Os hiperparâmetros são parâmetros externos ao modelo que não são aprendidos durante o treinamento, mas desempenham um papel crucial na definição do comportamento do algoritmo [30]. Exemplos comuns incluem a taxa de aprendizado em algoritmos de otimização ou o número de árvores em um modelo de floresta aleatória.

Assim, o ajuste de hiperparâmetros é frequentemente indispensável, embora tenham custos consideráveis, tais como o tempo e recursos computacionais necessários para explorar diversas combinações de configurações. Encontrar a combinação ideal de hiperparâmetros pode ser comparado a ajustar os controles de uma máquina para atingir o melhor desempenho possível. Esse processo de otimização é essencial para garantir que o modelo seja capaz de generalizar bem para novos dados.

Nesse sentido, a busca por hiperparâmetros adequados é muitas vezes realizada por meio de técnicas como busca em grade ou otimização bayesiana. A busca em grade envolve a exploração sistemática de diferentes combinações de hiperparâmetros, enquanto a otimização bayesiana utiliza métodos estatísticos para focar em regiões promissoras do espaço de hiperparâmetros, reduzindo assim o custo computacional associado à busca.

Portanto, embora o ajuste de hiperparâmetros possa ser um desafio, é uma etapa crucial para otimizar o desempenho de modelos de *machine learning* e garantir resultados mais precisos e confiáveis. Normalmente, esse ajuste exige que a *machine learning* esteja ajustada ao máximo desempenho, mas nem sempre é possível armazenar todos os dados, seja por privacidade, limitações de espaço ou outros. Mas se os dados tiverem que ser transferidos por meio de ligações de baixa largura de banda é possível reduzir

o tempo de ajuste. Model-Based Optimization (MBO) é um método de última geração para ajustar esses hiperparâmetros e otimizar o processo ao máximo [51].

[51] propõem um método MBO para ajustar os hiperparâmetros para algoritmos de *machine learning*, no qual utilizam uma caixa preta fictícia onde todo o sistema está distribuído, e otimizam o desempenho dessa caixa ao máximo, medindo a precisão média de previsão e a estabilidade estatística e/ou eficiência do tempo de execução.

Mas a aplicação de abordagens em modelos (model-based) depende muito das estatísticas iniciais, tais como a especificação de relação entre variáveis (por exemplo, independência) e as suposições específicas do modelo em relação às distribuições de probabilidade do processo (por exemplo, a variável de resultado pode ser necessária para ser binomial). Assim, o processo de classificação pode ser realizado com base nas probabilidades estimadas. Os investigadores devem examinar e confirmar cuidadosamente as suposições do modelo e escolher as funções de ligação que são efetivamente apropriadas. Como as suposições estatísticas nem sempre são válidas em problemas da vida real, especialmente para grandes dados incongruentes, como já referido, os métodos baseados em modelos podem não ser aplicáveis ou podem gerar resultados tendenciosos [27].

2.3.6 Discussão

Analisando os métodos apresentados acima, concluo que o mais adequado ao problema a ser tratado nesta dissertação é a árvore de decisão. Árvores de decisão possuem muitas vantagens quando comparadas com outros métodos de classificação: elas têm uma estrutura facilmente interpretável, sendo também menos suscetíveis a problemas de dimensionalidade. [31].

Árvores de decisão são mais simples de serem lidas e interpretadas, pois não exigem conhecimentos estatísticos. Além disso, elas são menos suscetíveis à *maldição da dimensionalidade*, que se refere ao aumento do erro à medida que o número de características aumenta.

Outro ponto levado em consideração para a sua escolha é que árvore de decisão requer menor esforço para preparação durante o pré-processamento. Muitos modelos de *machine learning* podem exigir um pré-processamento pesado de dados, como normalização, e podem exigir esquemas de regularização complexos.

Devido a estes benefícios e outros, como o facto de ser um modelo que precisa apenas ser construído uma vez para ser utilizado de maneira repetitiva, a árvore de decisão se torna a solução mais adequada a este trabalho.

2.4 Trabalhos Relacionados

Diferentes literaturas abordaram a problemática dos ataques de negação de serviço, durante os últimos anos, devido ao aumento de sua importância. Alguns desses fizeram uso de algoritmos de *machine learning*, tanto para prever se o acesso é legitimo ou ilegítimo, entre outros aspectos. Nesta secção serão mencionados alguns desses trabalhos e como foram desenvolvidos.

O trabalho [21] desenvolveu uma nova taxonomia para os ataques de DDoS. Baseado nesta taxonomia foi criado um *dataset* para diferentes ataques e aplicado algoritmos de *machine learning* para classificação. Este trabalho criou e recolheu os seus próprios dados para análise, utilizando duas redes distintas. Uma destas redes gerou os ataques e tráfego legítimo, enquanto a outra respondia a essas requisições. Os ataques de DDoS foram variados desde NTP, DNS, Lightweight Directory Access Protocol (LDAP), etc.. Após a construção do *dataset* foram aplicados algoritmos de *machine learning* para avaliação das seguintes métricas: precisão, *recall* e F1-score. Sendo que as técnicas *floresta aleatória* e ID3 alcançaram o maior acerto.

Já [23] focou o seu trabalho em ataques de DDoS em ambientes de IoT. Para construção de seu *dataset* foram utilizados 4 dispositivos (câmara, WeMo Smart, Servidor HD/RD e um telefone android) durante 10 minutos somados. E para simular o tráfego malicioso foi feito uso da ferramenta Hping3. Para classificação foram utilizados quatro diferentes algoritmos de *machine learning*: *K-nearest*, árvore de decisão, floresta aleatória, rede neuronal, sendo que todos os quatro algoritmos tiveram taxas de acerto superior a 0.99.

Outro trabalho que tinha como objetivo comparar a performance de diferentes algoritmos de *machine learning* foi o [47]. Este trabalho fez uso do DARPA *dataset* [**MIT**], sendo este público, e a sua premissa foi comparar o desempenho dos algoritmos de máquina de vetores de suporte e *deep feed forward*, nos seguintes indicadores: *recall*, precisão e F1-score.

O trabalho em [19] propôs um modelo que tinha como objetivo detetar o ataque DDoS, o mais longe possível do alvo, dentro do domínio do ISP. Este modelo utilizava os algoritmos de *machine learning* e redirecionava o tráfego suspeito para *honeypot* para uma segunda análise.

Outro trabalho que estudou a problemática DDoS foi o [40], em que são utilizados filtros em linux para mitigar os ataques. Neste estudo foram comparadas 3 diferentes ferramentas de filtragem (iptables, nftables e netmap). Para os testes foi utilizada

uma ferramenta apropriada para geração de pacotes em larga escala, spirent avalanche 3100b, emulando assim diferentes ataques de negação de serviço. Após análise foi constatado que o netmap apresentou um melhor desempenho geral para filtrar ataques.

Todos os estudos mencionados acima tratam do problema, os ataques de negação de serviço. Parte deles [21], [23], [47] focam-se apenas em classificar o tráfego usando algoritmos de *machine learning*. Outros lidam com a forma como tratar o tráfego maligno [19], [40]. Sendo também que em parte destas literaturas foram criados ambientes de teste para simular os problemas e colecionar as informações utilizadas para construção do *dataset*, outros já optaram por fazer uso de *datasets* públicos.

Este estudo diferencia-se dos anteriores no sentido de juntar ambas as fases do processo: a classificação do acesso, e também o bloqueio em caso de acesso não legítimo. Para tal será construída uma topologia para simular os ataques, colecionar os *datasets* de acessos legítimos e não legítimos. Serão depois utilizados algoritmos de *machine learning* para classificar os acessos e, após esta etapa, a aplicação de *scripts* em Python juntamente com a biblioteca netmiko para automatizar o *deploy* de políticas do *firewall* da Fortinet para bloquear requisições maliciosas.

2.5 Conclusões

Neste capítulo foram apresentados e discutidos os conceitos relevantes para o desenvolvimento deste projeto, incluindo os principais ataques de DDoS, conceitos de *machine learning* e uma análise detalhada aos trabalhos relacionados com a mesma temática. No próximo capítulo, iniciaremos a parte prática do projeto com a implementação da topologia, criação do conjunto de dados e a sua análise.

3

Aquisição e análise de dados

O Capítulo 3 deste trabalho tem como objetivo apresentar a metodologia utilizada para construir a topologia de redes que será utilizada para construir o *dataset* de acessos legítimos e não legítimos. A construção dessa topologia é fundamental para permitir que sejam simulados diferentes cenários de ataques DoS numa rede real, o que possibilitará o treino e teste dos modelos de deteção de ataques que serão posteriormente desenvolvidos.

O *dataset* a ser construído será baseado em dados colecionados a partir de simulações de ataques DoS em diferentes cenários. Esses dados serão utilizados para treinar e validar o modelo de deteção de ataques.

3.1 Topologia de rede

Para a construção do *dataset* a ser utilizado neste estudo, foi criado uma topologia de rede composta por diferentes dispositivo. Tendo como objetivo emular uma rede corporativa, onde os funcionários fazem acesso a serviços corporativos de maneira remota, sem o uso de uma Virtual Private Network (VPN). A Figura 3.1 ilustra a topologia de rede a utilizar nesta dissertação.

Este ambiente consiste em duas máquinas, uma representando um utilizador legítimo que solicitará aleatoriamente recursos HTTPs e File Transfer Protocol (FTP) do servidor e a outra utilizada para gerar tráfego malicioso. Este cenário ainda possui uma *firewall*

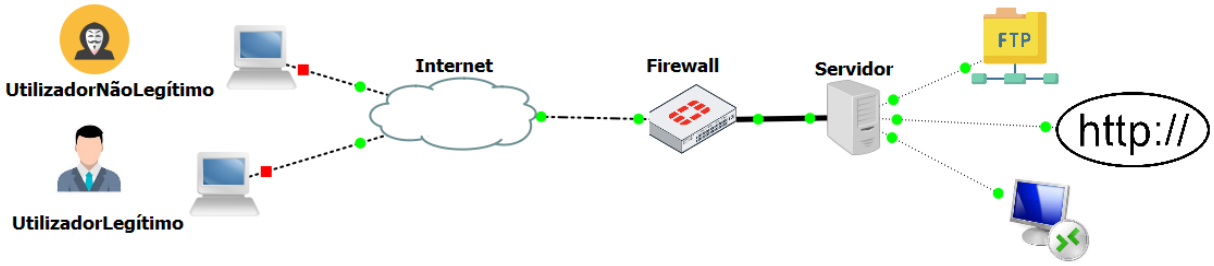


Figura 3.1: Topologia de rede base para o estudo.

que re-encaminha as requisições e o já mencionado servidor responsável por trata-las. A Tabela 3.1 detalha esta topologia.

Tabela 3.1: Detalhes da topologia.

Dispositivos	Sistema Operativo	Endereço IP
Firewall	FortiOs	200.0.0.2
Servidor	Debian	172.16.0.2
Utilizador Não Legítimo	Kali	Aleatório
Utilizador Legítimo	Ubuntu	12.12.12.1

3.2 Ferramentas utilizadas e ataques

Nesta secção são apresentadas as ferramentas e os tipos de ataques (acessos não legítimos) utilizados para construção do *dataset* que servirá de base ao desenvolvimento das heurísticas de deteção de ataques não legítimos.

3.2.1 Hping3

O Hping3 é uma ferramenta de linha de comando poderosa usada para pesquisar redes, criar pacotes personalizados e testar *firewalls*, entre outros [32]. Desenvolvido como uma extensão da ferramenta Hping original, o Hping3 foi projetado para ser usado por profissionais de segurança para testar a segurança de suas redes e identificar possíveis vulnerabilidades. Esta pode enviar pacotes TCP, UDP e ICMP e suporta vários tipos de pesquisa, incluindo pesquisa por Ping, *traceroutes* e pesquisa de

porta. Além disso, o hping3 suporta uma ampla variedade de opções de personalização, tornando-o uma ferramenta incrivelmente versátil para testes de segurança de rede.

Uma das características mais úteis do Hping3 é sua capacidade de criar pacotes personalizados com parâmetros específicos, como endereços IP de origem e destino, identificadores de portos e tipos de protocolo. Isso torna-o uma ferramenta valiosa para testar *firewalls*, sistemas de deteção de intrusão e outros mecanismos de segurança. O Hping3 também suporta *scripts*, permitindo que os utilizadores criem testes automatizados e os executem regularmente. Isso torna mais fácil identificar e remediar problemas de segurança antes que possam ser explorados por atacantes.

3.2.2 Slowloris

Slowloris é um tipo de ferramenta de ataque de negação de serviço que visa servidores web. Foi criado por Robert Hansen em 2009, e é amplamente utilizado nos dias de hoje. A ferramenta funciona estabelecendo várias ligações com um servidor web alvo e mantendo essas ligações abertas enviando solicitações HTTP incompletas. Isso faz com que o servidor use os seus recursos e eventualmente se torne incapaz de atender a solicitações legítimas.

