Aprendizado Profundo para a Predição de Ataques de Negação de Serviço Distribuído

Gabriel Lucas F. M. e Silva[1], **Anderson Bergamini de Neira**[2], **Michele Nogueira**[1 2]

[1]CCSC - Universidade Federal de Minas Gerais (UFMG)

[2]DINF - Universidade Federal do Paraná (UFPR)

{gabriel.silva,michele}@dcc.ufmg.br, abneira@inf.ufpr.br

Resumo. *Dentre as ameaças existentes no ciberespaço, o ataque de negação de serviço distribuído (DDoS) destaca-se por interromper serviços essenciais, negando o acesso a usuários legítimos e causando prejuízos econômicos. A literatura apresenta mecanismos para defender as vítimas. Contudo, os ataques DDoS nem sempre são detectados a tempo para que os mecanismos de defesa evitem os prejuízos. Para aumentar o tempo que a defesa terá para reagir ao ataque, este trabalho propõe um sistema baseado no aprendizado profundo supervisionado para identificar sinais da orquestração de ataques DDoS. Transformando o tráfego de rede em sinais precoces de alerta, este trabalho treinou uma rede neural profunda para identificar anomalias e predizer os ataques DDoS. O sistema proposto foi avaliado no conjunto de dados CTU-13 que contém o tráfego de dois ataques de DDoS. O modelo predisse o lançamento do ataque com 46 minutos de antecedência e uma baixa quantidade de erros.*

1. Introdução

A Internet e o constante desenvolvimento das tecnologias de redes permitem a conexão de milhares de pessoas em todo o mundo. Por meio dos mais diversos serviços e aplicações, ela proporciona a origem de negócios inovadores. Assim, os serviços conectados às redes de computadores desempenham cada vez mais um papel vital em nossas atividades pessoais e profissionais. A crescente importância dos serviços aumenta o incentivo a criminosos em atacá-los para obter ganhos pessoais ilicitamente. Uma das ameaças mais graves no ciberespaço é o ataque de negação de serviço distribuído (em inglês, *Distributed Denial of Service* – DDoS) [Jyoti and Behal 2021]. Esse pode interromper serviços e atualmente este ataque é comumente vendido como um serviço [Armor 2018] para pessoas que desejam prejudicar seus adversários [Nichols 2016].

Por ser uma ameaça repentina e lançar um alto volume de tráfego contra os seus alvos, os mecanismos de defesa possuem pouquíssimo tempo para detectar e reagir aos ataques DDoS. Isso se deve à agressividade do ataque. Em 2022, um ataque DDoS comprometeu o acesso a Internet de quase todos os habitantes de Andorra [Tyson 2022]. O ataque DDoS faz uso de múltiplos dispositivos comprometidos, conectados à Internet, para enviar milhões de mensagens, solicitações de conexão ou pacotes malformados de forma coordenada contra os seus alvos [Marrow and Stolyarov 2021]. O objetivo é consumir os recursos computacionais da vítima, reduzindo o desempenho e provocando a negação de serviços [Bhardwaj et al. 2016] aos usuários legítimos. Os mecanismos de defesa precisam identificar o lançamento do ataque o mais rápido possível para adotar as medidas necessárias, de modo a contê-lo e reduzir os prejuízos.

Assim como outras ameaças cibernéticas, é possível combater ataques DDoS a partir de diversas estratégias de defesa, tais como prevenção, mitigação, detecção, reação e, recentemente, predição [Keshariya and Foukia 2010]. Este trabalho foca na predição de ataques DDoS, que vem sendo desenvolvida para complementar os demais mecanismos de defesa e aumentar o tempo que as vítimas teriam para reagir ao ataque, como abordado em [Wang and Zhang 2017] e [Pelloso et al. 2018]. O Aprendizado Profundo (em inglês, *Deep Learning*) é um subconjunto do Aprendizado de Máquina e vem sendo utilizado para criar soluções de cibersegurança contra diferentes tipos de ameaças. As técnicas de aprendizado profundo foram utilizadas em estratégias de detecção, como em [Elsayed et al. 2020], [Sumathi and Karthikeyan 2021] e [Doriguzzi-Corin et al. 2020]. Neste artigo, defende-se que o aprendizado profundo pode contribuir expressivamente para a predição de ataques DDoS. Todavia, a literatura apresenta poucos trabalhos que utilizam aprendizado profundo para predizer os ataques.

Este trabalho apresenta um sistema baseado em aprendizado profundo para predizer ataques DDoS, evoluindo a literatura e aumentando o tempo de defesa contra esses ataques. Por meio da análise do tráfego de rede da vítima e da utilização do aprendizado profundo supervisionado, o sistema busca sinais precoces de alerta (em inglês, *Early Warning Signals* – EWS). Os EWSs são utilizados para identificar mudanças bruscas em sistemas dinâmicos e, com isso, antecipar fenômenos, em geral, da natureza. Este sistema é o primeiro na literatura que utiliza uma rede neural artificial treinada e embasada em EWSs, computados através de métricas estatísticas e de atributos extraídos dos pacotes do tráfego de rede. Após o treinamento, a rede neural é capaz de comprimir e reconstruir os sinais inseridos como entrada sob um pequeno custo de reconstrução. O valor desse custo é utilizado para definir um limiar e identificar EWSs. Isso é possível pois quando a rede recebe dados de entrada que são muito diferentes dos normais e tenta reconstruí-los, o custo de reconstrução é maior, visto que ela não foi treinada para recriar com precisão dados diferentes dos utilizados em treinamento. Assim, os sinais de orquestração dos ataques indicam o iminente lançamento de um ataque.

A avaliação do sistema utiliza o Cenário 10 do conjunto de dados CTU-13 [García et al. 2014] fornecido pela *Czech Technical University*. Este cenário contém uma captura de tráfego de rede de dois ataques simulados em laboratório. Extraem-se do cabeçalho de cada pacote o tamanho em bytes, o tempo decorrido entre o pacote anterior e o atual durante a comunicação do *Transmission Control Protocol* (TCP) e o tamanho da janela de comunicação TCP. Esses atributos são indicados como melhores atributos para detectar ataques de DDoS [Koay et al. 2019]. Para construir os sinais, os valores de *skewness* e *kurtosis* de cada um desses atributos são calculados para janelas de um segundo do tráfego. Por fim, os sinais construídos são usados para treinar e testar uma rede neural do tipo *Long Short-Term Memory* (LSTM) *Autoencoder* [Kromkowski et al. 2019]. Assim, apenas sinais que antecedem os ataques são utilizados durante o treinamento. O resultado indica a possibilidade de antecipar o primeiro ataque com 46 minutos antes do seu lançamento. A acurácia do sistema foi 99% contando apenas cinco falsos positivos.

