Comparação dos Algoritmos de Aprendizado de Máquina Rede Neural MLP e K-NN para a Detecção do Ataque Slowloris

Igo Henrique Silva Sousa[1], Carlos Alberto de Sousa Parente Rodrigues[1], Samuel Lima Carneiro[1], Gabriel Ferreira Cunha[1], Vinícius de Miranda Rios[2]

[1]Estudantes do Curso de Pós-graduação em Telemática – IFTO. e-mail: {igo.cstredes,carlos.ccomp,samuelcarneiro,gferreiracunha}@gmail.com
[2]Professor do Curso de Pós-graduação em Telemática – IFTO. e-mail: vinicius.rios@ifto.edu.br

Resumo: Os ataques do tipo slow DoS (*Denial of Service*) já são uma constante no cotidiano de empresas que fornecem serviços WEB. Dentre eles, o ataque *slowloris* possui uma crescente demanda para este fim. Baseado nisto, este trabalho apresenta uma proposta para avaliar dois algoritmos de aprendizado de máquina chamados rede neural MLP (*Multilayer Perceptron*) e *K-Nearest Neighbors* (K-NN), ambos utilizando dois atributos para classificação dos dados, denominados, quantidade de pacotes e entropia, em um ambiente emulado, com o objetivo de verificar qual dentre eles possui a melhor performance de classificação com a menor taxa de falso positivo e falso negativo. Os resultados obtidos mostram que o algoritmo K-NN é a melhor escolha por ter alcançado 100% de Precisão e 100% de Revogação, obtendo 100% de F1-score para ambos os rótulos de classificação, ataque e legítimo.

Palavras–chave: ddos, k-nn, rede neural mlp, slowloris

1 INTRODUÇÃO

Ao longo dos anos, os serviços fornecidos por empresas, órgãos públicos ou até mesmo usuários através da Internet, foram e ainda são alvos quase que diariamente de inúmeros tipos de ataques. Dentre estes, os ataques de negação de serviços - cujo objetivo é bloquear qualquer tipo de tentativa de acesso aos serviços online – estão entre os mais perigosos, tendo em vista que não possuem ainda uma defesa efetiva contra a maioria deles, havendo apenas meios de minimizar sua ação. Desta forma, podendo causar perdas financeiras e/ou enfraquecimento da marca da empresa/órgão público.

O ataque slowloris, objeto de estudo deste artigo, é um tipo de ataque de negação de serviço que pode ser desferido tanto no formato DoS ou DDoS (*Distributed Denial of Service*). Este é bem difundido na Internet por ser de código aberto e possuir simplicidade de codificação. Empresas como a Wordpress, por exemplo, são alvo constante deste tipo de ataque por fornecerem hospedagem de sítios WEB (STORM INTERNET, 2020). O seu funcionamento se dá pelo envio de requisições HTTP (*HyperText Transfer Protocol*), utilizando o método GET, em direção a um servidor WEB. Essas requisições são enviadas de forma ”lenta”, ou seja, uma conexão TCP (*Transmition Control Protocol*) com o cabeçalho HTTP mau formado é enviada juntamente com várias outras conexões TCP (DANTAS 2015; AQIL et al., 2015). O cabeçalho mau formado possui em seu final uma codificação \r\n na qual é interpretada pelo servidor WEB como uma solicitação de espera por novos dados. Assim, a cada requisição recebida, mais recursos são alocados até a parada total do servidor.

Portanto, o presente trabalho tem como objetivo detectar ataques slowloris, utilizando o algoritmo de aprendizado de máquina mais eficaz, selecionado entre os seguintes algoritmos: rede neural

MLP e K-NN. A escolha se dará pelo algoritmo que possuir a menor porcentagem de falso positivo e falso negativo na classificação do fluxo de tráfego. Todo esse processo será feito em um ambiente emulado, já que este elimina os altos custos para compra de computadores e dispositivos de rede, bem como permite que toda a configuração seja armazenada em um único computador. Assim, sendo possível ser executado em qualquer lugar.

A organização deste trabalho é descrita como se segue. A seção 2 apresenta o ambiente desenvolvido para verificar a eficiência dos algoritmos e o formato conceitual de cada algoritmo de aprendizado de máquina. Na seção 3, é apresentado os resultados obtidos, seguido pela seção 4 com a conclusão do estudo proposto.