O Slowloris funciona explorando uma fraqueza na forma como os servidores web lidam com as conexões HTTP [17]. Quando um cliente se liga a um servidor web, o servidor aloca recursos (como memória e capacidade de processamento) para lidar com a ligação. Tipicamente, o servidor espera que o cliente envie uma solicitação HTTP completa dentro de um determinado período de tempo. Se o cliente não enviar uma solicitação completa dentro desse tempo, o servidor fechará a ligação e libertará os recursos.

O atacante aproveita-se do comportamento descrito anteriormente estabelecendo várias ligações com um servidor web alvo e enviando solicitações HTTP incompletas. Especificamente, o Slowloris envia solicitações parciais que levam muito tempo para serem concluídas, como uma solicitação com um cabeçalho HTTP muito longo ou uma solicitação que está faltando os caracteres finais. Ao manter essas ligações abertas e enviar solicitações parciais, o Slowloris faz com que o servidor use os seus recursos e eventualmente se torne incapaz de atender a solicitações legítimas.

O Slowloris pode ser eficaz tanto contra servidores web apache quanto nginx, bem como outros tipos de servidores web. É um ataque de baixa largura de banda, o que significa que não requer muito tráfego de rede para ser eficaz. Além disso, como o

Slowloris não depende de uma vulnerabilidade específica no software do servidor, pode ser difícil se defender contra ele sem fazer mudanças na forma como o próprio servidor opera.

3.3 Experiências e coleção de dados

Por forma a gerar os *datasets* de análise através das ferramentas de *machine learning* foi gerado tráfego legitimo e ilegítimo durante 10 minutos, uma vez para cada ataque. Os acessos legítimos foram gerados de forma aleatória via *script*, utlizando a ferramenta wget para realizar as requisições HTTPs com o intuito de o tonar o mais fidedigno possível. Já o atacante realizava os seus ataques da seguinte maneira.

Num primeiro momento foi utilizada a ferramenta Hping3 para gerar ataques de inundação SYN, com a randomização de seu IP de origem, através da seguinte sintaxe:

sudo hping3 www.example.com -q -i 1 -c 1 -d 120 -S -p 443 –flood –rand-source

Em que cada argumento tem o seguinte significado:

- www.example.com: o endereço do site alvo;
- -q: informa ao Hping3 para executar no modo; silencioso, o que significa que não exibirá nenhuma saída além dos resultados finais;
- -i: define o intervalo entre os pacotes para 1 segundo;
- -c: define o número de pacotes a serem enviados para 1;
- -d: define o tamanho dos dados dos pacotes para 120 bytes;
- -S: define o sinalizador SYN no cabeçalhoTCP dos pacotes, que é usado para iniciar uma conexão TCP;
- -p: define o número da porta de destino como 443, que é a porta padrão para o tráfego https;
- –flood: define o envio de pacotes o mais rápido possível sem esperar por uma resposta;
- –rand-source: utiliza um endereço IP de origem aleatória para cada pacote, o que pode ajudar a evitar detecção ou filtragem por *firewalls*.

Num segundo momento foi utilizada a ferramenta Slowloris para realizar ataques de DoS de camada 7, através da seguinte sintaxe:

sudo python3 slowloris.py www.example.com -p 443

em que cada parâmetro tem o seguinte significado:

- www.example.com: o endereço do site alvo ;
- -p: define o número do porto de destino como 443, que é a porto padrão para o tráfego https.

3.4 Conversão

Utilizando as ferramentas mencionadas na Secção 3.3, foram geradas duas capturas no formato Packet Capture (PCAP), contendo todas as informações referentes ao fluxo de rede na topologia descrita, sendo importante referir que estas capturas foram realizadas na interface Wide Area Network (WAN) do *firewall*, interface que tem ligação com a Internet, e recebe os pacotes destinados ao servidor.

Após a captura de tráfego, foi necessários extrair as *features* a serem utilizadas neste estudo, e presentes na construção do *dataset*. Para este fim foi utilizado o *CICFlowMeter* que é uma ferramenta de análise de tráfego de rede de código aberto que permite a conversão de arquivos PCAP para arquivos do tipo Comma-Separated Values (CSV). Como resultado da conversão realizada, cada trama capturada é agora caracterizada por mais de 80 características nela observada. A lista completa das características de cada captura é apresentada no Anexo A, onde se podem encontrar:

- src_ip: Endereço IP de origem;
- dst_ip: Endereço IP de destino;
- src_port: Número da porta de origem;
- dst_port: Número da porta de destino;
- src_mac: Endereço MAC de origem;
- dst_mac: Endereço MAC de destino;
- protocol: Protocolo de comunicação usado (por exemplo, TCP, UDP, ICMP).

3.5 Pré-processamento de dados

O pré-processamento de dados é uma etapa crítica no processo de análise de dados. Consiste numa série de técnicas e procedimentos que visam preparar os dados para análises mais aprofundadas. O objetivo do pré-processamento de dados é melhorar a qualidade e a eficácia das análises, além de garantir que os dados sejam adequados para o modelo de análise escolhido. As técnicas de pré-processamento de dados incluem a limpeza de dados, a normalização, a redução de dimensionalidade, a discretização, a seleção de características, entre outras.

Conforme [6], alguns dos pontos a serem considerados no processo de redução de dimensionalidade são: *datasets* desorganizados, identificação e remoção de colunas que contém um único valor, identificação de colunas que têm poucos valores, identificação e remoção de colunas que têm baixa variância e identificação e remoção de linhas que contém dados duplicados.

Uma das questões frequentes em conjuntos de dados desorganizados, conhecidos como *"messy datasets"* (conjuntos de dados bagunçados), é a inclusão de colunas que mantêm constantemente o mesmo valor. Essas colunas não apresentam informações úteis e acabam ocupando espaço desnecessário no conjunto de dados. Para solucionar essa problemática, é crucial identificar essas colunas e removê-las, a fim de otimizar a utilidade e a eficiência do conjunto de dados.

Além das colunas que possuem apenas o mesmo valor, também é importante considerar as colunas que possuem poucos valores distintos. Essas colunas podem ser úteis para análises específicas, mas, em muitos casos, elas podem ser excluídas do conjunto de dados. A remoção dessas colunas pode tornar o conjunto de dados mais legível e mais fácil de ser analisado. Algumas *features* como exemplo *bwd_pkts_b_avg* e *bwd_blk_rate_avg* apresaram um único valor em todas suas linhas no *dataset*, neste caso 0, logo as mesmas foram removidas. A figura 3.2 ilustra alguns dos valores desses campos.

Outro problema que pode ser encontrado em conjuntos de dados desorganizados é a presença de colunas com baixa variância. Essas colunas não mudam o seu valor de forma considerável entre as diferentes amostras do conjunto de dados, o que significa que elas não fornecem muita informação útil. A remoção dessas colunas pode ajudar a tornar o conjunto de dados mais conciso e mais fácil de ser analisado.

Além das colunas, também é importante verificar as linhas do conjunto de dados em busca de dados duplicados. Essas linhas podem ser removidas do conjunto de dados para evitar a distorção dos resultados das análises realizadas com o conjunto de dados.

dst_mac	protocol	timestamp	flow_duration	flow_byts_s	...	bwd_pkts_b_avg	fwd_blk_rate_avg	bwd_blk_rate_avg	fwd_seg_size_avg	bwd_seg_size_avg	cwe_flag_count
:3e:b8:fd:00:02	17	2023-01-10 09:23:22	61.0	3.114754e+06	...	0.0	0.000000	0.0	95.000000	0.000000	0
:3e:b8:fd:00:02	6	2023-01-10 09:23:24	1903.0	1.040462e+05	...	0.0	0.000000	0.0	66.000000	66.000000	0
:3e:b8:fd:00:02	17	2023-01-10 09:23:27	47.0	4.255319e+06	...	0.0	0.000000	0.0	100.000000	0.000000	0
:3e:b8:fd:00:02	17	2023-01-10 09:23:32	48.0	4.166667e+06	...	0.0	0.000000	0.0	100.000000	0.000000	0
:3e:b8:fd:00:02	17	2023-01-10 09:23:37	50.0	3.800000e+06	...	0.0	0.000000	0.0	95.000000	0.000000	0
:3e:b8:fd:00:02	6	2023-01-10 09:23:24	15241116.0	1.006094e+03	...	0.0	0.000000	0.0	376.157895	481.588235	0
		2023-01-10									

Figura 3.2: Exemplo de colunas que possuem apenas um valor no *dataset* recolhido.

A remoção de linhas duplicadas também ajuda a tornar o conjunto de dados mais preciso e confiável para análises futuras.

Aplicadas as técnicas mencionadas anteriormente, o número de *features* foi reduzido de 80 para 64.

3.6 Correlação

A correlação é uma medida estatística que avalia a relação entre duas variáveis. A correlação pode ser positiva, negativa ou nula. Uma correlação positiva significa que quando o valor de uma variável aumenta, o valor da outra variável também aumenta. Por outro lado, uma correlação negativa significa que quando o valor de uma variável aumenta, o valor da outra variável diminui. Já uma correlação nula significa que não há relação entre as variáveis. A correlação é medida pelo coeficiente de correlação, que varia entre -1 e 1.

Selecionar *features* baseadas na correlação pode ser uma boa estratégia para construir modelos mais eficientes e precisos. Isso porque, quando duas variáveis têm uma correlação forte e positiva, por exemplo, é possível que ambas as variáveis forneçam informações semelhantes para o modelo. Isso pode resultar em *overfitting*, ou seja, o modelo pode estar a ajustar-se demais aos dados de treino e não ser capaz de generalizar para novos dados. Portanto, selecionar apenas uma das variáveis pode ajudar a evitar esse problema [18].

Partindo desse conceito, foram removidas as *features* que possuíam uma correlação forte entre elas: **-0.98 >= Correlação or <= 0.98** . Isso significa que eliminamos as *features* cujas correlações eram próximas de -1 (negativo) ou 1 (positivo). Essa escolha foi feita considerando valores menores ou iguais a -0.98 para correlações próximas a -1 (negativo) e maiores ou iguais a 0.98 para correlações próximas a 1 (positivo).

Utilizando como critério de desempate a *feature* com menor correlação em relação à coluna da "label", procedemos com a remoção. Essa "label"foi adicionada anteriormente ao *dataset*, contendo valores de 1 e 0 para identificar se o acesso é um ataque ou não. Abaixo, apresentamos um exemplo prático desse método:

- 1- As *features* **flow_pkts_s, fwd_pkts_s** tem uma correlação alta entre si :

  flow_pkts_s x fwd_pkts_s = 0.9871

- Sendo a correlação delas com a label a seguinte:

  flow_pkts_s x label = 0.009277
  fwd_pkts_s x label = 0.008989

  3- Logo a *features* **fwd_pkts_s** seria removida por possuir uma correlação mais fraca com a label.

Utilizando esta metodologia o número de *features* foi reduzido de 64 para 42, as mesmas estão listadas no fim do Anexo A. Sendo este o número final de *features* para a aplicação do modelo de *machine learning*.

3.7 Divisão do *dataset*

Dividir o conjunto de dados em treino, validação e teste é uma prática comum na área de *machine learning* e é essencial para o desenvolvimento de modelos precisos e abrangentes. Essa divisão permite avaliar o desempenho do modelo em dados não vistos e detetar possíveis problemas de *overfitting*, que ocorrem quando o modelo se ajusta demasiado aos dados de treino, mas não consegue generalizar para novos dados.

O conjunto de treino é usado para ajustar os parâmetros do modelo. O conjunto de validação é usado para ajustar os hiperparâmetros do modelo, como por exemplo o número de camadas ocultas numa rede neuronal ou o valor de regularização em um modelo de regressão. O conjunto de teste é usado para avaliar o desempenho final do modelo em dados não vistos. É importante que o conjunto de teste seja separado do conjunto de treino e validação desde o início do projeto, para que não haja vazamento de informação e para garantir que os resultados sejam imparciais.

A divisão correta do conjunto de dados pode ter um impacto significativo na qualidade do modelo final. Se o conjunto de treino for muito pequeno, o modelo pode não ser capaz de capturar todas as nuances dos dados e, portanto, pode ser muito simples e ter baixa precisão. Se o conjunto de validação for muito pequeno, pode haver uma

alta variância nos resultados do modelo e o processo de validação pode não ser fiável. Por outro lado, se o conjunto de teste for muito pequeno, pode ser difícil avaliar o desempenho do modelo em dados não vistos.

De forma contrária um conjunto de dados de treino muito grande pode trazer desafios significativos. O processo de treino pode exigir recursos computacionais consideráveis, como memória e tempo de processamento, o que pode dificultar ou impossibilitar a execução em algumas máquinas. Além disso, um conjunto de dados de treino muito grande pode resultar em *overfitting*, onde o modelo aprende a memorizar os exemplos de treino em vez de generalizar para novos dados [54].