Este trabalho procede como segue. A Seção 2 apresenta os trabalhos relacionados à predição de ataques DDoS. A Seção 3 detalha o sistema proposto. A Seção 4 apresenta a avaliação do sistema. Por fim, a Seção 5 conclui este trabalho.

2. Trabalhos Relacionados

Em [Santos et al. 2013], os autores apresentam um sistema para analisar dados encontrados na Internet. As redes sociais como o Twitter são fontes de dados relevantes para a análise proposta, pois é possível identificar campanhas para espalhar *malwares* ou identificar alvos de possíveis ataques. Assim, monitorando textos de mídias sociais como *tweets*, é possível usá-los para identificar potenciais alertas. Esses alertas antecipam ameaças cibernéticas. A solução filtra *tweets* relacionados a ataques DDoS.

[Liu et al. 2015] usam dados coletados na Internet para treinar a máquina de vetores de suporte (em inglês, *Support Vector Machine* - SVM) e predizer eventos de segurança. Os dados coletados na Internet são listas negras de reputação e eventos de segurança. As listas negras representam a reputação de dispositivos que podem estar envolvidos em ataques. Os eventos de segurança são ocorrências de ataques relatados na Internet. Os autores prevêem a ocorrência de eventos relacionados à segurança cibernética com uma média de verdadeiros positivos de 69%. O trabalho não foca em ataques DDoS.

O estudo de [Abaid et al. 2016] visa identificar e modelar o comportamento típico de *botnets* em uma cadeia de Markov. Deste modo, os autores apresentaram uma metodologia para predizer ataques com base na probabilidade de evolução do estado atual para um estado de ataque no futuro próximo. Para treinar a solução da cadeia de Markov, o autor obtém e processa de forma centralizada vários alertas. Durante os experimentos, a solução prediz a comunicação de C&C com precisão de 99%. Este resultado pode ser uma evidência para predizer futuros ataques DDoS. Os resultados ainda mostram que a predição do ataque pode variar de alguns segundos a 18 horas antes do início do ataque.

O estudo de [Pelloso et al. 2018] utiliza a teoria de metaestabilidade para descobrir sinais antes do início do ataque. De forma centralizada, a solução captura o tráfego da rede, prepara os dados capturados e calcula os indicadores. A solução analisa esses indicadores para obter evidências sobre ataques e produzir alertas de ataque DDoS. A solução foi avaliada experimentalmente em dois conjuntos de dados. Como resultado, a solução identificou evidências do ataque duas horas antes do atacante lançar o ataque.

[Holgado et al. 2020] apresentam o uso de alertas produzidos por um sistema de detecção de intrusão (*Intrusion Detection Systems - IDS*) para predizer ataques usando a Cadeia Oculta de Markov (em inglês, *Hidden Markov Model* - HMM). A proposta realiza o treinamento offline onde a solução compara alertas IDS com dados históricos previamente definidos. Após o treinamento, a solução define os estados e as probabilidades de mudanças. O ataque é predito quando existe a probabilidade da HMM evoluir de um estado sem ataque para um estado de ataque. Os autores verificaram que a técnica prediz um ataque DDoS cerca de 11 minutos de vantagem.

Apesar destes trabalhos, muito ainda pode ser feito para evoluir a área de predição de ataques DDoS. Neste cenário, soluções que utilizam aprendizado profundo podem evoluir o estado da arte da predição de ataques DDoS apresentando novas soluções com bons resultados. Essa técnica pode suprir a limitação de uso de dados rotulados.

3. Aprendizado Profundo para a Predição de ataques DDoS

Esta seção descreve a solução proposta para predizer ataques DDoS baseada em Aprendizado Profundo supervisionado. O sistema utiliza uma rede neural profunda do tipo

LSTM *Autoencoder* capaz de identificar EWSs. Os EWSs são sinais estatísticos extraídos de série temporais, capazes de antecipar transições críticas antes que elas aconteçam. Essas transições podem ser fenômenos ambientais, doenças e, em nosso trabalho, ataques de DDoS. Quando o sistema identifica um sinal precoce de alerta ele notifica os administradores de rede. Assim, a equipe terá mais tempo para lidar com ataques DDoS. A Subseção 3.1 descreve como a coleta os dados do tráfego de rede é realizada. A Subseção 3.2 detalha o pré-processamento do tráfego de rede. A Subseção 3.3 explica a configuração da rede neural implementada. A Subseção 3.4 apresenta como a identificação de anomalias é utilizada associando-as aos EWSs com o LSTM *Autoencoder*. Por fim, a Subseção 3.5 descreve a notificação dos administradores de rede.

3.1. Coleta dos Dados

A primeira etapa do sistema consiste na coleta do tráfego de rede. Essa etapa é importante pois o tráfego de rede será utilizado para treinar o modelo de aprendizado profundo e realizar a predição dos ataques. Para realizar a coleta dos dados, uma ferramenta de captura de tráfego de rede integrada ao *firewall* é utilizada para interceptar todo o tráfego de entrada e saída da rede do usuário e extrair os cabeçalhos dos pacotes trafegados. Após a extração, os cabeçalhos são exportados para para arquivos do tipo Captura de Pacotes (em inglês, *Packet Capture* – PCAP) e armazenados em um servidor de arquivos. O limite máximo de tráfego armazenado por captura pode ser configurado pelos usuários mas, por padrão, cada captura armazenada pelo sistema contém 1 segundo de tráfego. O tamanho máximo da captura também pode ser definido pela quantidade de pacotes. Uma vez salvos, os arquivos ficarão disponíveis no servidor de arquivos para serem pré-processados na etapa seguinte. O servidor de arquivos monitora periodicamente a capacidade de armazenamento e, por padrão, deleta as capturas mais antigas quando a capacidade de armazenamento está próxima do fim.