2 MATERIAL E MÉTODOS

Nesta seção, serão descritos os softwares e os hardwares utilizados na criação dos ambientes de teste, bem como suas características de funcionamento. O objetivo é criar *traces* de tráfego o mais próximo de um real, para extrair os atributos que serão utilizados pelos algoritmos de aprendizado de máquina para a classificação do fluxo de dados.

2.1 Geração do *trace* de tráfego do ambiente emulado

Todo o ambiente foi criado utilizando um software de emulação chamado netkit (NETKIT, 2020). Este software possui a capacidade de criar um computador virtual com todos os principais componentes de um computador real. Sua escolha foi baseada na facilidade de se criar uma rede virtual com vários dos ativos de uma rede física, com o intuito de se aproximar ao máximo do ambiente da Internet.

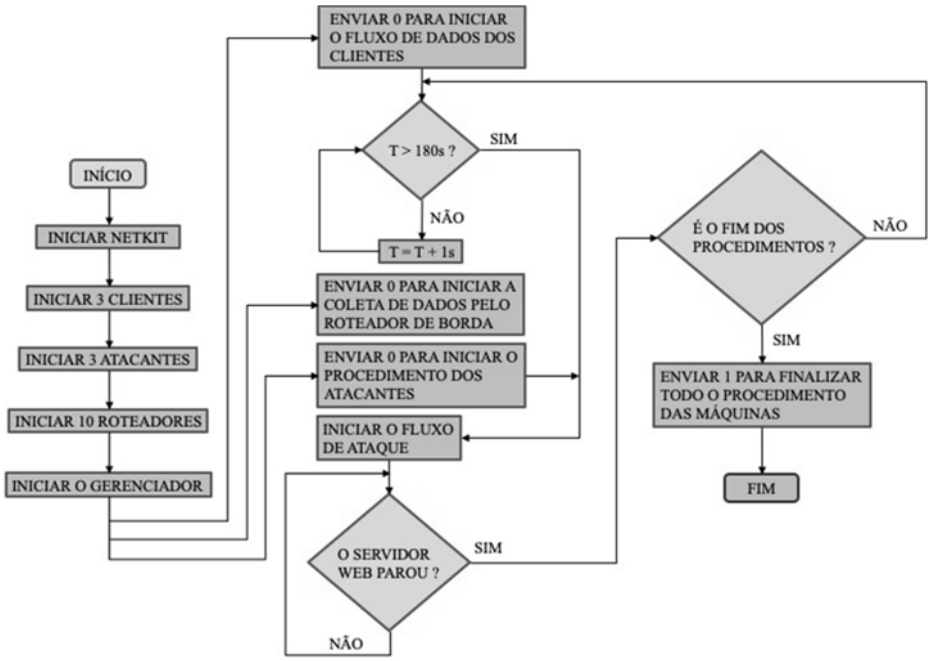
O cenário proposto foi criado contendo um total de sete computadores e dez roteadores virtuais interconectados. Dentre eles, o computador denominado gerenciador conduz toda a fase de início e fim da transmissão e da coleta de dados. Para isto, foram criados scripts em *shell* do tipo cliente/servidor utilizando o software netcat, com o intuito de iniciar/parar toda a condução do experimento por parte das máquinas virtuais. Os 3 computadores clientes, utilizaram os softwares curl-loader e siege com o objetivo de enviar requisições HTTP legítimas em direção à vítima. O uso destes dois softwares combinados se fez necessário, pelo fato do curl-loader não reproduzir uma conexão HTTP totalmente idêntica à original, embora possibilite a geração de IPs (*Internet Protocol*) falsos aleatórios, ao passo que o siege, embora não gere IPs falsos aleatórios, este cria uma conexão HTTP idêntica à original.

Desta forma, um software complementa o outra, possibilitando um ambiente com requisições HTTP de ”diferentes fontes” emulando um ataque próximo ao real. Cada computador cliente simulou o tráfego de 7 computadores reais. Os 3 computadores atacantes, tem como objetivo, obstruir o tráfego

dos clientes que chegam ao servidor WEB, injetando requisições HTTP com o seu cabeçalho mau formado, utilizando para este fim o software slowhttptest. O computador denominado vítima, alvo dos computadores clientes e atacantes, utiliza o software Apache para hospedar a página WEB. Por fim, o roteador de borda, será o local de coleta de todo o tráfego de dados que será analisado pelos algoritmos de aprendizado de máquina. Para a coleta, foi utilizado o software analisador de tráfego tshark.