Neste trabalho, e fazendo uso da biblioteca do Scikit-Learn em Python, o *dataset* original foi dividido em 3 *datasets* distintos, conforme Figura 3.3, um para treino, um para validação e um para teste. O *dataset* de treino é composto por 60 por cento dos dados recolhidos, obtidos de forma aleatória, e os outros dois *datasets* têm 20 por cento de informação, cada um. Sendo que conjunto de treino é utilizado para treinar o modelo, ajustando seus parâmetros e permitindo que ele aprenda padrões nos dados. Já o conjunto de validação é empregado para ajustar os hiperparâmetros do modelo e evitar o overfitting, enquanto o conjunto de teste é crucial para avaliar o desempenho real do modelo em dados não vistos anteriormente, fornecendo uma métrica objetiva de sua generalização para novos casos.

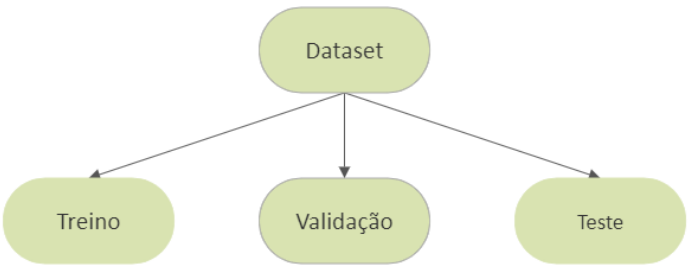


Figura 3.3: Divisão do *dataset*.

3.8 Aplicação dos algoritmos de *machine learning*

O uso de diferentes técnicas de aprendizagem de máquina é fundamental para a análise de dados. Nesse contexto, no âmbito deste projeto, foram utilizadas abordagens supervisionadas e não supervisionadas. Na categoria dos modelos supervisionados,

foi aplicado a *Floresta aleatória*, modelo que recebe dados rotulados como entrada e utiliza esses rótulos para aprender a fazer previsões precisas. Já na categoria de modelos não supervisionados, foi aplicado a *Floresta de Isolamento*. A *Floresta de Isolamento* é um modelo de deteção de anomalias que identifica pontos de dados que são incomuns em relação ao restante dos dados.

Essas técnicas são importantes, pois cada uma delas possui vantagens e limitações diferentes. Ao utilizar uma variedade de modelos, é possível obter uma visão mais completa e precisa dos dados, identificar padrões que podem não ser facilmente detetados com apenas um modelo e produzir resultados mais confiáveis.

3.8.1 Problema de classes não balanceadas

Conjunto de dados não balanceados podem ser um problema, pois o modelo pode aprender a prever a classe maioritária com precisão, mas ter um desempenho mau na classe minoritária, em alguns casos o modelo pode até ignorar completamente a classe minoritária [58]. O *dataset* construido para este trabalho possuía esta distorção conforme representado pela Tabela 3.2 abaixo, no qual a classe de acessos não legítimos compõem a maior parte do *dataset*, e a de legítimos representa menos de 1 por cento. Isso se deve ao facto dos ataques de DoS possuírem uma periodicidade muito maior que os acessos legítimos, e também de ter sido utilizado apenas um computador para gerar acesso legítimos.

Tabela 3.2: Número de ocorrências de cada classe.

Acessos Legítimos	Acessos Não Legítimos
1044342	58

Por forma a tratar esta problemática foram aplicadas técnicas de*upsampling* com Técnica de sobreamostragem sintética da classe minoritária (SMOTE). Esta técnica consiste em gerar amostras sintéticas para a classe minoritária, neste caso a classe de ataques. Os seguintes passos foram aplicados [58]:

1. Seleção de uma amostra aleatória da classe minoritária;
2. Identificação dos k vizinhos mais próximos da amostra selecionada;
3. Seleção aleatória de um dos k vizinhos e cálculo da diferença entre a amostra selecionada e o vizinho selecionado;

4. Multiplicação da diferença calculada por um número aleatório entre 0 e 1;

5. Adição do resultado da multiplicação a amostra selecionada para criar uma nova amostra sintética;

6. Repetição dos passos 1 a 5 até que o número especificado de amostras sintéticas seja gerado.

Depois de aplicar o SMOTE, a quantidade de valores nas classes de ataque e acesso legítimo foi igualada, o que resolveu artificialmente possíveis problemas de desequilíbrio do conjunto de dados. A Figura 3.4 demonstra a quantidade de dados em cada cada classe no final deste processo.

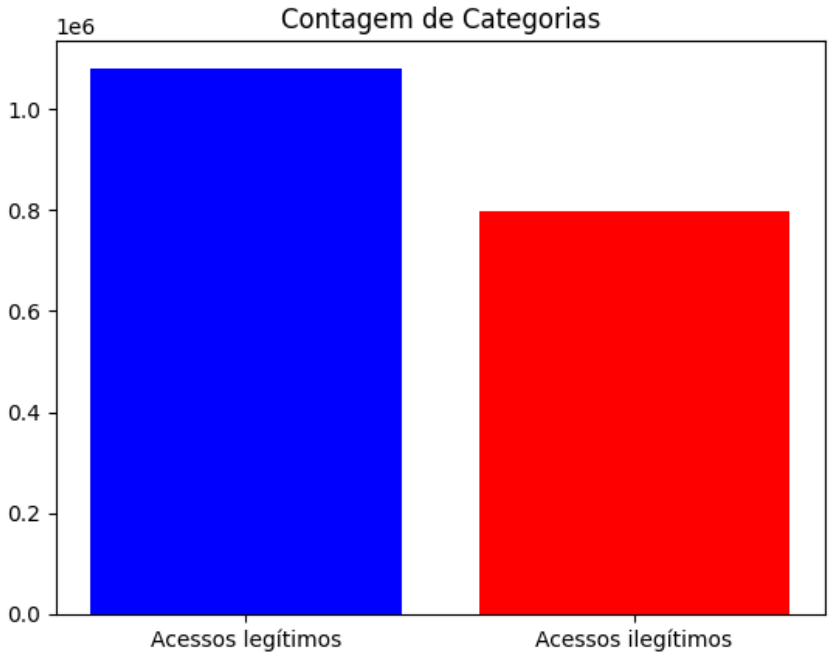


Figura 3.4: Quantidade de dados em cada classe após *upsampling*.

3.8.2 *Floresta aleatória*

A *floresta aleatória* é uma técnica de *machine learning* amplamente utilizada em problemas de classificação e regressão. Ela consiste num conjunto de árvores de decisão, onde cada árvore é construída com base numa amostra aleatória do conjunto de dados original. Essa abordagem permite que o modelo tenha uma maior capacidade de generalização e evite o sobreajuste, tornando-o mais eficiente na previsão de novos dados.

Este mesmo algoritmo foi selecionado devido aos resultados mencionados na Secção 2.3.4.4. Conforme discutido, ele apresenta tempos de treino inferiores em comparação com outros algoritmos. Além disso, é capaz de fazer previsões com alta precisão, mesmo para grandes conjuntos de dados, executando de forma eficiente. Além disso, sua capacidade de manter a precisão mesmo quando uma grande proporção de dados está faltando o torna uma escolha ideal para este contexto.

Na fase de treino com a *floresta aleatória* do scikit-learn, o mesmo precisou de cerca de 39 segundos para realizar o treino, o que valida a premissa anterior. Para o treino, os seguintes parâmetros para foram utilizados.

```
1 randomForest = RandomForestClassifier(n_estimators=50, max_depth=2, max_
      features=10, bootstrap=False)
2 randomForest.fit(X_train, y_train)
```

O primeiro parâmetro utilizado *n_estimators* refere-se ao número de estimadores (ou árvores de decisão) no algoritmo. Cada estimador contribui para a decisão final, e um maior número de estimadores geralmente resulta num modelo mais robusto, porém com maior tempo de treino e previsão.

O parâmetro *max_depth* controla a profundidade máxima e pode ajudar a evitar o sobreajuste (*overfitting*). Uma profundidade maior permite que as árvores sejam mais complexas e aprendam relações mais detalhadas nos dados, mas também pode levar a um modelo mais suscetível ao sobreajuste.

Já *max_features* define o número máximo de recursos (ou variáveis) que são considerados em cada divisão do nó durante a construção da árvore de decisão. No caso do algoritmo *floresta aleatória* do scikit-learn, a seleção de *features* é baseada numa técnica chamada *Feature Bagging* 2.3.4.4. A cada divisão de um nó da árvore, uma amostra aleatória de *features* é selecionada para determinar a melhor divisão. A Figura 3.5 representada as *features* escolhidas nas 5 primeiras divisões do algoritmo.

```
Árvore 1:
Features usadas: ['totlen_fwd_pkts', 'fwd_pkt_len_std', 'bwd_byts_b_avg', 'bwd_seg_size_avg']
Árvore 2:
Features usadas: ['totlen_fwd_pkts', 'fwd_pkt_len_std', 'bwd_pkt_len_std', 'bwd_seg_size_avg']
Árvore 3:
Features usadas: ['totlen_fwd_pkts', 'fwd_pkt_len_std', 'bwd_pkt_len_std', 'bwd_seg_size_avg']
Árvore 4:
Features usadas: ['totlen_fwd_pkts', 'fwd_pkt_len_std', 'pkt_len_max', 'bwd_seg_size_avg']
Árvore 5:
Features usadas: ['totlen_fwd_pkts', 'fwd_pkt_len_std', 'bwd_seg_size_avg', 'subflow_bwd_byts']
```

Figura 3.5: *Features* selecionadas no modelo *floresta aleatória*.

Por fim, o parâmetro *bootstrap* controla se a amostragem com reposição é usada para construir cada árvore na *floresta aleatória*. Se for definido como "False", a amostragem será realizada sem reposição, o que significa que cada árvore será construída a partir de uma amostra única dos dados. Definir como "True"permitirá a amostragem com reposição, o que significa que cada árvore pode conter várias instâncias dos mesmos dados.

Para avaliar o desempenho do modelo de *floresta aleatória*, utilizamos o relatório de classificação, que apresenta várias métricas para cada classe, bem como métricas agregadas.

Classe	Precisão	Recall	F1-Score	Support
0.0	1.00	1.00	1.00	216168
1.0	0.99	1.00	1.00	99804
Taxa de acerto			1.00	
Média Macro	1.00	1.00	1.00	315972
Média Ponderada	1.00	1.00	1.00	315972

Tabela 3.3: Tabela com os resultados obtidos com o modelo de *floresta aleatória*.

A precisão mede a proporção de instâncias classificadas corretamente em relação ao total de instâncias previstas numa determinada classe. No nosso caso, obtivemos uma precisão de 1.00 para a classe 0.0 e 0.99 para a classe 1.0.

O recall, também conhecido como taxa de verdadeiros positivos, mede a proporção de instâncias corretamente classificadas em relação ao total de instâncias reais numa determinada classe. Observamos um recall de 1.00 para ambas as classes.

O F1-Score é a média harmónica entre precisão e recall. É uma medida equilibrada que leva em consideração tanto a precisão quanto o recall. Neste caso, obtivemos um F1-Score de 1.00 para ambas as classes.

Além disso, podemos observar uma taxa de acerto geral de 1.00, o que significa que o modelo teve um desempenho muito bom em prever corretamente as classes em todo o conjunto de teste.

No geral, os resultados do modelo da *floresta aleatória* demonstraram um desempenho excecional, com altas taxas de acerto, recall e F1-Score em ambas as classes, indicando sua capacidade de lidar com o desbalanceamento dos dados e fazer previsões precisas. Esses resultados são encorajadores e validam a eficácia do algoritmo de *floresta aleatória* para a tarefa de classificação no nosso conjunto de dados.

Para avaliar o impacto dos parâmetros utilizados anteriormente, multiplicou-se os valores utilizados no treino anterior por 2, conforme ilustrado no seguinte excerto de código:

```
1    randomForest = RandomForestClassifier(n_estimators = 100, max_depth =
     4, max_features = 20, bootstrap = False)
2    randomForest.fit(X_train, y_train)
```

Observamos um aumento significativo no tempo de treino. Especificamente, o tempo necessário para treinar o modelo aumentou para 2 minutos, mais de 3 vezes em relação ao teste anterior. Esse aumento no tempo de treino ocorre devido ao aumento da complexidade do modelo.

Ao multiplicarmos os valores de "n_estimators", "max_depth"e "max_features"por 2, estamos a aumentar a complexidade do modelo *floresta aleatória*. Isso resulta na construção de um maior número de árvores de decisão, cada uma com uma profundidade maior e considerando um maior número de características para realizar as divisões. O algoritmo precisa de realizar mais cálculos e operações durante o treino, o que impacta diretamente no tempo necessário para concluir essa etapa.