3.2. Pré-Processamento dos Dados

Após a coleta dos dados do tráfego de rede, o servidor de pré-processamento fica responsável por verificar as capturas. Para isso, o servidor de pré-processamento monitora periodicamente o servidor de arquivos em busca de novas capturas. Quando uma nova captura está disponível, o servidor faz o *download* dela. Por padrão, o sistema realiza essa consulta a cada segundo. Em seguida, o pré-processamento carrega os dados na memória e extrai os atributos dos cabeçalhos dos pacotes: o tamanho em bytes, o tempo decorrido entre o pacote anterior e o atual durante a comunicação TCP e o tamanho da janela de comunicação TCP. A literatura cita esses atributos como os mais relevantes para detectar ataques de DDoS [Koay et al. 2019]. Os valores extraídos são armazenados em uma matriz bi-dimensional $m \times n$, onde m é igual ao número total de pacotes coletados e n a quantidade de atributos extraídos. Após a extração, a matriz é armazenada em um banco de dados e a captura de tráfego é deletada.

A próxima etapa de pré-processamento é transformar as matrizes obtidas em EWSs associadas às medidas estatísticas: *Skewness* e *Kurtosis*. Essas medidas foram utilizadas em outros trabalhos para realizar predições. [Dakos et al. 2012] utilizou essas medidas para antecipar mudanças ecológicas e [Xie et al. 2019] para predizer mudanças climáticas. Essas métricas baseiam-se no conceito de séries temporais. Uma série temporal é constituída de observações realizadas sequencialmente no tempo [Box et al. 2015].

Dadas as observações sequenciais x_1 e x_2,..., x_N a variável randômica x_1 representa o valor da primeira observação, o x_2 representa o valor na segundo ponto de observação, o x_N representa o valor da enésima observação [Shumway and Stoffer 2017]. *Skewness* é uma medida de assimetria de uma distribuição de probabilidade idealmente simétrica e indica o quanto a distribuição de probabilidade de uma variável aleatória desvia da sua distribuição normal. A Eq. 1 apresenta o cálculo do *Skewness*. O x_t refere-se a cada item observado na série temporal. O N representa a quantidade total de itens observados. O $\bar{x}$ refere-se a média aritmética simples e o termo s representa o desvio padrão. O *Kurtosis* é uma medida que caracteriza o achatamento da curva de uma distribuição. Ela indica o quanto uma variável se encontra nas caudas da distribuição. A Eq. 2 define o cálculo do *Kurtosis*. O termo N representa a quantidade de itens. Para calcular o *Kurtosis* ($\hat{y}^U$) é necessário o resultado de $\hat{y}$, definido por $\hat{y} = \frac{N\sum(x_t-\bar{x})^4}{[\sum(x_t-\bar{x})^2]^2}$. O termo x_t refere-se a cada item da série temporal. E o termo $\bar{x}$ refere-se a média aritmética simples.

$$g = \frac{N\sum_{t=1}^{N}(x_t-\bar{x})^3}{(N-1)(N-2)s^3} \qquad (1) \qquad \hat{y}^U = \frac{(N-1)}{(N-2)(N-3)}(N-1)\hat{y}+6 \qquad (2)$$

As pertubações ou oscilações afetam a distribuição dos dados e geram uma variação que caracteriza os sinais (EWS). Essa variação ocorre quando o sistema muda de estado, por exemplo, variando de um valores negativos para valores positivos. Como a transição de um estado para o outro apresenta uma desaceleração, é possível observar um aumento ou redução na *Skewness* de uma série temporal, uma vez que a distribuição dos valores na série temporal se tornará assimétrica. De forma análoga, as oscilações ou fortes perturbações também tornam mais provável que o estado de um sistema possa atingir valores extremos próximos a uma transição. Esses efeitos podem levar a um aumento na *Kurtosis* de uma série temporal anterior à transição. Desse modo, a cauda da distribuição da série temporal tornam-se mais largas devido ao aumento da presença de valores raros na série temporal. Assim, os atributos extraídos de cada pacote são transformados em sinais, utilizando *Skewness* e *Kurtosis*, com o intuito de identificar transições críticas durante a orquestração dos ataques e, dessa forma, antecipar seu lançamento.

Para gerar os sinais, as entradas da matriz são agregadas em janelas de 1 segundo do tráfego utilizando as funções de *Skewness* e *Kurtosis*. Dessa maneira, para cada um dos três atributos utilizados, dois novos valores são gerados para cada segundo. Como resultado, uma segunda matriz $m' \times n'$ é obtida, onde m' é igual ao número de segundos total da captura e n' igual ao número total de novos atributos depois da transformação. Com o objetivo de acelerar o processo de aprendizado e obter uma convergência mais rápida, nós normalizamos os dados para que todos os valores ficassem entre 0 e 1. Para isso, a Eq. 3 foi utilizada. Onde, o $X(j)$ representa o valor de cada amostra do atributo analisado, o min_A e o max_A são respectivamente menor e o maior valor observado para o atributo. O min'_A e o max'_A representam o novo intervalo. Por fim, para que os dados ficassem em um formato compatível com a entrada da rede neural, as entradas são formatadas em uma matriz tridimensional no formato $m' \times 1 \times n'$.

$$mm = \frac{X(j)-min_A}{max_A-min_A}(max'_A-min'_A)+min'_A \qquad (3)$$

3.3. Reconstrução dos Sinais Temporais

Após o pré-processamento, o sistema utiliza os dados para treinar e testar uma rede neural profunda chamada LSTM *Autoencoder*. Essa rede neural é capaz de aprender a gerar uma versão compactada dos dados processados e utilizá-la para reconstruir os sinais originais mediante um custo de reconstrução. Esse custo de reconstrução é o resultado da diferença entre o sinal original e o sinal reconstruído na saída da rede. Quando o valor do custo é igual a zero, o sinal reconstruído é idêntico ao sinal inserido na entrada, do contrário, se o custo de reconstrução for maior ou menor que zero, os valores são diferentes. O custo de reconstrução dos sinais pode ser utilizado para identificar EWSs e predizer os ataques de DDoS. Ao receber sinais com tráfego de orquestração do ataque, que não foram utilizados para treinar o modelo, o valor do custo se destaca com relação ao restante do tráfego.

As LSTMs são um subtipo das Redes Neurais Recorrentes (em inglês, *Recurrent Neural Networks* – RNNs) [Hochreiter and Schmidhuber 1997], projetadas para lidar com entradas de dados sequenciais. Como elas utilizam uma memória interna, essas redes possuem a capacidade de conservar informações, armazenando o estado das células para uso posterior. Elas fazem isso com um conjunto de portas (*gates*) utilizadas para controlar quando as informações entram, são emitidas ou apagadas das células. As LSTMs são adequadas para solucionar problemas que utilizam dados que evoluem ao longo do tempo e para realizar predições [Lindemann et al. 2021].