É importante salientar que, por padrão, as máquinas virtuais criadas pelo netkit possuem apenas 32MB. Sendo assim, foi necessário alterar o montante de memória das máquinas virtuais, pelo fato de novos softwares serem criados e/ou instalados, necessitando de mais memória. O roteador de borda precisou de mais memória que os demais, tendo em vista a quantidade de tráfego analisada e configurada para gravar em um formato específico.

Figura 1 – Fluxograma da produção do *trace* de tráfego do cenário proposto



Todo o processo de transmissão e coleta de dados do ambiente de teste para a geração do *trace* de tráfego de dados, está representado pela Figura 1. Inicialmente, o netkit cria todas as máquinas virtuais citadas anteriormente. Após isto, o computador gerenciador inicia o envio de uma mensagem UDP (*User Datagram Protocol*) com o seu conteúdo no valor 0, para que todas as máquinas contendo o *script* cliente, possam inicializar os procedimentos de ou coleta ou transmissão ou ataque. Os 3 computadores clientes ao receberem a mensagem, iniciam a transmissão das requisições HTTP. Concomitantemente, o roteador de borda iniciará a coleta do fluxo de dados. Após 180 segundos, os computadores atacantes iniciam a transmissão do tráfego de ataque. Os parâmetros e valores utilizados nas requisições HTTP dos computadores atacantes, são configurados por padrão como descrito em (SLOWHTTPTEST, 2020). O ataque durará enquanto o servidor WEB estiver ativo, caso contrário, eles

voltam a silenciarem-se, retornando a transmitir após 180 segundos e, assim, sucessivamente até o fim do processo. O tempo de duração de todo o processo é de 20 minutos, quando o gerenciador finaliza as transmissões e a coleta de dados, enviando uma mensagem UDP com o conteúdo 1 para todas as máquinas. Com o fim do processo, foi gerado um arquivo de 41.6MB de tamanho, contendo todo o tráfego transmitido. Este mesmo processo foi aplicado para a geração do *trace* de tráfego para o treinamento dos algoritmos de aprendizado de máquina, gerando um arquivo de 161.6MB de tamanho, contendo todo o tráfego transmitido durante o procedimento. Após isto, ambos os *traces* foram tratados e mantiveram somente os dados necessários para serem utilizados pelos algoritmos, sendo convertidos em *datasets*.

É importante mencionar que o software de hospedagem de páginas WEB escolhido foi o Apache, tendo em vista que, ainda, é o servidor WEB mais utilizado no mundo (NETCRAFT, 2020; W3TECHS, 2020).

2.2 Classificação dos atributos

Dois atributos foram selecionados para serem utilizados pelos algoritmos de aprendizado de máquina com o objetivo de classificar o fluxo de tráfego. O primeiro atributo escolhido foi a quantidade de pacotes, como utilizado por Nezhad et al. em (NEZHAD et al., 2016) e Singh e De em (SINGH e DE, 2017), devido ao montante destes, que o ataque gera, em direção à vítima. O segundo foi a entropia, visto que foi identificado que o campo porta de origem do cabeçalho TCP do tráfego de ataque, muda aleatoriamente a cada transmissão enviada da mesma conexão, gerando, assim, um valor alto para este atributo.

A entropia do fluxo de pacotes foi calculada baseado na tupla formada pelo quinteto: IP de origem, porta de origem, IP de destino, porta de destino e protocolo. Este atributo foi também uma escolha utilizada para detectar ataques de negação de serviço em KUMAR et al. em (KUMAR et al., 2007), LAWNICZAK et al. em (LAWNICZAK et al., 2009) e WANG et al. em (WANG et al., 2015), sendo, portanto, considerado neste trabalho. A entropia mede o grau de incerteza associado a uma variável aleatória (SHANNON, 2001). Quanto mais incerto o resultado de um experimento aleatório, maior é a informação obtida observando sua ocorrência, ou seja, quanto mais aleatório é o valor de um campo da tupla do fluxo de dados, mais alto é a sua entropia, ao passo que o contrário, menor é a sua entropia. A equação 1 (SHANNON, 2001) representa o cálculo da entropia:

$$H = \sum_{i=1}^{n} p_i \log p_i \quad (1)$$

Onde H é o resultado do cálculo da entropia, p_i é a probabilidade de cada elemento do quinteto ocorrer na janela deslizante de tempo e n é o número total de pacotes na janela deslizante de tempo do trace de tráfego.