No entanto, é interessante observar que, apesar do aumento no tempo de treino, as métricas de desempenho, como a taxa de acerto, a precisão, o recall e o f1-score, se mantiveram iguais em relação ao teste anterior. Isso demonstra que o conjunto de dados utilizado possivelmente não possui complexidade suficiente para se beneficiar do aumento na complexidade do modelo. Ou seja, as informações e padrões presentes nos dados podem ser corretamente identificados e generalizados através de um modelo menos complexo.

Esses resultados destacam a importância de encontrar um equilíbrio entre a complexidade do modelo e a complexidade dos dados em análise. Aumentar desnecessariamente a complexidade do modelo pode resultar num aumento significativo no tempo de treino sem oferecer ganhos adicionais nas métricas de desempenho. Portanto, é fundamental avaliar cuidadosamente a necessidade de ajustar a complexidade do modelo, considerando a natureza específica do conjunto de dados em questão.

3.8.3 *Floresta de isolamento*

A *floresta de isolamento* é um algoritmo de deteção de anomalias utilizado para identificar pontos de dados incomuns em conjuntos de dados. Ele é baseado no conceito de que as anomalias são mais facilmente isoladas do que as instâncias normais. O algoritmo cria várias árvores de decisão aleatórias, onde cada árvore divide o conjunto de

dados em partições binárias, de forma que as instâncias anómalas são isoladas em partições menores e mais rasas, exigindo menos divisões para serem isoladas. As anomalias são identificadas ao medir o número médio de divisões necessárias para isolar uma instância. Quanto menos divisões forem necessárias, maior a probabilidade de uma instância ser considerada anómala. Com a *floresta de isolamento*, é possível descobrir e identificar pontos de dados incomuns, o que pode ser útil em diversas áreas, como deteção de fraudes financeiras, deteção de intrusões em sistemas de segurança, identificação de comportamentos anómalos em redes de computadores, entre outros [1].

Na fase de treino, este algoritmo precisou de 83 segundos para realizar o treino, e os seguintes parâmetros para treino foram utilizados:

```
1 model = IsolationForest(n_estimators=100, contamination=0.00554, random_
      state=42)
2 model.fit(X)
3 y_pred = model.predict(X)
4 y_pred[y_pred == 1] = 0
5 y_pred[y_pred == -1] = 1
6 print("Numero de anomalias detectadas:", sum(y_pred))
```

O parâmetro *n_estimators*, determina o número de árvores de decisão a serem criadas. Algo semelhante ao algoritmo apresentando anteriormente.

O segundo parâmetro, *contamination* indica a proporção esperada de instâncias anómalas no conjunto de dados. Essa é uma estimativa do número de anomalias presentes nos dados. O valor padrão é 0.1, o que significa que 10% das instâncias são consideradas anómalas. No exemplo fornecido, **contamination**=0.00554, indica que cerca de 0.554% das instâncias são esperadas como anómalas.

E por ultimo *random_state* é usado para inicializar o gerador de números aleatórios interno do modelo. Ao definir um valor específico para *random_state*, como *random_state*=42, garante-se que o modelo produza resultados consistentes e reproduzíveis, tornando-o mais confiável para experimentação e comparação de resultados.

No total foram detetadas 5646 anomalias, número acima da quantidade de flag com o valor 1 que representa os acessos não legítimos.

3.9 Cálculo da importância das *features*

Para o cálculo da importância das *features* foi utilizado a classe *model.feature_importances* do algoritmo *floresta aleatória*. O *feature_importances* retorna um vetor que contém a

importância de cada *feature* utilizada no treino do modelo. Essas importâncias são calculadas com base na contribuição das *features* para a precisão do modelo. Valores mais altos indicam que a *feature* é mais importante para a tarefa de classificação. A Figura 3.6 apresenta o calculo das mesmas.

As 10 *features* mais importantes de acordo com o modelo aplicado foram as seguintes:

- **fwd_iat_min:** É o tempo mínimo interchegada (inter-arrival time) para pacotes que fluem no sentido de entrada. Representa o tempo decorrido entre a chegada de dois pacotes consecutivos.
- **bwd_pkts_b_avg:** É a média da taxa de pacotes que fluem no sentido de saída durante uma janela de tempo. Indica a quantidade média de pacotes recebidos numa conexão em sentido reverso.
- **pkt_len_var:** Essa *feature* representa a variância do comprimento dos pacotes. Indica a dispersão dos tamanhos dos pacotes numa conexão.
- **bwd_blk_rate_avg:** Representa a média da taxa de bloqueio (rate of blocking) em sentido de saúda durante uma janela de tempo. Representa a taxa média de pacotes bloqueados nma conexão em sentido de saída.
- **subflow_bwd_byts:** É a quantidade total de bytes recebidos num subfluxo (subflow) específico da conexão. Um subfluxo é uma subdivisão de uma conexão TCP.
- **bwd_seg_size_avg:** É o tamanho médio dos segmentos em sentido de saída durante uma janela de tempo. Representa o tamanho médio dos segmentos recebidos em uma conexão em sentido de saída.
- **pkt_len_std:** Caracteriza o desvio padrão do comprimento dos pacotes. Indica a dispersão dos tamanhos dos pacotes em relação à média numa conexão.
- **pkt_size_avg:** Simboliza tamanho médio dos pacotes em uma janela de tempo. Indica o tamanho médio dos pacotes numa conexão.
- **pkt_size_avg:** Representa o desvio padrão do comprimento dos pacotes em sentido de saída durante uma janela de tempo. Indica a dispersão dos tamanhos dos pacotes recebidos numa conexão em sentido de saída.

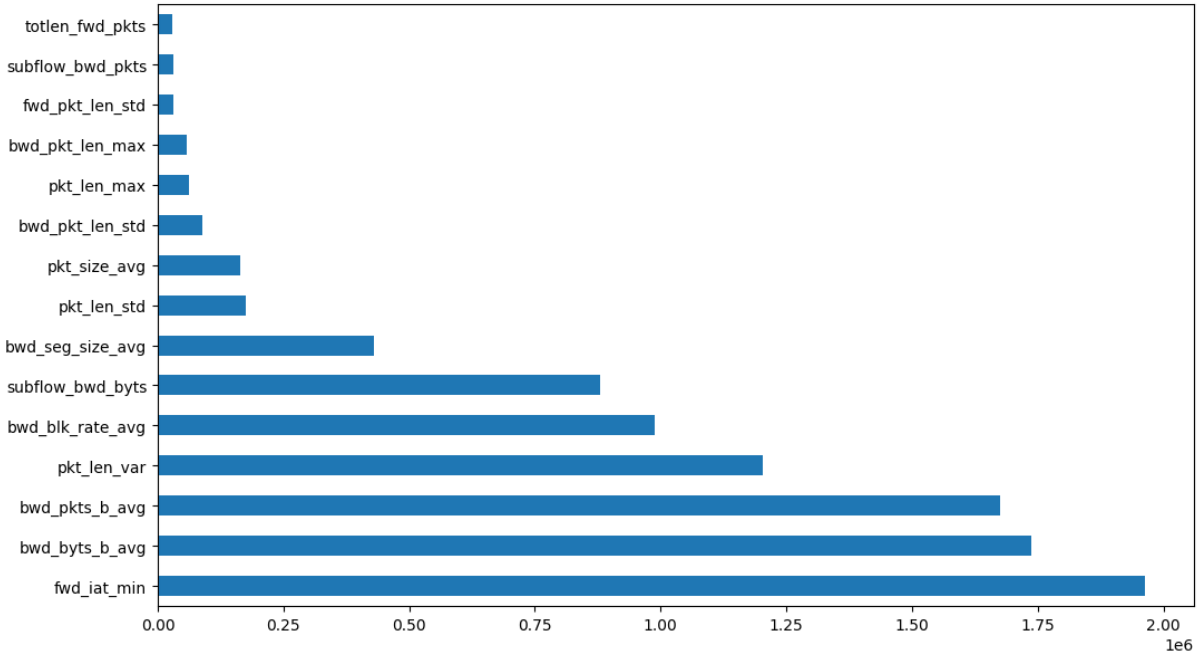


Figura 3.6: Calculo da importância das *features*.

3.10 Cálculo dos valores mínimos e máximos

Fazendo uso da função do Pandas *describe()* obtém-se um resumo estatístico descritivo dos dados do *dataset*. Esse método fornece informações úteis sobre as estatísticas básicas dos dados, como média, desvio padrão, valor mínimo, valor máximo e quartis.

O Pandas é uma biblioteca open-source que oferece estruturas de dados flexíveis e ferramentas de análise de dados para a linguagem de programação Python. A função *describe()* é uma das funcionalidades dessa biblioteca e é frequentemente empregada no início de uma análise exploratória de dados. Ela simplifica a visualização e interpretação das características fundamentais do conjunto de dados, proporcionando insights valiosos para a compreensão da distribuição e variabilidade dos dados [46]."

A Tabela 3.4 apresenta as estatísticas para a flag igual a 1, que representa a classe de acessos legítimos. As estatísticas incluem valores mínimos e máximos para cada uma das *features* selecionadas. Por exemplo, a *feature* "totlen_fwd_pkts"possui um valor mínimo de 132 e um valor máximo de 1667. Essas estatísticas fornecem uma visão geral das faixas de valores para cada *feature*, permitindo uma análise mais detalhada dos dados. A amplitude variada nas características, como "fwd_pkt_len_std"e "pkt_size_avg", é esperada em acessos legítimos, uma vez que diferentes tipos de tráfego podem exibir diferentes padrões de tamanho de pacotes e tempos de intervalo. É crucial compreender essas nuances para distinguir entre padrões normais e atividades

suspeitas.

Como parte da análise, destacamos a importância da feature "bwd_blk_rate_avg", que representa a taxa média de bloqueio bidirecional. Valores extremamente altos podem indicar tráfego anormalmente bloqueado, e a observação desses casos é vital para identificar potenciais eventos de ataque. Além disso, as estatísticas para "subflow_bwd_pkts"e "subflow_bwd_byts"revelam informações sobre os subfluxos bidirecionais, sendo cruciais para compreender padrões específicos de comunicação em acessos legítimos.

Tabela 3.4: Estatísticas para a flag igual a 1

Feature	**Valor Mínimo (Medida)**	**Valor Máximo (Medida)**
totlen_fwd_pkts	132 (bytes)	1667 (bytes)
fwd_pkt_len_std	0 (bytes)	160 (bytes)
bwd_pkt_len_max	66 (bytes)	1514 (bytes)
bwd_pkt_len_std	0 (bytes)	666 (bytes)
pkt_len_max	66 (bytes)	1514 (bytes)
pkt_len_std	0 (bytes)	684 (bytes)
pkt_len_var	0 (bytes)	468353 (bytes)
fwd_iat_min	0 (seconds)	11308 (seconds)
pkt_size_avg	66 (bytes)	740 (bytes)
bwd_byts_b_avg	0 (bytes)	11649 (bytes)
bwd_pkts_b_avg	0 (packets)	11 (packets)
bwd_blk_rate_avg	0 (bytes per second)	92798320 (bytes per second)
bwd_seg_size_avg	66 (bytes)	1096 (bytes)
subflow_bwd_pkts	1 (packets)	17 (packets)
subflow_bwd_byts	66 (bytes)	16595 (bytes)

A tabela 3.5 exibe as estatísticas para a flag igual a 0, que representa valores da classe de ataque. As estatísticas incluem os valores mínimos e máximos para cada uma das *features* consideradas mais importantes pelo modelo de *machine learning*. Por exemplo, a *feature* "totlen_fwd_pkts"possui um valor mínimo de 60 e um valor máximo de 901. Essas estatísticas permitem uma comparação direta com as estatísticas da flag igual a 1, classe de acessos legítimos", revelando possíveis diferenças significativas entre os dois estados.

Tabela 3.5: Estatísticas para a flag igual a 0

Feature	Valor Mínimo (Medida)	Valor Máximo (Medida)
totlen_fwd_pkts	60 (bytes)	901 (bytes)
fwd_pkt_len_std	0 (bytes)	0 (bytes)
bwd_pkt_len_max	0 (bytes)	61 (bytes)
bwd_pkt_len_std	0 (bytes)	0 (bytes)
pkt_len_max	60 (bytes)	174 (bytes)
pkt_len_std	0 (bytes)	56 (bytes)
pkt_len_var	0 (bytes)	3265 (bytes)
fwd_iat_min	0 (seconds)	0 (seconds)
pkt_size_avg	60 (bytes)	117 (bytes)
bwd_byts_b_avg	0 (bytes)	0 (bytes)
bwd_pkts_b_avg	0 (packets)	0 (packets)
bwd_blk_rate_avg	0 (bytes per second)	0 (bytes per second)
bwd_seg_size_avg	0 (bytes)	59 (bytes)
subflow_bwd_pkts	0 (packets)	1 (packets)
subflow_bwd_byts	0 (bytes)	64 (bytes)

Ao analisarmos as estatísticas para a flag igual a 0 na tabela 3.5, notamos diferenças notáveis em características específicas, como "bwd_byts_b_avg", "bwd_pkts_b_avg"e "bwd_blk_rate_avg", em comparação com a classe de acessos legítimos (flag igual a 1). A *features* "bwd_byts_b_avg", apresenta um valor mínimo de 0.00, indicando que em algumas instâncias de ataques, não houve bytes bidirecionais médios, o que pode sugerir uma ausência de tráfego significativo em determinados casos. Além disso, "bwd_pkts_b_avg"também possui valores mínimos e máximos distintos, indicando variações nas taxas médias de pacotes bidirecionais durante atividades maliciosas. A *features* "bwd_blk_rate_avg", que representa a taxa média de bloqueio bidirecional, mostra que em ataques, essa taxa pode variar consideravelmente, evidenciando a diversidade nas estratégias de bloqueio durante atividades maliciosas. Essas diferenças refletem a complexidade e variabilidade das características do tráfego em ataques, enfatizando a necessidade de compreender padrões específicos para a construção eficaz de modelos de detecção de anomalias.