Um *autoencoder* é um tipo de rede neural que utiliza aprendizado auto-supervisionado para aprender a gerar uma representação compacta de uma entrada de dados e, a partir dessa representação, reconstruir os dados inseridos na entrada. Para tal, a arquitetura do *autoencoder* é organizada em uma estrutura Codificador-Decodificador, que possui um gargalo em seu ponto médio. Esse gargalo funciona ao mesmo tempo como saída do Codificador e entrada do Decodificador. Para treinar o *autoencoder*, os pesos da rede são geralmente inicializados aleatoriamente e, em seguida, atualizados iterativamente durante o treinamento por meio de *backpropagation*. Uma vez ajustada, a saída da rede no gargalo é um vetor de comprimento fixo que fornece uma representação compacta dos dados de entrada. A saída da rede no Decodificador é a versão reconstruída dos dados inseridos no Codificador, a partir da representação compactada emitida no gargalo. Após o treinamento, o Decodificador pode ser descartado e o Codificador empilha junto aos outros tipos de redes neurais para aplicações diversas. Com isso, modelos do *autoencoder* já foram aplicados em muitos campos como, visão computacional, reconhecimento de fala e processamento de linguagem natural [Zhai et al. 2018].

O LSTM *Autoencoder* é uma rede LSTM organizada na arquitetura Codificador-Decodificador do *autoencoder*. Essa arquitetura potencializa as capacidades do *autoencoder* tradicional ao permitir a reconstrução de séries temporais através do aprendizado de conjuntos de dados sequenciais, possibilitado pelas células das camadas de rede LSTM. O LSTM *Autoencoder* foi utilizado para predizer anomalias em dados de séries temporais sobre vendas [Nguyen et al. 2021] e para predizer os quadros de um vídeo [Srivastava et al. 2015]. Para segurança de redes de computadores, o LSTM *autoencoder* foi aplicado para detectar atividades maliciosas em um tráfego de rede [Kromkowski et al. 2019].

A arquitetura do LSTM *Autoencoder* neste trabalho possui 5 camadas. As duas primeiras camadas da rede neural formam o Codificador e criam a representação com-

pacta dos dados de entrada. O Codificador empilha duas camadas LSTM contendo 16 e 4 neurônios, respectivamente. Em seguida, usa-se uma camada de vetor de repetição para distribuir o vetor compactado ao longo das etapas de tempo do Decodificador. O Decodificador possui mais duas camadas LSTM empilhadas com 4 e 16 neurônios, respectivamente. A saída do neurônio f segue a Unidade Linear Retificada (em inglês, *Rectified Linear Unit* – ReLU), como função de ativação ($f(x) = x^+ = \max(0, x)$). A camada de saída final do Decodificador fornece os dados de entrada reconstruídos. Por fim, a rede neural possui como otimizador Adam e o Erro Absoluto Médio para calcular a função de perda. A Figura 1 ilustra a arquitetura da rede neural.

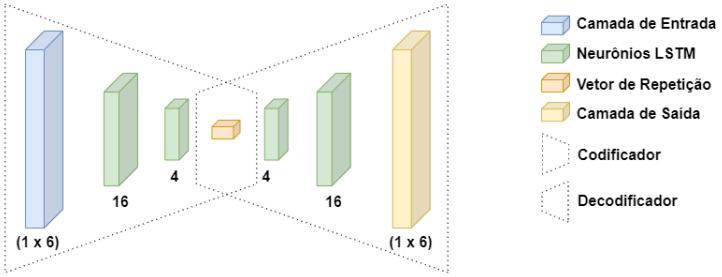


Figura 1. Arquitetura do LSTM *Autoencoder*

3.4. Identificação de Sinais Precoces de Alerta

O LSTM *Autoencoder* aprende a criar uma representação compacta de sinais temporais inseridos na entrada da rede neural e, a partir dessa representação, é capaz de recriar sinais temporais similares aos introduzidos na entrada sem gerar *overfitting*. No entanto, esses sinais não são exatamente iguais e um custo de reconstrução pode ser calculado ao comparar o sinal reconstruído na saída da rede neural com o sinal de entrada. Assim, este trabalho utiliza o Erro Médio Absoluto (em inglês, *Mean Absolute Error* – MAE) como o custo de reconstrução. O MAE (Eq. 4) é uma média dos erros absolutos $|e_i| = |y_i - x_i|$, onde y_i é o valor reconstruído e x_i o valor original. O termo n é a quantidade de sinais.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} = \frac{\sum_{i=1}^{n} |e_i|}{n} \tag{4}$$

Para identificar os sinais anteriores ao ataque e antecipá-los, nós monitoramos os resultados da função de custo de reconstrução, uma vez que o custo para reconstruir um sinal normal, sem sinais de orquestração do ataque, é pequeno, visto que possui dados parecidos com os que foram utilizados na etapa de treinamento. Contudo, quando o modelo recebe um sinal que possui tráfego malicioso, ou seja, com dados diferentes dos que foram utilizados no treinamento, o custo de reconstrução desse sinal é maior e pode apresentar valores discrepantes quando comparados com os valores de custo obtidos na etapa de treinamento. Através desse comportamento, nós definimos um limiar L para os resultados gerados pela função de custo de modo a separar o que é sinal normal, do que é sinal de orquestração do ataque. Esses sinais o sistema utiliza para antecipar os ataques DDoS. A Figura 2 detalha esse processo.