3 RESULTADOS E DISCUSSÕES

Nesta seção, será apresentado como os algoritmos de aprendizado de máquina classificaram cada fluxo de dados no *dataset* do ambiente emulado, o impacto dos casos de testes gerados, bem como a performance de cada um com a quantidade relacionada de falso positivo e falso negativo criados pela classificação geral.

3.1 Métricas de performance

A classificação como resultado das configurações de cada algoritmo de aprendizado de máquina foi feita através das seguintes métricas de performance (NGUYEN e ARMITAGE, 2008):

$$Precisão = \frac{VP}{VP+FP}, \quad (2)$$

$$Revocação = \frac{VP}{VP+FN}, \quad (3)$$

$$F1 - score = 2x\frac{Precisão * Revocação}{Precisão + Revocação}, \quad (4)$$

Em que o VP (Verdadeiro Positivo) é o número de casos positivos corretamente classificados como positivo, ou seja, rotula um fluxo de ataque como ataque. O FP (Falso Positivo) é o número de casos negativos marcados como positivos, ou seja, rotula um fluxo de ataque como legítimo. O VN (Verdadeiro Negativo) é o número de casos negativos classificados como positivo, ou seja, um fluxo de ataque ser rotulado como legítimo. O FN (Falso Negativo) é o número de casos positivos classificados como negativo, ou seja, um fluxo legítimo ser rotulado como ataque. A Precisão, representada pela equação 2, tem como objetivo verificar dentre todas as classificações marcadas como VP, que o modelo fez, quantas estão corretas. Já, a Revogação, representada pela equação 3, tem como objetivo verificar se dentre todas as situações de classificação VP como valor esperado, quantas estão corretas. O F1-score, representado pela equação 4, é a média harmônica entre Precisão e Revogação. E, por fim, é gerada a tabela Matriz Confusão, cujo objetivo é indicar os erros e acertos do modelo de classificação, comparando-os com o resultado esperado.

3.2 Performance da classificação dos algoritmos de aprendizado de máquina

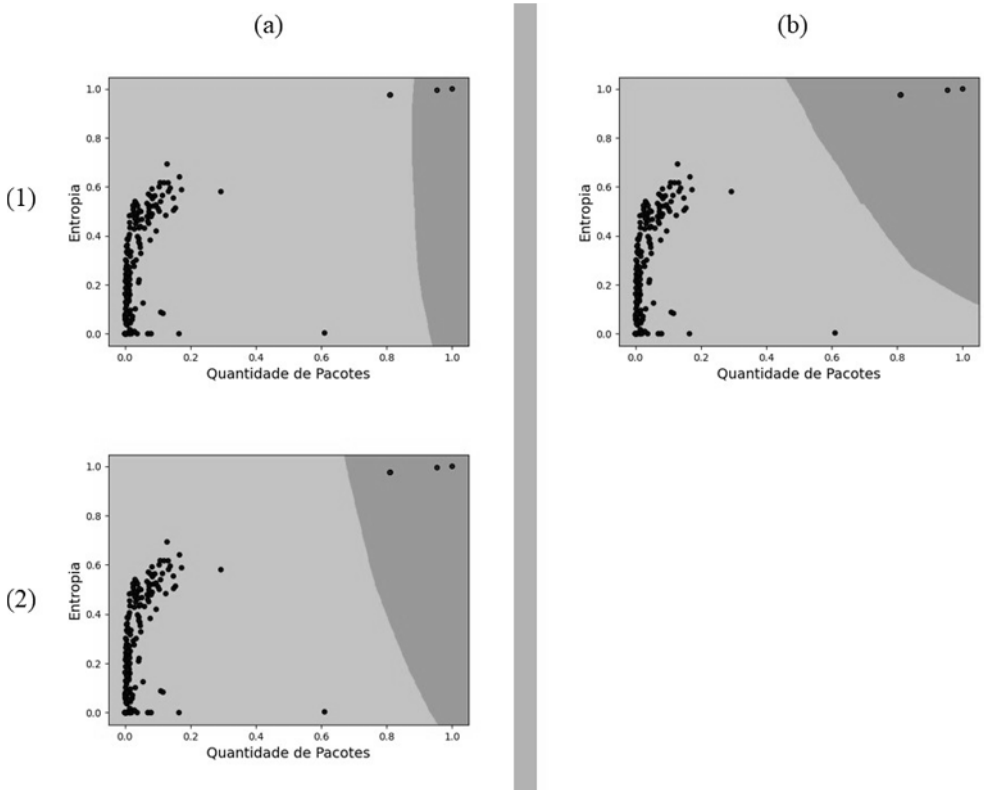
Após aplicar os algoritmos de aprendizado de máquina no *dataset* do ambiente emulado, o conjunto de atributos foram então classificados. Como pode ser observado na Tabela 1, o algoritmo K-NN obteve melhor performance de classificação em relação ao algoritmo rede neural MLP, possuindo 100% de eficiência em classificar o fluxo de dados do *dataset*.