3.11 Dataset público

Em [21], foi desenvolvida uma topologia de rede utilizando equipamentos reais, incluindo switches, computadores e servidores, para a geração de diversos tipos de ataques. Com base nessas capturas, foram criados diferentes *datasets*. Para fins comparativos, foi aplicado o algoritmo de *floresta aleatória* ao *dataset* referente aos ataques SYN

produzido por essa pesquisa. Os resultados obtidos mostraram uma taxa de acerto de 0.976, valor muito próximo ao apresentado anteriormente.

No entanto, os resultados obtidos com o algoritmo de *floresta aleatória* foram similares aos apresentados anteriormente. A taxa de acerto ficou em 0.97, o que sugere que pode haver uma relação direta com o fato das *features* selecionadas e os seus valores terem sido os mesmos utilizados anteriormente. Isso pode indicar a influência desses fatores na consistência dos resultados.

Ainda assim, os resultados obtidos são relevantes, pois reforçam a importância da escolha adequada das *features* e valores para obter um bom desempenho no contexto de deteção de ataques SYN. Além disso, o uso de equipamentos reais e a criação de *datasets* a partir de capturas de ataques contribuem para a validade e fiabilidade da pesquisa.

3.12 Conclusões

Neste capítulo foi apresentada a topologia de rede e as ferramentas utilizadas para a geração do *dataset* de análise relativamente à classificação de acessos. Foram classificados vários tipos de acesso, legítimo e não legítimo, e foram aplicados dois modelos de aprendizagem sobre os mesmos. Detalhou-se o processo de construção do *dataset*, incluindo todas as etapas de processamento de dados, e mostraram-se os resultados obtidos ao aplicar os algoritmos de *floresta aleatória* e *floresta de isolamento*. Os resultados mostraram que o modelo de *floresta aleatória* consegue obter bons resultados para ambas as classes. E por fim foi realizado um estudo breve de um *dataset* público.

4

Framework para automação e gestão preditiva de ataques de negação de serviço

Nesta secção, será desenvolvido um *framework* para validar a premissa inicial, na qual é possível utilizar as informações obtidas com os algoritmos de *machine learning*, e automatizar a criação de políticas em *firewall*. Este *framework* é composto por vários passos, desde a conversão do arquivo de captura até a aplicação do modelo sobre o *dataset* gerado. Em seguida, será feita a aplicação do modelo de automação, utilizando a informação obtida através *machine learning*.

4.1 Desenvolvimento do *framework*

O *framework* de automação desenvolvido neste estudo utiliza três *features* na tomada de decisão relativamente ao tipo de acesso em questão, nomeadamente, "bwd_byts_b_avg", "bwd_pkts_b_avg"e "bwd_blk_rate_avg", sendo estas calculadas utlizando o modelo de *machine learning* da *floresta aleatória*, confirme descrito 3.6. Estas características foram selecionadas devido à sua relevância na detecção de ameaças e ao seu potencial de identificação precoce de ataques e ao facto de possuírem valores mínimos e máximos iguais a zero quando a classe é identificada como um ataque. Levando isso em consideração, o *framework* foi projetado para receber como entrada um arquivo CSV

formatado no padrão *CICFlowMeter*, conforme descrito na Secção 3.4. Os valores dessas três características são validados, e se todas forem iguais a zero, o código classifica a atividade como um ataque e salva o endereço IP de origem, que será utilizado na criação de políticas de *firewall*.

O processo de implementação do *framework* consiste em receber um arquivo de captura do tipo PCAP que e convertido num arquivo CSV, formatado no padrão *CICFlowMeter*, que contém informações sobre o tráfego de rede. Em seguida, o *framework* aplica dois diferentes algoritmos de *machine learning* e valida os valores das características selecionadas para determinar se a atividade é considerada como um acesso não legítimo. Com base nas informações adquiridas por meio do modelo, o módulo implementado em Python é encarregado de extrair os endereços IP de origem e armazená-los para utilização posterior na elaboração de políticas de *firewall*. A Figura 4.1 esboça todos os passos desse procedimento.

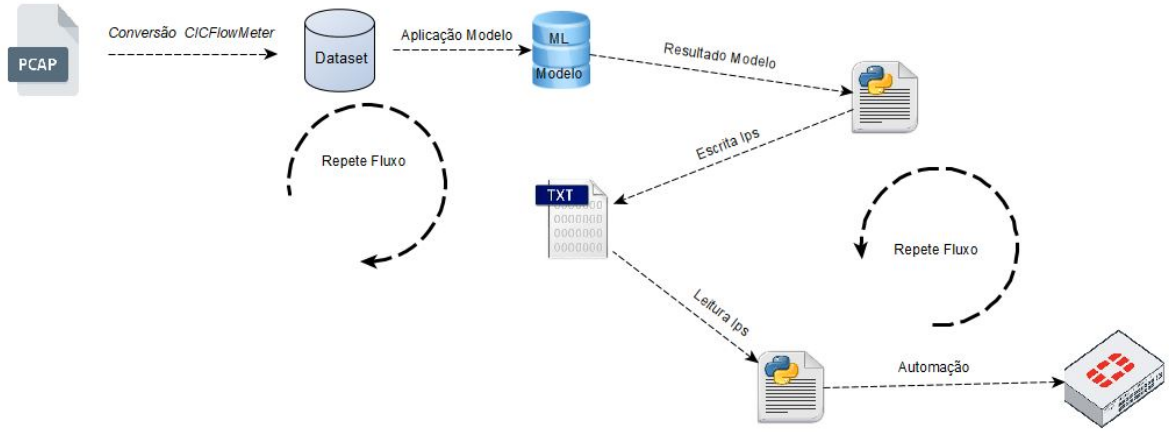


Figura 4.1: Sequência de eventos relativamente ao protótipo desenvolvido.

Em resumo, o *framework* de automação desenvolvido neste estudo utiliza três características do conjunto de dados para detecção de ataques em tempo real. Através da validação dos valores das características "bwd_byts_b_avg", "bwd_pkts_b_avg"e "bwd_blk_rate_avg", é possível identificar atividades suspeitas e classificá-las como ataques quando todas as características possuem valor zero. Essa abordagem visa agilizar a resposta a ameaças e contribuir para a implementação de políticas de segurança eficazes.

4.1.1 Código para o framework

Foram desenvolvidos dois códigos distintos, o primeiro código é responsável pela leitura do *dataset* e pela verificação do tipo de acessos: ele analisa os dados e determina se uma atividade é considerada um ataque ou um acesso legítimo. Em seguida, salva os endereços IP identificados como ataques num arquivo.

O segundo código é responsável por ler o arquivo contendo os endereços IPS não legítimos e, com base nessas informações, cria políticas e objetos no *firewall* para bloquear o tráfego malicioso. Este código utiliza os endereços IP salvos anteriormente para implementar medidas de segurança efetivas e mitigar potenciais ataques.

4.1.1.1 Módulo para processamento dos dados

O primeiro código desenvolvido é composto por 3 funções principais, a primeira delas *processar_arquivo(arquivo)*, recebe como entrada o *dataset* com o registo dos acessos. Esta função também é responsável por determinar quais características serão validadas. No presente caso, foram utilizadas as três características com valores mínimos e máximos iguais a zero quando ocorre um ataque, como calculado no capítulo anterior.

```
1
2  def processar_arquivo(arquivo):
3      with open(arquivo, "r") as arquivo_csv:
4          leitor_csv = csv.DictReader(arquivo_csv)
5          contador_ataques = 0
6          ips_nao_legitimos = set()  # Conjunto para armazenar os IPs não leg
       ítimos
7
8          for linha in leitor_csv:
9              valores = [
10                 float(linha["bwd_pkts_b_avg"]),
11                 float(linha["bwd_byts_b_avg"]),
12                 float(linha["bwd_blk_rate_avg"]),
13             ]
```

Outra função importante é a *verificar_atividade()* que recebe como argumento de entrada os valores para as 3 *features* mencionadas acima, e valida se as mesmas possuem valores igual a 0, e caso assim seja retorna/categoriza o acesso como "Ataque"; caso contrário retorna que se trata de um acesso legítimo. Por fim, em caso de ataque, o IP é guardado num vetor que contém todos osIPS não legítimos num arquivo do tipo texto

pela função *main()*. De seguida é representado o código das tarefas referidas anteriormente.

```
1
2  def verificar_atividade(valores):
3      if (
4          valores[0] == 0
5          and valores[1] == 0
6          and valores[2] == 0
7      ):
8          return "Ataque"
9      else:
10         return "Acesso legítimo"
11 def main():
12     i = 1
13     while True:
14         arquivo = f"testdataset-{i}.csv"
15         ips_nao_legitimos = processar_arquivo(arquivo)
16         write_ips_to_file(ips_nao_legitimos, "ips_nao_legitimos.txt")
17         i += 1
18         print("Aguardando 30 segundos antes da próxima execução...")
19         time.sleep(30)
```

4.1.1.2 Módulo para criação de políticas de firewall

Este módulo contém várias funções para automatizar tarefas de configuração de políticas num dispositivo *firewall* da Fortinet. A função *establish_connection()* estabelece a ligação Secure Shell (SSH) com o dispositivo Fortinet. Outra função *read_ips_from_file(file)* lê os endereços IPS de um arquivo do tipo texto, o mesmo que foi gerado pelo primeiro módulo contendo os IPS de origem de um atacante.

A função *validate_policy(connection)* verifica se uma determinada política de firewall já existe no dispositivo. A *função validate_object_group(connection)* verifica se um grupo de objetos específico já existe. A função *create_policy(connection)* cria uma nova política de *firewall* com configurações específicas, porém este apenas cria uma política na sua primeira interação, pois existe um validador, que caso já exista uma política, ele vai saltar esse passo.

```
1  # Validate if the policy exists
2  def validate_policy(connection):
3      output = connection.send_command('show firewall policy')
4      if 'edit 1' in output:
```

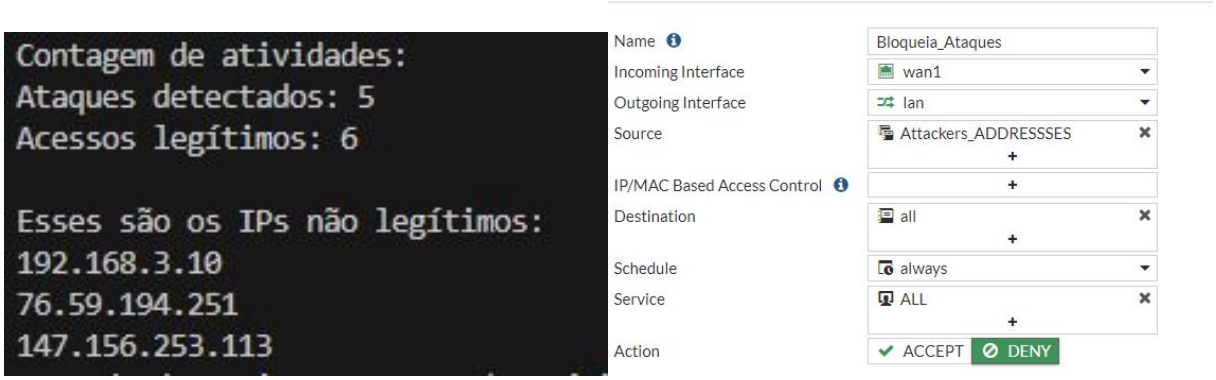
```
        return True
    else:
        return False
# Validate if the object group exists
def validate_object_group(connection):
    output = connection.send_command('show firewall addrgrp')
    if 'Attackers_ADDRESSSES' in output:
        return True
    else:
        return False
# Create the policy
def create_policy(connection):
    commands = [
        'config firewall policy',
        'edit 1',
        'set name "BLOCK"',
        'set srcintf "port1"',
        'set dstintf "port2"',
        'set srcaddr "Attackers_ADDRESSSES"',
        'set dstaddr "all"',
        'set schedule "always"',
        'set service "ALL"',
        'next',
        'end'
    ]
```

Já a função *create_object_group(connection)* cria um novo grupo de objetos, contendo os IPS que estavam no arquivo do tipo texto, e essa sempre será executada. Por último temos a função *append_object(connection, ip_address)* que adiciona esse objeto contendo o IP malicioso, no grupo que está referenciado na política.