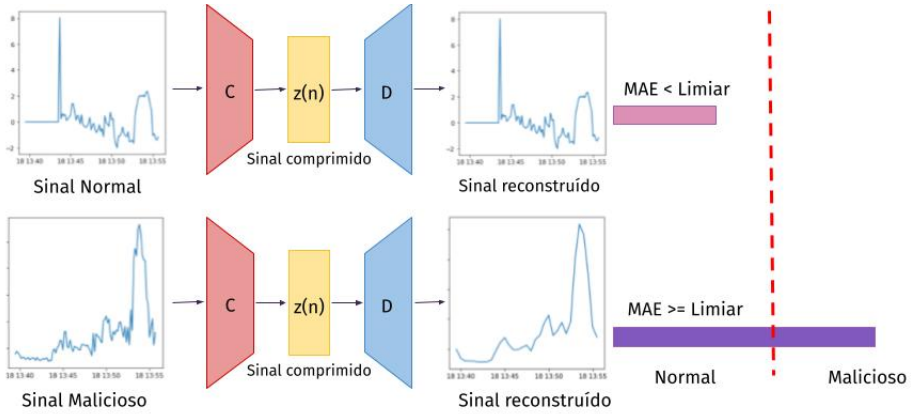


Figura 2. Identificação de Sinais Precoces de Alerta

3.5. Notificação do Ataque

Na última etapa, a solução utiliza a saída do LSTM *Autoencoder* para emitir notificações sobre a ocorrência de possíveis ataques DDoS. O sistema envia mensagens para os administradores quando os sinais analisados pelo LSTM *Autoencoder* ultrapassam o valor do limiar configurado. A notificação de possível ataque compreende ciclos temporais em que os EWSs variam anormalmente. O LSTM *Autoencoder* utiliza essa anomalia como sinal de um possível ataque DDoS. O ciclo normal indica que o ciclo não possui anomalias, portanto não existem sinais de que um possível ataque aconteça, assim nenhuma notificação é emitida. Se o sistema identificar algum sinal de ataque, ele notifica os administradores de rede para conduzirem as medidas cabíveis. Existem opções para essa notificação, como o envio de mensagens de texto em mensageiros instantâneos ou plataformas de comunicações empresariais como o Slack e o Microsoft Teams. Uma opção para automatizar a resposta a incidentes é se comunicar com uma interface de programação de aplicativos (em inglês, *Application Programming Interface* - API) web para transmitir os dados da ocorrência que pode ser utilizada como entrada para automatizar uma outra solução de cibersegurança.

4. Avaliação de Desempenho

Esta seção descreve a metodologia e os resultados da avaliação de desempenho do sistema proposto. A avaliação segue dois experimentos na mesma base de dados, com o intuito de analisar como diferentes técnicas de transformação dos dados afetam o desempenho do modelo. Os experimentos utilizaram a plataforma Google Colaboratory[1]. Esta disponibiliza o acesso a uma máquina virtual com 12.69 GB de memória RAM, 107.72 GB de armazenamento e Subsma GPU NVIDIA Tesla K80. Todo o código foi implementado utilizando a linguagem Python. A Subseção 4.1 descreve a base de dados utilizada e os procedimentos para pré-processar os dados, além das métricas para a avaliação dos experimentos. As Subseções 4.2 e 4.3 detalham os experimentos. Por fim, a Subseção 4.4 apresenta uma discussão dos resultados.

[1]https://colab.research.google.com/

4.1. Metodologia

A avaliação utiliza como entrada o Conjunto de Dados CTU-13. Esse conjunto foi criado por pesquisadores da Universidade Técnica Tcheca (em inglês, *Czech Technical University* – CTU) em 2011. O conjunto é dividido em 13 cenários e cada um contém o tráfego proveniente de máquinas benignas e de máquinas infectadas por diferentes *botnets*. O cenário utilizado foi o Cenário 10. Esse cenário possui 4,75 horas de duração e cerca de 1,3 milhões de fluxos de rede. Este cenário contém o tráfego de *botnet* antes dos ataques e depois do início do ataque DDoS.

Para simplificar a condução dos experimentos, o pré-processamento começa com a separação da captura de rede original em arquivos menores, contendo no máximo 1.000.000 de pacotes cada. Em seguida, o sistema extrai os atributos do cabeçalho de cada pacote dos arquivos de tráfego de rede. Como resultado, um arquivo do tipo CSV no formato $p \times q$ foi gerado para cada um dos 46 pedaços da captura, onde p é igual ao número de pacotes por arquivo e q igual ao número de atributos extraídos. Em seguida, o sistema pré-processa todos os arquivos CSV e transforma cada uma de suas entradas nos sinais temporais de *Skewness* e *Kurtosis* de acordo com os procedimentos descritos na Seção 3.2. Como resultado, uma matriz no formato $p' \times q'$ foi obtida, onde p' é igual a quantidade total de segundos que antecedem o ataque e q' é igual ao número de novos atributos. Neste caso, $q' = 6$, visto que os valores de *Skewness* e *Kurtosis* foram computados para cada um dos 3 atributos extraídos de cada pacote.

Para calcular as métricas de avaliação descritas anteriormente, é necessário rotular cada segundo do conjunto de dados como tráfego normal ou malicioso. Isto auxilia a verificar em quais momentos os *bots* influenciam nos atributos coletados no tráfego de rede. Ao analisar toda a base de dados, espera-se que o sistema apresente maior erro médio absoluto nos momentos em que os *bots* estão se preparando para atacar. Para rotular a base, todo segundo que possui pacotes enviados ou recebidos pelos *bots* foram considerados potenciais sinais de orquestração do ataque e, portanto, tráfego malicioso. Os demais segundos foram considerados como tráfego normal.

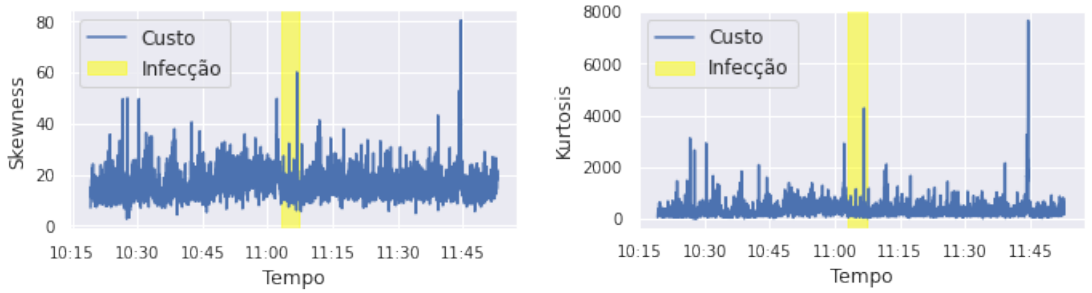
Como o intuito da solução é identificar os sinais iniciais do ataque, apenas o tráfego que precede o primeiro ataque de DDoS foi utilizado para treinar e testar a rede neural. Desse modo, selecionou-se todo o tráfego anterior ao primeiro ataque, que inicia às 10:19:13 de 18/08/2011 e termina às 11:52:53 do mesmo dia, totalizando 5680 segundos de tráfego. Esse sub-conjunto foi dividido em 3 partes iguais, onde a primeira parte foi utilizada para treino e as duas partes restantes para teste. Vale ressaltar também que, da primeira parte utilizada para treinar os modelos, 10% foi destinado para validação. Além disso, cada modelo foi treinado utilizando 100 épocas de treino.