Tabela 1. Resultado da avaliação da performance dos algoritmos rede neural MLP e K-NN baseado no *dataset* gerado

Algoritmo	Tráfego	Precisão	Revogação	F1-score	Matriz confusão $\begin{bmatrix} TP & FN \\ FP & TN \end{bmatrix}$		
Rede Neural MLP	Ataque	100%	95,94%	97,93%		ataque	legítimo
	Legítimo	99,86%	100%	99,93%	ataque	5,76	0,24
					legítimo	0	173
K-NN	Ataque	100%	100%	100%		ataque	legítimo
	Legítimo	100%	100%	100%	ataque	6	0
					legítimo	0	173

A representação dos valores gerados na Tabela 1, está caracterizada pelas fronteiras de decisão na Figura 2. O objetivo é destacar a separação da classificação dos dados que representam um tráfego de ataque, da classificação dos dados que representam um tráfego legítimo. Ainda, como forma de melhor representar os valores do campo Matriz confusão da Tabela 1, optou-se por colocar o resultado da média dos valores dos 1000 casos de testes gerados, por causa do início aleatório dos pesos do algoritmo rede neural MLP.

Figura 2. Fronteira de decisão dos algoritmos rede neural MLP (a) e K-NN (b) com os valores normalizados



Para o algoritmo K-NN foi gerado uma única amostra, representada pela Figura 2 (b), visto que seu resultado se manteve igual em todas as iterações de caso de teste. Isto foi possível, justamente porque o K-NN possui uma separação bem robusta, já que levou em consideração ambos os atributos simultaneamente, por causa da uniformidade dos pesos. Desta forma, obtendo como resultado para

ambos os rótulos de classificação uma Precisão de 100%, uma Revogação de 100%, gerando um F1-score de 100%. Este processo classificatório fica evidente ao observar os valores contidos no campo Matriz Confusão da Tabela 1, em que o algoritmo K-NN obteve o valor 0 para FP e FN.

O algoritmo rede neural MLP obteve uma fronteira de decisão diferente para alguns casos de teste. Assim, como forma de representar todo o processo de classificação das 1000 iterações, utilizou-se apenas duas amostras distintas, uma com e a outra sem falso negativo, como pode ser observado na Figura 2 (a). O atributo quantidade de pacotes foi utilizado mais vezes durante a classificação do conjunto de atributos do *dataset*, como pode ser observado na Figura 2 (a) (2), em que a divisão da fronteira está quase que totalmente horizontal, indicando uma forte classificação para este atributo. Isto ocorreu, muito provavelmente, pela configuração dos parâmetros e dos dados alimentados na rede durante o treinamento, levando o algoritmo a ”entender” que este atributo é mais relevante para a classificação do conjunto. Como resultado, este obteve para o tráfego rotulado como Ataque, uma Precisão de 100%, uma Revogação de 95,94%, gerando um F1-score de 97,93% e para o tráfego rotulado como Legítimo, obteve uma Precisão de 99,86%, uma Revogação de 100%, gerando um F1-score de 99,93%. Este processo classificatório fica evidente ao observar a Figura 2 (a) (1) juntamente com o campo Matriz Confusão da Tabela 1 do algoritmo em que, o valor de 0,24 para FN revela a presença de falsos negativos em alguns casos de teste, tendo uma relação de 1 modelo com caso de FN para 4 modelos em geral.