Este módulo tem um intervalo de 40 segundos entre cada interação, para poder assim estar sincronizado com a execução do primeiro módulo.

4.1.2 Resultados obtidos

Para a validação da solução proposta foi utilizada a topologia de rede descrita na secção 3.1. Nesse contexto, foi utilizada uma máquina estava conectada a essa topologia, utilizada para a criação das políticas de *firewall*. Numa segunda etapa, após a configuração das políticas com os endereços IP maliciosos devolvidos pelos modelos, foram realizados vários testes de tráfego da mesma origem maliciosa, com o objetivo



(a) Modulo 1. (b) Modulo 2.

Figura 4.2: Resultados obtidos.

de aceder a recursos específicos existentes dentro da rede. Como resultado, a *firewall* demonstrou eficácia ao bloquear essas tentativas como se poderá ver de seguida.

Com o primeiro módulo, foi possível verificar quais os IPS que são maliciosos e quais não são, salvando-os em um arquivo de texto. A Figura 4.2a ilustra a saída gerada durante uma das iterações desse processo.

Com o segundo módulo, estabelecemos uma conexão e criamos uma política de *firewall* no dispositivo Fortinet, conforme mostrado na Figura 4.2b. Através desse módulo, foi possível configurar as regras de segurança necessárias para proteger a rede contra ameaças.

Esses avanços na implementação dos módulos de verificação e configuração do *firewall* são relevantes para o contexto de segurança de redes. Os resultados obtidos demonstram a efetividade das soluções propostas, contribuindo para o aprimoramento da proteção contra ataques de DoS.

4.2 Conclusões

Neste capítulo, foi desenvolvido um *framework* destinado à automação das políticas de *firewall*. Esse *framework* é composto por dois módulos distintos: o primeiro tem a responsabilidade de validar se um fluxo corresponde a um ataque, utilizando informações de *Machine Learning*, e também de registar os endereços IP considerados como classe de ataque num arquivo de texto. Já o segundo módulo concentra-se na automatização da criação de políticas de *firewall*, que envolve a leitura das informações dos IPS, a ligação com o *firewall* e a criação ou atualização das políticas.

Através da implementação dessas políticas, alcançamos a proteção eficaz das redes

contra ataques de negação de serviço, uma vez que o *firewall* se tornou capaz de filtrar novos acessos originados dos mesmos IPS maliciosos. Esse mecanismo permitiu a defesa proativa das redes contra ameaças, tornando a infraestrutura mais segura e menos vulnerável a ataques.

5

Conclusões

Iniciamos este capítulo apresentando um resumo de todo o trabalho realizado nos capítulos anteriores, juntamente com as respetivas considerações finais. De discute-se o que consideramos ser os pontos cruciais a serem estudados a no futuro, dando continuidade ao trabalho desenvolvido nesta dissertação.

5.1 Resumo

Este trabalho teve como objetivo principal desenvolver um *framework* automatizado capaz de criar regras de firewall para proteger redes corporativas contra ataques do tipo DoS. Para alcançar esse objetivo, foi criado um *dataset* simulando esses tipos de ataque, e algoritmos de *machine learning* foram aplicados para obter os parâmetros necessários para o funcionamento do *framework*. O *framework* representa a principal diferença deste trabalho em relação aos estudos que serviram como base, pois sua conceção e implementação foram o foco central desta pesquisa.

O cerne deste trabalho consistiu no desenvolvimento de um *framework* automatizado com a finalidade de estabelecer regras de *firewall* visando a proteção de redes corporativas contra ataques do tipo *dos*. Para viabilizar a implementação do *framework* , um *dataset* simulando esses ataques foi criado, e em seguida, algoritmos de *machine learning* foram estudados para determinar os parâmetros essenciais para o funcionamento adequado do sistema.

No Capítulo 2 foi realizada uma introdução abrangente sobre os ataques de negação de serviço DoS, e o seu impacto em redes corporativas. Além disso, foram apresentados os principais tipos de ataques e seus respetivos protocolos de rede. Nesse contexto, foram também abordadas as principais técnicas de proteção contra tais ataques, incluindo deteção, desvio, filtragem e análise. Esse capítulo proporcionou uma visão abrangente dos desafios enfrentados pelas redes corporativas em relação à segurança contra ataques DoS e destacou a importância de soluções eficazes de proteção. No Capítulo 2, foram também explorados os diferentes modos de aprendizagem em *machine learning*, juntamente com seus componentes essenciais. Foram discutidos os algoritmos mais amplamente utilizados, bem como as técnicas empregadas para seleção de *features* durante a pré-análise de um *dataset*. Esse capítulo proporcionou uma base sólida para a compreensão do papel do *machine learning* no desenvolvimento do *framework* automatizado proposto neste trabalho. Com uma visão clara das ferramentas disponíveis em *Machine Learning*, tornou-se possível aplicar as abordagens mais adequadas para a criação de regras de *firewall* capazes de proteger efetivamente as redes corporativas contra ataques DoS. Ademais, o Capítulo 2 apresentou uma revisão detalhada dos trabalhos desenvolvidos na mesma temática, permitindo um contexto comparativo com o presente trabalho. Ao identificar lacunas e desafios nos estudos anteriores, tornou-se possível estabelecer a singularidade deste trabalho em relação aos trabalhos existentes. O destaque desta dissertação reside no desenvolvimento de um *framework* automatizado que emprega algoritmos de *machine learning* para criar regras de *firewall* adaptáveis e eficientes na proteção de redes corporativas contra ataques DoS.

No Capítulo 3, foi criada uma topologia de rede que simula uma infraestrutura corporativa. Essa topologia inclui um servidor responsivo capaz de atender às requisições HTTP/HTTPS e FTP, bem como um *firewall* que é responsável por receber e redirecionar essas requisições. Além disso, dois computadores foram inseridos na rede, sendo um deles destinado a realizar requisições legítimas, enquanto o outro é responsável por gerar ataques de negação de serviço DoS. Durante a simulação, o tráfego de rede foi gerado e capturado em pacotes PCAP, contendo tanto tráfego legítimo quanto não legítimo. Com o auxílio da ferramenta *CICFlowMeter*, foram extraídas informações dos fluxos de rede, assim como dados estatísticos relevantes. Essas informações foram utilizadas para criar um *dataset* que serviu como base para a análise e treinamento dos modelos de *machine learning*.

Em seguida, realizou-se a etapa de seleção de características do *dataset*, na qual foram descartadas *features* consideradas menos importantes para a análise, como os IPs de destino e endereços MAC, bem como outras *features* com correlação fraca ou nula. Esse processo permitiu otimizar o conjunto de dados e aumentar a eficiência dos modelos

de *machine learning*. Utilizando os algoritmos de *machine learning floresta aleatória* (supervisionado) e *floresta de ssolamento* (não supervisionado), foram obtidos resultados promissores em relação à predição de ataques e à identificação dos valores mínimos e máximos das *features* associados a situações de ataque, sendo estes valores importantes para o desenvolvimento do *framework*.

No Capítulo 4, utilizando os valores descobertos por meio do modelo de *machine learning*, foi desenvolvido um *framework* de automação. Esse *framework* consiste em dois códigos em Python. O primeiro código recebe como argumento de entrada um arquivo *dataset* e valida se se trata de um ataque ou de um fluxo normal, para cada uma das entradas existentes. Em caso de deteção de um ataque, o código regista o endereço IP de origem em um arquivo de texto. Já o segundo código lê o arquivo contendo os IPS considerados como atacantes e, com essa informação, estabelece uma conexão com o *firewall*. Ele verifica se já existe uma política para bloquear esse tráfego e, caso não exista, cria a política e adiciona os IPS maliciosos como bloqueados. Se já houver uma política existente, o código adiciona os IPS maliciosos à política existente, sem criar uma nova. Por meio dos testes realizados com esse *framework*, foi possível constatar sua eficácia em situações de ataque, uma vez que é capaz de bloquear um IP considerado malicioso em questão de segundos, fornecendo uma proteção significativa para uma rede corporativa.

O *framework* de automação desenvolvido destaca-se como uma ferramenta importante para fortalecer a segurança em redes corporativas. Agindo de forma proativa contra ameaças de ataques, ele identifica e bloqueia rapidamente IPs maliciosos, fornecendo uma camada adicional de proteção. A eficácia do *framework* demonstra os benefícios do uso de *machine learning* para criar soluções inovadoras na segurança cibernética das empresas.

5.2 Trabalho futuro

Após a conclusão do *framework* e a análise dos estudos prévios realizados, ainda podemos identificar algumas áreas que se mostram promissoras para um estudo mais aprofundado, bem como tarefas que poderiam ser desenvolvidas desde o início. Neste contexto, apresentamos a seguir três dessas áreas que merecem destaque para futuras pesquisas.

- A utilização de equipamentos reais e uma topologia mais complexa enriqueceria ainda mais o estudo. Com isso, ao invés de uma análise de DoS, poderia ser realizado um estudo sobre DDoS, algo que é ainda mais comum nos dias de hoje;

- Outro ponto que poderia ser considerado para trabalhos futuros é estudar outros tipos de ataques. O presente trabalho focou nos ataques do tipo SYN, porém os ataques de negação de serviço podem assumir diversas formas. Portanto, seria importante aplicar o mesmo modelo e *framework* para outros tipos de ataque;
- Por fim, a aplicação do *framework* desenvolvido em um ambiente real possibilitaria o estudo de como o mesmo reagiria diante de diferentes tipos de ataques, bem como os resultados obtidos. Essa investigação pode ser realizada para verificar e comparar os resultados com os obtidos neste trabalho.

Referências

[1] *Anomaly detection using Isolation Forest – A Complete Guide*, Acedido em: 2023-02-02. URL: https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/.

[2] *Como funciona o algoritmo Árvore de Decisão*, Acedido em: 2023-01-11. URL: https://didatica.tech/como-funciona-o-algoritmo-arvore-de-decisao/.

[3] Mariette Awad & Rahul Khanna, "Support vector machines for classification", em *Efficient Learning Machines*, Springer, 2015, páginas 39–66.

[4] Yalin Başţanlar & Mustafa Özuysal, "Introduction to machine learning", *miRNomics: MicroRNA Biology and Computational Analysis*, páginas 105–128, 2014.

[5] Alaa Alsaeedi, Omaimah Bamasag & Asmaa Munshi, "Real-Time DDoS flood Attack Monitoring and Detection (RT-AMD) Model for Cloud Computing", em *The 4th International Conference on Future Networks and Distributed Systems (ICFNDS)*, 2020, páginas 1–5.

[6] *Basic Data Cleaning for Machine Learning*, Acedido em: 2023-02-02. URL: https://machinelearningmastery.com/basic-data-cleaning-for-machine-learning/.

[7] Jason Brownlee, *Linear regression for machine learning*, 2020. URL: https://machinelearningmastery.com/linear-regression-for-machine-learning/.

[8] Francesco Camastra & Alessandro Vinciarelli, *Machine learning for audio, image and video analysis: theory and applications*. Springer, 2015.

[9] Jaime G Carbonell, Ryszard S Michalski & Tom M Mitchell, "Machine learning: A historical and methodological analysis", *AI Magazine*, vol. 4, n.º 3, páginas 69–79, 1983.

[10] *Check Point Research: Third quarter of 2022 reveals increase in cyberattacks and unexpected developments in global trends*, Acedido em: 2022-11-10. URL: https://blog.checkpoint.com/2022/10/26/third-quarter-of-2022-reveals-increase-in-cyberattacks/.

[11] *O que é ataque DDoS?*, Acedido em: 2022-11-10. URL: https://www.cloudflare.com/pt-br/learning/ddos/what-is-a-ddos-attack/.

[12] *What is a UDP flood attack?*, Acedido em: 2022-11-10. URL: https://www.cloudflare.com/learning/ddos/udp-flood-ddos-attack/.

[13] *Como funciona um ataque de ping da morte?*, Acedido em: 2022-11-10. URL: https://www.cloudflare.com/pt-br/learning/ddos/ping-of-death-ddos-attack/.

[14] *O que é um ataque por amplificação NTP?*, Acedido em: 2022-12-10. URL: https://www.cloudflare.com/pt-br/learning/ddos/ntp-amplification-ddos-attack/.

[15] *What is a Slowloris DDoS attack?*, Acedido em: 2022-12-10. URL: https://www.cloudflare.com/learning/ddos/ddos-attack-tools/slowloris/.

[16] *Como funciona um ataque de inundação Ping?*, Acedido em: 2022-12-10. URL: https://www.cloudflare.com/pt-br/learning/ddos/ping-icmp-flood-ddos-attack/.