As métricas acurácia, precisão e sensibilidade (em inglês, *Recall*) são aplicadas para avaliar o desempenho do sistema. Para obter as métricas é necessário calcular o total de acertos positivos (VP) e negativos (VN), o total de erros positivos (FP) e negativos (FN) e o total observações (N). A acurácia (Eq. 5) mede a proporção de previsões corretas entre o número total de casos examinados. A precisão (Eq. 6) mede a fração de previsões positivas que realmente pertencem ao conjunto de previsões positivas. A sensibilidade (Eq. 7) quantifica o número de predições positivas feitas entre todos os exemplos positivos no conjunto de dados.

$$a = \frac{VP + VN}{N} \quad (5) \qquad p = \frac{VP}{VP + FP} \quad (6) \qquad r = \frac{VP}{VP + FN} \quad (7)$$

4.2. Experimento 1

No experimento 1, instanciou-se o sistema utilizando o LSTM *Autoencoder* projetado para seis sinais temporais como entrada. Cada um dos sinais foi obtido agregando os valores do tamanho dos pacotes, tempo entre pacotes e tamanho da janela de comunicação TCP, por segundo do subconjunto de tráfego que precede o primeiro ataque e calculando o valor de *Skewness* e *Kurtosis* para cada segundo agregado. Os sinais construídos podem ser visualizados na Figura 3. Para simplificar a apresentação dos resultados a figura apresenta apenas o *skewness* (Figura 3(a)) e o *kurtosis* (Figura 3(b)) obtidos por meio da análise da variação do tempo entre os pacotes. Porém, todos os dados referentes aos valores do *skewness* e o *kurtosis* para os outros atributos estão disponibilizados online[2]. A partir dos gráficos, é possível verificar que os sinais gerados apresentam poucos valores contínuos, uma alta frequência e também alta granularidade das informações.



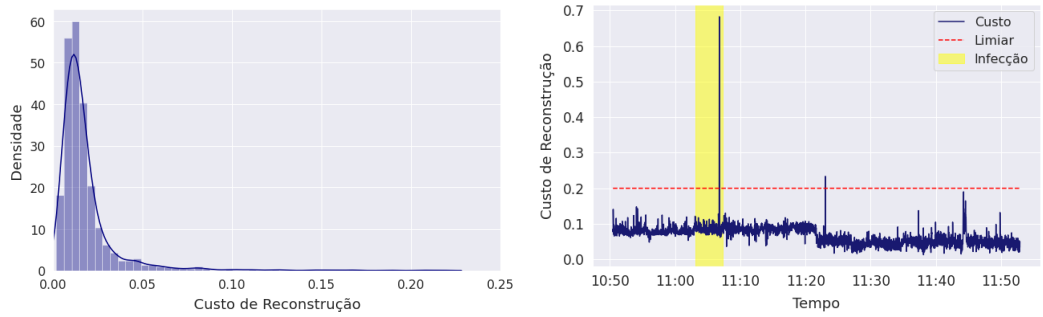
(a) *Skewness* (b) *Kurtosis*

Figura 3. *Skewness* e *Kurtosis* do tempo entre pacotes TCP no Experimento 1.

A Figura 4(a) apresenta a distribuição dos valores de custo para reconstruir o sinal utilizado durante o treinamento. Verifica-se que a maioria dos valores de custo de reconstrução fica entre 0,0 e 0,1. Além disso, em poucos casos, o modelo obteve um custo de reconstrução entre 0,1 e 0,2 para alguns segundos do tráfego. Desse modo, para garantir que apenas anomalias sejam identificadas como EWSs, configurou-se o limiar para notificar o ataque quando o custo para reconstruir os sinais ultrapassasse 0,2. Assim, todo segundo do sinal do subconjunto de teste que possuía um valor de custo de reconstrução acima de 0,2, foi considerado um sinal de orquestração do ataque e gera um alerta para os administradores de rede. A Figura 4(b) ilustra os valores do custo para reconstruir todos os sinais do conjunto de testes no Experimento 1. Os valores do custo para reconstruir alguns segundos dos sinais do conjunto de teste ficam acima do limiar configurado, porém a grande maioria fica abaixo de 0,2. Além disso, a maior parte dos valores de custo são próximos entre si e ficam entre 0,0 e 0,1, refletindo o mesmo comportamento indicado pela Figura 4(a) para reconstruir os sinais utilizados durante o treinamento.

Quando o custo de reconstrução ultrapassa o limiar, o sistema notifica os administradores do sistema sobre a possibilidade de um ataque DDoS iminente. Para verificar se essas notificações possuem tráfego malicioso, rotularam-se todos os segundos do tráfego

[2]https://github.com/gsilv4/sbrc-2022-silva-neira-nogueira



(a) *Custo de Reconstrução do Conjunto de Treino.* (b) *Custo de Reconstrução do Conjunto de Teste.*

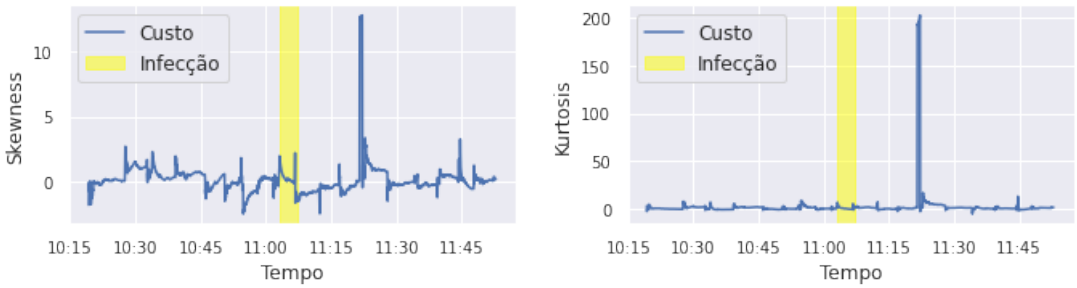
Figura 4. Custo de reconstrução no tempo para o Experimento 1

original que possuem tráfego de *bot* como maliciosos e os demais como tráfego normal. Todos os segundos que possuem pacotes enviados ou recebidos pelos *bots* foram considerados tráfego malicioso. Os demais segundos foram considerados tráfego normal. Em seguida, verificou-se se os segundos que geraram um custo de reconstrução acima do limiar (Figura 4(b)) possuíam tráfego malicioso. O sistema proposto identificou três segundos que possuem atividade de *bots* durante o período de infecção da *botnet*. Neste três segundos, o sistema notifica os administradores sobre a possibilidade de um ataque DDoS futuro. A primeira notificação ocorreu em 46 minutos antes do lançamento do primeiro ataque. Apesar deste resultado, o sistema gerou uma notificação de ataque quando não havia tráfego malicioso, sendo então um falso positivo. Além disso, 255 segundos são afetados pelo tráfego de bots e o custo de reconstrução não ultrapassou o limiar. Isso fez o sistema classificar esses segundos como normais, apresentando 255 falsos negativos. Acredita-se que isso seja causado pelo pouco impacto dos *bots* nas métricas.