4 CONSIDERAÇÕES FINAIS

Este trabalho apresenta a performance dos algoritmos de aprendizado de máquina rede neural MLP e K-NN em classificar o fluxo de dados em direção ao servidor WEB, sob a presença do ataque slowloris. Após a análise destes, pode-se observar que a melhor escolha para categorizar o fluxo de dados em tráfego de ataque ou tráfego legítimo, no cenário proposto, levando em consideração os atributos quantidade de pacotes e entropia, é o algoritmo K-NN, por apresentar uma eficácia superior à do algoritmo rede neural MLP. Uma possível causa disto, é que, dependendo do estado aleatório inicial da rede do algoritmo rede neural MLP, o modelo possa ter atingido um mínimo local durante o treinamento de alguns casos de teste, isto é, o algoritmo não convergiu para a melhor solução. Além disto, uma característica que difere estes dois algoritmos, é que, o K-NN, além de não possuir uma inicialização aleatória, não necessita de treinamento. No entanto, esta etapa de treinamento pode ser substituída pela utilização de algoritmos e estruturas de dados que descartam a necessidade de buscas por força bruta durante a classificação. Para trabalhos futuros, pretendemos inserir e avaliar novos algoritmos de aprendizado de máquina, bem como modificar algumas formas de classificar de outros, como por exemplo, a inicialização dos pesos do algoritmo rede neural serem gerados através do método algoritmo genético.

REFERÊNCIAS

AQIL, A.; ATYA, A. O. F.; JAEGER, T.; KRISHNAMURTHY, S. V.; LEVITT, K.; MCDANIEL, P. D.; ROWE, J.; SWAMI, A. Detection of stealthy tcp-based dos attacks. In**: IEEE Military Communications Conference**, p. 348–353, 2015.

DANTAS, Y. G. **Estratégias para tratamento de ataques de negação de serviço na camada de aplicação em redes IP**. Orientador: Vivek Nigam, 2015. 78 p. Dissertação (Mestrado em Ciência da Computação) Universidade Federal da Paraíba, João Pessoa, 2015.

HARRIS, B.; HUNT, R. TCP/IP security threats and attack methods. **Computer communications**, v. 22, n. 10, p. 885–897, 1999.

KUMAR, K.; JOSHI, R. C.; SINGH, K. A distributed Approach using entropy to detect DDoS attacks in ISP domain. In: **International Conference on Signal Processing, Communications and Networking**, p. 331–337, 2007.

LAWNICZAK, A. T.; WU, H.; STEFANO, B. Di. Entropy based detection of DDoS attacks in packet switching network models. **International Conference on Complex Sciences**, p. 1810–1822, 2009.

NETCRAFT. **Web Server Survey**. Disponível em: https://news.netcraft.com/archives/category/web-server-survey. Acesso em: 13 fev. 2020.

NETKIT. **Netkit**. Disponível em: http://wiki.netkit.org/. Acesso em:15 Jan. 2020.

NEZHAD, S. M. T.; NAZARI, M.; GHARAVOL, E. A. A novel dos and ddos attacks detection algorithm using arima time series model and chaotic system in computer networks. **IEEE Communications Letters**, v. 20, n. 4, p. 700–703, 2016.

NGUYEN, T. T. T.; ARMITAGE, G. J. A survey of techniques for internet traffic classification using machine learning. **IEEE Communications Surveys and Tutorials**, v. 10, n. 4, p. 56–76, 2008.

SHANNON, C. E. A mathematical theory of communication. **Mobile Computing and Communications Review**, v. 5, n. 1, p. 3–55, 2001.

SINGH, K. J.; DE, T. MLP-GA based algorithm to detect application layer ddos attack. **Journal of Information Security and Applications**, v. 36, p. 145–153, 2017.

SLOWHTTPTEST. **Slowhttptest**. Disponível em: https://github.com/shekyan/slowhttptest/wiki. Acesso em: 12 fev. 2019.

STORM INTERNET. **Wordpress attack by slowloris**. Disponível em: https://www.storminternet.co.uk/blog/Slowloris-outbreak-affecting-newly-updated-ordPress-sites/. Acesso em 03 Jan. 2020.

W3TECHS. **Comparison of the usage of Apache vs. Nginx for websites**. Disponível em: https://w3techs.com/technologies/comparison/ws-apache,ws-nginx. Acesso em: 13 Fev. 2020.

WANG, R.; JIA, Z.; JU, L. An Entropy-Based Distributed DDoS Detection Mechanism in Software-Defined Networking. **IEEE Trustcom/BigDataSE/ISPA**, v. 1, p. 310–317, 2015.