[17] *O que é um ataque DDoS Slowris??*, Acedido em: 2022-11-10. URL: https://www.cloudflare.com/pt-br/learning/ddos/ddos-attack-tools/slowloris/.

[18] *Feature selection — Correlation and P-value*, Acedido em: 2023-01-11. URL: https://towardsdatascience.com/feature-selection-correlation-and-p-value-da8921bfb3cf.

[19] Swati Sahu & Amit Verma, "DDoS attack detection in ISP domain using machine learning", em *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA)*, 2019, páginas 1–4. DOI: 10.1109/ICCUBEA47591.2019.9128624.

[20] *O que são ataques DDoS de inundação SYN?*, Acedido em: 2022-11-10, 2021. URL: https://www.akamai.com/pt/learn/what-are-syn-flood-ddos-attacks.

[21] Iman Sharafaldin, Arash Habibi Lashkari, Saqib Hakak & Ali A. Ghorbani, "Developing Realistic Distributed Denial of Service (DDoS) Attack Dataset and Taxonomy", em *2019 International Carnahan Conference on Security Technology (ICCST)*, 2019, páginas 1–8. DOI: 10.1109/CCST.2019.8888419.

[22] Giovanni Di Franco & Michele Santurro, "Machine learning, artificial neural networks and social research", *Quality & Quantity*, vol. 55, n.º 3, páginas 1007–1025, 2021.

[23] Rohan Doshi, Noah Apthorpe & Nick Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices", em *2018 IEEE Security and Privacy Workshops (SPW)*, 2018, páginas 29–35. DOI: 10.1109/SPW.2018.00013.

[24] Saher Esmeir & Shaul Markovitch, "Anytime Learning of Decision Trees.", *Journal of Machine Learning Research*, vol. 8, n.º 5, 2007.

[25] *Packet flow ingress and egress: FortiGates without network processor offloading*, Acedido em: 2022-11-10, 2021. URL: https://docs.fortinet.com/document/fortigate/6.4.0/parallel-path-processing-life-of-a-packet/86811/packet-flow-ingress-and-egress-fortigates-without-network-processor-offloading.

[26] *introduction what is a ddos attack?*, Acedido em: 2022-11-10. URL: https://www.fortinet.com/content/dam/fortinet/assets/white-papers/DDoS-Attack-Mitigation-Demystified.pdf.

[27] Chao Gao, Hanbo Sun, Tuo Wang, Ming Tang, Nicolaas I. Bohnen, Martijn L. T. M. Müller, Talia Herman, Nir Giladi, Alexandr Kalinin & Cathie Spino, *Model-based and model-free machine learning techniques for diagnostic prediction and classification of clinical outcomes in Parkinson's disease*, 2018. URL: https://www.nature.com/articles/s41598-018-24783-4.

[28] Akshat Gaurav, Brij B. Gupta & Prabin Kumar Panigrahi, "A novel approach for DDoS attacks detection in COVID-19 scenario for small entrepreneurs", *Technological Forecasting and Social Change*, vol. 177, pág. 121 554, 2022, ISSN: 0040-1625. DOI: https://doi.org/10.1016/j.techfore.2022.121554. URL: https://www.sciencedirect.com/science/article/pii/S0040162522000865.

[29] Luca M Ghiringhelli, Jan Vybiral, Sergey V Levchenko, Claudia Draxl & Matthias Scheffler, "Big data of materials science: critical role of the descriptor", *Physical review letters*, vol. 114, n.º 10, pág. 105 503, 2015.

[30] *O que é ajuste de hiperparâmetros?*, Acedido em: 2023-01-11. URL: https://aws.amazon.com/pt/what-is/hyperparameter-tuning/.

[31] Victoria Hodge & Jim Austin, "A survey of outlier detection methodologies", *Artificial intelligence review*, vol. 22, n.º 2, páginas 85–126, 2004.

[32] *Tool Documentation*, Acedido em: 2022-11-10. URL: https://www.kali.org/tools/hping3//.

[33] Rahmeh Fawaz Ibrahim, Qasem Abu Al-Haija & Ashraf Ahmad, "DDoS Attack Prevention for Internet of Thing Devices Using Ethereum Blockchain Technology", *Sensors*, vol. 22, n.º 18, pág. 6806, 2022.

[34] *DDoS Attacks*, Acedido em: 2022-11-10. URL: https://www.imperva.com/learn/ddos/ddos-attacks/.

[35] *Distributed Denial of Service (DDoS)*, Acedido em: 2022-11-10. URL: https://www.imperva.com/learn/ddos/denial-of-service/.

[36] *DDoS Mitigation: The Definitive Buyer's*, Acedido em: 2022-11-10. URL: https://www.imperva.com/learn/ddos/ddos-mitigation-services/.

[37] Weikuan Jia, Meili Sun, Jian Lian & Sujuan Hou, "Feature dimensionality reduction: A Review", *Complex amp; Intelligent Systems*, vol. 8, n.º 3, 2663–2693, 2022. DOI: 10.1007/s40747-021-00637-x.

[38] Yann Le Cun, Lionel D Jackel, Brian Boser, John S Denker, Henry P Graf, Isabelle Guyon, Don Henderson, Richard E Howard & William Hubbard, "Handwritten digit recognition: Applications of neural network chips and automatic learning", *IEEE Communications Magazine*, vol. 27, n.º 11, páginas 41–46, 1989.

[39] Andy Liaw, Matthew Wiener et al., "Classification and regression by randomForest", *R news*, vol. 2, n.º 3, páginas 18–22, 2002.

[40] Petr Blazek, Tomas Gerlich, Zdenek Martinasek & Jakub Frolka, "Comparison of Linux Filtering Tools for Mitigation of DDoS Attacks", em *2018 41st International Conference on Telecommunications and Signal Processing (TSP)*, 2018, páginas 1–5. DOI: 10.1109/TSP.2018.8441309.

[41] Stephen Marsland, *Machine Learning: An Algorithmic Perspective, Second Edition*, 2nd. Chapman amp; Hall/CRC, 2014, ISBN: 1466583282.

[42] João Paulo Abreu Maranhão, João Paulo Carvalho Lustosa da Costa, Edison Pignaton de Freitas, Elnaz Javidi & Rafael Timóteo de Sousa Júnior, "Error-Robust Distributed Denial of Service Attack Detection Based on an Average Common Feature Extraction Technique", *Sensors*, vol. 20, n.º 20, 2020, ISSN: 1424-8220. DOI: 10.3390/s20205845. URL: https://www.mdpi.com/1424-8220/20/20/5845.

[43] *O que é um ataque DDoS?*, Acedido em: 2022-11-10, 2021. URL: https://www.microsoft.com/pt-pt/security/business/security-101/what-is-a-ddos-attack.

[44] Tom Michael Mitchell, *Machine Learning*, 1ª ed. McGraw-Hill, mar. de 1997, ISBN: 0070428077.

[45] *O que é machine learning?*, Acedido em: 2022-11-10, 2022. URL: https://www.ibm.com/br-pt/topics/machine-learning.

[46] *Pandas DataFrame Describe*, Acessado em: 2023-01-11. URL: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.describe.html.

[47] Panida Khuphiran, Pattara Leelaprute, Putchong Uthayopas, Kohei Ichikawa & Wassapon Watanakeesuntorn, "Performance Comparison of Machine Learning Models for DDoS Attacks Detection", em *2018 22nd International Computer Science and Engineering Conference (ICSEC)*, 2018, páginas 1–4. DOI: 10.1109/ICSEC.2018.8712757.

[48] Hoa Thi Pham, Joseph Awange & Michael Kuhn, "Evaluation of Three Feature Dimension Reduction Techniques for Machine Learning-Based Crop Yield Prediction Models", *Sensors*, vol. 22, n.º 17, 2022, ISSN: 1424-8220. DOI: 10.3390/s22176609. URL: https://www.mdpi.com/1424-8220/22/17/6609.

[49] Hellen Geremias dos Santos, "Comparação da performance de algoritmos de machine learning para a análise preditiva em saúde pública e medicina", Tese de Doutoramento, Universidade de São Paulo, 2018.

[50] Gabriel R Schleder & Adalberto Fazzio, "Machine Learning in Physics, Chemistry, and Materials Science: Materials Discovery and Design", *Revista Brasileira de Ensino de Física*, vol. 43, 2021.

[51] Junjie Shi, Jiang Bian, Jakob Richter, Kuan-Hsun Chen, Jörg Rahnenführer, Haoyi Xiong & Jian-Jia Chen, "Modes: Model-based optimization on distributed embedded systems", *Machine Learning*, vol. 110, n.º 6, 1527–1547, 2021. DOI: 10.1007/s10994-021-06014-6.

[52] Felipe S. Dantas Silva, Esau Silva, Emidio P. Neto, Marcilio Lemos, Augusto J. Venancio Neto & Flavio Esposito, "A Taxonomy of DDoS Attack Mitigation Approaches Featured by SDN Technologies in IoT Scenarios", *Sensors*, vol. 20, n.º 11, 2020, ISSN: 1424-8220. DOI: 10.3390/s20113078. URL: https://www.mdpi.com/1424-8220/20/11/3078.

[53] *Redução da dimensionalidade na Machine Learning*, Acedido em: 2022-11-10. URL: https://softtek.eu/pt-pt/tech-magazine-pt/artificial-intelligence-pt/reducao-da-dimensionalidade-na-machine-learning/.

[54] *How to split data into three sets (train, validation, and test) And why?*, Acedido em: 2023-01-11. URL: https://towardsdatascience.com/how-to-split-data-into-three-sets-train-validation-and-test-and-why-e50d22d3e54c.

[55] Jeffrey M Stanton, "Galton, Pearson, and the peas: A brief history of linear regression for statistics instructors", *Journal of Statistics Education*, vol. 9, n.º 3, 2001.

[56] Gary Stein, Bing Chen, Annie S. Wu & Kien A. Hua, "Decision tree classifier for network intrusion detection with ga-based feature selection", *Proceedings of the 43rd annual Southeast regional conference - Volume 2*, 136–141, 2005. DOI: 10.1145/1167253.1167288.

[57] Ajay Taneja, *Feature selection and dimensionality reduction*, 2021. URL: https://www.linkedin.com/pulse/feature-selection-dimensionality-reduction-ajay-taneja/?trk=pulse-article_more-articles_related-content-card.

[58] *Upsampling with SMOTE for Classification Projects*, Acedido em: 2023-01-11. URL: https://towardsdatascience.com/upsampling-with-smote-for-classification-projects-e91d7c44e4bf.

Informações extraídas da captura wirehsark e informações utlizadas no *machine learning*

Fazendo uso da ferramenta CICFlowMeter foram extraídos os seguintes campos do arquivo PCAP. No total 80 campos que contem informações referentes ao fluxo de rede sendo ele legitimo ou malicioso.

- src_ip: Endereço IP de origem.
- dst_ip: Endereço IP de destino.
- src_port: Número da porta de origem.
- dst_port: Número da porta de destino.
- src_mac: Endereço MAC de origem.
- dst_mac: Endereço MAC de destino.
- protocol: Protocolo de comunicação usado (por exemplo, TCP, UDP, ICMP).
- timestamp: Horário em que o pacote foi capturado.
- flow_duration: Duração da conexão de rede.

- flow_byts_s: Taxa de transferência de bytes da conexão.
- flow_pkts_s: Taxa de transferência de pacotes da conexão.
- fwd_pkts_s: Taxa de transferência de pacotes para frente.
- bwd_pkts_s: Taxa de transferência de pacotes de retorno.
- tot_fwd_pkts: Número total de pacotes enviados para frente.
- tot_bwd_pkts: Número total de pacotes de retorno.
- totlen_fwd_pkts: Número total de bytes enviados para frente.
- totlen_bwd_pkts: Número total de bytes de retorno.
- fwd_pkt_len_max: Tamanho máximo do pacote para frente.
- fwd_pkt_len_min: Tamanho mínimo do pacote para frente.
- fwd_pkt_len_mean: Tamanho médio do pacote para frente.
- fwd_pkt_len_std: Desvio padrão do tamanho do pacote para frente.
- bwd_pkt_len_max: Tamanho máximo do pacote de retorno.
- bwd_pkt_len_min: Tamanho mínimo do pacote de retorno.
- bwd_pkt_len_mean: Tamanho médio do pacote de retorno.
- bwd_pkt_len_std: Desvio padrão do tamanho do pacote de retorno.
- pkt_len_max: Tamanho máximo do pacote.
- pkt_len_min: Tamanho mínimo do pacote.
- pkt_len_mean: Tamanho médio do pacote.
- pkt_len_std: Desvio padrão do tamanho do pacote.
- pkt_len_var: Variância do tamanho do pacote.
- fwd_header_len: Tamanho do cabeçalho para frente.
- bwd_header_len: Tamanho do cabeçalho de retorno.
- fwd_seg_size_min: Tamanho mínimo do segmento para frente.