Por fim, o sistema foi capaz de classificar corretamente 2737 segundos normais (VN). A partir desses resultados as métricas de avaliação descritas anteriormente são calculadas. Obtemos 91% de Acurácia, 75% de Precisão e 1% de Sensibilidade. A métricas demonstram que o modelo apresenta um alto percentual de acertos. Além disso, das quatro vezes que o sistema identificou uma anomalia, três predições foram realizadas corretamente. Porém, o baixo resultado da Sensibilidade demonstra a necessidade de ajustes para que o modelo consiga classificar mais sinais de orquestração do ataque e aumentar a quantidade de notificações.

4.3. Experimento 2

Com o intuito de reduzir a granularidade do conjunto de dados e aumentar o desempenho do modelo, um novo experimento utilizando janelas deslizantes foi realizado. Para tal, repetiu-se a fase de transformação do pré-processamento do conjunto de dados e calculou-se a média dos valores de cada atributo para cada segundo da captura. Em seguida, aplicou-se o algoritmo de janelas deslizantes. O tamanho da janela foi configurado para deslizar sobre 10% da duração total do tráfego que precede o ataque. Como essa fatia do conjunto possui 5680 segundos, o tamanho da janela definido foi de 568 segundos. O que significa que cada segundo dos sinais construídos foi calculado considerando os valores dos atributos dos pacotes enviados no momento atual e também nos 567 segundos anteriores. A Figura 5 apresenta que os sinais obtidos neste experimento destoam dos construídos no Experimento 1, visto que eles apresentam uma redução significativa na oscilação das medidas.

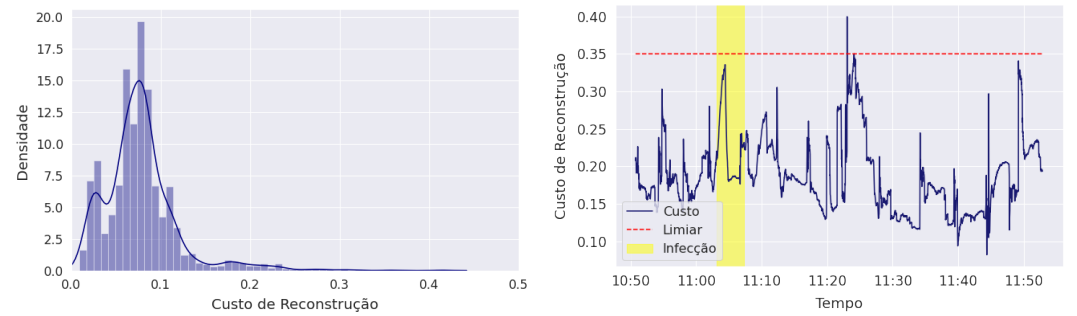


(a) *Skewness* (b) *Kurtosis*

Figura 5. *Skewness* e *Kurtosis* do tempo entre pacotes no Experimento 2

O modelo foi treinado utilizando a mesma arquitetura do experimento anterior, com a mesma divisão do conjunto de dados e quantidade de épocas de treino. Neste caso, foi considerada a distribuição dos valores do custo de reconstrução obtido durante o treinamento para definir o valor do limiar. A Figura 6(a) ilustra a distribuição do custo de reconstrução obtida nesta etapa. Assim como no primeiro experimento, a maior parte dos valores ficou distribuída entre 0,0 e 0,1. Porém, uma quantidade maior de instantes apresentou um custo de reconstrução entre 0,1 e 0,3, o que sugeriu que um limiar com um valor mais alto deveria ser utilizado para este modelo. Sendo assim, o limiar foi configurado para emitir notificações quando o custo de reconstrução fosse maior que 0,35.

A Figura 6(b) apresenta os valores de custo de reconstrução para reconstruir os sinais do conjunto de testes utilizando o novo modelo treinado. Ela indica que o custo de reconstrução acompanha as mesmas características dos sinais construídos na etapa de pré-processamento. Assim como no Experimento 1, é possível identificar um pico no custo de reconstrução durante a infecção das máquinas, mas esse pico não foi suficiente para predizer o ataque, uma vez que o valor escolhido para o limiar ficou mais alto. Além disso, duas predições de ataque são realizadas, a primeira às 11:23:03 e a segunda às 11:24:07, onde apenas a primeira estava correta. Diante disso, apesar de acertar na classificação de quase todos os sinais benignos e conseguir antecipar o ataque com 29 minutos de antecedência, a maioria dos instantes que possui tráfego malicioso ficaram abaixo do limiar, o que impactou diretamente na avaliação do modelo. Nesse caso, os resultados indicam 91% de Acurácia, 50% de Precisão e 0,4% de Sensibilidade.



(a) *Custo de Reconstrução do Conjunto de Treino.* (b) *Custo de Reconstrução do Conjunto de Teste.*

Figura 6. Custo de reconstrução no tempo para o Experimento 2

4.4. Discussão

O sistema proposto é capaz de antecipar o lançamento do ataque DDoS e aumentar o tempo em que os administradores de rede tem para reagir à ameaça. Contudo, muitos segundos que possuem tráfego malicioso não foram identificados, aumentando a taxa de erros do sistema. Uma alternativa para melhorar o desempenho do modelo é utilizar um limiar dinâmico, capaz de se adaptar a diferentes cenários e aumentar a quantidade de predições corretas. Além disso, outro ponto a se trabalhar é refinar a forma com que os sinais são rotulados para avaliação.