- fwd_act_data_pkts: Número de pacotes com dados de ação para frente.
- flow_iat_mean: Média do tempo inter-arrival (IAT) da conexão.
- flow_iat_max: Maior IAT da conexão.
- flow_iat_min: Menor IAT da conexão.
- flow_iat_std: Desvio padrão do IAT da conexão.
- fwd_iat_tot: Total de tempo entre dois pacotes enviados para frente consecutivos.
- fwd_iat_max: Maior tempo entre dois pacotes enviados para frente consecutivos.
- fwd_iat_min: Menor tempo entre dois pacotes enviados para frente consecutivos.
- fwd_iat_mean: Média do tempo entre dois pacotes enviados para frente consecutivos.
- fwd_iat_std: Desvio padrão do tempo entre dois pacotes enviados para frente consecutivos.
- bwd_iat_tot: Total de tempo entre dois pacotes de retorno consecutivos.
- bwd_iat_max: Maior tempo entre dois pacotes de retorno consecutivos.
- bwd_iat_min: Menor tempo entre dois pacotes de retorno consecutivos.
- bwd_iat_mean: o tempo médio entre pacotes recebidos em um fluxo, calculado apenas para pacotes que se movem na direção reversa.
- bwd_iat_std: o desvio padrão do tempo entre pacotes recebidos em um fluxo, calculado apenas para pacotes que se movem na direção reversa.
- fwd_psh_flags: o número de pacotes enviados com a bandeira PSH definida na direção para frente.
- bwd_psh_flags: o número de pacotes enviados com a bandeira PSH definida na direção para trás.
- fwd_urg_flags: o número de pacotes enviados com a bandeira URG definida na direção para frente.
- bwd_urg_flags: o número de pacotes enviados com a bandeira URG definida na direção para trás.

- fin_flag_cnt: o número de pacotes em um fluxo com a bandeira FIN definida.
- syn_flag_cnt: o número de pacotes em um fluxo com a bandeira SYN definida.
- rst_flag_cnt: o número de pacotes em um fluxo com a bandeira RST definida.
- psh_flag_cnt: o número de pacotes em um fluxo com a bandeira PSH definida.
- ack_flag_cnt: o número de pacotes em um fluxo com a bandeira ACK definida.
- urg_flag_cnt: o número de pacotes em um fluxo com a bandeira URG definida.
- ece_flag_cnt: o número de pacotes em um fluxo com a bandeira ECE definida.
- down_up_ratio: a razão de bytes na direção para frente para bytes na direção para trás.
- pkt_size_avg: o tamanho médio de pacotes em um fluxo.
- init_fwd_win_byts: o tamanho da janela inicial anunciado pelo receptor na direção para frente.
- init_bwd_win_byts: o tamanho da janela inicial anunciado pelo receptor na direção para trás.
- active_max: o tempo máximo em que um fluxo ficou ativo (em segundos).
- active_min: o tempo mínimo em que um fluxo ficou ativo (em segundos).
- active_mean: o tempo médio em que um fluxo permaneceu ativo (em segundos).
- active_std: o desvio padrão do tempo em que um fluxo permaneceu ativo (em segundos).
- idle_max: o tempo máximo em que um fluxo permaneceu inativo (em segundos).
- idle_min: o tempo mínimo em que um fluxo permaneceu inativo (em segundos).
- idle_mean: o tempo médio em que um fluxo permaneceu inativo (em segundos).
- idle_std: o desvio padrão do tempo em que um fluxo permaneceu inativo (em segundos).
- fwd_byts_b_avg: a média de bytes enviados na direção para frente em um fluxo.
- fwd_pkts_b_avg: a média de pacotes enviados na direção para frente em um fluxo.

- bwd_byts_b_avg: a média de bytes enviados na direção para trás em um fluxo.
- bwd_pkts_b_avg: a média de pacotes enviados na direção para trás em um fluxo.
- fwd_blk_rate_avg: a taxa média de blocos enviados na direção para frente em um fluxo.
- bwd_blk_rate_avg: a taxa média de blocos enviados na direção para trás em um fluxo.
- fwd_seg_size_avg: o tamanho médio dos segmentos enviados na direção para frente em um fluxo.
- bwd_seg_size_avg: o tamanho médio dos segmentos enviados na direção para trás em um fluxo.
- cwe_flag_count: o número de pacotes com a bandeira CWE definida em um fluxo.
- subflow_fwd_pkts: o número de pacotes em um subfluxo na direção para frente.
- subflow_bwd_pkts: o número de pacotes em um subfluxo na direção para trás.
- subflow_fwd_byts: o número de bytes em um subfluxo na direção para frente.
- subflow_bwd_byts: o número de bytes em um subfluxo na direção para trás.

Sendo que, após o pré-processamento, as seguintes *features* foram empregadas nos modelos de *machine learning*:

- src_ip: Endereço IP de origem.
- dst_ip: Endereço IP de destino.
- src_port: Porta de origem.
- dst_port: Porta de destino.
- dst_mac: Endereço MAC de destino.
- protocol: Protocolo utilizado.
- timestamp: Carimbo de data/hora.
- flow_pkts_s: Pacotes por segundo na fluxo.
- fwd_pkts_s: Pacotes por segundo no sentido direto.
- bwd_pkts_s: Pacotes por segundo no sentido inverso.
- totlen_fwd_pkts: Comprimento total dos pacotes no sentido direto.
- fwd_pkt_len_max: Comprimento máximo dos pacotes no sentido direto.
- fwd_pkt_len_std: Desvio padrão do comprimento dos pacotes no sentido direto.
- bwd_pkt_len_max: Comprimento máximo dos pacotes no sentido inverso.
- bwd_pkt_len_min: Comprimento mínimo dos pacotes no sentido inverso.
- bwd_pkt_len_std: Desvio padrão do comprimento dos pacotes no sentido inverso.
- pkt_len_max: Comprimento máximo dos pacotes.
- pkt_len_min: Comprimento mínimo dos pacotes.
- pkt_len_std: Desvio padrão do comprimento dos pacotes.
- pkt_len_var: Variância do comprimento dos pacotes.
- fwd_header_len: Comprimento do cabeçalho no sentido direto.
- flow_iat_mean: Média dos intervalos entre fluxos.

- flow_iat_min: Menor intervalo entre fluxos.
- fwd_iat_min: Menor intervalo entre pacotes no sentido direto.
- bwd_iat_tot: Soma dos intervalos entre pacotes no sentido inverso.
- bwd_iat_min: Menor intervalo entre pacotes no sentido inverso.
- bwd_iat_mean: Média dos intervalos entre pacotes no sentido inverso.
- fin_flag_cnt: Contagem de flags FIN.
- down_up_ratio: Razão entre pacotes de upload e download.
- pkt_size_avg: Tamanho médio dos pacotes.
- init_fwd_win_byts: Tamanho inicial da janela no sentido direto.
- idle_min: Menor intervalo de inatividade.
- idle_mean: Média dos intervalos de inatividade.
- idle_std: Desvio padrão dos intervalos de inatividade.
- bwd_byts_b_avg: Tamanho médio dos bytes no sentido inverso.
- bwd_pkts_b_avg: Número médio de pacotes no sentido inverso.
- bwd_blk_rate_avg: Taxa média de bloqueio no sentido inverso.
- fwd_seg_size_avg: Tamanho médio dos segmentos no sentido direto.
- bwd_seg_size_avg: Tamanho médio dos segmentos no sentido inverso.
- subflow_fwd_pkts: Número de pacotes no subfluxo no sentido direto.
- subflow_bwd_pkts: Número de pacotes no subfluxo no sentido inverso.
- subflow_bwd_byts: Número de bytes no subfluxo no sentido inverso.
- flag: Flag indicando ataque ou não.

B

Código desenvolvido para Automação

Neste anexo, apresentamos os códigos desenvolvidos para o framework de automação. O primeiro código se destina à verificação de se o fluxo corresponde a um ataque ou não. Já o segundo código é responsável por criar e atualizar a política de *firewall* com base nos parâmetros coletados no primeiro código.

```
1  import csv
2  import time
3
4  def verificar_atividade(valores):
5      if (
6          #valores[3] < 618 and
7          valores[0] == 0
8          and valores[1] == 0
9          and valores[2] == 0
10     ):
11         return "Ataque"
12     else:
13         return "Acesso legítimo"
14
15 def processar_arquivo(arquivo):
16     with open(arquivo, "r") as arquivo_csv:
17         leitor_csv = csv.DictReader(arquivo_csv)
18         contador_ataques = 0
19         ips_nao_legitimos = set()  # Conjunto para armazenar os IPs não leg
       ítimos
20
```

```
21          for linha in leitor_csv:
22              valores = [
23                  float(linha["bwd_pkts_b_avg"]),
24                  float(linha["bwd_byts_b_avg"]),
25                  float(linha["bwd_blk_rate_avg"]),
26                  #float(linha["fwd_iat_min"])
27              ]
28
29              atividade = verificar_atividade(valores)
30              print(atividade)
31              print()
32
33              if atividade == "Ataque":
34                  contador_ataques += 1
35                  ips_nao_legitimos.add(linha["src_ip"])  # Adicionar IP ao
    conjunto
36
37          print("Contagem de atividades:")
38          print("Ataques detectados:", contador_ataques)
39          print("Acessos legítimos:", leitor_csv.line_num - contador_ataques)
40
41          if ips_nao_legitimos:
42              print("\nEsses são os IPs não legítimos:")
43              for ip in ips_nao_legitimos:
44                  print(ip)
45
46          return ips_nao_legitimos
47
48  def write_ips_to_file(ips, nome_arquivo):
49      with open(nome_arquivo, "w") as arquivo:
50          for ip in ips:
51              arquivo.write(ip + "\n")  # Escrever IPs no arquivo
52
53  def main():
54      i = 1
55      while True:
56          arquivo = f"testdataset-{i}.csv"
57          ips_nao_legitimos = processar_arquivo(arquivo)
58          write_ips_to_file(ips_nao_legitimos, "ips_nao_legitimos.txt")
59          i += 1
60          print("Aguardando 90 segun antes da próxima execução...")
61          time.sleep(90)
62
63  if __name__ == "__main__":
64      main()
```

```
import time
from netmiko.fortinet import FortinetSSH
from netmiko import ConnectHandler

fortigate = {
    'device_type': 'fortinet',
    'host': '10.5.27.41',
    'username': 'admin2',
    'password': 'password',
    'port': 22,
}

# Function to establish the connection
def establish_connection():
    return ConnectHandler(**fortigate)

# Validate if the policy exists
def validate_policy(connection):
    output = connection.send_command('show firewall policy')
    if 'edit 1' in output:
        return True
    else:
        return False

# Validate if the object group exists
def validate_object_group(connection):
    output = connection.send_command('show firewall addrgrp')
    if 'Attackers_ADDRESSSES' in output:
        return True
    else:
        return False

# Create the policy
def create_policy(connection):
    commands = [
        'config firewall policy',
        'edit 1',
        'set name "BLOCK"',
        'set srcintf "port1"',
        'set dstintf "port2"',
        'set srcaddr "Attackers_ADDRESSSES"',
        'set dstaddr "all"',
```

```
        'set schedule "always"',
        'set service "ALL"',
        'next',
        'end'
    ]
    print("Creating firewall policy ....")
    output = connection.send_config_set(commands)
    print(output)

# Create the object group
def create_object_group(connection):
    commands = [
        'config firewall addrgrp',
        'edit "Attackers_ADDRESSSES"',
        'next',
        'end'
    ]
    print("Creating object address group....")
    output = connection.send_config_set(commands)
    print(output)

# Create the address group
def create_address_group(connection):
    commands = [
        'config firewall addrgrp',
        'edit "Attacker_address"',
        'next',
        'end'
    ]
    print("Creating address group....")
    output = connection.send_config_set(commands)
    print(output)

# Append an object to the object group
def append_object(connection, ip_address):
    commands = [
        'config firewall address',
        f'edit "{ip_address}"',
        f'set subnet {ip_address} 255.255.255.255',
        'next',
        'end',
        'config firewall addrgrp',
        'edit "Attacker_address"',
        f'append member "{ip_address}"',
        'next',
```

```
88          'end'
89      ]
90      print("Creating object address ....")
91      output = connection.send_config_set(commands)
92      print(output)
93
94  # Read IPs from file
95  def read_ips_from_file(file):
96      ips = []
97      with open(file, "r") as f:
98          for line in f:
99              ip = line.strip()
100             ips.append(ip)
101     return ips
102
103 # Main loop
104 while True:
105     try:
106         # Establish the connection
107         connection = establish_connection()
108
109         # Create the address group if it doesn't exist
110         if not validate_object_group(connection):
111             create_address_group(connection)
112
113         # Validate if the policy exists
114         if not validate_policy(connection):
115             create_policy(connection)
116
117         # Read IPs from file
118         ips_nao_legitimos = read_ips_from_file("ips_nao_legitimos.txt")
119         for ip in ips_nao_legitimos:
120             print(ip)
121             append_object(connection, ip)
122
123         #
```