5. Conclusão

A predição de ataques é um mecanismo de defesa que vem ganhando atenção na literatura. As técnicas de predição criam evidências da preparação de um ataque DDoS, para fornecer mais tempo para os administradores de rede lidarem com o ataque. O *aprendizado profundo* tem potencial para tratar com a predição de ataques DDoS. Este trabalho apresentou um sistema de predição de ataques DDoS utilizando o LSTM *Autoencoder* aliado à teoria do sinais antecipados de alerta. Os resultados da avaliação no cenário 51 da CTU-13 indicam que o sistema foi capaz de predizer o ataque DDoS com 46 minutos de antecedência. Esse resultado é expressivo pois os ataques estão evoluindo em sofisticação e volume. Além disso, as estratégias para lidar com os ataques demandam tempo para evitar os prejuízos. Assim, o sistema proposto auxilia os administradores de rede a evitarem prejuízos relacionados aos ataques DDoS. Como trabalhos futuros, vislumbra-se evoluir o sistema proposto através da criação de uma estratégia adaptativa para o limiar que é utilizado para diferenciar sinais diferentes dos preditos, indicando assim, possíveis sinais da orquestração de ataques DDoS. Também é possível evoluir a proposta automatizando a preparação dos dados o aprendizado de máquina autônomo.

Referências

Abaid, Z., Sarkar, D., Kaafar, M. A., and Jha, S. (2016). The early bird gets the botnet: A markov chain based early warning system for botnet attacks. In *LCN*, pages 61–68, UAE. IEEE.

Armor (2018). Armor's 'black market' report highlights the big business of cybercrime. Acesso em: 07/21. armor.com/resources/press-release/black-market-report-highlights-cybercrime/.

Bhardwaj, A., Subrahmanyam, G. V. B., Avasthi, V., Sastry, H., and Goundar, S. (2016). DDoS attacks, new DDoS taxonomy and mitigation solutions — A survey. In *SCOPES*, page 5.

Box, G. E., Jenkins, G. M., Reinsel, G. C., and Ljung, G. M. (2015). *Time series analysis: forecasting and control*. John Wiley & Sons.

Dakos, V., Carpenter, S. R., Brock, W. A., Ellison, A. M., Guttal, V., Ives, A. R., Kéfi, S., Livina, V., Seekell, D. A., van Nes, E. H., and Scheffer, M. (2012). Methods for detecting early warnings of critical transitions in time series illustrated using simulated ecological data. *PLOS ONE*, 7(7):1–20.

Doriguzzi-Corin, R., Millar, S., Scott-Hayward, S., Martínez-del Rincón, J., and Siracusa, D. (2020). Lucid: A practical, lightweight deep learning solution for DDoS attack detection. *TNSM*, 17(2):876–889.

Elsayed, M. S., Le-Khac, N.-A., Dev, S., and Jurcut, A. D. (2020). DDoSNet: A deep-learning model for detecting network attacks. In *WoWMoM*, pages 391–396.

García, S., Grill, M., Stiborek, J., and Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Holgado, P., Villagrá, V. A., and Vázquez, L. (2020). Real-time multistep attack prediction based on hidden markov models. *IEEE Trans. Dependable Secure Comput*, 17(1):134–147.

Jyoti, N. and Behal, S. (2021). A meta-evaluation of machine learning techniques for detection of DDoS attacks. In *INDIACom*, pages 522–526, India. IEEE.

Keshariya, A. and Foukia, N. (2010). DDoS defense mechanisms: A new taxonomy. In *DPM*, pages 222–236. Springer Berlin Heidelberg.

Koay, A., Welch, I., and Seah, W. (2019). (Short Paper) Effectiveness of entropy-based features in high- and low-intensity DDoS attacks detection. In *IWSEC*, pages 207–217.

Kromkowski, P., Li, S., Zhao, W., Abraham, B., Osborne, A., and Brown, D. E. (2019). Evaluating statistical models for network traffic anomaly detection. In *SIEDS*, pages 1–6.

Lindemann, B., Müller, T., Vietz, H., Jazdi, N., and Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP*, 99:650–655.

Liu, Y., Zhang, J., Sarabi, A., Liu, M., Karir, M., and Bailey, M. (2015). Predicting cyber security incidents using feature-based characterization of network-level malicious activities. In *IWSPA*.

Marrow, A. and Stolyarov, G. (2021). Russia's Yandex says it repelled biggest DDoS attack in history. [Acesso em: 10/2021]. https://www.reuters.com/technology/russias-yandex-says-it-repelled-biggest-ddos-attack-history-2021-09-09/.

Nguyen, H., Tran, K., Thomassey, S., and Hamad, M. (2021). Forecasting and anomaly detection approaches using lstm and lstm autoencoder techniques with the applications in supply chain management. *IJIM*, 57:38.

Nichols, D. (2016). The what and why of DDoS attacks. Acesso em: 07/2021. https://secura.cloud/insights/the-what-and-why-of-ddos-attacks.

Pelloso, M., Vergutz, A., Santos, A., and Nogueira, M. (2018). A self-adaptable system for DDoS attack prediction based on the metastability theory. In *GLOBECOM*, pages 1–6.

Santos, L. A. F., Campiolo, R., Gerosa, M. A., and Batista, D. M. (2013). Extração de alertas de segurança postados em mensagens de redes sociais. In *XXXI SBRC*, pages 791–804, Brasil.

Shumway, R. H. and Stoffer, D. S. (2017). *Characteristics of Time Series*, pages 1–44. Springer.

Srivastava, N., Mansimov, E., and Salakhutdinov, R. (2015). Unsupervised learning of video representations using lstms. *CoRR*.

Sumathi, S. and Karthikeyan, N. (2021). Detection of distributed denial of service using deep learning neural network. *JAIHC*, 12.

Tyson, M. (2022). Minecraft DDoS attack leaves small european country without Internet. Acesso em: 01/22. tomshardware.com/news/minecraft-ddos-attack-leaves-small-european-country-without-internet.

Wang, Z. and Zhang, Y. (2017). DDoS event forecasting using twitter data. In *IJCAI*, page 7.

Xie, X.-q., He, W.-P., Gu, B., Mei, Y., and Zhao, S.-s. (2019). Can kurtosis be an early warning signal for abrupt climate change? *Climate Dynamics*, 52.

Zhai, J.-H., Zhang, S., Chen, J., and He, Q. (2018). Autoencoder and its various variants. *SMC*, pages 415–419.